



ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ ΚΑΙ
ΤΕΧΝΟΛΟΓΙΑΣ (ΣΘΕΤ)

ΜΕΤΑΠΤΥΧΙΑΚΗ ΕΞΕΙΔΙΚΕΥΣΗ ΣΤΑ
ΠΛΗΡΟΦΟΡΙΑΚΑ ΣΥΣΤΗΜΑΤΑ

Διπλωματική Εργασία

Ανάπτυξη συστήματος συστάσεων για ομαδοποίηση φοιτητών σε
Διαδικτυακά περιβάλλοντα μάθησης

Ελένη Δαμάσκου

Επιβλέπων καθηγητής: Ροδάνθη Τσώνη

Πάτρα, Ιούνιος 2026

Η παρούσα εργασία αποτελεί πνευματική ιδιοκτησία του φοιτητή («συγγραφέας/δημιουργός») που την εκπόνησε. Στο πλαίσιο της πολιτικής ανοικτής πρόσβασης ο συγγραφέας/δημιουργός εκχωρεί στο ΕΑΠ, μη αποκλειστική άδεια χρήσης του δικαιώματος αναπαραγωγής, προσαρμογής, δημόσιου δανεισμού, παρουσίασης στο κοινό και ψηφιακής διάχυσής τους διεθνώς, σε ηλεκτρονική μορφή και σε οποιοδήποτε μέσο, για διδακτικούς και ερευνητικούς σκοπούς, άνευ ανταλλάγματος και για όλο το χρόνο διάρκειας των δικαιωμάτων πνευματικής ιδιοκτησίας. Η ανοικτή πρόσβαση στο πλήρες κείμενο για μελέτη και ανάγνωση δεν σημαίνει καθ' οιονδήποτε τρόπο παραχώρηση δικαιωμάτων διανοητικής ιδιοκτησίας του συγγραφέα/δημιουργού ούτε επιτρέπει την αναπαραγωγή, αναδημοσίευση, αντιγραφή, αποθήκευση, πώληση, εμπορική χρήση, μετάδοση, διανομή, έκδοση, εκτέλεση, «μεταφόρτωση» (downloading), «ανάρτηση» (uploading), μετάφραση, τροποποίηση με οποιονδήποτε τρόπο, τμηματικά ή περιληπτικά της εργασίας, χωρίς τη ρητή προηγούμενη έγγραφη συναίνεση του συγγραφέα/δημιουργού. Ο συγγραφέας/δημιουργός διατηρεί το σύνολο των ηθικών και περιουσιακών του δικαιωμάτων.



Ανάπτυξη συστήματος συστάσεων για ομαδοποίηση φοιτητών σε
Διαδικτυακά περιβάλλοντα μάθησης

Ελένη Δαμάσκου

Επιτροπή Επίβλεψης Διπλωματικής Εργασίας

Επιβλέπων Καθηγητής:

Ροδάνθη Τσώνη

ΣΕΠ ΕΑΠ

Συν-Επιβλέπων Καθηγητής:

Αθανάσιος Σύψας

ΣΕΠ ΕΑΠ

Πάτρα, Ιούνιος 2026

Περίληψη

Η παρούσα διπλωματική εργασία πραγματεύεται τον σχεδιασμό και την αξιολόγηση ενός πολυκριτηριακού συστήματος συστάσεων για τον σχηματισμό ομάδων σε περιβάλλοντα εξ αποστάσεως συνεργατικής μάθησης. Στόχος είναι η βελτιστοποίηση της σύνθεσης των ομάδων, λαμβάνοντας υπόψη τα πολυδιάστατα χαρακτηριστικά των φοιτητών. Η κεντρική συμβολή της εργασίας είναι τετραπλή: α) η πρόταση μιας ολοκληρωμένης μεθοδολογίας για την κατασκευή πολυδιάστατων προφίλ φοιτητών που συνδυάζουν δείκτες ακαδημαϊκής επίδοσης, συμμετοχής, κοινωνικής αλληλεπίδρασης και πιθανής απογοήτευσης, β) ο σχεδιασμός ενός ερμηνεύσιμου συστήματος συστάσεων, γ) η δημιουργία ενός συνθετικού συνόλου δεδομένων υψηλής δομικής πιστότητας για την αρχική αξιολόγηση και δ) η ανάπτυξη ενός πλαισίου αξιολόγησης βασισμένου σε μετρικές παιδαγωγικής λειτουργικότητας και δομικής ισορροπίας.

Για την υλοποίηση, η προτεινόμενη προσέγγιση αξιοποιεί μεθόδους Ανάλυσης Κοινωνικών Δικτύων (SNA) και Εκπαιδευτικής Εξόρυξης Δεδομένων και θεμελιώνεται στη Θεωρία της Κοινωνικής Αλληλεξάρτησης. Αναπτύχθηκε ένα συνθετικό σύνολο δεδομένων που προσομοιώνει αρχεία καταγραφής και βαθμολογίες από την πλατφόρμα Moodle. Για κάθε εικονικό φοιτητή κατασκευάστηκε ένα πολυδιάστατο προφίλ, το οποίο τροφοδοτεί έναν άπληστο (greedy) αλγόριθμο σχηματισμού ομάδων. Ο αλγόριθμος ενσωματώνει τα ευρήματα της ανάλυσης ομοφιλίας στα κριτήρια σύστασης και συγκρίνεται πειραματικά με δύο μεθόδους αναφοράς.

Στο πλαίσιο της υπολογιστικής αξιολόγησης, τα αποτελέσματα κατέδειξαν σαφή υπεροχή της προτεινόμενης μεθόδου. Όλες οι ομάδες που σχηματίστηκαν περιλάμβαναν τουλάχιστον ένα μέλος εκτός της κατηγορίας «Ανεπαρκής Αλληλεπίδραση». Παράλληλα, αξιοποίησε το σύνολο των διαθέσιμων φοιτητών του ενεργού πυρήνα (53 άτομα), επιτυγχάνοντας τον μέγιστο δυνατό αριθμό ομάδων με ενεργό πυρήνα. Παράλληλα, παρατηρήθηκαν σημαντικά χαμηλότερες αποκλίσεις στην κατανομή των χαρακτηριστικών των ομάδων. Η στατιστική ευστάθεια των αποτελεσμάτων διερευνήθηκε μέσω 100 τυχαίων εκτελέσεων, όπου η προτεινόμενη μέθοδος εμφάνισε σταθερή υπεροχή έναντι του τυχαίου σχηματισμού στις περισσότερες μετρικές αξιολόγησης. Η μεθοδολογία έχει σχεδιαστεί με στόχο τη γενίκευσή της, ενώ η μελλοντική έρευνα θα εστιάσει στην εφαρμογή και την παιδαγωγική αξιολόγησή της σε πραγματικά δεδομένα από το Moodle.

Λέξεις – Κλειδιά

Συστήματα συστάσεων, συνεργατική μάθηση, Moodle, ανάλυση κοινωνικών δικτύων, εκπαιδευτική εξόρυξη δεδομένων, σχηματισμός ομάδων

Development of a Recommender System for Grouping Students in Online Learning Environments

Eleni Damaskou

Abstract

This thesis addresses the design and evaluation of a multicriteria recommender system for student group formation in distance collaborative learning environments. Its aim is to optimize group composition by taking into account the multidimensional characteristics of students. The main contribution of the study is fourfold: (a) the proposal of an integrated methodology for constructing multidimensional student profiles that combine indicators of academic performance, participation, social interaction, and potential frustration; (b) the design of an interpretable recommender system; (c) the creation of a synthetic dataset with high structural fidelity for the initial evaluation of the methodology; and (d) the development of an evaluation framework based on metrics of pedagogical functionality and structural balance.

For its implementation, the proposed approach employs Social Network Analysis (SNA) and Educational Data Mining methods and is grounded in Social Interdependence Theory. A synthetic dataset was developed to simulate Moodle activity logs and grade records. A multidimensional profile was constructed for each virtual student and used as input to a greedy group formation algorithm. The algorithm incorporates the findings of the homophily analysis into its recommendation criteria and is experimentally compared with two baseline methods.

The computational evaluation demonstrated a clear advantage of the proposed method. All generated groups included at least one member who was not classified under the “Insufficient Interaction” category. In addition, the method distributed all 53 available active-core students across distinct groups, thereby achieving the maximum possible number of groups

with an active core. Substantially lower deviations were also observed in the distribution of group characteristics. The statistical stability of the results was examined through 100 random runs, in which the proposed method consistently outperformed random group formation across most evaluation metrics. The methodology was designed with generalizability in mind, while future research will focus on its application and pedagogical evaluation using real-world Moodle data.

Keywords

Recommender systems, collaborative learning, Moodle, social network analysis, educational data mining, group formation

Περιεχόμενα

Περίληψη.....	1
Περιεχόμενα	5
Κατάλογος Σχημάτων	10
Κατάλογος Πινάκων	11
Συντομογραφίες & Ακρωνύμια.....	13
1 Εισαγωγή.....	14
1.1 Το ερευνητικό πλαίσιο και το κενό	14
1.2 Σκοπός και επιμέρους στόχοι.....	15
1.3 Ερευνητικά ερωτήματα	17
1.4 Περιορισμοί της έρευνας.....	17
1.5 Δομή της διπλωματικής.....	19
2 Θεωρητικό υπόβαθρο.....	21
2.1 Συνεργατική μάθηση σε διαδικτυακά περιβάλλοντα	21
2.1.1 Βασικές αρχές και η θεωρία της Κοινωνικής Αλληλεξάρτησης.....	21
2.1.2 Ο σχηματισμός ομάδων ως κρίσιμος παράγοντας επιτυχίας	23
2.2 Το περιβάλλον Moodle ως πηγή δεδομένων αλληλεπίδρασης.....	25
2.2.1 Το Moodle ως περιβάλλον μάθησης και πεδίο εκπαιδευτικής έρευνας.....	25
2.2.2 Δεδομένα αλληλεπίδρασης (log files).....	26
2.3 Εκπαιδευτική Εξόρυξη Δεδομένων (Educational Data Mining – EDM).....	26
2.3.1 Ορισμός και κατηγορίες δεικτών	26
2.4 Ανάλυση κοινωνικών δικτύων (Social Network Analysis – SNA)	28
2.4.1 Αναπαράσταση της αλληλεπίδρασης των φοιτητών ως γράφος και μετρικές κεντρικότητας.....	28
2.4.2 Ανάδειξη λειτουργικών ρόλων.....	29
2.5 Ο συνδυασμός EDM και SNA ως ολοκληρωμένη προσέγγιση.....	30

2.6	Συστήματα συστάσεων στην εκπαίδευση	30
2.6.1	Εκπαιδευτικά Συστήματα Συστάσεων (Educational Recommender Systems - ERS)	30
2.6.2	Βασικές κατηγορίες συστημάτων συστάσεων	31
2.6.3	Συστήματα συστάσεων με επίγνωση πλαισίου (Context-Aware Recommender Systems - CARS)	35
2.6.4	Εφαρμογή σε συνεργατική μάθηση και σχηματισμό ομάδων	35
2.7	Σύνθεση: Ένα υβριδικό CARS πλαίσιο για βέλτιστο σχηματισμό ομάδων στο Moodle	36
3	Συγκριτική βιβλιογραφική ανασκόπηση προσεγγίσεων σχηματισμού ομάδων	38
3.1	Εισαγωγή και σκοπός της ανασκόπησης	38
3.2	Κατηγορίες προσεγγίσεων για σχηματισμό ομάδων.....	38
3.2.1	Προσεγγίσεις βάσει προφίλ και περιεχομένου (Profile/Content-Based)	40
3.2.2	Προσεγγίσεις βάσει συνεργατικού φιλτραρίσματος (Collaborative Filtering - CF)	41
3.2.3	Προσεγγίσεις βάσει κοινωνικού δικτύου (Social Network-Based)	41
3.2.4	Συστήματα βάσει γνώσης (Knowledge-Based Systems)	41
3.2.5	Υβριδικές προσεγγίσεις και συστήματα με επίγνωση πλαισίου	42
3.3	Αλγοριθμικές τεχνικές	42
3.3.1	Τεχνικές ομαδοποίησης (Clustering) για δημιουργία προφίλ.....	43
3.3.2	Αλγόριθμοι βελτιστοποίησης για σύνθεση ομάδων.....	43
3.4	Εντοπισμός ερευνητικού κενού.....	44
4	Μεθοδολογία ανάλυσης δεδομένων και δημιουργίας προφίλ φοιτητών	46
4.1	Δεδομένα και προεπεξεργασία.....	48
4.1.1	Χαρακτηριστικά των Moodle log files	48
4.1.2	Δημιουργία συνθετικού συνόλου δεδομένων	49
4.1.3	Καθαρισμός και φιλτράρισμα δεδομένων.....	52

4.2	Εξαγωγή χαρακτηριστικών και δημιουργία πολυδιάστατου προφίλ	55
4.2.1	Δείκτες συμμετοχής (Participation Features)	55
4.2.2	Δείκτες ακαδημαϊκής επίδοσης (Academic Performance Features)	56
4.2.3	Ανάλυση κοινωνικών δικτύων (Social Network Analysis)	58
4.2.3.1	Κατασκευή διμερούς γράφου (Bipartite Graph Construction)	58
4.2.3.2	Αποτελέσματα και ποιοτικός έλεγχος	60
4.2.3.3	Προβολή σε μονότροπο δίκτυο φοιτητών (One-mode Projection)	63
4.2.3.4	Υπολογισμός μετρικών κεντρικότητας	64
4.2.3.5	Αντιστοίχιση σε κοινωνικούς ρόλους	72
4.2.4	Σήματα απογοήτευσης (Frustration Signals)	81
4.2.5	Σύνθεση τελικού προφίλ	85
4.3	Στρωματοποίηση μαθησιακού προφίλ	90
4.3.1	Σκοπός και θεωρητικό πλαίσιο	90
4.3.2	Μεθοδολογία υπολογισμού σύνθετου δείκτη	90
4.3.3	Δημιουργία στρωμάτων μαθησιακού προφίλ	95
4.3.4	Αποτελέσματα και ερμηνεία στρωματοποίησης	96
5	Σχεδιασμός και υλοποίηση του συστήματος συστάσεων	104
5.1	Διατύπωση του προβλήματος ως πολυκριτηριακή σύσταση	104
5.2	Ομοφιλία και ετερογένεια στις προτεινόμενες ομάδες	105
5.2.1	Μεθοδολογία υπολογισμού	106
5.2.2	Εφαρμογή στο πειραματικό σύνολο δεδομένων	107
5.2.3	Συμπεράσματα για το συγκεκριμένο δίκτυο	107
5.3	Ο αλγόριθμος σύστασης ομάδων	108
5.3.1	Βασική λογική του αλγορίθμου	109
5.3.2	Είσοδος δεδομένων του αλγορίθμου	110
5.3.3	Υπολογισμός πλήθους και μεγέθους ομάδων	110
5.3.4	Υπολογισμός στόχων στρωματοποίησης	111

5.3.5	Ιεραρχική greedy τοποθέτηση βάσει ρόλων	111
5.3.6	Συνάρτηση κόστους: σκληροί περιορισμοί και ποινές	112
5.4	Πειραματική εφαρμογή και αξιολόγηση	117
5.4.1	Δεδομένα και παραμετροποίηση	117
5.4.2	Σενάρια σύγκρισης	117
5.4.3	Μετρικές αξιολόγησης	118
5.4.4	Αποτελέσματα για ομάδες 5 ατόμων	119
5.4.5	Αποτελέσματα για ομάδες 7 Ατόμων	121
5.4.6	Ανάλυση ευστάθειας και εμπειρική αξιολόγηση του τυχαίου σχηματισμού 122	
5.4.7	Συζήτηση και συμπεράσματα	126
6	Συμπεράσματα και Μελλοντικές Επεκτάσεις	128
6.1	Σύνοψη και απάντηση στα ερευνητικά ερωτήματα	128
6.2	Συμβολή της εργασίας	130
6.3	Περιορισμοί της έρευνας	131
6.4	Προτάσεις για μελλοντική έρευνα	132
6.5	Σχεδιασμός για μελλοντική ενσωμάτωση στο Moodle	134
6.5.1	Προϋποθέσεις ενσωμάτωσης	134
6.5.2	Ενδεικτική ροή λειτουργίας	135
	Βιβλιογραφία	136
	Παράρτημα Α Πηγαίος Κώδικας Υλοποίησης (Python Scripts)	145
A.1.	00_excel_to_csv.py	148
A.2.	01_filtering_quality_checks.py	149
A.3.	02_04_2_1_extract_participation_features.py	151
A.4.	03_04_2_2_extract_academic_performance_features.py	154
A.5.	04_sna_423_postcreated_edges_roles.py	157

A.6. 04b_sna_423_PREPARING_NODES_EDGES_FOR_GEPHI.py.....	161
A.7. 04c_4_2_3_student_student_weighted_pro.py	163
A.8. 04d_4_2_3_visualize_sna_metrics.py.....	167
A.9. 04e_4_2_3_assign_social_roles.py	169
A.10. 04f_4_2_3_validate_social_roles_kruskal_dunn.py	173
A.11. 04g_4_2_3_prepare_gephi_projected_roles_visualization.py	177
A.12. 06_frustration_signals.py	180
A.13. 07_04_2_5_final_student_profile.py	186
A.14. 07b_04_2_5_profile_visualization_REVISED.py	192
A.15. 07c_04_2_5_numerical_validation_report.py.....	197
A.16. 08_04_2_6_student_profile_stratification.py	204
A.17. 08b_04_3_strata_visualization_REVISED.py	213
A.18. 08c_04_3_2_weight_sensitivity_analysis.py	221
A.19. 09_HOMOPHILY_analysis.py	231
A.20. 10_05_3_grouping_ALGORITHM_v2.py	239
A.21. 11_05_4_experimental_evaluation.py	252
A.22. 12_05_4_repeated_random_evaluation.py	261
Παράρτημα Β Αναλυτικά αποτελέσματα στατιστικής και πειραματικής αξιολόγησης .	272
B.1 Πλήρη αποτελέσματα post-hoc ελέγχου Dunn	273
B.2 Πλήρη αποτελέσματα ανάλυσης ευστάθειας	275
B.2.1 Ομάδες 5 ατόμων.....	276
B.2.2 Ομάδες 7 ατόμων.....	278

Κατάλογος Σχημάτων

Σχήμα 4-1 Συνοπτική απεικόνιση της μεθοδολογικής ροής της εργασίας	47
Σχήμα 4-2 Δίκτυο αλληλεπιδράσεων στο forum σε διμερές μη-κατευθυνόμενο γράφο	62
Σχήμα 4-3 Boxplots μετρικών κεντρικότητας	70
Σχήμα 4-4 Ιστογράμματα μετρικών κεντρικότητας.....	71
Σχήμα 4-5 Κατανομή των μετρικών κεντρικότητας ανά κοινωνικό ρόλο.....	76
Σχήμα 4-6 Οπτικοποίηση του προβαλλόμενου και φιλτραρισμένου student–student δικτύου συνεργασίας στο Gephi.....	80
Σχήμα 4-7 Κατανομή χαρακτηριστικών συμμετοχής	87
Σχήμα 4-8 Κατανομή χαρακτηριστικών επίδοσης.....	88
Σχήμα 4-9 Κατανομή κοινωνικών ρόλων στο τελικό προφίλ.....	88
Σχήμα 4-10 Κατανομή χαρακτηριστικών απογοήτευσης	89
Σχήμα 4-11 Φοιτητές ανά στρώμα.....	98
Σχήμα 4-12 Κατανομή σύνθετου δείκτη προφίλ	99
Σχήμα 4-13 Κατανομή τυποποιημένου δείκτη προφίλ	99
Σχήμα 4-14 Σύνθετος δείκτης προφίλ ανά στρώμα	100
Σχήμα 4-15 Κοινωνικοί ρόλοι ανά στρώμα	100
Σχήμα 4-16 Μέσες τιμές διαστάσεων ανά στρώμα	101
Σχήμα 4-17 Μέσες τιμές βασικών μεταβλητών ανά στρώμα	101
Σχήμα 4-18 Κατανομή κοινωνικών ρόλων	102
Σχήμα 4-19 Σχέση σύνθετου δείκτη και τελικής επίδοσης.....	103
Σχήμα 5-1 Συνοπτική αναπαράσταση του αλγορίθμου ιεραρχικής greedy τοποθέτησης	116

Κατάλογος Πινάκων

Πίνακας 1-1 Αντιστοίχιση στόχων και μεθόδων	17
Πίνακας 2-1 Συγκριτική ανάλυση βασικών κατηγοριών συστημάτων συστάσεων για εκπαιδευτικούς σκοπούς	34
Πίνακας 3-1 Συγκριτική ανάλυση προσεγγίσεων σχηματισμού ομάδων σε εκπαιδευτικά περιβάλλοντα.....	40
Πίνακας 4-1 Βασικά πεδία ενός Moodle log file	49
Πίνακας 4-2 Χαρακτηριστικά του συνθετικού dataset	52
Πίνακας 4-3 Βήματα καθαρισμού δεδομένων	53
Πίνακας 4-4 Αποτελέσματα σταδίων φιλτραρίσματος δεδομένων.....	54
Πίνακας 4-5 Στατιστικά στοιχεία επεξεργασίας δεδομένων forum.....	61
Πίνακας 4-6 Στατιστικά στοιχεία προβαλλόμενου δικτύου φοιτητών	68
Πίνακας 4-7 Κριτήρια αντιστοίχισης κοινωνικών ρόλων.....	73
Πίνακας 4-8 Κατανομή κοινωνικών ρόλων φοιτητών στο προβαλλόμενο δίκτυο.....	74
Πίνακας 4-9 Αποτελέσματα ελέγχου Kruskal-Wallis.....	77
Πίνακας 4-10 Κατώτατες τιμές ορίων ανά κατηγορία.....	83
Πίνακας 4-11 Διαστάσεις και πεδία μαθησιακού προφίλ.....	87
Πίνακας 4-12 Σύνοψη αριθμητικής συνοχής του τελικού μαθησιακού προφίλ	89
Πίνακας 4-13 Βάρη χαρακτηριστικών ανά διάσταση	92
Πίνακας 4-14 Βάρη διαστάσεων σύνθετου δείκτη	93
Πίνακας 4-15 Συνοπτικά αποτελέσματα τοπικού ελέγχου ευαισθησίας βαρών.....	95
Πίνακας 4-16 Όρια στρωματοποίησης.....	96
Πίνακας 4-17 Κατανομή φοιτητών ανά στρώμα	96
Πίνακας 4-18 Μέσες τιμές βασικών μεταβλητών ανά στρώμα	97
Πίνακας 5-1 Δείκτες ομοφιλίας για κατηγορικά χαρακτηριστικά	107
Πίνακας 5-2 Δείκτες ομοφιλίας για αριθμητικά χαρακτηριστικά.....	107

Πίνακας 5-3 Βασικά στάδια αλγορίθμου ομαδοποίησης.....	109
Πίνακας 5-4 Σύνοψη παραμέτρων της συνάρτησης κόστους	115
Πίνακας 5-5 Συγκριτικά αποτελέσματα για ομάδες 5 ατόμων	120
Πίνακας 5-6 Συγκριτικά αποτελέσματα για ομάδες 7 ατόμων	121
Πίνακας 5-7 Στατιστικά πολλαπλών τυχαίων εκτελέσεων και συγκριτικά αποτελέσματα — ομάδες 5 ατόμων	123
Πίνακας 5-8 Στατιστικά πολλαπλών τυχαίων εκτελέσεων και συγκριτικά αποτελέσματα — ομάδες 7 ατόμων	124

Συντομογραφίες & Ακρωνύμια

API	Application Programming Interface (Διεπαφή Προγραμματισμού Εφαρμογών)
CARS	Context-Aware Recommender Systems (Συστήματα Συστάσεων με Επίγνωση Πλαισίου)
CF	Collaborative Filtering (Συνεργατικό Φιλτράρισμα)
CSV	Comma-Separated Values (Τιμές Διαχωριζόμενες με Κόμμα)
ΔΠΜ	Διαδικτυακά Περιβάλλοντα Μάθησης
EDM	Educational Data Mining (Εκπαιδευτική Εξόρυξη Δεδομένων)
ERS	Educational Recommender Systems (Εκπαιδευτικά Συστήματα Συστάσεων)
GA	Genetic Algorithms (Γενετικοί Αλγόριθμοι)
GDPR	General Data Protection Regulation (Γενικός Κανονισμός Προστασίας Δεδομένων)
LA	Learning Analytics (Αναλυτική Μάθησης)
LLM	Large Language Model (Μεγάλο Γλωσσικό Μοντέλο)
LMS	Learning Management System (Σύστημα Διαχείρισης Μάθησης)
NLP	Natural Language Processing (Επεξεργασία Φυσικής Γλώσσας)
RS	Recommender Systems (Συστήματα Συστάσεων)
SNA	Social Network Analysis (Ανάλυση Κοινωνικών Δικτύων)
ΕΑΠ	Ελληνικό Ανοικτό Πανεπιστήμιο

1 Εισαγωγή

1.1 Το ερευνητικό πλαίσιο και το κενό

Τα Διαδικτυακά Περιβάλλοντα Μάθησης (ΔΠΜ), και ειδικότερα τα Συστήματα Διαχείρισης Μάθησης (LMS) όπως το Moodle, αποτελούν θεμελιώδη εργαλεία της σύγχρονης τριτοβάθμιας εκπαίδευσης (Tsoni, Panagiotakopoulos, et al., 2022). Πέρα από τη διάθεση εκπαιδευτικού υλικού, υποστηρίζουν διαπροσωπικές αλληλεπιδράσεις μέσω φόρουμ, ομαδικών εργασιών και κουίζ, μετασχηματίζοντας το LMS σε ενεργό μαθησιακό χώρο.

Η συνεργατική μάθηση αναγνωρίζεται ως ιδιαίτερα αποτελεσματική παιδαγωγική προσέγγιση, καθώς ενισχύει την ενεργό συμμετοχή, την ανάπτυξη κοινωνικών δεξιοτήτων, την κριτική σκέψη, ενώ προάγει τη βαθύτερη κατανόηση μέσω της κοινωνικής κατασκευής της γνώσης, δηλαδή μιας διαδικασίας κατά την οποία οι εκπαιδευόμενοι οικοδομούν από κοινού τη γνώση μέσω της αλληλεπίδρασης και της ανταλλαγής απόψεων (Butera & Buchs, 2019). Στο ψηφιακό περιβάλλον, η εφαρμογή της καθίσταται δυνατή ακριβώς μέσω των λειτουργιών των LMS.

Ωστόσο, ένας κρίσιμος και συχνά υποτιμημένος παράγοντας για την επιτυχή εφαρμογή της συνεργατικής μάθησης είναι ο αποτελεσματικός σχηματισμός ομάδων. Ο παραδοσιακός, τυχαίος ή χειροκίνητος σχηματισμός αγνοεί παραμέτρους όπως η συμπεριφορά, η συμμετοχή, οι επιδόσεις ή οι κοινωνικές αλληλεπιδράσεις των φοιτητών ή απαιτεί πολύ καλή γνώση αυτών των παραμέτρων από τον καθηγητή το οποίο όμως είναι δύσκολο να συμβεί σε διαδικτυακά περιβάλλοντα μάθησης και ιδιαίτερα σε μεγάλα ακροατήρια. Αυτό μπορεί να οδηγήσει σε ανισορροπίες που υπονομεύουν τη συνεργατική δυναμική: ομάδες με παθητικά μέλη, έλλειψη συμπληρωματικών δεξιοτήτων ή κοινωνική απομόνωση.

Το Moodle, παρότι προσφέρει εκτενείς λειτουργίες διαχείρισης μαθημάτων, δεν διαθέτει εγγενή ευφυή εργαλεία για αυτόματο και βελτιστοποιημένο σχηματισμό ομάδων βασισμένο σε αντικειμενικά δεδομένα συμπεριφοράς. Παράλληλα, η συνεχής αλληλεπίδραση των φοιτητών με την πλατφόρμα παράγει μεγάλο όγκο δεδομένων καταγραφής (log files), τα οποία αποτυπώνουν λεπτομερώς τη μαθησιακή δραστηριότητα.

Η συστηματική ανάλυση αυτών των δεδομένων μέσω τεχνικών Εκπαιδευτικής Εξόρυξης Δεδομένων (EDM) και Ανάλυσης Μάθησης (Learning Analytics) (Long & Siemens, 2011)

θα μπορούσε να προσφέρει πολύτιμες πληροφορίες για τη συμπεριφορά των φοιτητών. Επιπλέον, η Ανάλυση Κοινωνικών Δικτύων (SNA) θα μπορούσε να αποτυπώσει τη δομή των αλληλεπιδράσεων και να αναδείξει λανθάνοντα χαρακτηριστικά και πρότυπα κοινωνικής συμπεριφοράς των φοιτητών (Tsoni, Panagiotakopoulos, et al., 2022). Ωστόσο, στη βιβλιογραφία διαπιστώνεται ότι οι τεχνικές EDM, SNA και οι αλγόριθμοι συστημάτων συστάσεων σπάνια αξιοποιούνται συνδυαστικά. Το ερευνητικό κενό εντοπίζεται στην απουσία ολοκληρωμένων υβριδικών προσεγγίσεων που να αξιοποιούν τα log files του Moodle συνδυάζοντας EDM, SNA και Συστήματα Συστάσεων με Επίγνωση Πλαισίου (CARS) (Adomavicius & Tuzhilin, 2011) για τον αυτόματο σχηματισμό συνεργατικών ομάδων βάσει πολλαπλών κριτηρίων.

1.2 Σκοπός και επιμέρους στόχοι

Κύριος σκοπός της διπλωματικής εργασίας είναι ο σχεδιασμός και η τεκμηρίωση ενός πλαισίου για ένα ευφυές σύστημα συστάσεων που υποστηρίζει τον αυτόματο, πολυκριτηριακό και παιδαγωγικά τεκμηριωμένο σχηματισμό ομάδων φοιτητών εντός του Moodle.

Επιμέρους στόχοι:

1. Να διερευνηθούν οι θεωρητικές βάσεις της συνεργατικής μάθησης, των συστημάτων συστάσεων σε εκπαιδευτικό πλαίσιο και των τεχνικών ανάλυσης δεδομένων από διαδικτυακά περιβάλλοντα μάθησης.
2. Να πραγματοποιηθεί συγκριτική επισκόπηση υφιστάμενων αλγοριθμικών προσεγγίσεων για τον σχηματισμό ομάδων σε εκπαιδευτικά περιβάλλοντα.
3. Να οριστεί μεθοδολογία για την εξαγωγή χαρακτηριστικών μέσω EDM από τα log files του Moodle και την κατασκευή πολυδιάστατων προφίλ φοιτητών, που να ενσωματώνουν μετρικές συμμετοχής, επιδόσεων, κοινωνικής αλληλεπίδρασης και συμπεριφορικών δεικτών απογοήτευσης.
4. Να σχεδιαστεί ο αλγοριθμικός πυρήνας ενός υβριδικού συστήματος συστάσεων που διατυπώνει τον σχηματισμό ομάδων ως πολυκριτηριακή διαδικασία σύστασης, επιδιώκοντας ομοιογένεια ως προς το κίνητρο και ετερογένεια ως προς τις δεξιότητες και τους κοινωνικούς ρόλους (Butera & Buchs, 2019).
5. Να προταθεί πλαίσιο αξιολόγησης του συστήματος με τεχνικές, συμπεριφορικές και ακαδημαϊκές μετρικές.

6. Να διερευνηθούν οι τεχνικές προϋποθέσεις για την πιθανή ενσωμάτωση του συστήματος στο Moodle, είτε ως πρόσθετο (plugin) είτε ως εξωτερική υπηρεσία (API).

Η αντιστοίχιση των επιμέρους στόχων με το θεωρητικό υπόβαθρο, τη μεθοδολογική προσέγγιση και το αναμενόμενο αποτέλεσμα παρουσιάζεται στον παρακάτω πίνακα (Πίνακας 1-1).

Επιμέρους στόχος	Θεωρητικό Υπόβαθρο	Μεθοδολογική Προσέγγιση	Αναμενόμενο Αποτέλεσμα
Στόχος 1	Βιβλιογραφική ανάλυση log files	Συστηματική μελέτη	Σύνθεση θεωρητικού πλαισίου για συνεργατική μάθηση, συστήματα συστάσεων και EDM/SNA
Στόχος 2	Συγκριτική ανάλυση	Βιβλιογραφική επισκόπηση	Συγκριτική ανασκόπηση υφιστάμενων προσεγγίσεων σχηματισμού ομάδων
Στόχος 3	EDM, SNA	Ανάλυση log files	Δημιουργία ενιαίου, πολυδιάστατου προφίλ φοιτητή με δείκτες συμμετοχής, επιδόσεων, κοινωνικής αλληλεπίδρασης και συναισθηματικών δεικτών
Στόχος 4	CARS, βελτιστοποίηση	Σχεδιασμός αλγορίθμου	Αλγοριθμικό μοντέλο που διατυπώνει τον σχηματισμό ομάδων ως πολυκριτηριακό πρόβλημα βελτιστοποίησης
Στόχος 5	Μεθοδολογία αξιολόγησης	Ορισμός μετρικών	Πλαίσιο αξιολόγησης με τεχνικές, συμπεριφορικές και ακαδημαϊκές μετρικές

Στόχος 6	Αρχιτεκτονική λογισμικού	Τεχνική ανάλυση	Προτάσεις για πρακτική ενσωμάτωση στο Moodle (plugin/API)
----------	-----------------------------	-----------------	---

Πίνακας 1-1 Αντιστοίχιση στόχων και μεθόδων

1.3 Ερευνητικά ερωτήματα

Η έρευνα διαμορφώνει πέντε κύρια ερευνητικά ερωτήματα (ΕΕ):

ΕΕ1: Ποια δεδομένα αλληλεπίδρασης του Moodle είναι καταλληλότερα για την εξαγωγή αξιόπιστων δεικτών μαθησιακής και κοινωνικής συμπεριφοράς φοιτητών;

ΕΕ2: Πώς μπορούν οι δείκτες αυτοί να συνδυαστούν σε ένα ενιαίο, πολυδιάστατο προφίλ φοιτητή που να υποστηρίζει τον σχηματισμό συνεργατικών ομάδων;

ΕΕ3: Πώς μπορεί η Ανάλυση Κοινωνικών Δικτύων να συμβάλει στον εντοπισμό λειτουργικών ρόλων και στην αποτύπωση του τρόπου διασύνδεσης και συμμετοχής των φοιτητών στη μαθησιακή κοινότητα;

ΕΕ4: Πώς μπορεί να διαμορφωθεί ένα αλγοριθμικό μοντέλο που διατυπώνει τον σχηματισμό ομάδων ως πολυκριτηριακό πρόβλημα σύστασης, λαμβάνοντας υπόψη παιδαγωγικούς και συμπεριφορικούς παράγοντες;

ΕΕ5: Με ποιες πολυδιάστατες μετρικές μπορεί να αξιολογηθεί η αποτελεσματικότητα του προτεινόμενου συστήματος συστάσεων;

1.4 Περιορισμοί της έρευνας

Η έρευνα επικεντρώνεται στο περιβάλλον του Moodle και στα δεδομένα δραστηριότητας και επίδοσης που αυτό παράγει. Η εφαρμογή της προτεινόμενης μεθοδολογίας σε άλλα Συστήματα Διαχείρισης Μάθησης (LMS) απαιτεί τεχνικές προσαρμογές, κυρίως ως προς την αντιστοίχιση της δομής και της σημασιολογίας των πεδίων των αρχείων καταγραφής.

Λόγω νομικών, ηθικών και θεσμικών περιορισμών πρόσβασης σε πραγματικά εκπαιδευτικά δεδομένα, καθώς και των απαιτήσεων προστασίας προσωπικών δεδομένων, η ανάλυση βασίζεται σε συνθετικά δεδομένα. Τα δεδομένα αυτά παρουσιάζουν υψηλή δομική πιστότητα

ως προς τα αυθεντικά αρχεία του Moodle, καθώς η παραγωγή τους βασίστηκε σε σαφείς προδιαγραφές από την επίσημη τεκμηρίωση της πλατφόρμας. Ωστόσο, η δομική πιστότητα δεν συνεπάγεται πλήρη συμπεριφορική εγκυρότητα: τα συνθετικά δεδομένα δεν αποτυπώνουν πλήρως την πολυπλοκότητα της πραγματικής ανθρώπινης συμπεριφοράς, ιδίως ως προς τη χρονική κατανομή της δραστηριότητας και τα πρότυπα αλληλεπίδρασης. Η βασική αξία της εργασίας έγκειται στη μεθοδολογία που προτείνεται, η οποία είναι σχεδιασμένη ώστε να εφαρμόζεται σε πραγματικά δεδομένα Moodle υπό τις απαιτούμενες διαδικασίες ψευδοανωνυμοποίησης και προστασίας προσωπικών δεδομένων.

Η επεξεργασία αξιοποιεί τα δύο βασικά αρχεία που καταγράφονται και εξάγονται μέσω των τυπικών λειτουργιών του Moodle: το αρχείο καταγραφής δραστηριότητας και το αρχείο βαθμολογιών. Η χρήση μόνο αυτών των δύο αρχείων συνιστά περιορισμό, καθώς δεν αξιοποιούνται δεδομένα αυτοαναφοράς (π.χ., κίνητρα, στάσεις, συναισθήματα). Οι διαστάσεις αυτές προσεγγίζονται έμμεσα μέσω παρατηρήσιμων συμπεριφορικών δεικτών (συχνότητα και ένταση αλληλεπίδρασης, συνέπεια συμμετοχής), οι οποίοι αποτελούν ενδείξεις και όχι άμεσες μετρήσεις της υποκείμενης εμπλοκής των φοιτητών.

Ένας επιπλέον σημαντικός περιορισμός είναι ότι η αξιολόγηση του συστήματος είναι αποκλειστικά υπολογιστική (a priori) και βασίζεται σε μετρικές που αξιολογούν τη σύνθεση των ομάδων πριν από την έναρξη της συνεργασίας. Δεν εξετάζεται σε πραγματικό εκπαιδευτικό περιβάλλον αν οι προτεινόμενες ομάδες συνεργάστηκαν αποτελεσματικότερα ή πέτυχαν υψηλότερα μαθησιακά αποτελέσματα. Η παιδαγωγική επικύρωση της μεθόδου σε πραγματικές συνθήκες αποτελεί, επομένως, απαραίτητη κατεύθυνση για μελλοντική έρευνα.

Παρά τους περιορισμούς αυτούς, η προτεινόμενη μεθοδολογία προσφέρει ένα αυτοματοποιημένο, επαναλήψιμο και μεταφέρσιμο πλαίσιο για τον σχηματισμό ομάδων, το οποίο μπορεί να εφαρμοστεί σε διαφορετικά σύνολα δεδομένων Moodle μετά από κατάλληλη προσαρμογή, χωρίς να απαιτείται πρόσθετη συλλογή δεδομένων μέσω ερωτηματολογίων ή συνεντεύξεων, γεγονός που ενισχύει τη συμβατότητά του με τις απαιτήσεις προστασίας προσωπικών δεδομένων.

1.5 Δομή της διπλωματικής

Η εργασία οργανώνεται σε έξι κεφάλαια, ακολουθώντας μια λογική πορεία από τη θεμελίωση του προβλήματος έως την εξαγωγή συμπερασμάτων.

Κεφάλαιο 2: Θεωρητικό υπόβαθρο – Παρουσιάζει αναλυτικά τις θεωρητικές βάσεις που θεμελιώνουν την προτεινόμενη προσέγγιση. Εξετάζονται οι αρχές της συνεργατικής μάθησης και η θεωρία Κοινωνικής Αλληλεξάρτησης, το περιβάλλον Moodle ως πηγή δεδομένων αλληλεπίδρασης, οι μεθοδολογίες Εκπαιδευτικής Εξόρυξης Δεδομένων (EDM) και Ανάλυσης Κοινωνικών Δικτύων (SNA) και τα Συστήματα Συστάσεων με Επίγνωση Πλαισίου (CARS). Η σύνθεση των θεωριών αυτών αναδεικνύει την αναγκαιότητα μιας υβριδικής προσέγγισης για τον σχηματισμό ομάδων.

Κεφάλαιο 3: Συγκριτική βιβλιογραφική ανασκόπηση – Αναλύει και συγκρίνει υφιστάμενες προσεγγίσεις σχηματισμού ομάδων σε εκπαιδευτικά περιβάλλοντα. Μέσω συστηματικής κατηγοριοποίησης και κριτικής ανάλυσης, αναδεικνύονται τα πλεονεκτήματα και οι περιορισμοί κάθε προσέγγισης, τεκμηριώνοντας το ερευνητικό κενό που η παρούσα εργασία καλείται να καλύψει.

Κεφάλαιο 4: Μεθοδολογία ανάλυσης δεδομένων και δημιουργίας προφίλ φοιτητών – Περιγράφει λεπτομερώς τη διαδικασία επεξεργασίας των log files του Moodle, τη δημιουργία συνθετικών δεδομένων υψηλής πιστότητας, και την εξαγωγή δεικτών συμμετοχής, επίδοσης, κοινωνικών ρόλων (μέσω SNA) και σημάτων απογοήτευσης. Η ενότητα ολοκληρώνεται με τη σύνθεση των επιμέρους δεικτών σε ένα ενιαίο πολυδιάστατο προφίλ και τη στρωματοποίηση των φοιτητών.

Κεφάλαιο 5: Σχεδιασμός και υλοποίηση του συστήματος συστάσεων – Αναλύει τον σχεδιασμό του greedy αλγορίθμου σύστασης ομάδων τεσσάρων σταδίων, ο οποίος ενσωματώνει πολυκριτηριακά κριτήρια βασισμένα στη θεωρία Κοινωνικής Αλληλεξάρτησης. Παρουσιάζεται η πειραματική εφαρμογή του αλγορίθμου σε δύο σενάρια μεγέθους ομάδας 5 και 7 ατόμων και η συγκριτική αξιολόγηση έναντι ενός τυχαίου αλλά και ενός μονοκριτηριακού σχηματισμού, με χρήση μετρικών παιδαγωγικής λειτουργικότητας και δομικής ισορροπίας.

Κεφάλαιο 6: Συμπεράσματα και μελλοντικές επεκτάσεις – Συνοψίζει τα κύρια ευρήματα της έρευνας, απαντά στα ερευνητικά ερωτήματα, αναδεικνύει τη συμβολή της εργασίας στη

βιβλιογραφία, συζητά τους περιορισμούς της προσέγγισης και προτείνει συγκεκριμένες κατευθύνσεις για μελλοντική έρευνα, όπως η εφαρμογή σε πραγματικά δεδομένα, η διερεύνηση εναλλακτικών αλγορίθμων και η πρακτική ενσωμάτωση στο Moodle.

2 Θεωρητικό υπόβαθρο

Το παρόν κεφάλαιο παρουσιάζει το θεωρητικό πλαίσιο που θεμελιώνει την υβριδική προσέγγιση της εργασίας. Εξετάζονται διαδοχικά οι αρχές της συνεργατικής μάθησης, το περιβάλλον Moodle ως πηγή δεδομένων αλληλεπίδρασης, οι μεθοδολογίες Εκπαιδευτικής Εξόρυξης Δεδομένων (EDM) και Ανάλυσης Κοινωνικών Δικτύων (SNA), και τα Συστήματα Συστάσεων με Επίγνωση Πλαισίου (CARS), τα οποία αποτελούν τη βάση για τον σχεδιασμό του προτεινόμενου συστήματος.

2.1 Συνεργατική μάθηση σε διαδικτυακά περιβάλλοντα

2.1.1 Βασικές αρχές και η θεωρία της Κοινωνικής Αλληλεξάρτησης

Η συνεργατική μάθηση αποτελεί δομημένη παιδαγωγική στρατηγική κατά την οποία οι εκπαιδευόμενοι εργάζονται σε μικρές ομάδες για την επίτευξη κοινού μαθησιακού στόχου, με σκοπό την ενίσχυση τόσο της ατομικής μάθησης όσο και της συλλογικής απόδοσης. Αν και η χρήση μικρών ομάδων για μάθηση ανάγεται στην αρχαιότητα, η σύγχρονη εκπαιδευτική εφαρμογή της συνεργατικής μάθησης άρχισε να αναπτύσσεται από τη δεκαετία του 1960 και γνώρισε ραγδαία διάδοση από τη δεκαετία του 1980 και μετά, αποτελώντας σήμερα μία από τις πλέον τεκμηριωμένες παιδαγωγικές προσεγγίσεις (Johnson & Johnson, 2009).

Η θεωρητική θεμελίωση της συνεργατικής μάθησης στηρίζεται κατά κύριο λόγο στη Θεωρία της Κοινωνικής Αλληλεξάρτησης (Social Interdependence Theory). Οι ρίζες της ανιχνεύονται στις αρχές του 20ού αιώνα, στη σχολή της ψυχολογίας Gestalt, όπου ο Kurt Koffka διατύπωσε την ιδέα ότι οι ομάδες αποτελούν δυναμικά σύνολα, στα οποία η αλληλεξάρτηση μεταξύ των μελών μπορεί να ποικίλλει (Deutsch, 1968). Ο Kurt Lewin επέκτεινε αυτή τη σκέψη, προτείνοντας ότι η ουσία μιας ομάδας έγκειται στην αλληλεξάρτηση των μελών της, η οποία δημιουργείται μέσω κοινών στόχων και καθιστά την ομάδα ένα «δυναμικό όλον», όπου μια αλλαγή στην κατάσταση οποιουδήποτε μέλους επηρεάζει την κατάσταση όλων των άλλων (Lewin, 1948· Johnson & Johnson, 2009).

Ο Morton Deutsch, μαθητής του Lewin, θεμελίωσε τη σύγχρονη θεωρία στα τέλη της δεκαετίας του 1940, εξετάζοντας τον τρόπο με τον οποίο η επίτευξη των στόχων διαφορετι-

κών ατόμων μπορεί να συνδέεται θετικά ή αρνητικά (Deutsch, 1949, 1962). Ο Deutsch διέκρινε δύο θεμελιώδεις τύπους κοινωνικής αλληλεξάρτησης, οι οποίοι καθορίζουν τη δυναμική της συνεργασίας ή του ανταγωνισμού:

- **Θετική Αλληλεξάρτηση (Συνεργασία):** Υπάρχει όταν υπάρχει θετική συσχέτιση μεταξύ της επίτευξης των στόχων των μελών. Το άτομο αντιλαμβάνεται ότι μπορεί να επιτύχει τον στόχο του μόνο εφόσον επιτύχουν και οι συνεργάτες του (Deutsch, 1962). Αυτή η μορφή αλληλεξάρτησης οδηγεί σε προωθητική αλληλεπίδραση (promotive interaction), όπου τα μέλη διευκολύνουν το ένα το έργο του άλλου (Johnson & Johnson, 2009).
- **Αρνητική Αλληλεξάρτηση (Ανταγωνισμός):** Υπάρχει όταν υπάρχει αρνητική συσχέτιση μεταξύ της επίτευξης των στόχων — η επιτυχία του ενός συνεπάγεται την αποτυχία των άλλων. Αυτή η μορφή οδηγεί σε αντιτιθέμενη αλληλεπίδραση (oppositional interaction), όπου τα μέλη αποθαρρύνουν ή παρεμποδίζουν το έργο των άλλων (Deutsch, 1962· Johnson & Johnson, 2009).

Εκτός από τους δύο αυτούς τύπους, η θεωρία εξετάζει και την κατάσταση μη αλληλεξάρτησης (no interdependence), όπου η επίτευξη του στόχου ενός ατόμου είναι ανεξάρτητη από τους στόχους των άλλων, οδηγώντας σε ατομικιστική συμπεριφορά (Deutsch, 1962· Johnson & Johnson, 2009). Σύμφωνα με τη βασική παραδοχή της θεωρίας, ο τρόπος με τον οποίο διαρθρώνονται οι στόχοι των συμμετεχόντων καθορίζει τον τρόπο αλληλεπίδρασής τους και, συνεπώς, τα αποτελέσματα της μαθησιακής κατάστασης (Deutsch, 1949, 1962).

Για να είναι μια συνεργατική κατάσταση αποτελεσματική, οι Johnson και Johnson (2009) προσδιορίζουν πέντε βασικά στοιχεία που πρέπει να δομηθούν κατάλληλα: (α) θετική αλληλεξάρτηση (οι στόχοι των μελών είναι συνδεδεμένοι θετικά), (β) ατομική υπευθυνότητα (κάθε μέλος είναι υπεύθυνο για τη συμβολή του), (γ) προωθητική αλληλεπίδραση (τα μέλη υποστηρίζουν και ενθαρρύνουν το έργο του άλλου), (δ) κοινωνικές δεξιότητες (τα μέλη διαθέτουν τις απαραίτητες δεξιότητες επικοινωνίας και συνεργασίας) και (ε) ομαδική επεξεργασία (τα μέλη αναστοχάζονται και αξιολογούν τη λειτουργία της ομάδας).

Το εκτενές σώμα ερευνών και οι συνθετικές αναλύσεις που παρουσιάζουν οι Johnson και Johnson (1989, 2009) υποστηρίζουν ότι η θετική αλληλεξάρτηση και η προωθητική αλληλεπίδραση συνδέονται με υψηλότερα μαθησιακά αποτελέσματα, ισχυρότερες διαπροσωπικές σχέσεις και θετικότερα αποτελέσματα ψυχολογικής προσαρμογής, σε σύγκριση με τις ανταγωνιστικές ή ατομικιστικές δομές.

Η παρούσα εργασία επιλέγει τη Θεωρία της Κοινωνικής Αλληλεξάρτησης ως το κεντρικό παιδαγωγικό πλαίσιο για τον σχηματισμό των ομάδων για δύο βασικούς λόγους. Πρώτον, η θεωρία παρέχει ένα ολοκληρωμένο πλαίσιο για την κατανόηση των συνθηκών που ευνοούν την αποτελεσματική συνεργασία, μέσω των πέντε βασικών στοιχείων που προσδιορίζουν οι Johnson και Johnson (2009). Δεύτερον, η θεωρία αναγνωρίζει ότι η δομή των στόχων —η οποία καθορίζεται από τον σχεδιασμό της δραστηριότητας, τις οδηγίες, την κατανομή εργασιών και πόρων, τον τρόπο αξιολόγησης και την ατομική και συλλογική ευθύνη— αποτελεί καθοριστικό παράγοντα για την ποιότητα της συνεργασίας. Στην παρούσα εργασία, ο αλγόριθμος σχηματισμού ομάδων επιδιώκει να δημιουργήσει ευνοϊκές προϋποθέσεις για θετική αλληλεξάρτηση, μέσω της ισορροπημένης κατανομής των επιπέδων συμμετοχής και επίδοσης. Παράλληλα, η Ανάλυση Κοινωνικών Δικτύων (SNA) αξιοποιείται για την ανάδειξη λανθανόντων χαρακτηριστικών και προτύπων κοινωνικής συμπεριφοράς των φοιτητών (Tsoni, Panagiotakopoulos, et al., 2022), τα χαρακτηριστικά αυτά αξιοποιούνται στη συνέχεια για τη σύνθεση ομάδων με λειτουργική συμπληρωματικότητα ρόλων. Η προσέγγιση αυτή ευθυγραμμίζεται με τη λογική ότι ο σχηματισμός των ομάδων δεν εγκαθιδρύει από μόνος του τη θετική αλληλεξάρτηση, αλλά μπορεί να δημιουργήσει τις συνθήκες εκείνες που την ευνοούν, σε συνδυασμό με τον παιδαγωγικό σχεδιασμό της συνεργατικής δραστηριότητας.

2.1.2 Ο σχηματισμός ομάδων ως κρίσιμος παράγοντας επιτυχίας

Ο τρόπος σχηματισμού των ομάδων επηρεάζει σημαντικά τη δυναμική και την αποτελεσματικότητά τους. Ο τυχαίος σχηματισμός δεν λαμβάνει συστηματικά υπόψη τα χαρακτηριστικά των εκπαιδευομένων, ενώ ο χειροκίνητος σχηματισμός προϋποθέτει ότι ο διδάσκων διαθέτει επαρκή και επικαιροποιημένη γνώση για τους φοιτητές, προϋπόθεση που καθίσταται ιδιαίτερα δύσκολο να ικανοποιηθεί σε μεγάλα ή διαδικτυακά ακροατήρια (Maqtary et al., 2019).

Με βάση τη σύνθεση της σχετικής βιβλιογραφίας και τις παιδαγωγικές αρχές της Θεωρίας της Κοινωνικής Αλληλεξάρτησης, στην παρούσα εργασία επιλέγονται τρεις άξονες ως ιδιαίτερα σχετικοί με τη σύνθεση και λειτουργία των συνεργατικών ομάδων:

- **Ισορροπία και ετερογένεια ακαδημαϊκών χαρακτηριστικών:** Η σύνθεση μιας ομάδας με μέλη που παρουσιάζουν διαφορετικά επίπεδα προηγούμενη γνώσης και επίδοσης μπορεί να δημιουργήσει δυνατότητες αλληλοσυμπλήρωσης και αμοιβαίας

υποστήριξης (Butera & Buchs, 2019). Η προηγούμενη γνώση και η επίδοση συγκαταλέγονται στα χαρακτηριστικά που χρησιμοποιούνται συχνά στις προσεγγίσεις σχηματισμού συνεργατικών ομάδων (Maqtary et al., 2019).

- **Δομή κοινωνικής αλληλεπίδρασης και λειτουργικές θέσεις συμμετοχής:** Η ανάλυση των προτύπων αλληλεπίδρασης μεταξύ των φοιτητών σε διαδικτυακά περιβάλλοντα μάθησης μπορεί να αναδείξει λανθάνοντα χαρακτηριστικά και πρότυπα κοινωνικής συμπεριφοράς (Tsoni, Panagiotakopoulos, et al., 2022). Οι μετρικές κεντρικότητας που υπολογίζονται μέσω Ανάλυσης Κοινωνικών Δικτύων (SNA) αποτυπώνουν διαφορετικές όψεις της δομικής θέσης κάθε φοιτητή και μπορούν συνδυαστικά να υποστηρίξουν τη μελέτη της συμμετοχής και των ρόλων σε διαδικτυακά φόρουμ (Rabbany et al., 2014).
- **Ισορροπία στη συμμετοχή:** Η έντονα άνιση κατανομή της δραστηριότητας εντός μιας ομάδας μπορεί να περιορίσει τις ευκαιρίες ουσιαστικής συνεργασίας και να συνδεθεί με χαμηλότερη ικανοποίηση των μελών (Strauß & Rummel, 2021). Η ισορροπία δεν προϋποθέτει απολύτως ίση ποσοτική συμμετοχή, αλλά την αποφυγή καταστάσεων στις οποίες ορισμένα μέλη κυριαρχούν συστηματικά, ενώ άλλα παραμένουν περιφερειακά ή παθητικά.

Παράλληλα, η συναισθηματική κατάσταση των φοιτητών κατά τη διάρκεια της μάθησης αποτελεί μια σημαντική διάσταση της μαθησιακής εμπειρίας. Η παρατεταμένη ή μη ρυθμιζόμενη απογοήτευση (frustration) μπορεί να περιορίσει την επιμονή και την εμπλοκή, επηρεάζοντας αρνητικά τη μαθησιακή διαδικασία (Pekrun, 2006· D'Mello & Graesser, 2012). Αν και τα αρχεία καταγραφής του Moodle δεν μετρούν άμεσα το συναίσθημα, μπορούν να αποτυπώσουν συμπεριφορικά μοτίβα που ερμηνεύονται ως πιθανά σήματα απογοήτευσης (Tsoni et al., 2026). Για τον λόγο αυτό, η διάσταση αυτή προστίθεται ως συμπληρωματική συνιστώσα του πολυδιάστατου προφίλ, εμπλουτίζοντας την αποτύπωση της μαθησιακής συμπεριφοράς των φοιτητών.

Οι τρεις αυτοί άξονες και η συμπληρωματική συναισθηματική διάσταση δεν προέρχονται από μία μόνο μελέτη, αλλά προκύπτουν από τη σύνθεση της σχετικής θεωρητικής και εμπειρικής βιβλιογραφίας. Παρέχουν τη θεωρητική βάση για την επιλογή των διαστάσεων που θα ενσωματωθούν στο πολυδιάστατο προφίλ των φοιτητών στα επόμενα κεφάλαια.

2.2 Το περιβάλλον Moodle ως πηγή δεδομένων αλληλεπίδρασης

2.2.1 Το Moodle ως περιβάλλον μάθησης και πεδίο εκπαιδευτικής έρευνας

Το Moodle (Modular Object-Oriented Dynamic Learning Environment) είναι ένα από τα πλέον διαδεδομένα Συστήματα Διαχείρισης Μάθησης (LMS) ανοικτού κώδικα παγκοσμίως. Σύμφωνα με την επίσημη ανακοίνωση του Δεκεμβρίου 2025, οι χρήστες σε καταχωρισμένες εγκαταστάσεις Moodle ξεπέρασαν τα 500 εκατομμύρια (Loganbill, 2025), ενώ τα δυναμικά στατιστικά στοιχεία της πλατφόρμας καταγράφουν περισσότερες από 148.000 ενεργές καταχωρισμένες εγκαταστάσεις Moodle σε πάνω από 235 χώρες (Moodle, χ.χ.-e). Πρόκειται για μια διαδικτυακή πλατφόρμα που υποστηρίζει τη δημιουργία και τη διαχείριση μαθημάτων, παρέχοντας ένα ευρύ φάσμα εργαλείων για τη διάθεση εκπαιδευτικού υλικού, την επικοινωνία και τη συνεργασία μεταξύ διδασκόντων και εκπαιδευομένων. Η παιδαγωγική φιλοσοφία του Moodle συνδέεται με τον κονστρουξιονισμό του Papert, σύμφωνα με τον οποίο η μάθηση ενισχύεται όταν οι εκπαιδευόμενοι δημιουργούν και μοιράζονται δημιουργήματα με νόημα (Papert & Harel, 1991). Παράλληλα, η έμφαση στη συνεργατική οικοδόμηση της γνώσης μέσα σε μαθησιακές κοινότητες εδράζεται στις αρχές του κοινωνικού κονστρουξιονισμού (Dougiamas & Taylor, 2003). Μεταξύ των βασικών λειτουργιών του περιλαμβάνονται η ανάρτηση πολυμεσικού περιεχομένου, η οργάνωση δραστηριοτήτων αξιολόγησης (κουίζ, εργασίες), η δημιουργία χώρων συζήτησης (φόρουμ), η διαχείριση βαθμολογιών και η παραγωγή αναφορών δραστηριότητας.

Η συστηματική καταγραφή των ενεργειών των χρηστών και των αποτελεσμάτων αξιολόγησης καθιστά το Moodle όχι μόνο περιβάλλον μάθησης, αλλά και πηγή ψηφιακών ιχνών για την εκπαιδευτική έρευνα και τα Learning Analytics. Η επιλογή του Moodle ως πλατφόρμας αναφοράς για την παρούσα έρευνα αιτιολογείται από τρεις βασικούς λόγους: (α) την εκτεταμένη διεθνή χρήση του σε διαφορετικά εκπαιδευτικά περιβάλλοντα, (β) την αρθρωτή αρχιτεκτονική του και τον χαρακτήρα του ως λογισμικό ανοικτού κώδικα, που διευκολύνουν την προσαρμογή και την επέκταση του λογισμικού χωρίς τέλη άδειας χρήσης για το βασικό σύστημα (Moodle, χ.χ.-c· Moodle, χ.χ.-d), και (γ) τη δυνατότητα εξαγωγής των αρχείων καταγραφής δραστηριότητας και του αρχείου βαθμολογιών, τα οποία αποτελούν τις πρωτογενείς πηγές δεδομένων για την εξαγωγή δεικτών μαθησιακής συμπεριφοράς, επίδοσης και κοινωνικής αλληλεπίδρασης (Moodle, χ.χ.-a· Moodle, χ.χ.-b).

2.2.2 Δεδομένα αλληλεπίδρασης (log files)

Κάθε δράση οποιουδήποτε χρήστη στο Moodle καταγράφεται αυτόματα στη βάση δεδομένων του συστήματος ως μια δομημένη εγγραφή στα αρχεία καταγραφής (log files). Κάθε τέτοια εγγραφή είναι ένα συμβάν που περιέχει βασικά μεταδεδομένα, όπως: το αναγνωριστικό χρήστη (*user ID*), τον τύπο της δραστηριότητας (event context, π.χ., *forum_post_created*, *assign_submission_submitted*), το αντικείμενο στο οποίο αφορά (*instance ID*) και μια χρονική σήμανση (*timestamp*).

Αυτά τα ακατέργαστα δεδομένα αλληλεπίδρασης αποτελούν τον κρίσιμο πόρο για την παρούσα έρευνα. Μέσω συστηματικής ανάλυσης και σύνθεσης, μετατρέπονται από απλά αρχεία καταγραφής σε ποιοτικές και ποσοτικές πληροφορίες που αποτυπώνουν τη μαθησιακή διαδικασία σε τρεις διαστάσεις:

- **Συμμετοχή:** Συχνότητα σύνδεσης, πλήθος και τύπος πρόσβασης σε πόρους, συμμετοχή σε δραστηριότητες, χρονική συνέπεια.
- **Ακαδημαϊκή επίδοση:** Βαθμοί από εργασίες και κουίζ, ιστορικό βαθμολογιών.
- **Κοινωνική αλληλεπίδραση:** Δεδομένα από συζητήσεις φόρουμ, τα οποία αποτελούν τη βασική πρώτη ύλη για την εφαρμογή της Ανάλυσης Κοινωνικών Δικτύων (SNA).

2.3 Εκπαιδευτική Εξόρυξη Δεδομένων (Educational Data Mining – EDM)

2.3.1 Ορισμός και κατηγορίες δεικτών

Η Εκπαιδευτική Εξόρυξη Δεδομένων (EDM) ορίζεται ως η επιστημονική περιοχή που ασχολείται με την ανάπτυξη μεθόδων για την εξαγωγή γνώσης από δεδομένα εκπαιδευτικών περιβαλλόντων (Romero & Ventura, 2013). Βασικός της στόχος είναι η αναγνώριση παραγόντων και προτύπων που σχετίζονται με την επιτυχή συνεργασία, όπως τάσεις συμμετοχής και συσχετισμοί μεταξύ δραστηριότητας και επίδοσης, υποστηρίζοντας έτσι τη λήψη αποφάσεων, την προσωποποίηση της μάθησης και την έρευνα.

Η εφαρμογή τεχνικών EDM στα δεδομένα του Moodle επιτρέπει την εξαγωγή ποσοτικών και ποιοτικών δεικτών (indicators/metrics), οι οποίοι κατηγοριοποιούνται ανάλογα με τη διάσταση της συμπεριφοράς που μετρούν (Romero & Ventura, 2013). Οι βασικές κατηγο-

ρίες δεικτών που αξιοποιούνται στην EDM περιγράφονται και συστηματοποιούνται σε θεμελιώδεις και επικαιροποιημένες ανασκοπήσεις του πεδίου (Romero & Ventura, 2010, 2020), ενώ αντίστοιχοι δείκτες έχουν εφαρμοστεί σε αναλύσεις της μαθησιακής και κοινωνικής συμπεριφοράς φοιτητών σε περιβάλλοντα εξ αποστάσεως μάθησης (π.χ., Tsoni, Sakkoropoulos, et al., 2022). Στην παρούσα εργασία, οι κατηγορίες δεικτών που τροφοδοτούν τη δημιουργία του προφίλ κάθε φοιτητή είναι οι εξής:

1. **Δείκτες συμμετοχής (Engagement Indicators):** Ποσοτικοί δείκτες που μετρούν την παρουσία και τη δραστηριότητα στην πλατφόρμα. Περιλαμβάνουν μετρικές όπως η συχνότητα σύνδεσης, ο αριθμός προσβάσεων σε διάφορους πόρους, η συμμετοχή σε δραστηριότητες (π.χ., προβολές σελίδων, υποβολές) και η χρονική συνέπεια της δραστηριότητας.
2. **Δείκτες ακαδημαϊκής επίδοσης (Performance indicators):** Δείκτες που αντικατοπτρίζουν τα μαθησιακά αποτελέσματα. Αυτοί προέρχονται κυρίως από τα εργαλεία αξιολόγησης του Moodle και περιλαμβάνουν τον μέσο όρο βαθμών, ιστορικά στοιχεία βαθμολογίας, τάσεις βελτίωσης και δείκτες συμμόρφωσης με προθεσμίες.
3. **Σύνθετοι δείκτες και προφίλ συμπεριφοράς (Composite behavioral indicators/profiles):** Οι δείκτες αυτοί συχνά προκύπτουν μέσω τεχνικών όπως η ομαδοποίηση (clustering) όπου πολλαπλοί απλοί δείκτες συνδυάζονται για τη δημιουργία υψηλότερου επιπέδου τυπολογιών ή προφίλ (Romero & Ventura, 2013). Αυτό επιτρέπει, για παράδειγμα, την ταξινόμηση των φοιτητών σε κατηγορίες όπως «ο ενεργός συνεργάτης», «ο αυτόνομος μαθητής» ή «ο περιφερειακός συμμετέχων».

Εκτός από τους τυπικούς δείκτες συμμετοχής και επίδοσης, η σύγχρονη έρευνα αναγνωρίζει τη σημασία της συναισθηματικής διάστασης της μάθησης. Η απογοήτευση (frustration) αποτελεί σημαντική συναισθηματική κατάσταση, η οποία μπορεί να επηρεάσει τη μαθησιακή διαδικασία και την εμπλοκή των εκπαιδευομένων (Pekrun, 2006· D'Mello & Graesser, 2012). Ειδικότερα, έρευνες σε ψηφιακά μαθησιακά περιβάλλοντα και Ευφυή Διδακτικά Συστήματα έχουν εξετάσει τόσο τη δυναμική της απογοήτευσης κατά τη μάθηση όσο και τη δυνατότητα πρόβλεψής της μέσω δεδομένων αλληλεπίδρασης, με στόχο την έγκαιρη υποστήριξη των εκπαιδευομένων (D'Mello & Graesser, 2012· Zambrano et al., 2024). Στην παρούσα εργασία, η διάσταση αυτή προσεγγίζεται μέσω ειδικών συμπεριφορικών δεικτών, οι οποίοι ερμηνεύονται ως ενδείξεις πιθανής απογοήτευσης.

2.4 Ανάλυση κοινωνικών δικτύων (Social Network Analysis – SNA)

2.4.1 Αναπαράσταση της αλληλεπίδρασης των φοιτητών ως γράφος και μετρικές κεντρικότητας

Η Ανάλυση Κοινωνικών Δικτύων (SNA) είναι μια ισχυρή αναλυτική προσέγγιση για τη συστηματική μελέτη των σχέσεων και αλληλεπιδράσεων μεταξύ των μελών μιας κοινότητας. Βασική αρχή της είναι η αναπαράσταση του κοινωνικού συστήματος ως γράφου. Στο πλαίσιο ενός εκπαιδευτικού περιβάλλοντος όπως το Moodle, κάθε συμμετέχων αντιπροσωπεύεται ως κόμβος (node), και κάθε αλληλεπίδραση (π.χ., απάντηση σε δημοσίευση φόρουμ, κοινοποίηση πόρου) αποτυπώνεται ως ακμή (edge) που συνδέει τους αντίστοιχους κόμβους. Η αναπαράσταση αυτή επιτρέπει την ποσοτική διερεύνηση της δομής των σχέσεων, των προτύπων επικοινωνίας και της θέσης των συμμετεχόντων στο δίκτυο, αναδεικνύοντας πτυχές της κοινωνικής δυναμικής της τάξης που δεν είναι άμεσα ορατές (Grunspan et al., 2014· Tsoni et al., 2019).

Η SNA μετασχηματίζει τη δομή του γράφου σε ένα σύνολο ποσοτικών μετρικών που αποτυπώνουν τη θέση και τη σχετική σημασία κάθε κόμβου στο δίκτυο (Wasserman & Faust, 1994). Οι βασικότερες από αυτές, οι οποίες αξιοποιούνται στην παρούσα εργασία, είναι οι κλασικές μετρικές κεντρικότητας βαθμού, διαμεσολάβησης και εγγύτητας (Freeman, 1979), καθώς και το PageRank (Brin & Page, 1998):

- **Κεντρικότητα Βαθμού (Degree Centrality):** Αριθμός των άμεσων συνδέσεων (ακμών) ενός κόμβου. Σε ένα φόρουμ, υψηλή κεντρικότητα βαθμού υποδηλώνει τον πιο δραστήριο συμμετέχοντα (πολλές απαντήσεις/δημοσιεύσεις).
- **Κεντρικότητα Διαμεσολάβησης (Betweenness Centrality):** Μετρά το πόσο συχνά ένας κόμβος λειτουργεί ως απαραίτητος «σύνδεσμος» ή «γέφυρα» μεταξύ δύο άλλων κόμβων. Δείχνει μέλη που συνδέουν διαφορετικές υποομάδες και ελέγχουν τη ροή πληροφοριών (Freeman, 1979).
- **Κεντρικότητα Εγγύτητας (Closeness Centrality):** Μετρά το πόσο «κοντά» είναι ένας κόμβος προς όλους τους υπόλοιπους βάσει των συντομότερων διαδρομών που τον συνδέουν με αυτούς. Υψηλή τιμή υποδηλώνει ότι ο φοιτητής μπορεί να προσεγγίσει τους άλλους σχετικά γρήγορα, μέσω λίγων ενδιάμεσων βημάτων (Wasserman & Faust, 1994).
- **PageRank:** Αποτιμά τη σημασία ενός κόμβου λαμβάνοντας υπόψη τη σημασία των γειτόνων του (Brin & Page, 1998). Ένας φοιτητής έχει υψηλό PageRank όχι απλώς

αν είναι δραστήριος, αλλά αν οι αλληλεπιδράσεις του σχετίζονται με άλλους εξίσου κεντρικούς και δραστήριους φοιτητές, υποδηλώνοντας αναγνωρισμένο κύρος και ουσιαστική συνεισφορά.

Επιπλέον, τεχνικές όπως η ανίχνευση κοινοτήτων επιτρέπουν τον εντοπισμό φυσικά σχηματιζόμενων υποομάδων εντός του ευρύτερου δικτύου.

2.4.2 Ανάδειξη λειτουργικών ρόλων

Ο συνδυασμός των μετρικών κεντρικότητας —και όχι η μεμονωμένη χρήση τους— μπορεί να υποστηρίξει την αναγνώριση διακριτών λειτουργικών θέσεων και μορφών συμμετοχής στο δίκτυο (de Nooy et al., 2018). Στη βιβλιογραφία των διαδικτυακών μαθησιακών κοινοτήτων, η Ανάλυση Κοινωνικών Δικτύων έχει αξιοποιηθεί τόσο για τη μελέτη των ρόλων και της συμμετοχής των φοιτητών σε διαδικτυακά φόρουμ (Rabbany et al., 2014) όσο και για την ανάδειξη λανθανόντων χαρακτηριστικών και προτύπων κοινωνικής συμπεριφοράς (Tsoni, Panagiotakopoulos, et al., 2022).

Οι επιμέρους μετρικές αποτυπώνουν διαφορετικές και συμπληρωματικές όψεις της θέσης ενός συμμετέχοντα στο δίκτυο (Vignery & Laurier, 2020). Ο συνδυασμός τους μπορεί να υποστηρίξει τη διάκριση λειτουργικών θέσεων: υψηλές τιμές σε πολλαπλές μετρικές μπορεί να υποδηλώνουν κεντρική θέση, η υψηλή κεντρικότητα διαμεσολάβησης αναδεικνύει διαμεσολαβητικές θέσεις (Freeman, 1979· Burt, 1992), ενώ ο συνδυασμός υψηλής κεντρικότητας βαθμού με χαμηλότερες τιμές στις άλλες μετρικές υποδηλώνει ενεργή συμμετοχή. Αντίθετα, χαμηλές τιμές στις περισσότερες μετρικές συνδέονται με περιφερειακές θέσεις συμμετοχής.

Οι παραπάνω συσχετίσεις δεν συνιστούν καθολικά αποδεκτή ή μηχανιστική αντιστοίχιση μετρικών και ρόλων, αλλά λειτουργικές ερμηνείες που βασίζονται στη συνδυαστική εξέταση των μετρικών και της συνολικής δικτυακής θέσης των συμμετεχόντων. Η αναγνώρισή τους συμβάλλει στην κατανόηση της κοινωνικής δυναμικής της μαθησιακής κοινότητας και παρέχει τη θεωρητική βάση για την επιχειρησιακή διαμόρφωση των κοινωνικών ρόλων που παρουσιάζεται στο Κεφάλαιο 4.

2.5 Ο συνδυασμός EDM και SNA ως ολοκληρωμένη προσέγγιση

Ενώ η Εκπαιδευτική Εξόρυξη Δεδομένων (EDM) προσφέρει ισχυρούς, ποσοτικούς δείκτες ατομικής συμπεριφοράς και επίδοσης, παραμένει περιορισμένη στην ανάλυση των σχέσεων και της δομής των αλληλεπιδράσεων εντός της μαθησιακής κοινότητας. Αυτό το κενό καλύπτεται από την Ανάλυση Κοινωνικών Δικτύων (SNA), η οποία εξετάζει πώς οι φοιτητές συνδέονται, επικοινωνούν και σχηματίζουν λανθάνοντα δίκτυα.

Ο συνδυασμός των δύο μεθοδολογιών είναι, επομένως, απαραίτητος για μια ολιστική και πολυδιάστατη μοντελοποίηση του φοιτητή: η EDM μετρά το «πόσο» ενεργεί ένας φοιτητής (συμμετοχή, επίδοση), ενώ η SNA αποκαλύπτει το «με ποιον» αλληλεπιδρά και «σε ποιον λειτουργικό ρόλο» εντάσσεται. Στην παρούσα εργασία, αυτός ο συμπληρωματικός συνδυασμός θεμελιώνει τη δημιουργία του ενιαίου, πολυδιάστατου προφίλ, συνδέοντας ατομικά χαρακτηριστικά με την κοινωνική τους διάσταση.

2.6 Συστήματα συστάσεων στην εκπαίδευση

2.6.1 Εκπαιδευτικά Συστήματα Συστάσεων (Educational Recommender Systems - ERS)

Τα Συστήματα Συστάσεων (Recommender Systems - RS) είναι ευφυή συστήματα που προβλέπουν τις ανάγκες ή τις προτιμήσεις των χρηστών και προτείνουν σχετικές ενέργειες ή αντικείμενα (Ricci et al., 2015). Στην εκπαίδευση, εξειδικεύονται ως Εκπαιδευτικά Συστήματα Συστάσεων (Educational Recommender Systems - ERS), στα οποία ο στόχος μετατοπίζεται από την πρόβλεψη καταναλωτικών προτιμήσεων στην υποστήριξη της μαθησιακής διαδικασίας και την επίτευξη μαθησιακών αποτελεσμάτων, όπως η προσωποποίηση της μάθησης και η ενίσχυση της συνεργασίας (Drachsler et al., 2015· Verbert et al., 2012). Τα ERS έχουν εφαρμοστεί σε ποικίλα εκπαιδευτικά πλαίσια, όπως η σύσταση μαθησιακών πόρων, η εξατομίκευση μαθησιακών διαδρομών και η υποστήριξη συνεργατικών δραστηριοτήτων (Manouselis et al., 2011).

Πολλές από τις ευρέως χρησιμοποιούμενες προσεγγίσεις συστημάτων συστάσεων βασίζονται στην αναζήτηση ομοιότητας, είτε μεταξύ των χρηστών (συνεργατικό φιλτράρισμα) είτε μεταξύ του προφίλ του χρήστη και των χαρακτηριστικών των αντικειμένων (φιλτράρισμα βάσει περιεχομένου) (Pazzani & Billsus, 2007· Su & Khoshgoftaar, 2009). Όταν, όμως, το αντικείμενο της σύστασης είναι ο σχηματισμός ομάδων, το πρόβλημα καθίσταται ιδιαίτερα

σύνθετο, καθώς απαιτείται η επιλογή κατάλληλων συνδυασμών φοιτητών και η εξισορρόπηση πολλαπλών, συχνά ανταγωνιστικών, κριτηρίων, όπως η κοινωνική συνοχή, η συμπληρωματικότητα των ρόλων και η ισορροπία επιδόσεων (Maqtary et al., 2019). Στην παρούσα εργασία, η προσέγγιση δεν περιορίζεται σε ένα ενιαίο κριτήριο συνολικής ομοιότητας μεταξύ των φοιτητών, αντίθετα, οι επιμέρους διαστάσεις του προφίλ αξιοποιούνται με διαφορετικό τρόπο και καθεμία συμβάλλει διακριτά σε μια πολυεπίπεδη, πολυκριτηριακή και υβριδική προσέγγιση ανάλυσης και σύστασης. Η αξιοποίηση δεικτών συμμετοχής, επίδοσης και κοινωνικής αλληλεπίδρασης, καθώς και συμπεριφορικών ενδείξεων πιθανής απογοήτευσης, δεν βασίζεται σε ένα ενιαίο μέτρο ομοιότητας, αλλά αποδίδει σε κάθε διάσταση διακριτή λειτουργία στην ανάλυση και στη διαδικασία σύστασης.

2.6.2 Βασικές κατηγορίες συστημάτων συστάσεων

Στη βιβλιογραφία έχουν διαμορφωθεί διάφορες βασικές κατηγορίες συστημάτων συστάσεων, οι οποίες διαφέρουν ως προς τον μηχανισμό παραγωγής των συστάσεων και τις πηγές πληροφόρησης που αξιοποιούν (Burke, 2002· Ricci et al., 2015). Οι κυριότερες από αυτές παρουσιάζονται συνοπτικά στη συνέχεια:

- **Συνεργατικό φιλτράρισμα (Collaborative Filtering - CF):** Βασίζεται στην παραδοχή ότι χρήστες με παρόμοια ιστορικά συμπεριφορές ή αξιολογήσεις θα έχουν παρόμοιες προτιμήσεις στο μέλλον (Su & Khoshgoftaar, 2009). Κατ' αναλογία, στο πλαίσιο σχηματισμού ομάδων, η αρχή αυτή θα μπορούσε να αξιοποιηθεί για τη σύσταση συνεργασίας μεταξύ φοιτητών με παρόμοια χαρακτηριστικά συμμετοχής ή επίδοσης. Το βασικό μειονέκτημά του είναι το «πρόβλημα ψυχρής εκκίνησης» (cold start) για νέους φοιτητές χωρίς ιστορικό καθώς και ο κίνδυνος δημιουργίας υπερβολικά ομοιογενών ομάδων, γεγονός που μπορεί να περιορίσει τη συμπληρωματικότητα δεξιοτήτων (Adomavicius & Tuzhilin, 2005· Ricci et al., 2015).
- **Φιλτράρισμα βάσει περιεχομένου (Content-Based Filtering):** Προτείνει αντικείμενα παρόμοια με αυτά που ένας χρήστης έχει προτιμήσει στο παρελθόν, με βάση τα χαρακτηριστικά τους (Pazzani & Billsus, 2007). Κατ' αναλογία στον σχηματισμό ομάδων, η ίδια λογική θα μπορούσε να αξιοποιηθεί για τη σύσταση συνεργασίας μεταξύ φοιτητών με παρόμοια χαρακτηριστικά (π.χ., δεξιότητες, ενδιαφέροντα). Το βασικό πλεονέκτημα της προσέγγισης είναι η ανεξαρτησία από δεδομένα άλλων

χρηστών. Ωστόσο, η εφαρμογή αυστηρά ομοιογενών κριτηρίων επιλογής ενέχει τον κίνδυνο δημιουργίας ομάδων με περιορισμένη ποικιλότητα δεξιοτήτων και απόψεων, γεγονός που μπορεί να λειτουργήσει ανασταλτικά για τη συνεργατική μάθηση (Maqtary et al., 2019).

- **Συστήματα βάσει γνώσης και περιορισμών (Knowledge-Based / Constraint-Based Systems):** Αξιοποιούν ρητά κανόνες, περιορισμούς και γνώση του πεδίου εφαρμογής για να συνδέσουν τις ανάγκες των χρηστών με τα χαρακτηριστικά των διαθέσιμων αντικειμένων (Felfernig & Burke, 2008). Στο πλαίσιο σχηματισμού ομάδων, μια τέτοια προσέγγιση θα μπορούσε να ενσωματώνει παιδαγωγικούς κανόνες, όπως «κάθε ομάδα πρέπει να περιλαμβάνει τουλάχιστον έναν φοιτητή με υψηλή επίδοση σε συγκεκριμένο γνωστικό αντικείμενο». Το βασικό πλεονέκτημα της προσέγγισης είναι ότι δεν απαιτεί ιστορικό αξιολογήσεων ή προηγούμενων αλληλεπιδράσεων, καθώς οι συστάσεις παράγονται βάσει ρητών κανόνων και περιορισμών. Ωστόσο, η σχεδίαση και η συντήρηση του συστήματος κανόνων είναι συχνά πολύπλοκη και δαπανηρή, ιδιαίτερα όταν το πεδίο εφαρμογής είναι ευρύ ή τα κριτήρια μεταβάλλονται (Felfernig & Burke, 2008· Burke, 2000).
- **Υβριδικά συστήματα (Hybrid Systems):** Συνδυάζουν δύο ή περισσότερες από τις παραπάνω προσεγγίσεις για να μετριάσουν τα μειονεκτήματα της κάθε μίας και να ενισχύσουν τα πλεονεκτήματά τους (Burke, 2002· Adomavicius & Tuzhilin, 2005). Για παράδειγμα, ένα σύστημα μπορεί να χρησιμοποιήσει Content-Based Filtering για αρχική σύσταση και Collaborative Filtering για τελική βελτίωση. Η τάση προς τα υβριδικά συστήματα είναι ιδιαίτερα έντονη σε πολύπλοκα πεδία όπως η εκπαίδευση, όπου οι απαιτήσεις είναι πολυδιάστατες και απαιτούν ταυτόχρονη εξισορρόπηση πολλαπλών κριτηρίων (Drachsler et al., 2015).

Κατηγορία	Βασική αρχή λειτουργίας	Προτεινόμενο κριτήριο σχηματισμού ομάδων	Κύρια πλεονεκτήματα	Κύριοι περιορισμοί (στο πλαίσιο σχηματισμού ομάδων)
Συνεργατικό φίλτράρισμα (CF)	Προβλέπει προτιμήσεις βάσει ομοιότητας στο ιστορικό συμπεριφοράς	Ομοιότητα στη συνεργατική συμπεριφορά	• Αποτελεσματικό με πλούσιο ιστορικό • Εμπειρικά τεκμηριωμένο	• Πρόβλημα ψυχρής εκκίνησης • Ευαίσθητο σε αραιά δεδομένα
Φιλτράρισμα βάσει περιεχομένου (Content-Based)	Προτείνει στοιχεία παρόμοια με αυτά που ο χρήστης έχει προτιμήσει	Ομοιότητα στα χαρακτηριστικά του προφίλ	• Ανεξάρτητο από δεδομένα άλλων χρηστών • Διαφανής λογική	• Υπερβολική ομοιογένεια • Περιορισμένη ικανότητα ανακάλυψης διαφορετικών ή συμπληρωματικών επιλογών • Απαιτεί επαρκή χαρακτηριστικά του προφίλ
Συστήματα βάσει γνώσης (Knowledge-Based)	Χρησιμοποιούν ρητούς κανόνες ή γνώση τομέα	Ισορροπία & Συμπληρωματικότητα βάσει κανόνων	• Ιδανικά για cold start • Εγγυώνται τήρηση κριτηρίων • Διαφανής σχεδιασμός	• Δυσκίνητη δημιουργία/συντήρηση κανόνων • Περιορισμένη προσαρμοστικότητα

Υβριδικά συστήματα	Συνδυασμός δύο ή περισσότερων προσεγγίσεων	Πολυκριτηριακή βελτιστοποίηση	• Μετριάζει περιορισμούς απλών μεθόδων • Ευέλικτο και προσαρμοσимо • Ισορροπεί ανταγωνιστικά κριτήρια	• Αυξημένη πολυπλοκότητα • Δυσκολία αξιολόγησης συμβολής κάθε συνιστώσας
---------------------------	--	-------------------------------	---	---

Πίνακας 2-1 Συγκριτική ανάλυση βασικών κατηγοριών συστημάτων συστάσεων για εκπαιδευτικούς σκοπούς

Όπως αναδεικνύεται από τη συγκριτική ανάλυση στον παραπάνω πίνακα (Πίνακας 2-1) καμία από τις βασικές κατηγορίες συστημάτων συστάσεων δεν επαρκεί αυτόνομα για την πλήρη αντιμετώπιση του πολυδιάστατου προβλήματος του σχηματισμού ομάδων. Η ανάγκη για ταυτόχρονη διασφάλιση τόσο μιας κοινής βάσης συνεργασίας (που προϋποθέτει ορισμένη ομοιογένεια κινήτρων) όσο και λειτουργικής συμπληρωματικότητας (που απαιτεί ετερογένεια δεξιοτήτων και ρόλων) καθιστά τις μονοδιάστατες προσεγγίσεις ανεπαρκείς. Το γεγονός αυτό οδηγεί στην υιοθέτηση υβριδικών προσεγγίσεων, οι οποίες μπορούν να συνδυάσουν τα πλεονεκτήματα διαφορετικών μεθόδων για την κάλυψη των πολλαπλών αυτών απαιτήσεων (Drachsler et al., 2015· Adomavicius & Tuzhilin, 2005).

2.6.3 Συστήματα συστάσεων με επίγνωση πλαισίου (Context-Aware Recommender Systems - CARS)

Μια ιδιαίτερα σημαντική εξέλιξη των RS αποτελούν τα Συστήματα Συστάσεων με Επίγνωση Πλαισίου (CARS), τα οποία αναγνωρίζουν ότι μια αποτελεσματική σύσταση δεν πρέπει να βασίζεται αποκλειστικά σε χαρακτηριστικά χρήστη και αντικειμένου, αλλά να λαμβάνει υπόψη το πλαίσιο (context) χρήσης (Adomavicius & Tuzhilin, 2011).

Στην εκπαίδευση, αυτό το πλαίσιο μπορεί να περιλαμβάνει:

- Τον τύπο της μαθησιακής δραστηριότητας (π.χ., εργαστηριακή άσκηση, ερευνητική εργασία).
- Τους συγκεκριμένους μαθησιακούς στόχους του μαθήματος.
- Τις συλλογικές ιδιότητες μιας ομάδας (π.χ., ομαδική δυναμική, ισορροπία δεξιοτήτων).

Τα CARS ευθυγραμμίζονται άμεσα με την πρόσφατη έρευνα που υπογραμμίζει την ανάγκη για υβριδικά και ευαίσθητα στο πλαίσιο συστήματα, ως μια προηγμένη κατεύθυνση για την αντιμετώπιση πολύπλοκων εκπαιδευτικών προβλημάτων (Mateos & Bellogín, 2025).

2.6.4 Εφαρμογή σε συνεργατική μάθηση και σχηματισμό ομάδων

Η εφαρμογή των RS για τον σχηματισμό ομάδων (Group Recommendation/Formation) αποτελεί μια εξειδικευμένη υποκατηγορία. Σε αυτή την περίπτωση, το αντικείμενο της σύστασης δεν είναι ένα μεμονωμένο στοιχείο για έναν μεμονωμένο χρήστη, αλλά μια σύνθετη

ομάδα ατόμων (Jameson & Smyth, 2007) η οποία πρέπει να βελτιστοποιήσει ένα σύνολο συχνά ανταγωνιστικών κριτηρίων.

Στο πλαίσιο της συνεργατικής μάθησης, οι προσεγγίσεις διακρίνονται σε τρεις κύριες κατηγορίες (Maqtary et al., 2019):

- **Μέθοδοι βάσει προφίλ (Profile-Based):** Ομαδοποιούν φοιτητές με βάση την ομοιότητα των χαρακτηριστικών τους (π.χ., κίνητρα, προηγούμενες γνώσεις).
- **Μέθοδοι βάσει επιδόσεων (Performance-Based):** Σχηματίζουν ομάδες με στόχο την εξισορρόπηση των επιδόσεων (π.χ., διασφαλίζοντας ότι κάθε ομάδα έχει παρόμοιο μέσο όρο).
- **Μέθοδοι βάσει κοινωνικού δικτύου (Social Network-Based):** Επωφελούνται της δομής των αλληλεπιδράσεων, για παράδειγμα μέσω ανίχνευσης κοινοτήτων, αξιοποιώντας δομικές ιδιότητες του δικτύου, όπως κοινότητες και μετρικές κεντρικότητας, για τον εντοπισμό κοινωνικών ρόλων και σχέσεων.

Ο σχηματισμός βέλτιστων ομάδων απαιτεί την ταυτόχρονη βελτιστοποίηση πολλαπλών, συχνά ανταγωνιστικών κριτηρίων που ορίζονται από το εκπαιδευτικό πλαίσιο. Η ανάγκη για ομοιογένεια ως προς το κίνητρο και ετερογένεια ως προς τις δεξιότητες/ρόλους μεταφράζεται σε ένα πολυκριτηριακό μοντέλο βελτιστοποίησης που λαμβάνει υπόψη το πλαίσιο εντός ενός CARS.

2.7 Σύνθεση: Ένα υβριδικό CARS πλαίσιο για βέλτιστο σχηματισμό ομάδων στο Moodle

Συνθέτοντας τις θεωρητικές αρχές που αναλύθηκαν στις προηγούμενες ενότητες, η εργασία προτείνει ένα υβριδικό πλαίσιο CARS για τον σχηματισμό ομάδων στο Moodle. Η προσέγγιση αυτή συνδυάζει στοιχεία από υφιστάμενες μεθόδους και επιχειρεί να αντιμετωπίσει τέσσερα επιμέρους ερευνητικά κενά που αναδεικνύονται από τη σχετική βιβλιογραφία (Maqtary et al., 2019):

1. **Αξιοποίηση εγγενών δεδομένων του LMS:** Το σύστημα βασίζεται αποκλειστικά σε δεδομένα καταγραφής του Moodle ενισχύοντας έτσι την αυτοματοποίηση και επεκτασιμότητα.

2. **Σύνθεση προφίλ με EDM και SNA:** Ενσωμάτωση δυναμικών κοινωνικών χαρακτηριστικών (κεντρικότητες, ρόλοι) μέσω SNA, δημιουργώντας ενιαία, πολυδιάστατα προφίλ που αντανακλούν τόσο ακαδημαϊκή όσο και κοινωνική συμπεριφορά.
3. **Υβριδικός σχεδιασμός στο πλαίσιο CARS:** Διατυπώνει το πρόβλημα ως πολυκριτηριακό (multi-criteria) και ευαίσθητο στο πλαίσιο. Το πλαίσιο ορίζεται από τις παιδαγωγικές απαιτήσεις της συνεργατικής μάθησης (ομοιογένεια κινήτρων, ετερογένεια ρόλων).
4. **Προδιαγραφές για μελλοντική ενσωμάτωση:** Η προσέγγιση συνοδεύεται από τεχνικές προδιαγραφές για ενσωμάτωση στο Moodle (ως πρόσθετο ή εξωτερική υπηρεσία).

Με αυτόν τον τρόπο, η προτεινόμενη μέθοδος εντάσσεται στις σύγχρονες ερευνητικές προσεγγίσεις για τα ευφυή συστήματα συστάσεων στην εκπαίδευση, προσφέροντας ένα ολοκληρωμένο, αυτοματοποιημένο και παιδαγωγικά τεκμηριωμένο πλαίσιο για τον ισορροπημένο σχηματισμό ομάδων. Η σύνθεση δεικτών ακαδημαϊκής και μαθησιακής συμπεριφοράς, οι οποίοι προκύπτουν μέσω τεχνικών Εκπαιδευτικής Εξόρυξης Δεδομένων (EDM), με κοινωνικά χαρακτηριστικά του δικτύου αλληλεπίδρασης, τα οποία αναδεικνύονται μέσω της Ανάλυσης Κοινωνικών Δικτύων (SNA), συγκροτεί ένα ενιαίο πλαίσιο συστάσεων που υποστηρίζει τη διερεύνηση των ερευνητικών ερωτημάτων της εργασίας.

3 Συγκριτική βιβλιογραφική ανασκόπηση προσεγγίσεων σχηματισμού ομάδων

3.1 Εισαγωγή και σκοπός της ανασκόπησης

Η παρούσα ενότητα επιχειρεί μια συγκριτική βιβλιογραφική ανασκόπηση υφιστάμενων προσεγγίσεων για τον αυτοματοποιημένο σχηματισμό ομάδων σε εκπαιδευτικά περιβάλλοντα. Ο σκοπός δεν είναι η απλή καταγραφή μεθόδων, αλλά η κριτική ανάλυση της ικανότητάς τους να ανταποκριθούν στα πολυδιάστατα και συχνά ανταγωνιστικά κριτήρια που απορρέουν από τις παιδαγωγικές αρχές της συνεργατικής μάθησης, όπως διατυπώθηκαν στο Κεφάλαιο 2. Ειδικότερα, η ανασκόπηση επικεντρώνεται στο πόσο αποτελεσματικά οι υπάρχουσες τεχνικές αντιμετωπίζουν το δίλημμα της ομοιογένειας σε κίνητρα και ενδιαφέροντα έναντι της ετερογένειας σε δεξιότητες και κοινωνικούς ρόλους, όπως υποδεικνύει η Θεωρία της Κοινωνικής Αλληλεξάρτησης.

Η ανάλυση διεξάγεται σε δύο επίπεδα. Στο πρώτο επίπεδο, εξετάζονται θεωρητικές προσεγγίσεις και ερευνητικά πρωτότυπα από τη διεθνή βιβλιογραφία. Στο δεύτερο επίπεδο, διερευνάται η υφιστάμενη κατάσταση στο Moodle ως το πλέον διαδεδομένο ανοιχτού κώδικα LMS, εξετάζοντας τους ενσωματωμένους μηχανισμούς και τα διαθέσιμα πρόσθετα (plugins) που σχετίζονται με τη διαχείριση ομάδων.

3.2 Κατηγορίες προσεγγίσεων για σχηματισμό ομάδων

Στη βιβλιογραφία έχουν προταθεί διάφορες κατηγορίες προσεγγίσεων για τον αυτοματοποιημένο σχηματισμό ομάδων σε εκπαιδευτικά περιβάλλοντα. Ο παρακάτω πίνακας (Πίνακας 3-1) συνοψίζει αντιπροσωπευτικές μελέτες από κάθε κατηγορία, παρουσιάζοντας τα δεδομένα που αξιοποιούν, την αλγοριθμική τεχνική, τα κύρια ευρήματα και τους περιορισμούς τους. Στις υποενότητες 3.2.1-3.2.5 οι προσεγγίσεις αυτές αναλύονται λεπτομερέστερα.

Κατηγορία / Μέθοδος	Χρησιμοποιούμενα δεδομένα / Χαρακτηριστικά	Αλγόριθμος / Τεχνική	Κύρια ευρήματα & Περιορισμοί	Βιβλιογραφική αναφορά
Προφίλ / Περιεχομένου	Αυτοαναφερόμενα δεδομένα (δεξιότητες, ενδιαφέροντα)	Ιεραρχική ομαδοποίηση, k-means	<i>Ευρήματα:</i> Δημιουργεί ομάδες με κοινά ενδιαφέροντα. <i>Περιορισμοί:</i> Υπερβολική ομοιογένεια, αγνοεί δυναμική αλληλεπιδράσεων.	Maqtary et al. (2019). Drachsler et al. (2015)
Συνεργατικό φιλτράρισμα	Ιστορικό συνεργασιών (π.χ. προηγούμενες ομάδες)	Υβριδικά μοντέλα CF	<i>Ευρήματα:</i> Αποτελεσματικό όταν υπάρχει πλούσιο ιστορικό. <i>Περιορισμοί:</i> Πρόβλημα ψυχρής εκκίνησης, ενίσχυση υπαρχουσών ομάδων (echo chambers).	Bellogín et al. (2014)
Βάσει κοινωνικού δικτύου (SNA)	Αλληλεπιδράσεις σε φόρουμ (δημοσιεύσεις, απαντήσεις)	Ανίχνευση κοινοτήτων (π.χ. Louvain), μετρικές κεντρικότητας	<i>Ευρήματα:</i> Αναδεικνύει λειτουργικούς ρόλους (Ηγέτης, Γέφυρα). <i>Περιορισμοί:</i> Αγνοεί ακαδημαϊκή επίδοση, η ανίχνευση κοινοτήτων διατηρεί υπαρχουσες ομάδες.	Rabbany et al. (2014). Maqtary et al. (2019)

Βάσει γνώ- σης	Ρητοί παιδα- γωγικοί κα- νόνες (π.χ. "μία ομάδα πρέπει να έ- χει τουλάχισ- τον έναν υ- ψηλόβαθμο φοιτητή")	Συστήματα κα- νόνων, λογική προγραμματι- σμού	<i>Ευρήματα:</i> Εγγυάται τήρηση συγκεκριμέ- νων κριτηρίων. <i>Περιορισμοί:</i> Δυσκί- νητη δημιουργία/συ- ντήρηση κανόνων, χα- μηλή προσαρμοστικό- τητα.	Adomavicius & Tuzhilin (2011)
Υβριδικές / CARS	Συνδυασμός δεδομένων: επίδοση, συμμετοχή, SNA	Γενετικοί αλγό- ριθμοι, greedy βελτιστοποίηση	<i>Ευρήματα:</i> Ισορροπεί ομοιογένεια και ετε- ρογένεια, μετριάζει περιορισμούς απλών μεθόδων. <i>Περιορισμοί:</i> Αυξη- μένη πολυπλοκότητα σχεδιασμού και αξιο- λόγησης.	Maqtary et al. (2019)

Πίνακας 3-1 Συγκριτική ανάλυση προσεγγίσεων σχηματισμού ομάδων σε εκπαιδευτικά περιβάλλοντα

3.2.1 Προσεγγίσεις βάσει προφίλ και περιεχομένου (Profile/Content-Based)

Οι προσεγγίσεις αυτής της κατηγορίας στοχεύουν πρωταρχικά στην ομοιογένεια, ομαδοποιώ-
ντας φοιτητές με βάση την ομοιότητα στα χαρακτηριστικά ή στα προφίλ τους, όπως προηγού-
μενες γνώσεις, ενδιαφέροντα ή ακαδημαϊκές επιδόσεις. Το κύριο πλεονέκτημά τους είναι η
ανεξαρτησία από δεδομένα άλλων χρηστών και η καλή αντιμετώπιση του προβλήματος «ψυ-
χρής εκκίνησης». Ωστόσο, το βασικό τους μειονέκτημα για τη συνεργατική μάθηση είναι ότι
τείνουν να δημιουργούν υπερβολικά ομοιογενείς ομάδες, περιορίζοντας τη συμπληρωματικό-
τητα δεξιοτήτων και την ετερογένεια που είναι απαραίτητη για την αποτελεσματική συνεργα-
τική αλληλεπίδραση (Maqtary et al., 2019).

3.2.2 Προσεγγίσεις βάσει συνεργατικού φιλτραρίσματος (Collaborative Filtering - CF)

Οι μέθοδοι CF βασίζονται στην υπόθεση ότι φοιτητές με παρόμοιο ιστορικό συμπεριφοράς ή επιδόσεων μπορούν να συνεργαστούν αποτελεσματικά. Μπορούν να αξιοποιήσουν δεδομένα από προηγούμενες ομαδικές δραστηριότητες ή από ατομικές επιδόσεις σε παρόμοιες εργασίες. Παρότι είναι ισχυρές σε συστήματα με πλούσιο ιστορικό, αντιμετωπίζουν το πρόβλημα «ψυχής εκκίνησης» για νέους φοιτητές χωρίς ιστορικό (Adomavicius & Tuzhilin, 2011). Επιπλέον, λόγω της εξάρτησής τους από υπάρχουσες συνεργασίες, τείνουν να ενισχύουν υπάρχουσες ομαδικές δυναμικές και μπορεί να περιορίζουν την ανακάλυψη νέων, πιθανώς πιο συμπληρωματικών συνεργασιών (Bellogin et al., 2017).

3.2.3 Προσεγγίσεις βάσει κοινωνικού δικτύου (Social Network-Based)

Αυτές οι προσεγγίσεις επωφελούνται από τη δομή των αλληλεπιδράσεων μεταξύ των φοιτητών (π.χ., από φόρουμ) για να οδηγήσουν τον σχηματισμό ομάδων. Ανάλογα με τη χρησιμοποιούμενη τεχνική, μπορούν να υπηρετήσουν τόσο τον στόχο της ομοιογένειας όσο και της ετερογένειας. Μέθοδοι όπως η ανίχνευση κοινοτήτων τείνουν να ομαδοποιούν φοιτητές που ήδη αλληλεπιδρούν έντονα μεταξύ τους, δημιουργώντας ομάδες με υψηλή κοινωνική συνοχή και ομοιογένεια. Αντίθετα, η ανάλυση μετρικών κεντρικότητας για την ανάδειξη λειτουργικών ρόλων (π.χ., του «συντονιστή» με υψηλό degree ή της «γέφυρας» με υψηλή betweenness centrality) μπορεί να αποσκοπήσει στη δημιουργία ομάδων με ετερογένεια και συμπληρωματικότητα ρόλων. Η βασική δύναμη της SNA έγκειται στην ικανότητα να αποκαλύπτει λανθάνοντα μοτίβα στις σχέσεις που δεν είναι προφανή από επιφανειακή παρατήρηση. Ωστόσο, όταν εφαρμόζονται μεμονωμένα, οι προσεγγίσεις αυτές συχνά αγνοούν κρίσιμες ακαδημαϊκές διαστάσεις (π.χ., επίδοση) και, στην περίπτωση της ανίχνευσης κοινοτήτων, μπορεί να ενισχύσουν υπάρχουσες κοινωνικές ομάδες χωρίς να προάγουν τη διασύνδεση ή τη συμπληρωματικότητα πέρα από τη κοινωνική συνοχή.

3.2.4 Συστήματα βάσει γνώσης (Knowledge-Based Systems)

Οι προσεγγίσεις αυτής της κατηγορίας επιδιώκουν ρητά την ετερογένεια και την ισορροπία, βασιζόμενες σε προκαθορισμένους κανόνες που κωδικοποιούν παιδαγωγικές αρχές. Στο πλαίσιο σχηματισμού ομάδων, αυτό σημαίνει ότι ο σχηματισμός γίνεται σύμφωνα με ρητά ορισμένους κανόνες όπως «κάθε ομάδα πρέπει να έχει τουλάχιστον έναν φοιτητή με υψηλό βαθμό» ή «οι ομάδες πρέπει να έχουν ισορροπημένη κατανομή δεξιοτήτων» (Adomavicius & Tuzhilin,

2011). Έτσι, δημιουργούν ομάδες με δομημένη συμπληρωματικότητα. Ωστόσο, ένας κρίσιμος περιορισμός είναι ότι, επικεντρωμένες στην ετερογένεια, μπορεί να παραμελήσουν τη δημιουργία μιας φυσικής κοινής βάσης συνεργασίας (ομοιογένεια κινήτρων) μεταξύ των μελών. Επιπλέον, απαιτούν τη δημιουργία και διατήρηση σύνθετων κανόνων, οι οποίοι μπορεί να είναι δυσκίνητοι στην επέκταση και να μην προσαρμόζονται εύκολα σε διαφορετικά εκπαιδευτικά πλαίσια.

3.2.5 Υβριδικές προσεγγίσεις και συστήματα με επίγνωση πλαισίου

Οι υβριδικές προσεγγίσεις συνδυάζουν δύο ή περισσότερες από τις παραπάνω μεθόδους για να μετριάσουν τα μειονεκτήματα της κάθε μίας και να ενισχύσουν τα πλεονεκτήματά τους (Drachsler et al., 2015). Παράλληλα, μια διακριτή διάσταση των σύγχρονων συστημάτων συστάσεων αποτελεί η επίγνωση πλαισίου (context-awareness), η οποία αναφέρεται στη συνεκτίμηση των συνθηκών υπό τις οποίες πραγματοποιείται η σύσταση (Adomavicius & Tuzhilin, 2011). Τα συστήματα που λαμβάνουν υπόψη τέτοιες πληροφορίες χαρακτηρίζονται ως Συστήματα Συστάσεων με Επίγνωση Πλαισίου (Context-Aware Recommender Systems – CARS). Ένα σύστημα μπορεί να είναι υβριδικό, να διαθέτει επίγνωση πλαισίου ή να συνδυάζει και τα δύο χαρακτηριστικά.

Στην παρούσα εργασία, το πλαίσιο αφορά τις παιδαγωγικές και οργανωτικές συνθήκες υπό τις οποίες πραγματοποιείται η σύσταση ομάδων εντός του Moodle. Οι συνθήκες αυτές μεταφράζονται σε παραμέτρους, κριτήρια και περιορισμούς του αλγορίθμου, όπως το επιθυμητό μέγεθος των ομάδων και οι κανόνες σύνθεσης που απορρέουν από τις αρχές της συνεργατικής μάθησης. Οι κανόνες αυτοί αξιοποιούν τα χαρακτηριστικά του πολυδιάστατου προφίλ κάθε φοιτητή για την παραγωγή των προτεινόμενων ομάδων.

3.3 Αλγοριθμικές τεχνικές

Αυτή η ενότητα αναλύει τις συγκεκριμένες αλγοριθμικές τεχνικές που χρησιμοποιούνται στη βιβλιογραφία για τις δύο κρίσιμες φάσεις της διαδικασίας σχηματισμού ομάδων: (α) την ανάλυση και κατηγοριοποίηση των φοιτητών βάσει χαρακτηριστικών, και (β) τον βέλτιστο σχηματισμό των ομάδων.

3.3.1 Τεχνικές ομαδοποίησης (Clustering) για δημιουργία προφίλ

Για την κατηγοριοποίηση φοιτητών σε ομάδες με βάση την ομοιότητα χαρακτηριστικών, έχουν εφαρμοστεί διάφοροι αλγόριθμοι ομαδοποίησης στην εκπαιδευτική βιβλιογραφία:

- **K-means:** Ο πιο διαδεδομένος αλγόριθμος ομαδοποίησης, ο οποίος διαιρεί τα δεδομένα σε προκαθορισμένο αριθμό k ομάδων με βάση την ελαχιστοποίηση της απόστασης από το κέντρο κάθε ομάδας (centroid). Είναι απλός και αποτελεσματικός για μεγάλα σύνολα δεδομένων, αλλά απαιτεί τον προκαθορισμό του k και είναι ευαίσθητος στις αρχικές συνθήκες και στις ακραίες τιμές (outliers) (Jain, 2010)
- **Ιεραρχική ομαδοποίηση (Hierarchical Clustering):** Δημιουργεί ένα δέντρο ομάδων (dendrogram) που επιτρέπει την ανάλυση των δεδομένων σε διαφορετικά επίπεδα λεπτομέρειας, χωρίς προκαθορισμό του αριθμού των ομάδων. Είναι ιδιαίτερα χρήσιμος για την εξερεύνηση της εσωτερικής δομής των δεδομένων, αλλά είναι υπολογιστικά απαιτητικότερος από τον K-means για μεγάλα σύνολα δεδομένων (Murtagh & Contreras, 2012).
- **DBSCAN (Density-Based Spatial Clustering):** Βρίσκει ομάδες βασισμένες στην πυκνότητα των σημείων στο χώρο των χαρακτηριστικών. Το βασικό του πλεονέκτημα είναι ότι δεν απαιτεί προκαθορισμένο αριθμό ομάδων και μπορεί να εντοπίσει και «θόρυβο» (outliers), δηλαδή φοιτητές που δεν ταιριάζουν εύκολα σε καμία κύρια ομάδα (Ester et al., 1996). Αυτό τον καθιστά κατάλληλο για ανίχνευση ασυνήθιστων προτύπων συμπεριφοράς σε εκπαιδευτικά δεδομένα.

3.3.2 Αλγόριθμοι βελτιστοποίησης για σύνθεση ομάδων

Ο σχηματισμός ομάδων μπορεί να διατυπωθεί ως πρόβλημα πολυκριτηριακής βελτιστοποίησης (multi-objective optimization), όπου πρέπει να μεγιστοποιηθούν ή ελαχιστοποιηθούν πολλαπλοί, συχνά ανταγωνιστικοί στόχοι (π.χ., ομοιογένεια κινήτρων, ετερογένεια δεξιοτήτων, ισορροπία ρόλων).

- **Άπληστοι αλγόριθμοι (Greedy Algorithms):** Κάνουν τη βέλτιστη τοπική επιλογή σε κάθε βήμα της διαδικασίας σχηματισμού ομάδων. Είναι γρήγοροι και απλοί στην υλοποίηση, κάτι που τους καθιστά κατάλληλους για πραγματικές εφαρμογές. Ωστόσο, δεν εγγυώνται το γενικότερο βέλτιστο αποτέλεσμα, καθώς οι τοπικά βέλτιστες επιλογές μπορεί να οδηγήσουν σε ένα αποτέλεσμα που δεν είναι το καλύτερο δυνατό για ολόκληρο το πρόβλημα (Cormen et al., 2009).

- **Γενετικοί αλγόριθμοι (Genetic Algorithms - GA):** Είναι μετα-ευρετικοί αλγόριθμοι που μιμούνται τη φυσική εξέλιξη, δουλεύοντας με πληθυσμούς πιθανών λύσεων που εξελίσσονται κατά τη διάρκεια του χρόνου. Είναι ιδιαίτερα κατάλληλοι για προβλήματα με πολλά κριτήρια και μεγάλο χώρο αναζήτησης, καθώς μπορούν να βρουν καλές προσεγγιστικές λύσεις χωρίς να εξαντλούν όλες τις πιθανότητες. Το βασικό τους μειονέκτημα είναι το υψηλό υπολογιστικό κόστος, ιδιαίτερα για μεγάλα σύνολα δεδομένων (Moreno et al., 2012).
- **Αλγόριθμοι αντιστοίχισης ή ανάθεσης (Matching/Assignment Algorithms):** Διατυπώνουν το πρόβλημα ως αντιστοίχιση φοιτητών σε προκαθορισμένες θέσεις, με στόχο την ελαχιστοποίηση μιας συνάρτησης κόστους ή τη μεγιστοποίηση μιας συνάρτησης ωφέλειας. Στο κλασικό γραμμικό πρόβλημα ανάθεσης, μέθοδοι όπως ο ουγγρικός αλγόριθμος μπορούν να υπολογίσουν τη βέλτιστη αντιστοίχιση (Kuhn, 1955). Ωστόσο, ο σχηματισμός ομάδων περιλαμβάνει συχνά αλληλεξαρτήσεις μεταξύ περισσότερων από δύο μελών, πολλαπλά κριτήρια και σύνθετους περιορισμούς, με αποτέλεσμα να απαιτούνται επεκτάσεις του βασικού μοντέλου ή συνδυασμός του με άλλες μεθόδους.

3.4 Εντοπισμός ερευνητικού κενού

Από τη συγκριτική ανάλυση των προσεγγίσεων αναδεικνύεται ότι οι υπάρχουσες κατηγορίες, όταν εφαρμόζονται μεμονωμένα, δεν επαρκούν για την πλήρη αντιμετώπιση του πολυδιάστατου προβλήματος σχηματισμού ομάδων. Οι προσεγγίσεις που στοχεύουν στην ομοιογένεια τείνουν να παραμελούν τη συμπληρωματικότητα, ενώ αυτές που στοχεύουν στην ετερογένεια ενδέχεται να υποβαθμίζουν την ανάγκη για ένα κοινό πλαίσιο αναφοράς και αμοιβαίας κατανόησης μεταξύ των μελών.

Επιπλέον, στην υπάρχουσα βιβλιογραφία διαπιστώνονται τέσσερα επιμέρους κενά. Παρά τη διαθεσιμότητα πλούσιων δεδομένων καταγραφής σε πλατφόρμες όπως το Moodle, αρκετές από τις προσεγγίσεις που εξετάστηκαν είτε βασίζονται σε εξωτερικά ερωτηματολόγια είτε αξιοποιούν περιορισμένο υποσύνολο των διαθέσιμων δεδομένων. Συναπώς, παρατηρείται έλλειψη μεθοδολογιών που να αξιοποιούν τα δεδομένα αυτά για τη σύνθεση ποσοτικών δεικτών συμμετοχής και επίδοσης, λειτουργικών κοινωνικών ρόλων που αναδεικνύονται από την ανάλυση των αλληλεπιδράσεων, καθώς και συμπεριφορικών ενδείξεων πιθανής απογοήτευσης, σε ένα ενιαίο, πολυδιάστατο προφίλ φοιτητή. Παράλληλα, ενώ τα Συστήματα Συστάσεων με Επίγνωση Πλαισίου (CARS) έχουν αναγνωριστεί ως κατάλληλο θεωρητικό πλαίσιο,

η εφαρμογή τους για τη διατύπωση του σχηματισμού ομάδων ως προβλήματος πολυκριτηριακής σύστασης και λήψης απόφασης παραμένει περιορισμένη. Τέλος, αρκετές ερευνητικές προτάσεις παραμένουν σε θεωρητικό ή εργαστηριακό επίπεδο, χωρίς να εξετάζουν τις τεχνικές προϋποθέσεις για την πρακτική ενσωμάτωσή τους σε ένα λειτουργικό περιβάλλον LMS όπως το Moodle.

Συνεπώς, το κύριο ερευνητικό κενό που αναδεικνύεται είναι η έλλειψη ενός ολοκληρωμένου πλαισίου που να συνδυάζει τα παραπάνω στοιχεία σε μια συνεκτική, αυτοματοποιημένη διαδικασία. Η παρούσα εργασία επιχειρεί να καλύψει το κενό αυτό, προτείνοντας ένα υβριδικό πλαίσιο που αξιοποιεί τεχνικές Εκπαιδευτικής Εξόρυξης Δεδομένων (EDM) για την εξαγωγή δεικτών συμμετοχής και επίδοσης, Ανάλυση Κοινωνικών Δικτύων (SNA) για την ανάδειξη λειτουργικών κοινωνικών ρόλων, και πολυκριτηριακή σύσταση στο πλαίσιο των CARS, ενώ παράλληλα ενσωματώνει συμπεριφορικές ενδείξεις πιθανής απογοήτευσης ως επιπλέον διάσταση του πολυδιάστατου προφίλ. Η αναλυτική περιγραφή της προτεινόμενης προσέγγισης παρουσιάζεται στα Κεφάλαια 4 και 5.

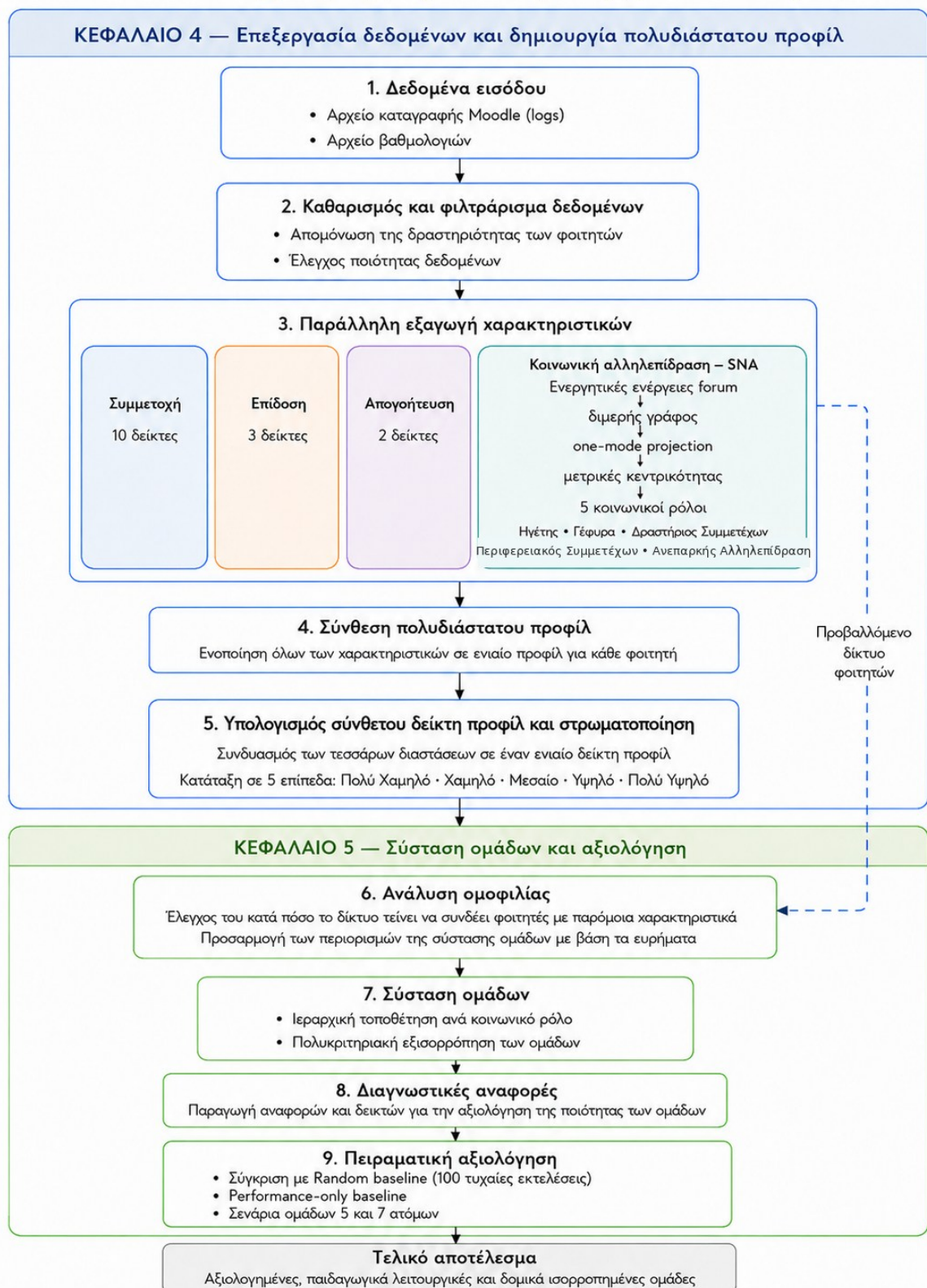
4 Μεθοδολογία ανάλυσης δεδομένων και δημιουργίας προφίλ φοιτητών

Στο κεφάλαιο αυτό παρουσιάζεται η μεθοδολογία που ακολουθήθηκε για την επεξεργασία των δεδομένων και τη δημιουργία των προφίλ των φοιτητών. Η προσέγγιση εντάσσεται στο πεδίο των Learning Analytics, καθώς αξιοποιεί τα ψηφιακά ίχνη που αφήνουν οι φοιτητές κατά την αλληλεπίδρασή τους με το Moodle.

Βασική αφετηρία αποτέλεσε η μελέτη της δομής των πραγματικών αρχείων καταγραφής του Moodle, τα οποία αποτυπώνουν λεπτομερώς τη μαθησιακή δραστηριότητα. Λόγω των περιορισμών που θέτει το GDPR και της αδυναμίας άμεσης πρόσβασης σε πραγματικά δεδομένα, κρίθηκε αναγκαία η δημιουργία ενός συνθετικού συνόλου δεδομένων υψηλής πιστότητας, η διαδικασία παραγωγής του οποίου τεκμηριώνεται διεξοδικά.

Πριν από την αναλυτική παρουσίαση των επιμέρους σταδίων, παρατίθεται συνοπτική απεικόνιση της συνολικής μεθοδολογικής ροής (Σχήμα 4-1). Το διάγραμμα αυτό αποτυπώνει τη ροή της μεθοδολογίας στο σύνολό της, από την επεξεργασία των δεδομένων (Κεφάλαιο 4) έως τη σύσταση και αξιολόγηση των ομάδων (Κεφάλαιο 5). Οι λεπτομέρειες κάθε σταδίου αναλύονται εκτενώς στις αντίστοιχες ενότητες.

Συνολική ροή της προτεινόμενης μεθοδολογίας



Σχήμα 4-1 Συνοπτική απεικόνιση της μεθοδολογικής ροής της εργασίας

Ακολουθεί η περιγραφή των σταδίων επεξεργασίας: καθαρισμός δεδομένων, εξαγωγή χαρακτηριστικών συμμετοχής, επίδοσης, κοινωνικής αλληλεπίδρασης (SNA) και σημάτων απογοήτευσης, και τελικά η σύνθεση ενός ενιαίου πολυδιάστατου προφίλ για κάθε φοιτητή.

4.1 Δεδομένα και προεπεξεργασία

4.1.1 Χαρακτηριστικά των Moodle log files

Τα αρχεία καταγραφής (log files) του Moodle αποτελούν την πρωταρχική πηγή πληροφορίας για την ανάλυση της μαθησιακής συμπεριφοράς, καθώς καταγράφουν με λεπτομέρεια κάθε ενέργεια των χρηστών. Κάθε γραμμή του αρχείου αντιστοιχεί σε ένα μοναδικό γεγονός (event) και περιλαμβάνει μια σειρά από πεδία, η κατανόηση των οποίων είναι κρίσιμη για την ορθή επεξεργασία και εξαγωγή χαρακτηριστικών.

Στον παρακάτω πίνακα (Πίνακας 4-1) παρουσιάζονται συνοπτικά τα βασικότερα πεδία ενός τυπικού log file του Moodle.

Πεδίο	Περιγραφή
Timestamp	Η χρονοσφραγίδα του γεγονότος, που περιλαμβάνει ημερομηνία και ώρα με ακρίβεια λεπτού.
User ID	Το μοναδικό αναγνωριστικό του χρήστη που πραγματοποίησε την ενέργεια.
Full name	Το πλήρες όνομα του χρήστη. Για λόγους προστασίας προσωπικών δεδομένων, το πεδίο αυτό αντικαθίσταται κατά τη διαδικασία ανωνυμοποίησης με το μοναδικό αναγνωριστικό.
Event context	Η ευρύτερη περιοχή ή δέσμη ενεργειών στην οποία εντάσσεται το γεγονός (π.χ., μάθημα, assignment, forum).
Component	Το υποσύστημα του Moodle που σχετίζεται με το γεγονός (π.χ., system, log, file).
Event name	Η συγκεκριμένη ενέργεια που καταγράφηκε (π.χ., course viewed, assignment submitted, forum post created).

Description	Το πλουσιότερο σε πληροφορία πεδίο. Περιγράφει αναλυτικά το γεγονός, αποτυπώνοντας ποιος χρήστης αλληλεπιδρά με ποιον πόρο (π.χ., "The user with id 'user107' has created post in discussion '67890'", "The user with id 'user107' has submitted assignment 'WA1'").
Origin	Η προέλευση του γεγονότος (π.χ., web interface, cli, ws).
IP	Η διεύθυνση IP του χρήστη.

Πίνακας 4-1 Βασικά πεδία ενός Moodle log file

4.1.2 Δημιουργία συνθετικού συνόλου δεδομένων

Η πλατφόρμα Moodle παρέχει στους χρήστες με ρόλο tutor τη δυνατότητα εξαγωγής (export) των αρχείων καταγραφής (log files) σε επεξεργάσιμες μορφές (π.χ., Excel, CSV) απευθείας από το περιβάλλον διαχείρισης του μαθήματος. Ωστόσο, η πρόσβαση σε πραγματικά εκπαιδευτικά δεδομένα για ερευνητικούς σκοπούς προϋποθέτει αυστηρές διαδικασίες αδειοδότησης και συμμόρφωσης με το Γενικό Κανονισμό Προστασίας Δεδομένων (GDPR). Για το λόγο αυτό, στο πλαίσιο της παρούσας εργασίας επιλέχθηκε η χρήση συνθετικού συνόλου δεδομένων υψηλής πιστότητας, προκειμένου να διασφαλιστεί τόσο η ηθική συμμόρφωση όσο και ο πλήρης έλεγχος της δομής των δεδομένων.

Η παραγωγή του συνθετικού dataset πραγματοποιήθηκε με τη χρήση του μεγάλου γλωσσικού μοντέλου (Large Language Model – LLM) της OpenAI, ChatGPT (έκδοση 5.4 Thinking), το οποίο χρησιμοποιήθηκε ως εργαλείο παραγωγής δομημένων δεδομένων βάσει προκαθορισμένων προδιαγραφών. Οι πειραματικές δοκιμές πραγματοποιήθηκαν κατά τη διάρκεια του Ιανουαρίου 2026, ενώ το τελικό αρχείο οριστικοποιήθηκε στις 27 Ιανουαρίου 2026. Η διαδικασία βασίστηκε σε μια σειρά από σαφείς προδιαγραφές (specifications), οι οποίες αντλήθηκαν από τη μελέτη του επίσημου τεχνικού οδηγού (documentation) του Moodle για τα αρχεία καταγραφής. Οι προδιαγραφές αυτές διασφάλισαν ότι το παραγόμενο σύνολο θα προσομοιάζει πιστά ένα πραγματικό αρχείο καταγραφής και ήταν οι εξής:

- **Δομική πιστότητα:** Τα συνθετικά δεδομένα έπρεπε να έχουν ακριβώς την ίδια μορφή και δομή στηλών με ένα αυθεντικό log file του Moodle, όπως αυτή τεκμηριώνεται στο επίσημο documentation, ώστε να είναι δυνατή η εφαρμογή της ίδιας μεθοδολογίας επεξεργασίας.

- **Χρονική ακρίβεια:** Το πεδίο timestamp θα έπρεπε να περιλαμβάνει ημερομηνία και ώρα με ακρίβεια λεπτού, καθώς η πληροφορία αυτή είναι κρίσιμη για μετέπειτα αναλύσεις (π.χ., ανίχνευση γρήγορων διαδοχικών ενεργειών).
- **Χρονική περίοδος:** Ζητήθηκε η προσομοίωση δραστηριότητας για το ακαδημαϊκό έτος 2023-2024.
- **Ανωνυμοποίηση:** Αντί για πραγματικά ονόματα, τα δεδομένα έπρεπε να περιλαμβάνουν μοναδικά αναγνωριστικά (user IDs) για κάθε φοιτητή. Για την επίτευξη συνεπούς αντιστοίχισης, ζητήθηκε το user ID που θα αντικαταστήσει το όνομα να αντλείται από το αντίστοιχο αναγνωριστικό που εμφανίζεται στο πεδίο *description* των log files. Με τον τρόπο αυτό, διασφαλίζεται ότι κάθε φοιτητής έχει ένα μοναδικό ID, το οποίο εμφανίζεται με συνέπεια τόσο στη στήλη ταυτοποίησης όσο και μέσα στις περιγραφές των γεγονότων.
- **Αφαίρεση περιττών πεδίων:** Η στήλη της διεύθυνσης IP παραλείφθηκε σκόπιμα, καθώς δεν προσφέρει αξιοποιήσιμη πληροφορία για τους σκοπούς της ανάλυσης που πρόκειται να διεξαχθεί.
- **Προσομοίωση ρόλων μέσω ID:** Μια ιδιαίτερα σημαντική προδιαγραφή αφορούσε την προσομοίωση της διαφοροποίησης των ρόλων των χρηστών μέσω του μήκους του ID. Έχει παρατηρηθεί ότι το Moodle αποδίδει διαφορετικού μήκους αναγνωριστικά ανάλογα με το ρόλο του χρήστη. Συγκεκριμένα, τα ID των φοιτητών έχουν συνήθως μήκος επτά χαρακτήρων, ενώ τα ID των διδασκόντων ή των διαχειριστών έχουν μεγαλύτερο μήκος. Η ιδιότητα αυτή ενσωματώθηκε στα συνθετικά δεδομένα, επιτρέποντας τον μετέπειτα εύκολο διαχωρισμό των φοιτητών από άλλες κατηγορίες χρηστών κατά τη φάση του φιλτραρίσματος.
- **Πληθυσμός:** Το σύνολο δεδομένων δημιουργήθηκε ώστε να περιλαμβάνει δραστηριότητα για 382 φοιτητές, ένα μέγεθος αντιπροσωπευτικό ενός τυπικού μεταπτυχιακού προγράμματος ή μεγάλου προπτυχιακού μαθήματος. Το αρχείο βαθμολογιών περιλάμβανε 383 μοναδικούς φοιτητές. Ένας φοιτητής εμφανιζόταν μόνο στο αρχείο βαθμολογιών και διατηρήθηκε κατά τη σύνθεση του τελικού μαθησιακού προφίλ.

Η διαδικασία παραγωγής κατέληξε στη δημιουργία δύο διακριτών αρχείων:

1. **Κύριο log file:** Περιλαμβάνει το σύνολο των γεγονότων με τα πεδία που περιγράφηκαν, αποτελώντας την κύρια πηγή για την εξαγωγή δεικτών συμμετοχής και ανάλυσης κοινωνικών δικτύων.

2. Αρχείο βαθμολογιών: Περιλαμβάνει τα user IDs και τις αντίστοιχες βαθμολογίες των φοιτητών, επιτρέποντας τη συσχέτιση της ακαδημαϊκής επίδοσης με τα υπόλοιπα χαρακτηριστικά.

Για τον έλεγχο της δομικής συνέπειας, πραγματοποιήθηκε δειγματοληπτικός έλεγχος των παραγόμενων εγγραφών, ώστε να διαπιστωθεί η ορθότητα της μορφής των timestamps, η μοναδικότητα των user IDs και η συνέπεια των αναφορών στα πεδία *description*.

Η τεκμηριωμένη αυτή προσέγγιση παραγωγής συνθετικών δεδομένων, βασισμένη σε δημόσια διαθέσιμες προδιαγραφές, επέτρεψε την ανάπτυξη και δοκιμή της μεθοδολογίας εξαγωγής χαρακτηριστικών και ανάλυσης κοινωνικών δικτύων σε ένα ρεαλιστικό, αν και συνθετικό, περιβάλλον. Η μεθοδολογία που αναπτύσσεται στα επόμενα στάδια είναι σχεδιασμένη ώστε να είναι, κατ' αρχήν, εφαρμόσιμη σε πραγματικά δεδομένα Moodle, με την προϋπόθεση προσαρμογής στις εκάστοτε διαθέσιμες στήλες και την ακριβή δομή τους. Η έλλειψη πεδίων όπως το IP ή η τροποποίηση άλλων (π.χ., η αντικατάσταση του ονόματος από το user ID του πεδίου *description*) δεν επηρεάζει την εγκυρότητα των αναλύσεων, καθώς τα πεδία αυτά δεν αξιοποιούνται στην εξαγωγή των κρίσιμων χαρακτηριστικών (συμμετοχή, επίδοση, κοινωνική αλληλεπίδραση).

Είναι σημαντικό να αναγνωριστούν οι εγγενείς περιορισμοί της χρήσης συνθετικών δεδομένων. Παρά τη δομική συνέπεια του συνθετικού συνόλου δεδομένων που διασφαλίστηκε μέσω των προδιαγραφών, τα συνθετικά δεδομένα δεν μπορούν να αποτυπώσουν πλήρως την πολυπλοκότητα της πραγματικής ανθρώπινης συμπεριφοράς σε ένα εκπαιδευτικό περιβάλλον. Συγκεκριμένα, η χρονική κατανομή της δραστηριότητας, τα πρότυπα αλληλεπίδρασης μεταξύ φοιτητών και η συναισθηματική διάσταση της μάθησης προσεγγίζονται μέσω στατιστικών κανόνων και όχι μέσω πραγματικών συμπεριφορικών δεδομένων. Επιπλέον, το LLM που χρησιμοποιήθηκε για την παραγωγή των δεδομένων βασίζεται σε στατιστικές συσχετίσεις από το corpus εκπαίδευσής του και όχι σε πραγματικές μαθησιακές ακολουθίες. Η επιλογή αυτή δημιουργεί ένα εγγενές trade-off (αντιστάθμιση): από τη μία πλευρά, επιτυγχάνεται υψηλή ακρίβεια και λεπτομέρεια των δεδομένων, από την άλλη, η προσέγγιση βασίζεται σε εφαρμόσιμες τακτικές που σέβονται τα ευαίσθητα προσωπικά δεδομένα. Ως εκ τούτου, τα αποτελέσματα της παρούσας εργασίας θα πρέπει να θεωρηθούν ως ένδειξη της δυνατότητας και της αποτελεσματικότητας της προτεινόμενης μεθοδολογίας υπό ιδανικές συνθήκες δομικής συνέπειας, με την κατανόηση ότι η εφαρμογή σε πραγματικά δεδομένα ενδέχεται να απαιτήσει πρόσθετες προσαρμογές.

Τα βασικά χαρακτηριστικά του συνθετικού συνόλου δεδομένων συνοψίζονται στον παρακάτω πίνακα (Πίνακας 4-2).

Χαρακτηριστικό	Τιμή
Μοναδικοί φοιτητές στο αρχείο καταγραφής	382
Μοναδικοί φοιτητές στο αρχείο βαθμολογιών	383
Περίοδος καταγραφής	Ακαδημαϊκό έτος 2023–2024
Συνολικός αριθμός events	786833
Πεδία log file	Time, User full name, Event context, Component, Event name, Description, Origin
Αφαιρεθέντα πεδία	IP

Πίνακας 4-2 Χαρακτηριστικά του συνθετικού dataset

4.1.3 Καθαρισμός και φιλτράρισμα δεδομένων

Η προεπεξεργασία των δεδομένων των αρχείων καταγραφής αποτελεί απαραίτητο στάδιο πριν από την εξαγωγή χαρακτηριστικών μαθησιακής δραστηριότητας, καθώς διασφαλίζει ότι οι επόμενες αναλύσεις βασίζονται αποκλειστικά σε έγκυρες και συνεπείς εγγραφές που αφορούν τη δραστηριότητα των φοιτητών. Στόχος είναι η απομάκρυνση εγγραφών που σχετίζονται με άλλους τύπους χρηστών ή τεχνικές/διοικητικές ενέργειες, καθώς και ο εντοπισμός τυχόν ασυνεπειών που θα μπορούσαν να αλλοιώσουν τα αποτελέσματα.

Η διαδικασία επεξεργασίας δεδομένων υλοποιήθηκε σε Python με χρήση της βιβλιοθήκης pandas. Τα ενδιάμεσα αποτελέσματα αποθηκεύονται σε μορφή CSV, εξασφαλίζοντας ταχεία ανάγνωση μεγάλου όγκου δεδομένων και αποφεύγοντας την ανάγκη για επιπρόσθετες βιβλιοθήκες ανάγνωσης αρχείων Excel, όπως οι openpyxl ή xlrd. Η επιλογή αυτή ενισχύει την αρθρωτή δομή (modularity) του συστήματος, επιτρέποντας σε κάθε στάδιο του pipeline να λειτουργεί αυτόνομα.

Κατ' εξαίρεση, το αρχείο αναφοράς (Report_4_1_3) που συνοψίζει τα αποτελέσματα των σταδίων επεξεργασίας εξάγεται επιπλέον σε μορφή .xlsx, για λόγους ευκολίας ανάγνωσης και άμεσης επιθεώρησης.

Η διαδικασία περιλαμβάνει τα στάδια που συνοψίζονται στον Πίνακα 4-3

Βήμα	Περιγραφή
Φιλτράρισμα origin	Αφαίρεση γεγονότων με προέλευση cli ή ws
Φιλτράρισμα ρόλων	Αφαίρεση εγγραφών που αντιστοιχούν σε διδάσκοντες ή διαχειριστές
Έλεγχος συνέπειας	Έλεγχος timestamps, IDs και δομής description
Τελικό dataset	Διατήρηση μόνο έγκυρων εγγραφών δραστηριότητας φοιτητών

Πίνακας 4-3 Βήματα καθαρισμού δεδομένων

Το πρώτο στάδιο αφορά τον αποκλεισμό εγγραφών που δεν σχετίζονται με τη μαθησιακή δραστηριότητα των φοιτητών στο περιβάλλον Moodle. Συγκεκριμένα, διατηρήθηκαν μόνο οι εγγραφές όπου η στήλη Origin έχει τιμή "web", αποκλείοντας τεχνικές/διαχειριστικές ενέργειες που προέρχονται από cli (command line interface) ή ws (web service). Οι εγγραφές αυτές συνήθως αντιστοιχούν σε αυτοματοποιημένες διεργασίες του συστήματος ή σε ενέργειες διαχειριστών και δεν αντανakλούν άμεσα τη μαθησιακή δραστηριότητα των φοιτητών. Από τις 786.833 αρχικές εγγραφές, οι 780.158 αντιστοιχούν σε ενέργειες που πραγματοποιήθηκαν μέσω της διεπαφής χρήστη.

Στη συνέχεια, πραγματοποιήθηκε φιλτράρισμα για τη διατήρηση μόνο των εγγραφών που αντιστοιχούν σε φοιτητές. Αξιοποιώντας τη σχεδιαστική επιλογή που περιγράφεται στην Ενότητα 4.1.2 (διαφοροποίηση ρόλων μέσω του μήκους του αναγνωριστικού), εφαρμόστηκε φίλτρο στη στήλη User full name με βάση το τριψήφιο αναγνωριστικό της μορφής "userXXX" που φέρουν οι φοιτητές. Με τον τρόπο αυτό, το σύνολο δεδομένων μειώθηκε σε 686.275 γραμμές, που αντιστοιχούν σε 382 μοναδικούς φοιτητές.

Για τη διασφάλιση της αξιοπιστίας των δεδομένων, πραγματοποιήθηκαν τρεις βασικοί έλεγχοι ποιότητας. Πρώτον, έλεγχος εγκυρότητας των timestamps: όλες οι εγγραφές ελέγχθηκαν ως προς τη δυνατότητα μετατροπής τους σε έγκυρη μορφή ημερομηνίας-ώρας με βάση το πρότυπο "%d/%m/%y, %H:%M", ώστε να εντοπιστούν τυχόν αλλοιωμένες ή μη έγκυρες χρονο-

σφραγίδες που θα μπορούσαν να επηρεάσουν αναλύσεις χρονικής ακολουθίας. Δεύτερον, έλεγχος συνέπειας του πεδίου Description: από το πεδίο αυτό εξήχθη, μέσω κανονικής έκφρασης, το αναγνωριστικό του χρήστη που πραγματοποίησε την ενέργεια (π.χ., από τη φράση "The user with id 'user107' ..."). Στη συνέχεια, ελέγχθηκε αν το εξαγόμενο αναγνωριστικό ταυτίζεται με την τιμή της στήλης User full name. Η συνέπεια αυτή είναι κρίσιμη για την ορθή συσχέτιση των ενεργειών με τους φοιτητές. Τρίτον, έλεγχος μοναδικότητας αντιστοίχισης: για κάθε μοναδικό User full name, ελέγχθηκε αν οι τιμές που εξάγονται από το πεδίο Description ως actor_user (το αναγνωριστικό του χρήστη που εκτέλεσε την ενέργεια) είναι πάντα οι ίδιες. Με άλλα λόγια, εξετάστηκε αν ένας συγκεκριμένος φοιτητής εμφανίζεται στο πεδίο Description με διαφορετικά αναγνωριστικά στο σύνολο του αρχείου καταγραφής, γεγονός που θα υποδήλωνε ασυνέπεια στα δεδομένα και θα καθιστούσε αναξιόπιστη τη συσχέτιση των ενεργειών με τον σωστό φοιτητή.

Η εφαρμογή των παραπάνω ελέγχων ποιότητας δεν ανέδειξε ασυνεπείς εγγραφές στο σύνολο των 686.275 γραμμών που αντιστοιχούν σε δραστηριότητα φοιτητών. Το αποτέλεσμα αυτό επιβεβαιώνει ότι το συνθετικό σύνολο δεδομένων που δημιουργήθηκε σύμφωνα με τις προδιαγραφές της ενότητας 4.1.2 παρουσιάζει υψηλό βαθμό συνέπειας των δεδομένων. Τα αποτελέσματα των επιμέρους σταδίων συνοψίζονται στον παρακάτω πίνακα (Πίνακας 4-4).

Στάδιο	Αριθμός γραμμών
Αρχικό σύνολο δεδομένων (logfile)	786.833
Μετά από φιλτράρισμα Origin = "web"	780.158
Μετά από φιλτράρισμα φοιτητών (μοτίβο "userXXX")	686.275
Μετά από ποιοτικούς ελέγχους (τελικό σύνολο)	686.275

Πίνακας 4-4 Αποτελέσματα σταδίων φιλτραρίσματος δεδομένων

Λεπτομέρειες υλοποίησης παρουσιάζονται στο Παράρτημα Α.

4.2 Εξαγωγή χαρακτηριστικών και δημιουργία πολυδιάστατου προφίλ

Μετά την ολοκλήρωση του φιλτραρίσματος και των ποιοτικών ελέγχων, το σύνολο δεδομένων είναι έτοιμο για το επόμενο στάδιο: την εξαγωγή χαρακτηριστικών (feature extraction) τα οποία θα αποτυπώσουν ποσοτικά τη μαθησιακή συμπεριφορά κάθε φοιτητή.

4.2.1 Δείκτες συμμετοχής (Participation Features)

Οι δείκτες συμμετοχής αποτυπώνουν την ποσότητα και την ποικιλία της αλληλεπίδρασης κάθε φοιτητή με το μαθησιακό περιβάλλον. Για τον υπολογισμό τους αναπτύχθηκε αλγόριθμος σε Python, ο οποίος αξιοποιεί τα φιλτραρισμένα δεδομένα και κατηγοριοποιεί κάθε εγγραφή βάσει των πεδίων *Event name* και *Component* εξάγοντας 10 χαρακτηριστικά συμμετοχής ανά φοιτητή:

- **Forum posts (forum_posts):** Ενέργειες δημιουργίας νέου θέματος ή απάντησης σε υπάρχον. Αντιστοιχεί σε ενεργητική κοινωνική αλληλεπίδραση και παραγωγή περιεχομένου.
- **Forum views (forum_views):** Προβολές θεμάτων ή μηνυμάτων στο forum. Αντιστοιχεί σε παθητική κοινωνική αλληλεπίδραση και κατανάλωση περιεχομένου.
- **Quiz attempts (quiz_attempts):** Προσπάθειες συμμετοχής σε κουίζ (έναρξη ή υποβολή). Αποτυπώνει την εξάσκηση και αυτοαξιολόγηση.
- **Assignment submissions (assignment_submissions):** Υποβολές εργασιών. Αποτελεί δείκτη ολοκλήρωσης υποχρεώσεων.
- **Resource views (resource_views):** Προβολές εκπαιδευτικού υλικού (αρχεία, φάκελοι, ιστοσελίδες, κ.ά.). Αντιστοιχεί σε μελέτη υλικού.
- **Course views (course_views):** Επισκέψεις στην κεντρική σελίδα του μαθήματος. Αποτελεί γενικό δείκτη παρουσίας και πλοήγησης.
- **Συνολικές ενέργειες (total_events):** Γενικός δείκτης κινητικότητας.
- **Ενεργές ημέρες (active_days):** Αριθμός μοναδικών ημερομηνιών με τουλάχιστον μία καταγεγραμμένη ενέργεια. Αποτυπώνει τη συνέπεια και την κατανεμημένη ενασχόληση.

- **Μέσος όρος ενεργειών ανά ενεργή ημέρα (avg_events_per_active_day):** Δείκτης έντασης δραστηριότητας, που σε συνδυασμό με τις ενεργές ημέρες διαφοροποιεί τη «σταθερή μελέτη» από την «πυροσβεστική» ενασχόληση.
- **Αναλογία posts προς views (forum_engagement_ratio):** Δείκτης ενεργητικότητας στο forum, με υψηλές τιμές να υποδηλώνουν φοιτητές που περισσότερο γράφουν παρά διαβάζουν.

Οι δείκτες αυτοί δεν αποτυπώνουν μόνο την ποσότητα δραστηριότητας των φοιτητών, αλλά επιτρέπουν και τη διάκριση διαφορετικών προτύπων συμμετοχής, όπως φοιτητές που αλληλεπιδρούν ενεργά στο forum, φοιτητές που επικεντρώνονται κυρίως στην πρόσβαση στο εκπαιδευτικό υλικό ή φοιτητές που παρουσιάζουν εντατική αλλά περιορισμένης διάρκειας δραστηριότητα.

Ο αλγόριθμος υλοποιήθηκε σε Python με χρήση αποδοτικών τεχνικών επεξεργασίας δεδομένων που επιτρέπουν την ταχεία επεξεργασία του μεγάλου όγκου εγγραφών (686.275 events). Παράλληλα, ενσωματώθηκαν έλεγχοι ποιότητας που παράγουν ξεχωριστό αρχείο αναφοράς με στατιστικά στοιχεία και εντοπισμό ακραίων ή δυνητικά προβληματικών περιπτώσεων (π.χ., φοιτητές με υποβολές εργασιών αλλά καμία προσπάθεια σε κούιζ).

Το τελικό προϊόν της διαδικασίας είναι ένα αρχείο, το οποίο περιέχει για καθέναν από τους 382 φοιτητές το αναγνωριστικό χρήστη (user ID) και τις 10 μετρικές συμμετοχής. Το σύνολο δεδομένων αυτό παρέχει τα χαρακτηριστικά που συγκροτούν την πρώτη διάσταση του πολυδιάστατου προφίλ.

Λεπτομέρειες υλοποίησης παρουσιάζονται στο Παράρτημα Α.

4.2.2 Δείκτες ακαδημαϊκής επίδοσης (Academic Performance Features)

Για την εξαγωγή των δεικτών επίδοσης αξιοποιήθηκε ένα αρχείο με συνθετικό dataset το οποίο περιέχει τις βαθμολογίες για σύνολο 383 φοιτητών. Η δομή του αρχείου περιλαμβάνει έξι στήλες βαθμολογίας: πέντε εργασίες (*WAI* έως *WA5*) και την τελική εξέταση (*Final Exam*). Οι βαθμοί κυμαίνονται στην κλίμακα 0-10, με την τιμή 0 να υποδηλώνει είτε μη παράδοση της εργασίας είτε μηδενική βαθμολογία.

Οι δείκτες που υπολογίζονται ανά φοιτητή είναι οι εξής:

- **final_exam:** Ο βαθμός της τελικής εξέτασης. Αποτελεί βασικό δείκτη της συνολικής κατάκτησης της γνώσης στο τέλος της εκπαιδευτικής διαδικασίας.
- **completed_assignments:** Ο αριθμός των εργασιών που παραδόθηκαν (βαθμός μεγαλύτερος του μηδενός). Η τιμή κυμαίνεται από 0 έως 5 και αποτυπώνει τη συνέπεια και οργάνωση του φοιτητή ως προς την ολοκλήρωση των υποχρεώσεων του.
- **avg_assignments_nonzero:** Ο μέσος όρος βαθμολογίας μόνο για τις εργασίες που παραδόθηκαν. Ο δείκτης αυτός διαχωρίζει την ποιότητα της εργασίας από την απλή παράδοση, δίνοντας μια καθαρή εικόνα της επίδοσης του φοιτητή όταν συμμετέχει ενεργά.

Ιδιαίτερη προσοχή δόθηκε στον χειρισμό των μηδενικών τιμών. Η απλή χρήση του μέσου όρου όλων των εργασιών (συμπεριλαμβανομένων των μηδενικών) θα μπορούσε να οδηγήσει σε παραπλανητικά συμπεράσματα, τιμωρώντας ουσιαστικά τους φοιτητές που δεν παρέδωσαν κάποια εργασία, αλλά ενδέχεται να είχαν υψηλή επίδοση σε όσες παρέδωσαν. Για τον λόγο αυτό, υιοθετήθηκε η διάκριση μεταξύ:

- Ποσοτικού δείκτη συμμετοχής στις εργασίες (*completed_assignments*): πόσες εργασίες παραδόθηκαν
- Ποιοτικού δείκτη επίδοσης στις εργασίες (*avg_assignments_nonzero*): πόσο καλή ήταν η επίδοση στις παραδοθείσες

Αναφορικά με την τελική εξέταση, η τιμή 0 αντιμετωπίζεται ως πραγματική βαθμολογία ανεξάρτητα από την αιτία της — είτε αντικατοπτρίζει παρουσία με μηδενική επίδοση είτε απουσία λόγω αδυναμίας ή αδιαφορίας. Από παιδαγωγική σκοπιά, και στις δύο περιπτώσεις το αποτέλεσμα είναι ισοδύναμο: ο φοιτητής δεν απέδειξε ότι κατέκτησε τη γνώση που αξιολογείται, και αυτό αποτυπώνεται αυτούσιο στο προφίλ του.

Παράλληλα, πραγματοποιήθηκε έλεγχος συμβατότητας των αναγνωριστικών χρηστών (user IDs) μεταξύ του αρχείου βαθμολογιών και του αρχείου δεικτών συμμετοχής που υπολογίστηκε στο προηγούμενο στάδιο. Από τον έλεγχο συμβατότητας διαπιστώθηκε ότι το αρχείο βαθμολογιών περιείχε 383 μοναδικά user IDs, ενώ το φιλτραρισμένο σύνολο δεδομένων συμμετοχής περιείχε 382. Οι 382 φοιτητές ήταν κοινοί στα δύο αρχεία, ενώ ένας επιπλέον φοιτητής εμφανιζόταν μόνο στο αρχείο βαθμολογιών. Ο φοιτητής αυτός διατηρήθηκε κατά τη σύνθεση του τελικού προφίλ, με μηδενικές τιμές στα χαρακτηριστικά που εξάγονται από το αρχείο καταγραφής.

Το τελικό προϊόν της διαδικασίας είναι ένα αρχείο, το οποίο περιέχει για κάθε φοιτητή τις τρεις μετρικές επίδοσης. Το σύνολο των δεδομένων αποτελεί τη δεύτερη διάσταση του πολυδιάστατου προφίλ.

4.2.3 Ανάλυση κοινωνικών δικτύων (Social Network Analysis)

Στην παρούσα ενότητα, στόχος είναι η ανάλυση της κοινωνικής δομής που διαμορφώνεται μέσω της συμμετοχής των φοιτητών στα forums του μαθήματος. Η ανάλυση αυτή θα επιτρέψει την εξαγωγή μετρικών που ποσοτικοποιούν τον ρόλο κάθε φοιτητή στο δίκτυο και θα αποτελέσουν τη βάση για την αντιστοίχιση σε κοινωνικούς ρόλους εντός της μαθησιακής κοινότητας.

4.2.3.1 Κατασκευή διμερούς γράφου (Bipartite Graph Construction)

Για τη μοντελοποίηση της αλληλεπίδρασης των φοιτητών με το περιεχόμενο του forum, επιλέχθηκε η προσέγγιση του διμερούς γράφου (bipartite graph). Σε έναν διμερή γράφο, οι κόμβοι ανήκουν σε δύο διακριτά σύνολα και οι ακμές συνδέουν πάντα κόμβους από διαφορετικά σύνολα. Στην περίπτωση της παρούσας ανάλυσης:

- Το πρώτο σύνολο κόμβων αποτελείται από τους φοιτητές (source)
- Το δεύτερο σύνολο κόμβων αποτελείται από τα νήματα συζητήσεων (threads) (target)

Μια ακμή δημιουργείται όταν ένας φοιτητής συνεισφέρει ενεργητικά σε ένα thread, δηλαδή όταν δημοσιεύει περιεχόμενο ή δημιουργεί νέο θέμα συζήτησης. Στα αρχεία καταγραφής του Moodle οι ενέργειες αυτές εμφανίζονται ως *post created*, *discussion created* και *some content has been posted*.

Για την εξαγωγή των ακμών του διμερούς γράφου από τα δεδομένα καταγραφής, αναπτύχθηκε εξειδικευμένο script σε Python. Η διαδικασία που ακολουθήθηκε περιλαμβάνει τα ακόλουθα στάδια:

Φιλτράρισμα ενεργητικής συμμετοχής

Από το σύνολο των 686.275 καταγεγραμμένων ενεργειών των φοιτητών, οι 151.178 αφορούσαν το component του forum. Από αυτές, διατηρήθηκαν αποκλειστικά εκείνες που αντιστοιχούν σε ενεργητική συνεισφορά στο forum, δηλαδή 1.107 εγγραφές. Ως ενεργητική συνεισφορά ορίστηκαν οι ακόλουθες ενέργειες:

- post created: δημιουργία νέας απάντησης σε υπάρχον thread (372 εγγραφές)
- discussion created: δημιουργία νέου θέματος συζήτησης (137 εγγραφές)
- some content has been posted: γενική περιγραφή που αντιστοιχεί σε δημοσίευση περιεχομένου (598 εγγραφές)

Αντιθέτως, αποκλείστηκαν ενέργειες όπως:

- views: απλή προβολή περιεχομένου (παθητική συμπεριφορά)
- subscriptions: εγγραφή για ειδοποιήσεις
- updates: επεξεργασία υπάρχουσας δημοσίευσης
- deletes: διαγραφή δημοσίευσης

Οι απλές προβολές αποκλείστηκαν από τη δικτυακή ανάλυση, επειδή καταγράφουν πρόσβαση στο περιεχόμενο και όχι ενεργητική συνεισφορά σε συγκεκριμένο νήμα συζήτησης.

Εξαγωγή source και target από το πεδίο description

Το πεδίο *Description* των αρχείων καταγραφής περιέχει μια περιγραφική πρόταση που αναλύει το γεγονός, π.χ.:

"The user with id 'user107' has created post with id '12345' in discussion '67890'"

Για την εξαγωγή των απαραίτητων αναγνωριστικών, χρησιμοποιήθηκαν κανονικές εκφράσεις (regular expressions):

- USER_RE: εντοπίζει το αναγνωριστικό του χρήστη (user with id 'user107')
- DISCUSSION_RE: εντοπίζει το αναγνωριστικό του thread (discussion '67890')
- POST_ID_RE: εντοπίζει το αναγνωριστικό της συγκεκριμένης δημοσίευσης (χρησιμοποιείται για αποφυγή διπλοτύπων)

Αποφυγή διπλοτύπων (Deduplication)

Κατά την επεξεργασία των δεδομένων, διαπιστώθηκε ότι η ίδια ενεργητική συνεισφορά μπορεί να καταγραφεί με διαφορετικές περιγραφές ή να εμφανίζεται σε πολλαπλές εγγραφές. Για παράδειγμα, ένα post μπορεί να καταγραφεί τόσο ως "post created" όσο και ως "some content has been posted". Για την αποφυγή υπερεκτίμησης της δραστηριότητας, εφαρμόστηκε μηχανισμός deduplication με βάση:

- Για δημοσιεύσεις (posts): το ζεύγος (*user_id*, *post_id*) εξασφαλίζει ότι κάθε post μετράται μία φορά.
- Για νέα θέματα (discussions): το ζεύγος (*user_id*, *discussion_id*) εξασφαλίζει ότι κάθε νέο θέμα μετράται μία φορά.

Χάρη στον μηχανισμό αυτό αποφεύχθηκαν 461 διπλότυπα που θα οδηγούσαν σε υπερεκτίμηση της δραστηριότητας των χρηστών, ενώ δεν καταγράφηκε καμία αποτυχία ανάλυσης (parse fail), επιβεβαιώνοντας την ποιότητα των δεδομένων και την ορθότητα των κανονικών εκφράσεων.

Υπολογισμός βαρών

Για κάθε ζεύγος (φοιτητής, thread), καταγράφονται τρία διακριτά βάρη, ανάλογα με τον τύπο της συνεισφοράς:

- *W_post_created*: αριθμός απαντήσεων που δημιούργησε ο φοιτητής στο thread
- *W_discussion_created*: 1 αν ο φοιτητής δημιούργησε το thread (αλλιώς 0)
- *W_some_content_posted*: αριθμός δημοσιεύσεων που καταγράφηκαν με το event "some content has been posted"
- *Weight_active*: συνολικό βάρος ενεργητικής συμμετοχής. Το συνολικό βάρος αποτελεί μια συνοπτική μέτρηση της έντασης της αλληλεπίδρασης του φοιτητή με το συγκεκριμένο thread.

4.2.3.2 Αποτελέσματα και ποιοτικός έλεγχος

Η εφαρμογή του αλγορίθμου παρήγαγε τα αποτελέσματα που συνοψίζονται στον Πίνακα 4-5.

Μέγεθος	Τιμή
Σύνολο εγγραφών στο φιλτραρισμένο αρχείο	686.275

Εγγραφές forum	151.178
Εγγραφές ενεργητικής συμμετοχής που εντοπίστηκαν	1.107
Επιτυχής ανάλυση (parsed OK)	646
Αποτυχίες ανάλυσης (parse fail)	0
Διπλότυπα που αποφεύχθηκαν (dedup skips)	461
Τελικές ακμές διμερούς γράφου	364
Μοναδικοί φοιτητές	135
Μοναδικά threads	193

Πίνακας 4-5 Στατιστικά στοιχεία επεξεργασίας δεδομένων forum

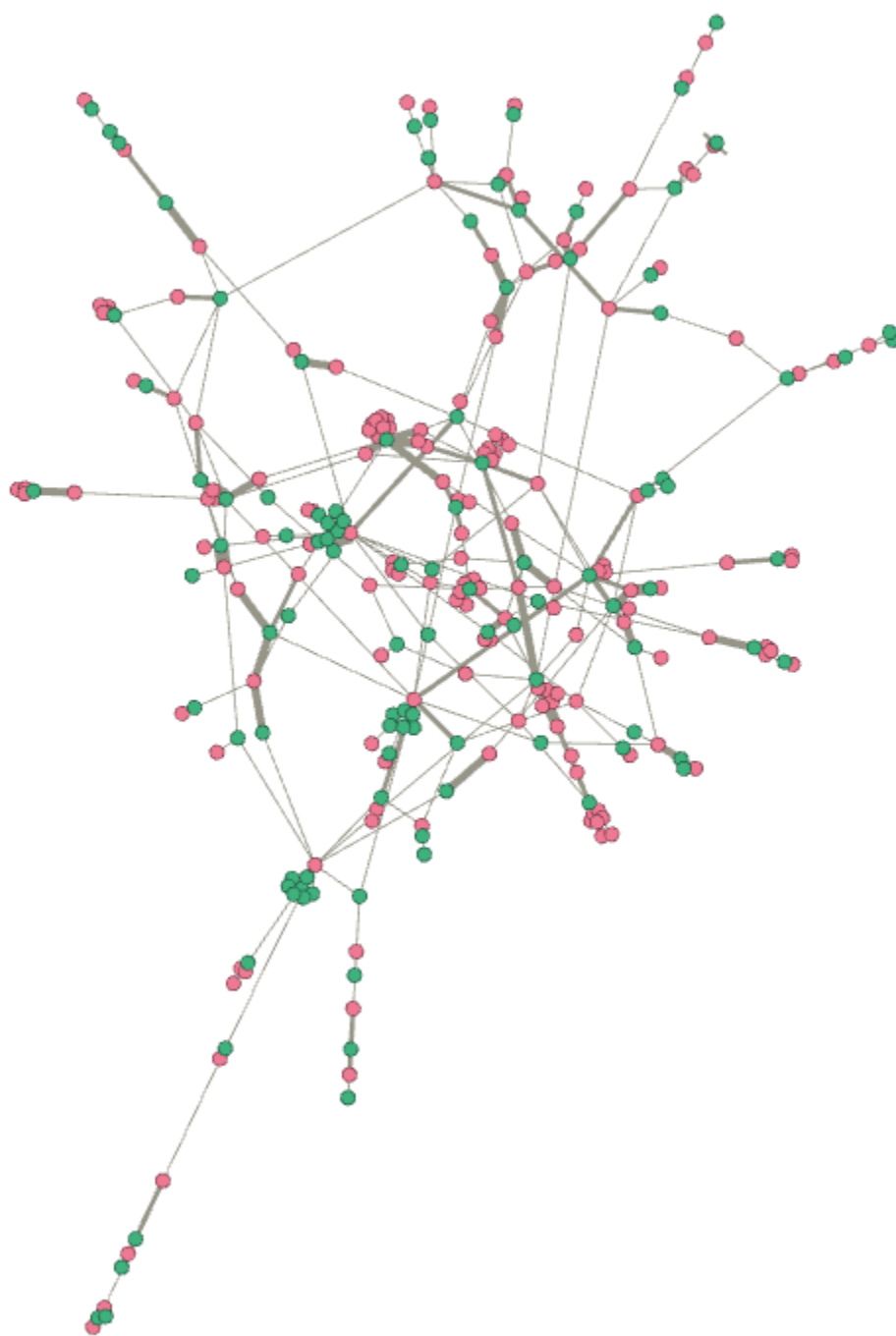
Στο πλαίσιο της ανάλυσης, προβλέφθηκε η δημιουργία του αρχείου unparsed_active_forum_samples.csv, το οποίο σχεδιάστηκε για την καταγραφή δειγμάτων εγγραφών που δεν ήταν δυνατό να αναλυθούν (π.χ., λόγω αποτυχίας των κανονικών εκφράσεων να εξάγουν τα απαραίτητα αναγνωριστικά). Ο μηχανισμός αυτός αποτελεί σημαντικό εργαλείο ποιοτικού ελέγχου, επιτρέποντας τη διερεύνηση τυχόν απρόβλεπτων μορφών δεδομένων και τη συνεχή βελτίωση της διαδικασίας εξαγωγής.

Κατά την εκτέλεση του αλγορίθμου, δεν εντοπίστηκε καμία εγγραφή που να απέτυχε στην ανάλυση (parse fail), όπως φαίνεται και από τα στατιστικά αποτελέσματα. Το γεγονός αυτό επιβεβαιώνει:

- Την υψηλή ποιότητα και συνέπεια των συνθετικών δεδομένων
- Την ορθότητα των κανονικών εκφράσεων που χρησιμοποιήθηκαν
- Την αποτελεσματικότητα προηγούμενης φάσης φιλτραρίσματος (4.1.3)

Παρόλο που το αρχείο δεν δημιουργήθηκε λόγω μηδενικών αποτυχιών, η ύπαρξή του ως προβλεπόμενου μηχανισμού στον κώδικα τεκμηριώνει τη μεθοδολογική προσήλωση στην ποιότητα και την πληρότητα της ανάλυσης.

Το δίκτυο αλληλεπιδράσεων που προέκυψε απεικονίζεται στο Σχήμα 4-2. Οι κόμβοι φοιτητές αποτυπώνονται με πράσινο χρώμα, ενώ οι κόμβοι συζητήσεις αποτυπώνονται με κόκκινο χρώμα. Η οπτικοποίηση έγινε με τη χρήση του λογισμικού Gephi όπου δημιουργήθηκε διμερές, μη-κατευθυνόμενο και σταθμισμένο δίκτυο.



Σχήμα 4-2 Δίκτυο αλληλεπιδράσεων στο forum σε διμερές μη-κατευθυνόμενο γράφο

Λεπτομέρειες υλοποίησης παρουσιάζονται στο Παράρτημα Α.

4.2.3.3 Προβολή σε μονότροπο δίκτυο φοιτητών (One-mode Projection)

Ο αρχικός διμερής γράφος φοιτητών-συζητήσεων παρουσιάζει περιορισμούς ως προς τον υ-πολογισμό κοινωνικά ερμηνεύσιμων μετρικών κεντρικότητας μεταξύ φοιτητών. Συγκεκριμένα, οι shortest-path διαδικασίες επηρεάζονται έντονα από την παρουσία των κόμβων-συζητήσεων (discussion nodes), καθιστώντας δύσκολη την άμεση αποτίμηση των κοινωνικών σχέσεων μεταξύ ομοειδών κόμβων (Borgatti & Everett, 1997). Για τον λόγο αυτό, πραγματοποιήθηκε one-mode projection του διμερούς γράφου στον χώρο των φοιτητών, οδηγώντας σε ένα μη-κατευθυνόμενο, σταθμισμένο δίκτυο που περιλαμβάνει αποκλειστικά κόμβους-φοιτητές.

Ορισμός του προβαλλόμενου δικτύου:

Έστω $G_b = (U \cup V, E)$ ο αρχικός διμερής γράφος, όπου:

- U : σύνολο φοιτητών, $|U| = 135$
- V : σύνολο νημάτων συζητήσεων (threads), $|V| = 193$
- $E \subseteq U \times V$: ακμές ενεργητικής συμμετοχής, $|E| = 364$

Το προβαλλόμενο δίκτυο $G_p = (U, E_p)$ ορίζεται ως ένα μη-κατευθυνόμενο, σταθμισμένο γράφημα επί των φοιτητών, όπου:

$$E_p = \{(u_i, u_j) \mid \exists v \in V: (u_i, v) \in E \wedge (u_j, v) \in E, i \neq j\}$$

Δύο φοιτητές u_i και u_j συνδέονται εάν έχουν συμμετάσχει ενεργά σε τουλάχιστον ένα κοινό thread. Το βάρος της ακμής w_{ij} ορίζεται ως το άθροισμα των *Weight_active* των δύο φοιτητών στο κοινό thread.

Το προβαλλόμενο δίκτυο ορίστηκε ως μη-κατευθυνόμενο, καθώς η μελέτη επικεντρώνεται στην ύπαρξη και ένταση της συνεργατικής αλληλεπίδρασης και όχι στη ροή πληροφορίας μεταξύ αποστολέα και παραλήπτη. Η κοινή συμμετοχή σε discussion thread θεωρήθηκε αμοιβαία κοινωνική σχέση. Η προβολή διμερούς γράφου σε μονότροπο δίκτυο είναι τεκμηριωμένη μέθοδος για την ανάλυση έμμεσων αλληλεπιδράσεων (Borgatti & Everett, 1997).

Το βάρος της ακμής w_{ij} ορίζεται ως:

$$w_{ij} = \sum_{v \in \Gamma(u_i) \cap \Gamma(u_j)} (w(u_i, v) + w(u_j, v))$$

όπου:

- $\Gamma(u)$: το σύνολο των discussions (threads) στα οποία ο φοιτητής u έχει ενεργητική συμμετοχή
- $w(u, v)$: η ενεργητική συνεισφορά του φοιτητή u στο thread v

Η επιλογή του αθροίσματος βασίζεται στη λογική των weighted collaboration networks (Newman, 2004· Opsahl, 2013), όπου η συνολική ένταση της αλληλεπίδρασης μεταξύ δύο φοιτητών αποτυπώνεται από το άθροισμα των επιμέρους συνεισφορών τους στα κοινά threads.

Από την προβολή προέκυψε ένας γράφος με 135 κόμβους και 587 ακμές.

Φιλτράρισμα βαρών

Για την απομάκρυνση χαμηλής έντασης συνεργατικών συνδέσεων που ενδέχεται να αντιστοιχούν σε ευκαιριακές αλληλεπιδράσεις, εφαρμόστηκε φιλτράρισμα βάσει του 75ού εκατοστημορίου της κατανομής των βαρών. Η επιλογή αυτή αποτελεί εμπειρική απόφαση βασισμένη στην κατανομή των δεδομένων: διατηρεί ακμές με επαρκή ένταση αλληλεπίδρασης ($w_{ij} \geq 3,0$) και αποκόπτει συνδέσεις που προκύπτουν από μεμονωμένη συμμετοχή σε ένα μόνο thread, με αποτέλεσμα ποσοστό διατήρησης 50,1% των αρχικών ακμών.

Το φιλτραρισμένο δίκτυο περιλαμβάνει 294 ακμές. Από τους 135 φοιτητές, 28 δεν διατήρησαν ισχυρές συνδέσεις μετά το φιλτράρισμα (degree = 0). Οι φοιτητές αυτοί συμμετείχαν ενεργά στο forum, αλλά είτε αποκλειστικά σε threads χωρίς άλλη ενεργητική συμμετοχή (μοναδικοί συμμετέχοντες), είτε είχαν συνδέσεις με άλλους φοιτητές με βάρος μικρότερο του κατωφλίου ($w_{ij} < 3$). Στο τελικό μαθησιακό προφίλ που παρουσιάζεται στην Ενότητα 4.2.5, οι φοιτητές αυτοί λαμβάνουν μηδενικές τιμές στις βασικές μετρικές SNA (degree, betweenness, closeness), ενώ το PageRank τους λαμβάνει μικρές μη μηδενικές τιμές καθώς ο αλγόριθμος αποδίδει ένα ελάχιστο σκορ σε όλους τους κόμβους ανεξαρτήτως συνδεσιμότητας.

4.2.3.4 Υπολογισμός μετρικών κεντρικότητας

Για την ανάλυση της κοινωνικής δομής του δικτύου αλληλεπίδρασης των φοιτητών στο forum, υπολογίστηκαν τέσσερις βασικές μετρικές κεντρικότητας. Οι μετρικές αυτές βασίζονται στην κλασική προσέγγιση της ανάλυσης κοινωνικών δικτύων (SNA), όπως θεμελιώθηκε από τον Freeman (1979), ενώ συμπληρώνονται από το PageRank ως αλγοριθμική εκτίμηση επιρροής

(Brin & Page, 1998). Η οπτικοποίηση του δικτύου πραγματοποιήθηκε με τη χρήση του λογισμικού Gephi (Bastian et al., 2009), το οποίο χρησιμοποιήθηκε αποκλειστικά για τη γραφική απεικόνιση της δομής του δικτύου. Οι υπολογισμοί των μετρικών κεντρικότητας, αντιθέτως, πραγματοποιήθηκαν αλγοριθμικά στην Python αξιοποιώντας τη βιβλιοθήκη networkx, όπως περιγράφεται αναλυτικά στη συνέχεια.

Για το Betweenness και το Closeness, λόγω της απαίτησης των αλγορίθμων για αποστάσεις, πραγματοποιήθηκε μετατροπή του βάρους σε απόσταση σύμφωνα με τον τύπο $d_{ij} = 1/w_{ij}$ (Opsahl, 2013), εξασφαλίζοντας ότι ισχυρότερες συνεργασίες (μεγαλύτερο w_{ij}) αντιστοιχούν σε μικρότερες αποστάσεις.

Degree Centrality (Κεντρικότητα βαθμού)

Μαθηματικός ορισμός (απλός βαθμός):

Σε ένα μη-κατευθυνόμενο γράφημα $G = (U, E)$, η κεντρικότητα βαθμού ενός κόμβου u_i ορίζεται ως:

$$\deg(u_i) = \sum_{j=1}^n a_{ij}$$

όπου $n = |U|$ και $a_{ij} = 1$ εάν υπάρχει ακμή μεταξύ u_i και u_j , αλλιώς 0.

Η κανονικοποιημένη κεντρικότητα βαθμού (Freeman, 1979) δίνεται από τον τύπο:

$$C_D(u_i) = \frac{\deg(u_i)}{n - 1}$$

Σταθμισμένος βαθμός (Weighted Degree / Strength):

Παράλληλα υπολογίστηκε και ο σταθμισμένος βαθμός, ο οποίος ορίζεται ως το άθροισμα των βαρών των ακμών του κόμβου:

$$s(u_i) = \sum_j w_{ij}$$

Ο δείκτης αυτός αποτυπώνει την ένταση της συνεργασίας και όχι μόνο τον αριθμό των συνεργατών. Δύο φοιτητές με τον ίδιο απλό βαθμό μπορεί να διαφέρουν σημαντικά ως προς τη συνολική ένταση των αλληλεπιδράσεών τους.

Ερμηνεία στο προβαλλόμενο δίκτυο:

Ο απλός βαθμός $\deg(u_i)$ αντιστοιχεί στον αριθμό των άλλων φοιτητών με τους οποίους ο φοιτητής u_i συνεργάζεται (κοινή συμμετοχή σε thread με επαρκή ένταση, $w_{ij} \geq 3$). Αποτελεί μέτρο τοπικής κεντρικότητας και δείκτη κοινωνικής δραστηριότητας. Ο σταθμισμένος βαθμός $s(u_i)$ συμπληρώνει την ανάλυση αποτυπώνοντας τη συνολική ένταση των συνεργασιών.

Betweenness Centrality (Κεντρικότητα διαμεσολάβησης)

Μαθηματικός ορισμός:

Η betweenness centrality ενός κόμβου u_i (Freeman, 1977) ορίζεται ως:

$$C_B(u_i) = \sum_{s \neq i, t \neq i, s \neq t} \frac{\sigma_{st}(u_i)}{\sigma_{st}}$$

όπου:

- σ_{st} : συνολικός αριθμός συντομότερων διαδρομών μεταξύ των κόμβων s και t
- $\sigma_{st}(u_i)$: αριθμός από αυτές τις συντομότερες διαδρομές που διέρχονται από τον u_i

Η κανονικοποιημένη εκδοχή (για μη-κατευθυνόμενα γραφήματα) είναι:

$$C_B^{\text{norm}}(u_i) = \frac{C_B(u_i)}{(n-1)(n-2)/2}$$

Ερμηνεία στο προβαλλόμενο δίκτυο:

Ένας φοιτητής με υψηλή betweenness centrality λειτουργεί ως «γέφυρα» (bridge) μεταξύ διαφορετικών υποομάδων. Υψηλές τιμές υποδεικνύουν ισχυρό ρόλο διαμεσολάβησης, όπου ο φοιτητής ενώνει άλλους φοιτητές που διαφορετικά δεν θα συνδέονταν, διευκολύνοντας τη ροή πληροφορίας και τη διάχυση της γνώσης σε ολόκληρο το δίκτυο.

Closeness Centrality (Κεντρικότητα εγγύτητας)

Μαθηματικός ορισμός:

Η closeness centrality ενός κόμβου u_i (Freeman, 1979) ορίζεται ως το αντίστροφο του αθροίσματος των αποστάσεων προς όλους τους άλλους κόμβους:

$$C_C(u_i) = \frac{1}{\sum_{j \neq i} d(u_i, u_j)}$$

όπου $d(u_i, u_j)$ είναι η απόσταση (minimum path length) μεταξύ u_i και u_j . Στην παρούσα σταθμισμένη υλοποίηση, η απόσταση ορίζεται ως το ελάχιστο άθροισμα αποστάσεων που προκύπτουν από τον μετασχηματισμό $d_{ij} = 1/w_{ij}$.

Η κανονικοποιημένη εκδοχή, που λαμβάνει υπόψη το μέγεθος του δικτύου, είναι:

$$C_C^{\text{norm}}(u_i) = \frac{n-1}{\sum_{j \neq i} d(u_i, u_j)}$$

Ερμηνεία στο προβαλλόμενο δίκτυο:

Η closeness centrality αντανακλά την ταχύτητα διάχυσης πληροφορίας από έναν φοιτητή προς ολόκληρη την κοινότητα. Φοιτητές με υψηλή τιμή βρίσκονται σε στρατηγική θέση, επιτρέποντάς τους να έχουν πρόσβαση στην πληροφορία γρηγορότερα και να μεταδίδουν γνώση αποτελεσματικά.

Σημείωση: Για τους 28 φοιτητές με degree = 0 μετά το φιλτράρισμα, η τιμή closeness είναι μηδενική λόγω απουσίας συνδέσεων. Η ερμηνεία της μετρικής αφορά τους μη απομονωμένους κόμβους (107 φοιτητές) και τα επιμέρους συνδεδεμένα τμήματα του δικτύου.

PageRank (Εκτίμηση επιρροής)

Μαθηματικός ορισμός (weighted PageRank):

Ο PageRank (Brin & Page, 1998) είναι ένας επαναληπτικός αλγόριθμος που αποδίδει σκορ σε κάθε κόμβο βάσει της δομής των συνδέσεων. Στην παρούσα εργασία χρησιμοποιήθηκε αποκλειστικά η σταθμισμένη εκδοχή (weighted PageRank), όπου η κατανομή της επιρροής στα γειτονικά nodes σταθμίστηκε με βάση το βάρος συνεργασίας w_{ij} :

$$PR(u_i) = \frac{1-p}{n} + p \sum_{u_j \in \mathcal{N}(u_i)} \frac{w_{ji} \cdot PR(u_j)}{\sum_{u_k \in \mathcal{N}(u_j)} w_{jk}}$$

όπου:

- p : παράγοντας αποκατάστασης (damping factor), με $p = 0,85$
- $\mathcal{N}(u_i)$: σύνολο γειτόνων του u_i
- w_{ij} : βάρος της ακμής μεταξύ u_i και u_j

Ερμηνεία στο προβαλλόμενο δίκτυο:

Το PageRank αποτυπώνει την κοινωνική επιρροή κάθε φοιτητή. Ένας φοιτητής αποκτά υψηλό PageRank όταν συνεργάζεται με άλλους φοιτητές που έχουν οι ίδιοι υψηλό Page Rank (αρχή της «μετάδοσης φήμης»), υποδεικνύοντας έναν πιθανό πυρήνα επιδραστικών μελών. Το άθροισμα των τιμών PageRank σε όλους τους κόμβους ισούται με 1. Ακόμη και οι απομονωμένοι κόμβοι (degree = 0) λαμβάνουν μικρές μη μηδενικές τιμές καθώς ο αλγόριθμος αποδίδει ένα ελάχιστο σκορ σε όλους τους κόμβους ανεξαρτήτως συνδεσιμότητας (ο παράγοντας $\frac{1-p}{n}$ εξασφαλίζει ότι κανένας κόμβος δεν λαμβάνει μηδενική τιμή).

Μέγεθος	Τιμή
Φοιτητές με ενεργή συμμετοχή ($\ U \ $)	135
Μοναδικά threads ($\ V \ $)	193
Ακμές διμερούς γράφου ($\ E \ $)	364
Ακμές προβαλλόμενου δικτύου (full)	587
Κατώφλι φιλτραρίσματος (75ο εκατοστημόριο) θ	3,0
Ακμές μετά το φιλτράρισμα	294
Μη απομονωμένοι φοιτητές μετά το φιλτράρισμα	107
Απομονωμένοι φοιτητές (degree = 0)	28

Πίνακας 4-6 Στατιστικά στοιχεία προβαλλόμενου δικτύου φοιτητών

Τα βασικά στατιστικά στοιχεία του προβαλλόμενου δικτύου συνοψίζονται στον Πίνακα 4-6.

Κατανομή των μετρικών κεντρικότητας

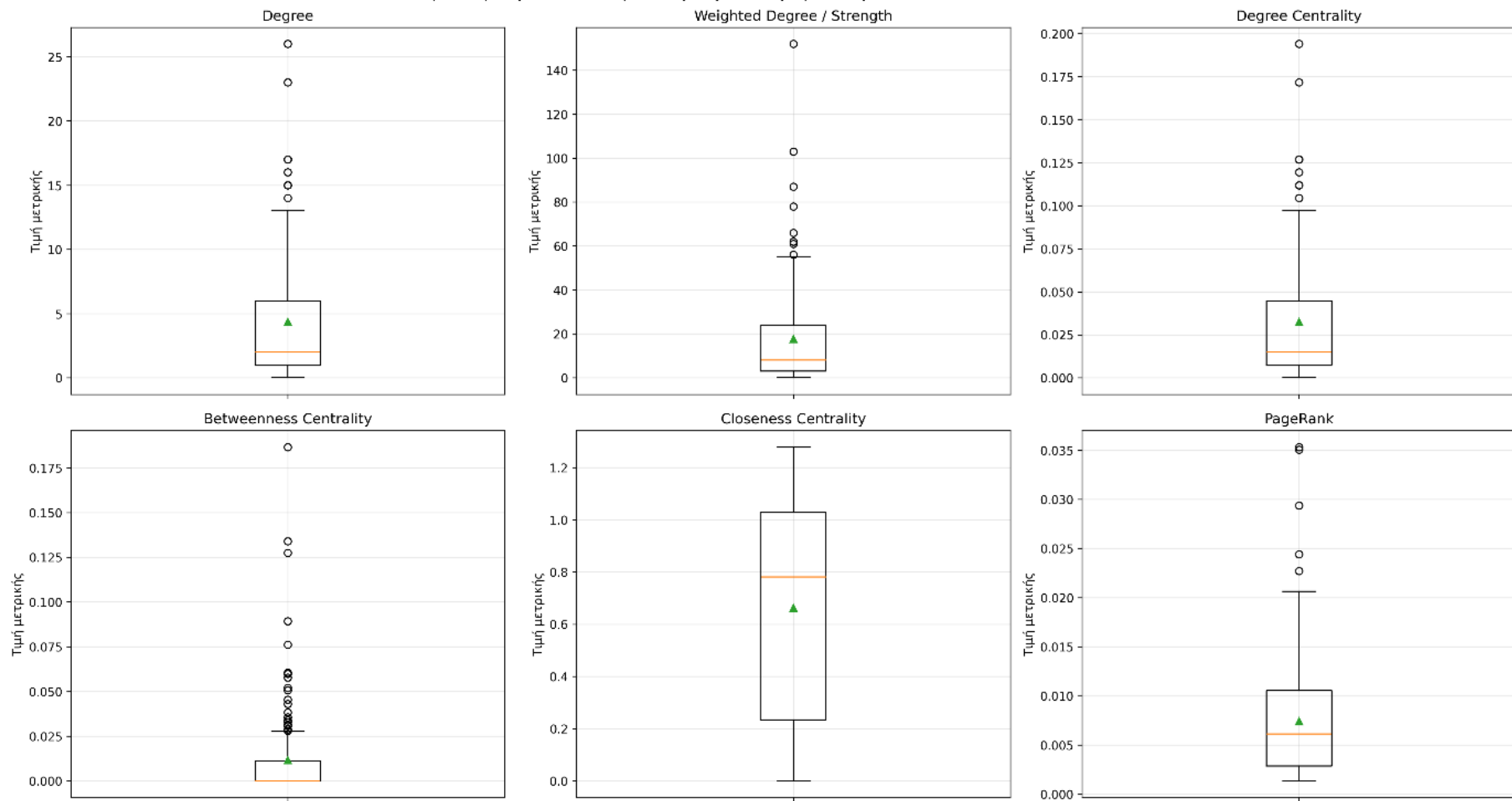
Η ανάλυση της κατανομής των μετρικών κεντρικότητας (που ορίστηκαν στην προηγούμενη υποενότητα) αποκαλύπτει σημαντικές πληροφορίες για τη δυναμική της μαθησιακής κοινότητας.

Παρατηρείται ότι σε όλες τις μετρικές ένας μικρός αριθμός φοιτητών συγκεντρώνει υψηλές τιμές, ενώ η πλειονότητα εμφανίζει χαμηλές έως μέτριες τιμές. Το εύρημα αυτό είναι συνεπές

με προηγούμενες μελέτες σε διαδικτυακές μαθησιακές κοινότητες, όπου συνήθως μια μειοψηφία συμμετεχόντων αναλαμβάνει τον κύριο όγκο της δραστηριότητας και της αλληλεπίδρασης (Bonafini, 2018).

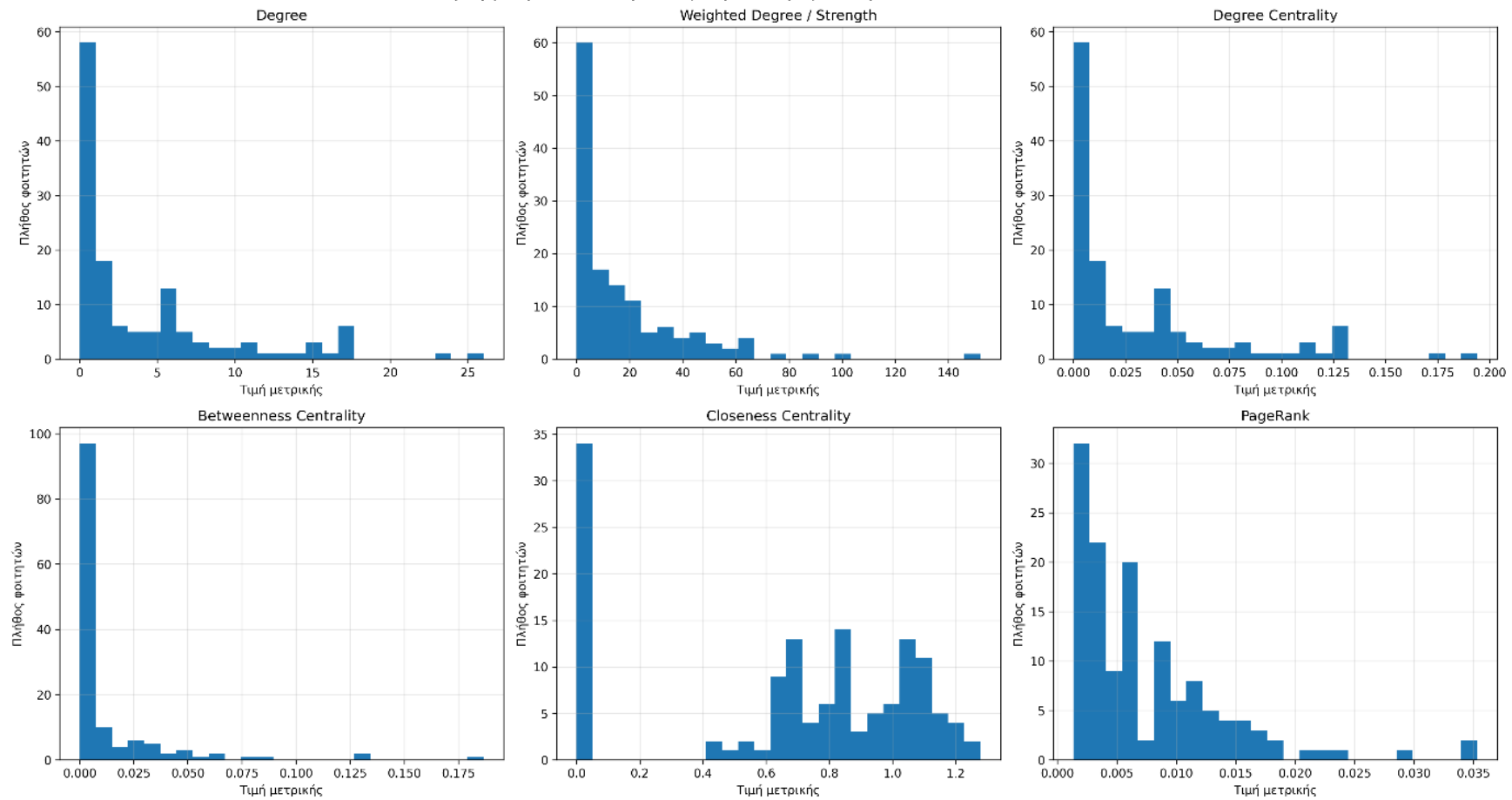
Τα διαγράμματα boxplot (Σχήμα 4-3) και ιστογράμματα (Σχήμα 4-4) των μετρικών κεντρικότητας παρουσιάζονται στη συνέχεια.

Boxplots μετρικών κεντρικότητας στο προβαλλόμενο student-student δίκτυο



Σχήμα 4-3 Boxplots μετρικών κεντρικότητας

Κατανομές μετρικών κεντρικότητας στο προβαλλόμενο student-student δίκτυο



Σχήμα 4-4 Ιστογράμματα μετρικών κεντρικότητας

4.2.3.5 Αντιστοίχιση σε κοινωνικούς ρόλους

Μετά τον υπολογισμό των μετρικών κεντρικότητας στο προβαλλόμενο δίκτυο φοιτητών, ακολούθησε η αντιστοίχιση των φοιτητών σε ποιοτικούς κοινωνικούς ρόλους. Η κατηγοριοποίηση αντλείται από τη βιβλιογραφία των μαθησιακών κοινοτήτων, η οποία εξετάζει διαφορετικές μορφές συμμετοχής, από περισσότερο περιφερειακές έως περισσότερο κεντρικές θέσεις (Wenger, 1998), καθώς και από μελέτες που αξιοποιούν την Ανάλυση Κοινωνικών Δικτύων για τη διερεύνηση προτύπων αλληλεπίδρασης και αναδυόμενων ρόλων σε διαδικτυακά μαθησιακά περιβάλλοντα (de Laat et al., 2007· Rabbany et al., 2014). Στόχος της διαδικασίας είναι η ταξινόμηση των φοιτητών σε διακριτούς κοινωνικούς ρόλους, οι οποίοι θα αξιοποιηθούν στον σχηματισμό ομάδων.

Μεθοδολογική προσέγγιση

Για την κατηγοριοποίηση αναπτύχθηκε αλγόριθμος σε Python ο οποίος αξιοποιεί το σύνολο των μετρικών κεντρικότητας (degree, weighted degree, betweenness, closeness, PageRank). Η λογική ταξινόμησης είναι ιεραρχική (priority-based): η προτεραιότητα δόθηκε αρχικά στους ρόλους υψηλής δομικής σημασίας («Ηγέτης», «Γέφυρα»), και στη συνέχεια στους ρόλους που περιγράφουν το γενικό επίπεδο δραστηριότητας. Η ιεραρχική αυτή διαδικασία εφαρμόστηκε ώστε κάθε φοιτητής να αντιστοιχίζεται σε έναν μόνο κυρίαρχο ρόλο, αποφεύγοντας την πολλαπλή ταξινόμηση.

Η κανονικοποίηση των μετρικών πραγματοποιήθηκε με τη μέθοδο z-score. Για τις μετρικές weighted degree, closeness και PageRank εφαρμόστηκε τυπική z-score κανονικοποίηση σε όλο το δείγμα. Για τη betweenness, λόγω της έντονα ασύμμετρης κατανομής της (πολλές μηδενικές τιμές), ο z-score υπολογίστηκε μόνο στις μη μηδενικές τιμές, αποφεύγοντας τη στατιστική παραμόρφωση που προκαλεί ο μεγάλος αριθμός μηδενικών τιμών.

Κριτήρια αντιστοίχισης ρόλων

Τα κριτήρια αντιστοίχισης των κοινωνικών ρόλων συνοψίζονται στον Πίνακα 4-7

Ρόλος	Κριτήρια	Παιδαγωγική ερμηνεία
-------	----------	----------------------

«Ηγέτης» (Leader)	$\text{weighted_degree_z} > 1,0$ και $\text{pagerank_z} > 1,0$ και $\text{closeness_z} \geq 0$	Υψηλή ένταση συνεργασίας, κοινωνική επιρροή και κεντρική θέση στο δίκτυο
«Γέφυρα» (Bridge)	$\text{betweenness} > 0$ και $\text{betweenness_z} > 0$	Διαμεσολαβητικός ρόλος, συνδέει διαφορετικές υποομάδες
Δραστήριος Συμμετέχων	$\text{weighted_degree_z} > 0,0$	Άνω του μέσου όρου στην ένταση συνεργασίας
Περιφερειακός Συμμετέχων	Κανένα από τα παραπάνω	Χαμηλή έως μέτρια δραστηριότητα, αλλά μη μηδενική
Ανεπαρκής Αλληλεπίδραση	$\text{degree} = 0$	Απομονωμένοι κόμβοι

Πίνακας 4-7 Κριτήρια αντιστοίχισης κοινωνικών ρόλων

Τα κατώφλια (π.χ. $z > 1,0$) δεν επιλέχθηκαν αυθαίρετα αλλά με σκοπό την ανάδειξη φοιτητών που αποκλίνουν ουσιαστικά από τον μέσο όρο της κοινότητας. Η χρήση $z\text{-score} > 1$ αντιστοιχεί σε τιμές άνω της μίας τυπικής απόκλισης από τον μέσο όρο και αποτελεί συντηρητικό κριτήριο εντοπισμού υψηλής κεντρικότητας. Η επιλογή αυτή προτιμήθηκε έναντι ποσοστιαίων (percentile-based) κατωφλίων ώστε η μέθοδος να μπορεί να γενικεύεται σε διαφορετικά σύνολα δεδομένων ανεξαρτήτως μεγέθους και πυκνότητας δικτύου.

Η συνθήκη $\text{closeness_z} \geq 0$ για τον ρόλο του «Ηγέτη» λειτουργεί ως ήπιος περιορισμός συνοχής (αποκλείοντας φοιτητές με αρνητική εγγύτητα, δηλαδή δομικά απομακρυσμένους), χωρίς να αποτελεί πρωτεύον κριτήριο ταξινόμησης. Ο όρος «Ηγέτης» χρησιμοποιείται λειτουργικά για να περιγράψει φοιτητές με υψηλή κοινωνική επιρροή και ένταση αλληλεπίδρασης στο δίκτυο, χωρίς να υπονοεί επίσημο ή παιδαγωγικά αναγνωρισμένο ηγετικό ρόλο.

Αποτελέσματα

Εφαρμόζοντας τα παραπάνω κριτήρια στο σύνολο των 135 φοιτητών, προέκυψε η ακόλουθη κατανομή:

Ρόλος	Πλήθος	Ποσοστό
Ηγέτης	18	13,33%

Γέφυρα	7	5,19%
Δραστήριος Συμμετέχων	28	20,74%
Περιφερειακός Συμμετέχων	54	40,00%
Ανεπαρκής Αλληλεπίδραση	28	20,74%
Σύνολο	135	100%

Πίνακας 4-8 Κατανομή κοινωνικών ρόλων φοιτητών στο προβαλλόμενο δίκτυο

Η κατανομή είναι συνεπής με τη βιβλιογραφία: μια μειοψηφία συμμετεχόντων (Ηγέτες + Γέφυρες, 18,52%) συγκεντρώνει την υψηλότερη δραστηριότητα και επιρροή, ενώ η πλειονότητα εντάσσεται σε κατηγορίες μέτριας ή χαμηλής ενεργητικότητας (Bonafini, 2018· Rabbany et al., 2014).

Διευκρίνιση για την απόδοση SNA τιμών στο σύνολο των φοιτητών

Η ανάλυση κοινωνικού δικτύου και η αντιστοίχιση των ρόλων του Πίνακα 4.8 αφορά αποκλειστικά τους 135 φοιτητές που εμφάνισαν ενεργητική συμμετοχή στο forum και αποτέλεσαν κόμβους του προβαλλόμενου δικτύου (Ενότητα 4.2.3.4). Για τους φοιτητές αυτούς, οι μετρικές κεντρικότητας (degree, betweenness, closeness, PageRank) υπολογίστηκαν αλγοριθμικά βάσει της δομής του δικτύου.

Ωστόσο, στο τελικό πολυδιάστατο μαθησιακό προφίλ (Ενότητα 4.2.5) αποδίδονται τιμές SNA σε όλους τους 383 φοιτητές, ως εξής:

Για τους 107 φοιτητές με μη μηδενικές συνδέσεις στο φιλτραρισμένο δίκτυο, οι τιμές προκύπτουν από την παραπάνω ανάλυση.

Για τους 28 φοιτητές που συμμετείχαν ενεργητικά στο forum, αλλά δεν διατήρησαν ακμές μετά την εφαρμογή του κατωφλίου (degree = 0), οι μετρικές degree, betweenness και closeness τίθενται στο μηδέν, ενώ το PageRank λαμβάνει την ελάχιστη μη μηδενική τιμή που αποδίδει ο αλγόριθμος στους απομονωμένους κόμβους του δικτύου.

Για τους 248 φοιτητές που δεν εμφάνισαν καμία ενεργητική συμμετοχή στο forum, οι SNA μετρικές δεν υπολογίζονται αλγοριθμικά, καθώς οι φοιτητές αυτοί δεν αποτελούν κόμβους του προβαλλόμενου δικτύου. Κατά τη σύνθεση του τελικού προφίλ, οι αντίστοιχες τιμές τίθενται κατασκευαστικά στο μηδέν.

Και για τις δύο τελευταίες κατηγορίες, ο κοινωνικός ρόλος ορίζεται ως «Ανεπαρκής Αλληλεπίδραση». Η απόδοση αυτή γίνεται για λόγους πληρότητας και ομοιογένειας του τελικού προφίλ και δεν προϋποθέτει την ύπαρξη του φοιτητή ως κόμβου στο προβαλλόμενο δίκτυο. Με τον τρόπο αυτό, το Κεφάλαιο 5 μπορεί να αξιοποιήσει το σύνολο των 383 φοιτητών στη διαδικασία ομαδοποίησης, χωρίς αποκλεισμούς λόγω απουσίας SNA δεδομένων.

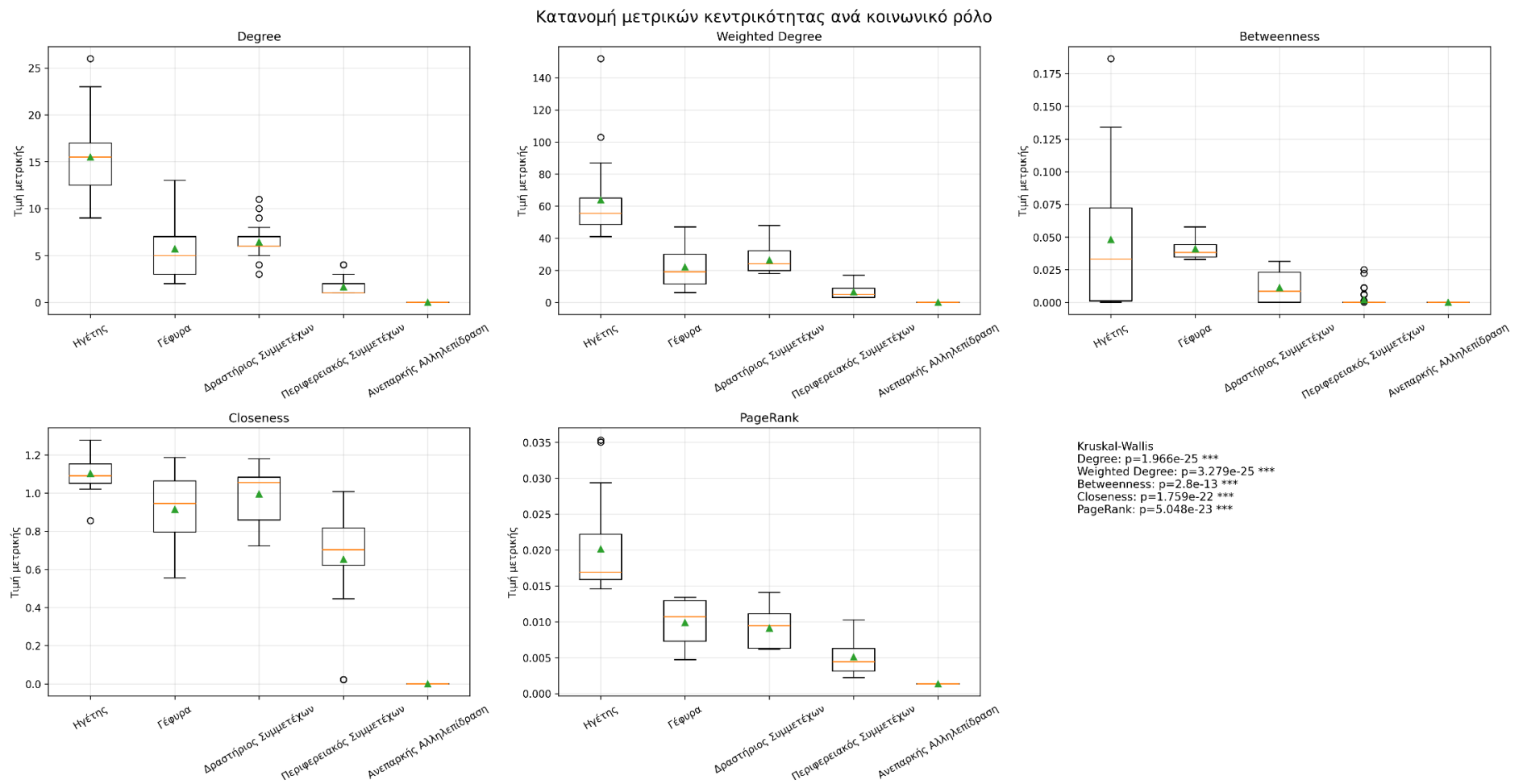
Μη ενεργοί συμμετέχοντες

Η απουσία έντονης αλληλεπίδρασης στο Moodle δεν συνεπάγεται απαραίτητα έλλειψη συνεργασίας. Μέρος της επικοινωνίας των φοιτητών είναι πιθανό να πραγματοποιείται μέσω εξωτερικών καναλιών (π.χ. κοινωνικά δίκτυα, εφαρμογές ανταλλαγής μηνυμάτων, πλατφόρμες τηλεδιάσκεψης), τα οποία δεν καταγράφονται από την πλατφόρμα. Συνεπώς, οι μετρικές κοινωνικής αλληλεπίδρασης αντανακλούν αποκλειστικά την ορατή δραστηριότητα εντός του LMS.

Παρατήρηση: Η μοναδική θέση των Γεφυρών

Αξίζει να σημειωθεί ότι τρεις από τους επτά φοιτητές που χαρακτηρίστηκαν ως Γέφυρες παρουσιάζουν αρνητικό z-score στον σταθμισμένο βαθμό ($sna_weighted_degree_z < 0$), δηλαδή η συνολική ένταση συνεργασίας τους είναι κάτω του μέσου όρου. Παρά τη χαμηλή τους δραστηριότητα, η θετική τους betweenness centrality αποκαλύπτει ότι κατέχουν στρατηγική θέση στο δίκτυο. Η περίπτωση αυτή καταδεικνύει ότι η betweenness προσθέτει μη πλεονάζουσα πληροφορία σε σχέση με τις απλές μετρικές δραστηριότητας: ένας φοιτητής μπορεί να είναι «κοινωνικά κρίσιμος» χωρίς να είναι υπερδραστήριος, όταν η θέση του λειτουργεί ως συνδετικός κρίκος μεταξύ διαφορετικών τμημάτων του δικτύου (Burt, 1992· Borgatti, 2005).

Η κατανομή των μετρικών κεντρικότητας ανά κοινωνικό ρόλο παρουσιάζεται στο Σχήμα 4-5.



Σχήμα 4-5 Κατανομή των μετρικών κεντρικότητας ανά κοινωνικό ρόλο

Στατιστική επικύρωση της κατηγοριοποίησης σε κοινωνικούς ρόλους

Για να διαπιστωθεί εάν οι πέντε ρόλοι αντιστοιχούν σε διακριτές ομάδες φοιτητών, διενεργήθηκε στατιστική ανάλυση. Δεδομένου ότι οι τιμές των μετρικών κεντρικότητας δεν ακολουθούν κανονική κατανομή, επιλέχθηκε ο μη παραμετρικός έλεγχος Kruskal-Wallis. Ως όριο στατιστικής σημαντικότητας ορίστηκε $\alpha = 0,05$.

Τα αποτελέσματα έδειξαν στατιστικά σημαντικές διαφορές μεταξύ των πέντε ρόλων για όλες τις μετρικές:

Μετρική	Τιμή H	p-value
Βαθμός (Degree)	122,03	< 0,001
Σταθμισμένος βαθμός	120,99	< 0,001
Διαμεσολάβηση (Betweenness)	64,83	< 0,001
Εγγύτητα (Closeness)	108,20	< 0,001
Επιρροή (PageRank)	110,75	< 0,001

Πίνακας 4-9 Αποτελέσματα ελέγχου Kruskal-Wallis

Τα αποτελέσματα του ελέγχου, τα οποία συνοψίζονται στον Πίνακα 4-9, έδειξαν ότι οι υψηλές τιμές του στατιστικού H (όπου μεγαλύτερο H σημαίνει μεγαλύτερη απόκλιση μεταξύ των ομάδων) υποδηλώνουν έντονες διαφοροποιήσεις μεταξύ των ρόλων. Ακόμη και για τη betweenness, όπου οι απόλυτες τιμές είναι μικρές, η στατιστικά σημαντική διαφορά ($p < 0,001$, δηλαδή πιθανότητα μικρότερη από 0,1% οι διαφορές να οφείλονται σε τύχη) επιβεβαιώνει ότι οι φοιτητές που χαρακτηρίστηκαν ως Γέφυρες κατέχουν διακριτή θέση στο δίκτυο.

Post-hoc έλεγχος Dunn

Τα αποτελέσματα του Kruskal-Wallis υπέδειξαν την ύπαρξη στατιστικά σημαντικών διαφορών μεταξύ των ρόλων. Για τον προσδιορισμό των συγκεκριμένων ζευγών ρόλων που διαφέρουν σημαντικά, εφαρμόστηκε post-hoc έλεγχος Dunn (Dunn, 1964) με διόρθωση Bonferroni για τον μετριασμό του σφάλματος τύπου I λόγω πολλαπλών συγκρίσεων.

Λόγω του μεγάλου αριθμού συγκρίσεων, παρουσιάζονται εδώ μόνο τα σημαντικότερα ευρήματα:

- Οι συγκρίσεις μεταξύ διακριτών ρόλων με μεγάλη διαφορά (π.χ. Ηγέτης–Περιφερειακός Συμμετέχων) ήταν στατιστικά σημαντικές για όλες τις μετρικές.
- Οι συγκρίσεις μεταξύ Ηγέτη και Γέφυρας εμφάνισαν περιορισμένες διαφοροποιήσεις σε ορισμένες μετρικές, γεγονός που είναι αναμενόμενο, καθώς και οι δύο ρόλοι αντιστοιχούν σε φοιτητές με αυξημένη δομική σημασία, αλλά διαφοροποιούνται ως προς τη φύση της θέσης τους στο δίκτυο.
- Οι συγκρίσεις μεταξύ Γέφυρας και Δραστήριου Συμμετέχοντα **δεν** ήταν στατιστικά σημαντικές για τον απλό βαθμό ($p > 0,05$), γεγονός αναμενόμενο καθώς ο ρόλος της Γέφυρας βασίζεται στην betweenness, όχι στον αριθμό συνεργασιών.

Τα πλήρη αποτελέσματα του ελέγχου Dunn παρατίθενται στο Παράρτημα Β.

Συμπερασματικά, η στατιστική ανάλυση επιβεβαιώνει ότι η κατηγοριοποίηση των φοιτητών στους πέντε κοινωνικούς ρόλους δεν είναι αυθαίρετη, αλλά αποτυπώνει στατιστικά σημαντικές και μετρήσιμες διαφοροποιήσεις ως προς τις εξεταζόμενες μετρικές κεντρικότητας. Η στατιστική επικύρωση αποτελεί στάδιο ελέγχου της παραγόμενης κοινωνικής διάστασης πριν από την ενσωμάτωσή της στο πολυδιάστατο προφίλ. Σε περίπτωση που δεν διαπιστώνονταν σημαντικές διαφοροποιήσεις, η διαδικασία αντιστοίχισης των κοινωνικών ρόλων θα επαναλαμβανόταν με αναθεωρημένα κριτήρια. Με βάση τα αποτελέσματα του ελέγχου, οι κοινωνικοί ρόλοι διατηρούνται στην επόμενη φάση της μεθοδολογικής ροής και μπορούν να αξιοποιηθούν ως στατιστικά τεκμηριωμένη είσοδος στο σύστημα συστάσεων για τον σχηματισμό ομάδων (Κεφάλαιο 5).

Οπτικοποίηση στο Gephi

Για την οπτικοποίηση του προβαλλόμενου δικτύου, τα δεδομένα εξήχθησαν σε μορφή κατάλληλη για το λογισμικό Gephi (Bastian et al., 2009). Συγκεκριμένα, δημιουργήθηκαν δύο αρχεία: ένα αρχείο κόμβων που περιλαμβάνει το σύνολο των 135 φοιτητών με ενεργητική συμμετοχή στο forum, συνοδευόμενο από τις μετρικές κεντρικότητας (*sna_degree*, *sna_weighted_degree*, *sna_betweenness*, *sna_closeness*, *sna_pagerank*) και τον κοινωνικό ρόλο (*network_role_el*), και ένα αρχείο ακμών που περιλαμβάνει τις 294 συνδέσεις του φιλτραρισμένου δικτύου με το αντίστοιχο βάρος (*Weight*) (Σχήμα 4-6). Η διαδικασία αυτή διασφάλισε ότι τόσο οι συνδεδεμένοι όσο και οι 28 απομονωμένοι κόμβοι ($\text{degree} = 0$) συμπεριλήφθηκαν στην οπτικοποίηση, ώστε να αποτυπωθεί τόσο η παρουσία όσο και η απουσία συνδέσεων στο δίκτυο.

Η διάταξη των κόμβων πραγματοποιήθηκε με τον αλγόριθμο ForceAtlas2 (Jacomy et al., 2014), ο οποίος, μέσω δυνάμεων έλξης και απώθησης, τείνει να τοποθετεί εγγύτερα τους ισχυρότερα συνδεδεμένους κόμβους, αναδεικνύοντας τη συνολική δομή των σχέσεων. Οι οπτικές μεταβλητές του σχήματος αντιστοιχούν σε διαφορετικές διαστάσεις της κοινωνικής ανάλυσης:

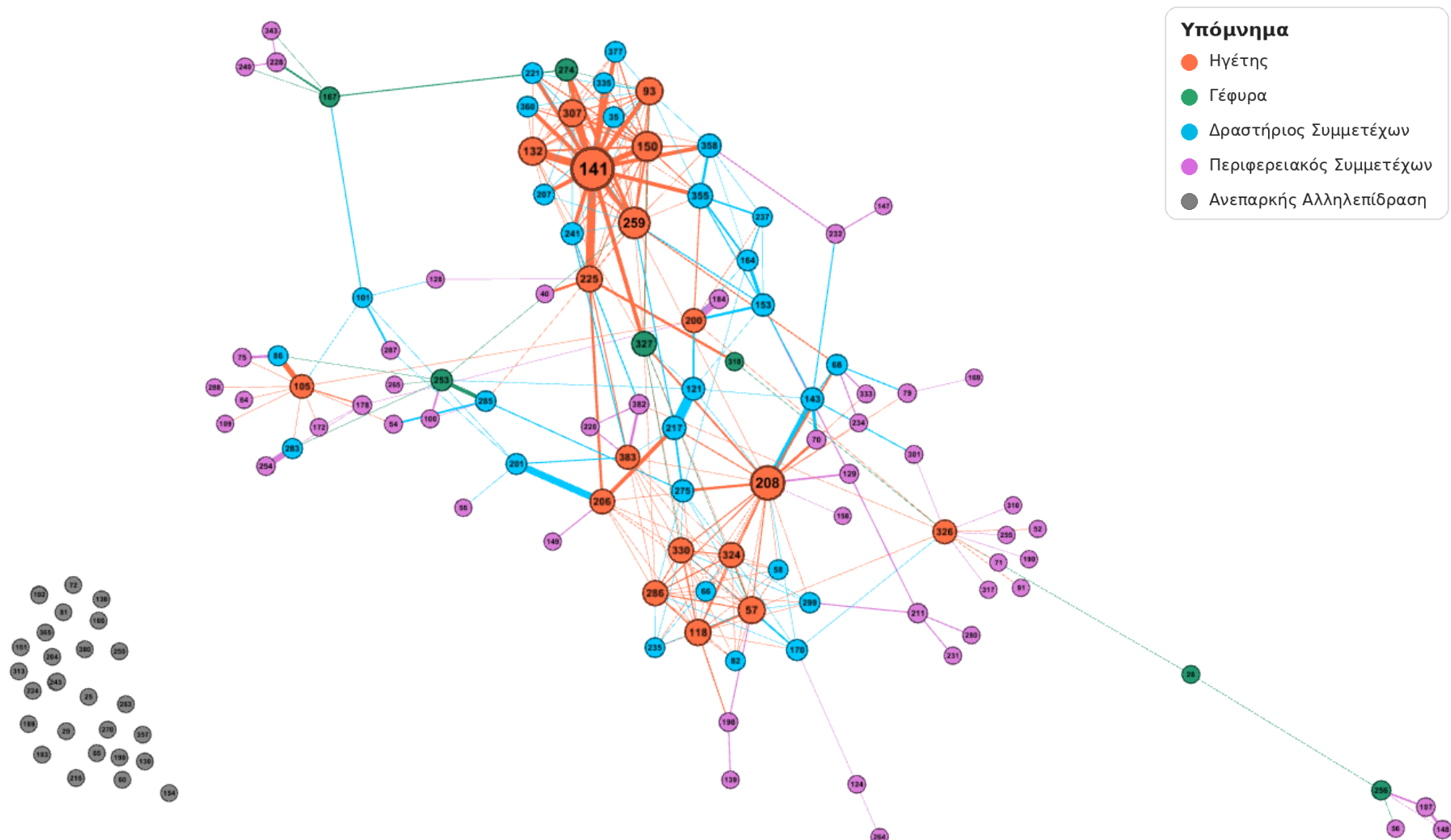
Χρώμα κόμβων: Καθορίζεται από τον κοινωνικό ρόλο κάθε φοιτητή (*network_role_el*), όπως προέκυψε από την αντιστοίχιση σε ρόλους (Ενότητα 4.2.3.6). Οι πέντε διακριτοί ρόλοι (Ηγέτης, Γέφυρα, Δραστήριος Συμμετέχων, Περιφερειακός Συμμετέχων, Ανεπαρκής Αλληλεπίδραση) αποτυπώνονται με διαφορετικά χρώματα, επιτρέποντας την άμεση οπτική αναγνώριση της ποιοτικής θέσης κάθε φοιτητή στο δίκτυο.

Μέγεθος κόμβων: Ανάλογο του σταθμισμένου βαθμού (*sna_weighted_degree*) κάθε φοιτητή, ο οποίος αποτυπώνει τη συνολική ένταση της συνεργασίας του. Μεγαλύτεροι κύκλοι υποδηλώνουν φοιτητές με υψηλότερη ένταση συνδέσεων.

Πάχος ακμών: Ανάλογο του βάρους της σχέσης (*Weight*) μεταξύ δύο φοιτητών, όπως προέκυψε από το άθροισμα των ενεργητικών συνεισφορών τους σε κοινά threads. Παχύτερες γραμμές αντιστοιχούν σε ισχυρότερες συνεργασίες.

Το δίκτυο απεικονίστηκε ως μη-κατευθυνόμενο. Οι ετικέτες των κόμβων εμφανίζονται, ώστε να είναι δυνατή η ταυτοποίηση των φοιτητών κατά την οπτική επιθεώρηση του δικτύου.

Η οπτικοποίηση αναδεικνύει έναν πυκνό πυρήνα συνεργασιών στο κέντρο του δικτύου, όπου συγκεντρώνονται οι φοιτητές με υψηλή κεντρικότητα (Ηγέτες), ενώ οι Γέφυρες εντοπίζονται σε στρατηγικές θέσεις μεταξύ διαφορετικών υποομάδων. Περιφερειακοί φοιτητές με χαμηλότερη συνδεσιμότητα τοποθετούνται στα άκρα του σχήματος. Στην περιφέρεια διακρίνονται επίσης οι 28 απομονωμένοι κόμβοι ($\text{degree} = 0$), οι οποίοι δεν συνδέονται με καμία ακμή μετά το φιλτράρισμα. Δεν παρατηρούνται πολλαπλά μεγάλα αποσυνδεδεμένα συστατικά πέραν των μεμονωμένων απομονωμένων κόμβων, γεγονός που υποστηρίζει την ερμηνεία των μετρικών κεντρικότητας (*betweenness*, *closeness*) στο κύριο συνδεδεμένο τμήμα του δικτύου (107 φοιτητές, 79,3% του συνόλου).



Σχήμα 4-6 Οπτικοποίηση του προβαλλόμενου και φιλτραρισμένου student-student δικτύου συνεργασίας στο Gephi

Λεπτομέρειες υλοποίησης παρουσιάζονται στο Παράρτημα Α.

4.2.4 Σήματα απογοήτευσης (Frustration Signals)

Η απογοήτευση (frustration) αναγνωρίζεται ως μια ιδιαίτερα επιζήμια συναισθηματική κατάσταση, καθώς μπορεί να μειώσει τα κίνητρα, να περιορίσει τη δέσμευση των φοιτητών και να επηρεάσει αρνητικά την ακαδημαϊκή τους επίδοση (Pekrun, 2006· Rajendran et al., 2013).

Η παρούσα εργασία υιοθετεί μια μη παρεμβατική προσέγγιση που βασίζεται αποκλειστικά στα αρχεία καταγραφής των αλληλεπιδράσεων των φοιτητών με την πλατφόρμα μάθησης.

Η προσέγγιση αυτή στηρίζεται στην παρατήρηση ενός ευρέως τεκμηριωμένου συμπεριφορικού προτύπου: όταν οι χρήστες αντιμετωπίζουν δυσκολίες κατά την αλληλεπίδραση με ένα ψηφιακό σύστημα, τείνουν να πραγματοποιούν αυξημένο αριθμό ενεργειών σε σύντομο χρονικό διάστημα, προσπαθώντας επανειλημμένα να ολοκληρώσουν μια ενέργεια ή να εντοπίσουν την επιθυμητή λειτουργία (Tsoni et al., 2026· Klein, 1998· Hertzum & Hornbæk, 2023).

Ο αλγόριθμος που αναπτύχθηκε βασίζεται στην έννοια της έντονης δραστηριότητας σε μικρά χρονικά διαστήματα.

Προεπεξεργασία δεδομένων

Αρχικά, οι χρονικές σημάνσεις που περιείχονταν στη στήλη Time μετατράπηκαν σε μορφή ημερομηνίας-ώρας (datetime). Η μετατροπή αυτή ήταν απαραίτητη ώστε να καταστεί δυνατή η χρονική ανάλυση των αλληλεπιδράσεων.

Στη συνέχεια, δημιουργήθηκε η μεταβλητή Minute, η οποία αντιστοιχεί στο χρονικό λεπτό στο οποίο πραγματοποιήθηκε κάθε ενέργεια. Η μεταβλητή αυτή χρησιμοποιήθηκε ως βασική χρονική μονάδα ανάλυσης, επιτρέποντας την ομαδοποίηση των ενεργειών των φοιτητών σε μικρά χρονικά διαστήματα.

Κατηγοριοποίηση Ενεργειών

Για την καλύτερη ερμηνεία της συμπεριφοράς των φοιτητών, οι καταγεγραμμένες ενέργειες ταξινομήθηκαν σε έξι κατηγορίες δραστηριοτήτων. Η κατηγοριοποίηση πραγματοποιήθηκε με αλγόριθμο που εξετάζει το όνομα του συμβάντος (Event name) και, όπου κρίνεται απαραίτητο, το πλαίσιο στο οποίο εντάσσεται (Event context).

Η διαδικασία κατηγοριοποίησης επιτρέπει την ανάλυση της απογοήτευσης σε διαφορετικά μαθησιακά πλαίσια, όπως η πλοήγηση στο υλικό, η συμμετοχή σε συζητήσεις ή η ολοκλήρωση δραστηριοτήτων αξιολόγησης.

Οι έξι κατηγορίες που χρησιμοποιήθηκαν είναι:

- **Login:** ενέργειες εισόδου και εξόδου από την πλατφόρμα
- **Quiz:** δραστηριότητες που σχετίζονται με κουίζ
- **Submission:** υποβολή εργασιών
- **Forum:** συμμετοχή σε συζητήσεις
- **Navigation:** πλοήγηση στο υλικό του μαθήματος
- **Other:** λοιπές ενέργειες που δεν εντάσσονται στις παραπάνω κατηγορίες

Σύνοψη δραστηριότητας ανά λεπτό

Για την ανίχνευση της έντασης της δραστηριότητας σε επίπεδο λεπτού, τα δεδομένα ομαδοποιήθηκαν με βάση τρία κριτήρια:

- τον φοιτητή,
- το λεπτό
- την κατηγορία ενέργειας

Για κάθε ομάδα υπολογίστηκε το πλήθος των καταγεγραμμένων συμβάντων (*event_count*).

Η προσέγγιση αυτή επιτρέπει την ανίχνευση σύντομων εξάρσεων δραστηριότητας που θα ήταν δύσκολο να εντοπιστούν σε μεγαλύτερα χρονικά διαστήματα.

Υπολογισμός προσαρμοστικών ορίων

Κρίσιμο σημείο της μεθοδολογίας αποτελεί η εξατομίκευση των ορίων ανίχνευσης. Η εφαρμογή ενός καθολικού, ενιαίου ορίου για όλους τους φοιτητές θα ήταν προβληματική, καθώς η ετερογένεια που χαρακτηρίζει τη συμπεριφορά των χρηστών σε ψηφιακά περιβάλλοντα μάθησης είναι ιδιαίτερα έντονη (Saqr, 2024). Ένας φοιτητής με υψηλή γενική δραστηριότητα θα δημιουργούσε πλήθος ψευδώς θετικών σημάτων, ενώ ένας πιο συνεσταλμένος χρήστης με χαμηλότερη αλληλεπίδραση θα περνούσε απαρατήρητος ακόμη κι αν βίωνε έντονη απογοήτευση.

Ο υπολογισμός των προσαρμοστικών ορίων πραγματοποιήθηκε ξεχωριστά για κάθε φοιτητή και για κάθε κατηγορία δραστηριότητας. Η επιλογή αυτή επιτρέπει την προσαρμογή της ανίχνευσης όχι μόνο

στη γενική συμπεριφορά κάθε χρήστη, αλλά και στα διαφορετικά πρότυπα αλληλεπίδρασης που χαρακτηρίζουν κάθε τύπο δραστηριότητας στην πλατφόρμα. Για παράδειγμα, η πλοήγηση στο εκπαιδευτικό υλικό συνοδεύεται συχνά από μεγαλύτερο αριθμό ενεργειών σε σχέση με δραστηριότητες όπως η υποβολή εργασιών ή η συμμετοχή σε συζητήσεις. Η εκτίμηση των ορίων σε επίπεδο κατηγορίας διασφαλίζει ότι οι αποκλίσεις εντοπίζονται σε σχέση με το κατάλληλο πλαίσιο δραστηριότητας.

Για τον υπολογισμό του ορίου για κάθε φοιτητή και για κάθε κατηγορία δραστηριότητας υπολογίστηκαν δύο στατιστικά μεγέθη:

- Ο μέσος όρος (*mean_count*) του αριθμού των ενεργειών ανά λεπτό
- Η τυπική απόκλιση (*std_count*), που αποτυπώνει τη διακύμανση της δραστηριότητάς του

Το προσαρμοστικό όριο ορίστηκε ως:

$$raw_threshold = mean_count + 2 \times std_count$$

Η επιλογή του πολλαπλασιαστή 2 βασίζεται στον εμπειρικό κανόνα της κανονικής κατανομής, σύμφωνα με τον οποίο περίπου το 95% των παρατηρήσεων βρίσκεται εντός δύο τυπικών αποκλίσεων από τον μέσο όρο. Συνεπώς, τιμές που υπερβαίνουν το όριο αυτό μπορούν να αντιμετωπιστούν ως στατιστικά ασυνήθιστες, υποδηλώνοντας πιθανή απόκλιση από το τυπικό πρότυπο συμπεριφοράς του φοιτητή (Montgomery & Runger, 2011).

Για την αποφυγή εξαιρετικά χαμηλών ορίων σε περιπτώσεις περιορισμένης δραστηριότητας (π.χ. φοιτητές με ελάχιστες αλληλεπιδράσεις), εφαρμόστηκαν κατώτατα όρια ανά κατηγορία. Τα όρια επιλέχθηκαν εμπειρικά με βάση τη φύση των αλληλεπιδράσεων στην πλατφόρμα Moodle, λειτουργούν ως μηχανισμός σταθεροποίησης του αλγορίθμου και ορίστηκαν όπως παρουσιάζεται στον Πίνακα 4-10.

Κατηγορία	Ελάχιστο όριο
login	2
quiz	3
submission	3
forum	3
navigation	4
other	3

Πίνακας 4-10 Κατώτατες τιμές ορίων ανά κατηγορία

Σε περιπτώσεις περιορισμένης δραστηριότητας, ο στατιστικός υπολογισμός θα παρήγαγε πολύ χαμηλά όρια, τα οποία θα οδηγούσαν σε μεγάλο αριθμό ψευδώς θετικών σημάτων. Με την εφαρμογή των προσαρμοστικών ορίων το τελικό όριο δεν μπορεί να πέσει κάτω από μια ελάχιστη τιμή που εξασφαλίζει ότι η ανίχνευση έχει νόημα. Το όριο αυτό ορίστηκε ως το μέγιστο μεταξύ του στατιστικά υπολογισμένου ορίου και του κατώτατου ορίου ανά κατηγορία.

Ανίχνευση σημάτων απογοήτευσης

Ως σήμα απογοήτευσης ορίστηκε κάθε λεπτό όπου ο αριθμός των ενεργειών ενός φοιτητή σε μια συγκεκριμένη κατηγορία υπερέβαινε το αντίστοιχο προσαρμοστικό όριο. Με τον τρόπο αυτό, εντοπίζονται στιγμές όπου η δραστηριότητα του φοιτητή αποκλίνει σημαντικά από το συνηθισμένο του πρότυπο.

Φίλτρο επιμονής

Η καταγραφή μεμονωμένων λεπτών αυξημένης δραστηριότητας ενέχει τον κίνδυνο να συμπεριληφθούν τυχαίες διακυμάνσεις που δεν αντιστοιχούν σε γνήσια συναισθηματική φόρτιση. Για τον λόγο αυτό, εφαρμόστηκε φίλτρο επιμονής (persistence filter) το οποίο ενισχύει την αξιοπιστία της ανίχνευσης.

Το φίλτρο λειτουργεί ως εξής: ένα λεπτό χαρακτηρίζεται ως «επίμονο σήμα» και θεωρείται έγκυρο μόνο εάν, εντός χρονικού παραθύρου ± 5 λεπτών γύρω από αυτό, υπάρχουν τουλάχιστον δύο λεπτά με σήμα απογοήτευσης.

Συγχώνευση σημάτων σε επεισόδια

Για τη συγχώνευση των επίμονων σημάτων σε επεισόδια, εφαρμόστηκε διαφορετικό κριτήριο: δύο σήματα θεωρούνται μέρος του ίδιου επεισοδίου αν η μεταξύ τους χρονική απόσταση δεν υπερβαίνει τα 2 λεπτά. Το όριο αυτό είναι μικρότερο από το παράθυρο του φίλτρου επιμονής (± 5 λεπτά) και εξυπηρετεί την ομαδοποίηση συνεχόμενων λεπτών έντονης δραστηριότητας.

Η συγχώνευση αυτή αποτυπώνει καλύτερα τη συνεχόμενη φύση της συναισθηματικής φόρτισης, όπου η απογοήτευση εκδηλώνεται ως μια παρατεταμένη περίοδος έντονης δραστηριότητας και όχι ως μεμονωμένα, απομονωμένα λεπτά.

Για κάθε επεισόδιο καταγράφηκαν:

- Η χρονική σήμανση έναρξης και λήξης
- Το συνολικό πλήθος λεπτών με σήμα απογοήτευσης εντός του επεισοδίου
- Η μέγιστη και η μέση ένταση δραστηριότητας

Εξαγωγή δεικτών απογοήτευσης και ενσωμάτωση στο μαθησιακό προφίλ

Η εφαρμογή της μεθοδολογίας οδήγησε στην εξαγωγή δεικτών απογοήτευσης από τα δεδομένα αλληλεπίδρασης των φοιτητών. Δημιουργήθηκε αρχείο, το οποίο περιέχει αναλυτικά στοιχεία για κάθε επεισόδιο απογοήτευσης που εντοπίστηκε. Από το αρχείο αυτό, εξάγονται για κάθε φοιτητή δύο νέοι δείκτες:

1. **Σύνολο σημάτων απογοήτευσης:** Ο αριθμός των λεπτών όπου η δραστηριότητά του υπερέβη το αντίστοιχο προσαρμοστικό όριο (μετά την εφαρμογή του φίλτρου επιμονής)
2. **Πλήθος επεισοδίων απογοήτευσης:** Ο αριθμός των διακριτών επεισοδίων έντονης δραστηριότητας που προκύπτουν από τη συγχώνευση διαδοχικών σημάτων απογοήτευσης

Οι δείκτες αυτοί ενσωματώνονται στο πολυδιάστατο μαθησιακό προφίλ των φοιτητών.

Λεπτομέρειες υλοποίησης παρουσιάζονται στο Παράρτημα Α.

4.2.5 Σύνθεση τελικού προφίλ

Στόχος της διαδικασίας αυτής ήταν η δημιουργία μιας ολοκληρωμένης αναπαράστασης της μαθησιακής συμπεριφοράς, η οποία θα αποτελέσει την είσοδο για τον αλγόριθμο σύστασης ομάδων στο Κεφάλαιο 5.

Για τη σύνθεση αυτή, συγκεντρώθηκαν τα δεδομένα από τις τέσσερις διακριτές διαστάσεις που αναλύθηκαν προηγουμένως:

- **Συμμετοχή:** δείκτες όπως ενεργές ημέρες, ένταση δραστηριότητας, συμμετοχή σε φόρουμ και προσπάθειες σε κουίζ
- **Επίδοση:** βαθμοί εργασιών (μόνο παραδοθείσες), αριθμός εργασιών που παραδόθηκαν και βαθμός τελικής εξέτασης
- **Κοινωνική διάσταση (SNA):** ο λειτουργικός ρόλος κάθε φοιτητή στο προβαλλόμενο δίκτυο αλληλεπιδράσεων, όπως προέκυψε από την ανάλυση κοινωνικών ρόλων (Ενότητα 4.2.3.5)
- **Συναισθηματική διάσταση:** δείκτες απογοήτευσης (σύνολο σημάτων και αριθμός επεισοδίων)

Τα δεδομένα από όλες τις επιμέρους πηγές ενοποιήθηκαν σε έναν ενιαίο πίνακα, μέσω διαδοχικών συγχωνεύσεων χρησιμοποιώντας ως κοινό κλειδί το μοναδικό αναγνωριστικό κάθε φοιτητή (*user_id*). Η διαδικασία αυτή διασφάλισε ότι για κάθε φοιτητή υπάρχει πλέον ένα πλήρες, πολυδιάστατο διάνυσμα χαρακτηριστικών που αποτυπώνει τη μαθησιακή του συμπεριφορά.

Από την ενοποίηση προέκυψαν δύο σύνολα δεδομένων:

1. **Πλήρες προφίλ:** Περιλαμβάνει το σύνολο των **383 φοιτητών** (382 από το αρχείο καταγραφής συμμετοχής + 1 φοιτητής που υπήρχε μόνο στο αρχείο βαθμολογιών). Το αρχείο αυτό περιλαμβάνει όλα τα εξαχθέντα χαρακτηριστικά (27 στήλες) για τους 382 φοιτητές, ενώ για τον επιπλέον φοιτητή οι τιμές των χαρακτηριστικών που προέρχονται από το αρχείο καταγραφής συμμετοχής έχουν τεθεί στο μηδέν. Αποτελεί την πλήρη, μη συμπιεσμένη αναπαράσταση της μαθησιακής συμπεριφοράς και μπορεί να αξιοποιηθεί για τεκμηρίωση και περαιτέρω ερευνητικές αναλύσεις.
2. **Προφίλ για ομαδοποίηση:** Περιλαμβάνει 12 πεδία, εκ των οποίων τα 11 αποτελούν χαρακτηριστικά εισόδου και το 1, το μοναδικό αναγνωριστικό του φοιτητή (`user_id`), τα οποία αντιστοιχούν στις τέσσερις βασικές διαστάσεις του μαθησιακού προφίλ (Πίνακας 4-11). Στο προφίλ αυτό, η κοινωνική διάσταση εκπροσωπείται αποκλειστικά από τον κοινωνικό ρόλο (`network_role_el`), ο οποίος συμπυκνώνει τις επιμέρους μετρικές κεντρικότητας. Η επιλογή αυτή έγινε για την αποφυγή πλεονασμού και υπερεκπροσώπησης της κοινωνικής διάστασης στη διαδικασία ομαδοποίησης. Από το στάδιο αυτό και έπειτα, όλες οι αναλύσεις και η διαδικασία σχηματισμού ομάδων αφορούν το σύνολο των 383 φοιτητών.

Η επιλογή των 11 αυτών πεδίων έγινε με κριτήριο:

- Την αντιπροσωπευτικότητα: Κάθε πεδίο αποτυπώνει μια διακριτή πτυχή της μαθησιακής συμπεριφοράς.
- Την ερμηνευσιμότητα: Ο αριθμός των πεδίων είναι διαχειρίσιμος για ανάλυση και οπτικοποίηση.
- Τη συμβατότητα με τη βιβλιογραφία: Τα πεδία αντιστοιχούν σε δείκτες που χρησιμοποιούνται ευρέως σε μελέτες Learning Analytics, Educational Data Mining και Social Network Analysis.

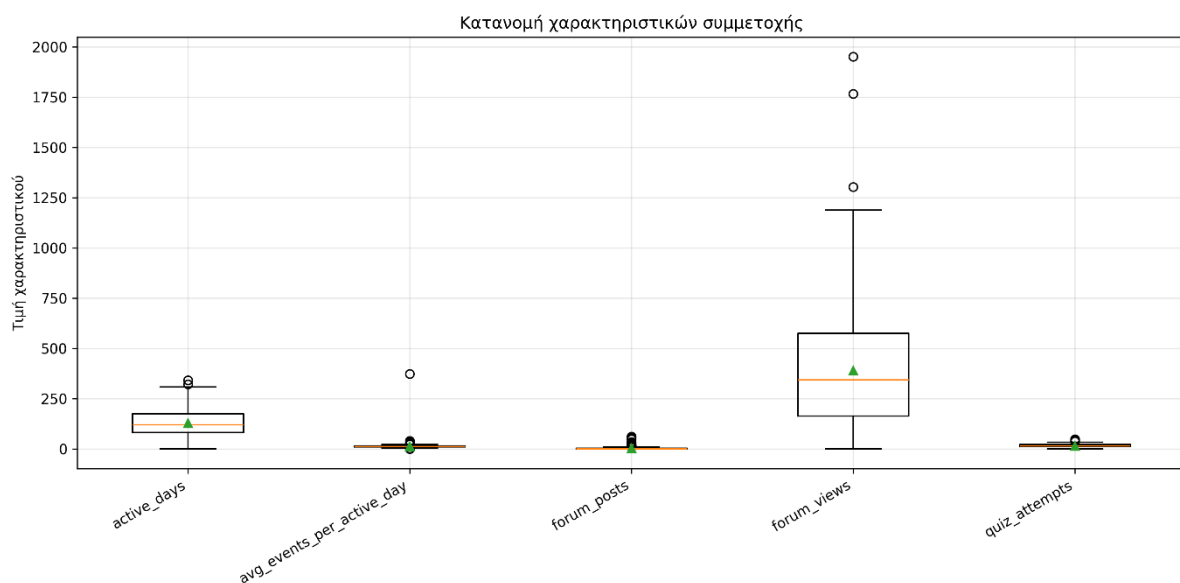
Για τη διασφάλιση της ποιότητας των δεδομένων πριν από την εφαρμογή του αλγορίθμου σύστασης ομάδων, ελέγχθηκε η μοναδικότητα του `user_id` για όλους τους φοιτητές. Οι κενές τιμές που εντοπίστηκαν μετά τη συγχώνευση συμπληρώθηκαν με μηδέν για τα αριθμητικά πεδία και με την ένδειξη «Ανεπαρκής Αλληλεπίδραση» για τα πεδία που περιέχουν χαρακτηρισμούς. Έτσι, το τελικό σύνολο δεδομένων είναι πλήρες και ομοιογενές, έτοιμο για την επόμενη φάση ανάλυσης.

Διάσταση	Πεδία	Περιγραφή
Συμμετοχή	<code>active_days</code>	Ημέρες ενεργής παρουσίας
	<code>avg_events_per_active_day</code>	Μέση ένταση δραστηριότητας

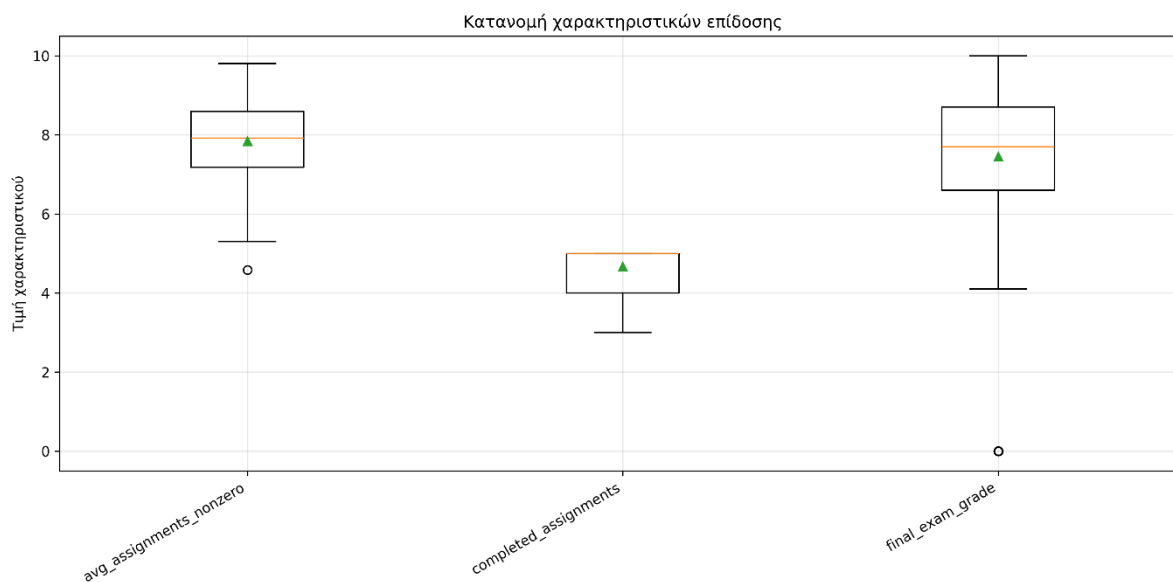
	forum_posts	Αριθμός αναρτήσεων
	forum_views	Αριθμός προβολών
	quiz_attempts	Προσπάθειες σε κουίζ
Επίδοση	avg_assignments_nonzero	Μέσος όρος εργασιών (μόνο παραδοθείσες)
	completed_assignments	Αριθμός εργασιών που παραδόθηκαν
	final_exam_grade	Βαθμός τελικής εξέτασης
Κοινωνική	network_role_el	Λειτουργικός ρόλος στο προβαλλόμενο δίκτυο
Συναισθηματική	frustration_signal_count	Σύνολο σημάτων απογοήτευσης
	frustration_episode_count	Αριθμός επεισοδίων απογοήτευσης

Πίνακας 4-11 Διαστάσεις και πεδία μαθησιακού προφίλ.

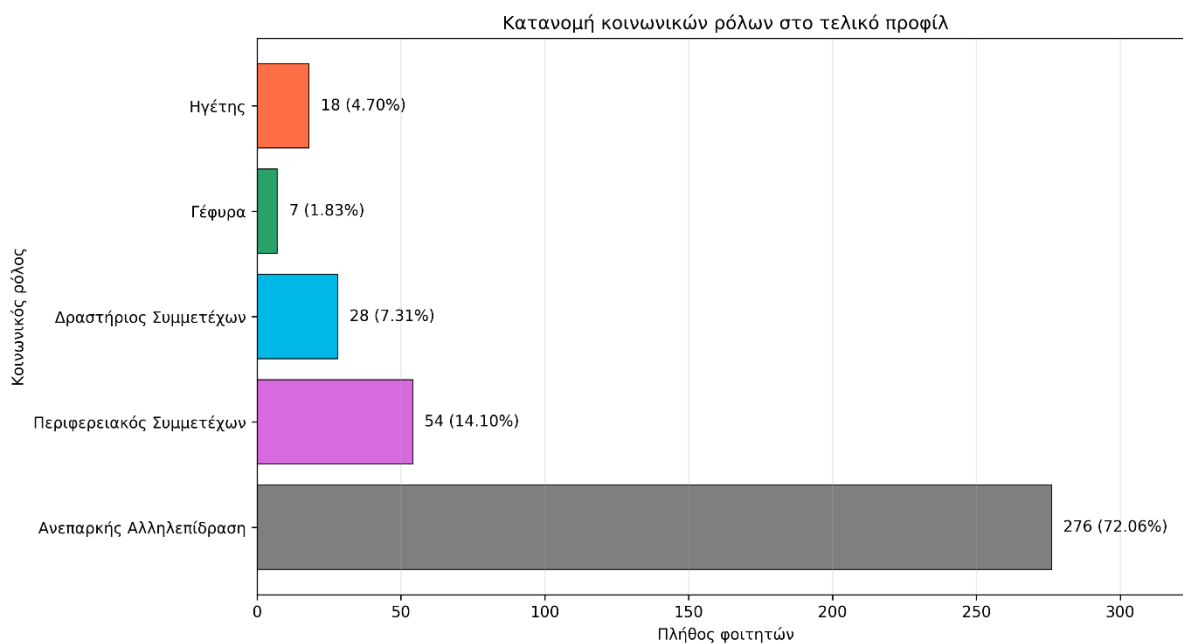
Η κατανομή των διαστάσεων του ενιαίου προφίλ απεικονίζεται στα Σχήματα 4-7 έως 4-10.



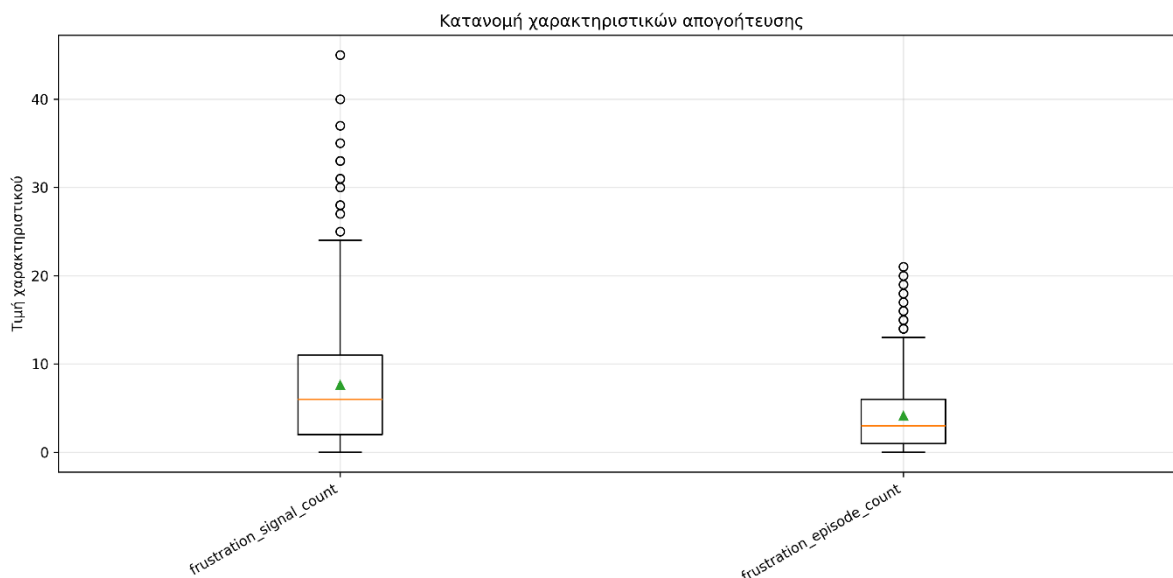
Σχήμα 4-7 Κατανομή χαρακτηριστικών συμμετοχής



Σχήμα 4-8 Κατανομή χαρακτηριστικών επίδοσης



Σχήμα 4-9 Κατανομή κοινωνικών ρόλων στο τελικό προφίλ



Σχήμα 4-10 Κατανομή χαρακτηριστικών απογοήτευσης

Η αριθμητική συνοχή του τελικού μαθησιακού προφίλ συνοψίζεται στον Πίνακα 4-12

Κατηγορία	Πλήθος	Ποσοστό
Ενεργοί με $sna_degree > 0$	107	27,9%
Συμμετείχαν αλλά $degree = 0$ (απομονωμένοι)	28	7,3%
Χωρίς καμία ενεργητική συμμετοχή στο forum	248	64,8%
Σύνολο	383	100%

Πίνακας 4-12 Σύνοψη αριθμητικής συνοχής του τελικού μαθησιακού προφίλ

Παρατηρήσεις επί της διαδικασίας

Κατά τη διαδικασία συγχώνευσης διαπιστώθηκε ότι από τους 383 φοιτητές:

- 107 (27,9%) είχαν ενεργητική συμμετοχή στο forum και διατήρησαν ισχυρές συνδέσεις στο προβαλλόμενο δίκτυο ($sna_degree > 0$). Οι φοιτητές αυτοί κατανέμονται στους ρόλους Ηγέτης, Γέφυρα, Δραστήριος Συμμετέχων και Περιφερειακός Συμμετέχων, όπως προέκυψε από την ανάλυση της Ενότητας 4.2.3.5.
- 28 (7,3%) συμμετείχαν στο forum αλλά δεν διατήρησαν ισχυρές συνδέσεις μετά το φιλτράρισμα ($degree = 0$), χαρακτηριζόμενοι ως «Ανεπαρκής Αλληλεπίδραση» (βλ. Ενότητα 4.2.3.4).
- 248 (64,8%) δεν εμφάνισαν ενεργητική συμμετοχή στο forum ικανή να δημιουργήσει συνδέσεις στο προβαλλόμενο δίκτυο, με αποτέλεσμα ο ρόλος τους να καταχωρείται επίσης ως «Ανεπαρκής Αλληλεπίδραση».

Λεπτομέρειες υλοποίησης παρουσιάζονται στο Παράρτημα Α.

4.3 Στρωματοποίηση μαθησιακού προφίλ

4.3.1 Σκοπός και θεωρητικό πλαίσιο

Η στρωματοποίηση (stratification) αποτελεί σημαντική τεχνική στην αναλυτική μάθηση (Learning Analytics) για τον εντοπισμό ομάδων φοιτητών με διακριτά χαρακτηριστικά συμπεριφοράς και επίδοσης (Baker & Inventado, 2014). Στόχος της παρούσας ενότητας είναι η κατηγοριοποίηση των φοιτητών σε επίπεδα μαθησιακού προφίλ με βάση ένα σύνθετο δείκτη προφίλ, ο οποίος ενσωματώνει πολλαπλές διαστάσεις της μαθησιακής δραστηριότητας.

Η προσέγγιση βασίζεται στο πολυδιάστατο μοντέλο εμπλοκής των Fredricks et al. (2004), το οποίο διακρίνει τρεις διαστάσεις: συμπεριφορική, συναισθηματική και γνωστική εμπλοκή. Στην παρούσα ανάλυση, οι διαστάσεις αυτές αποτυπώνονται μέσω τεσσάρων δεικτών:

- **Συμμετοχή (engagement):** αποτυπώνει τη συμπεριφορική εμπλοκή (*active_days*, *avg_events_per_active_day*, *forum_posts*, *forum_views*, *quiz_attempts*, *completed_assignments*)
- **Επίδοση (performance):** αποτυπώνει τη γνωστική εμπλοκή (*final_exam_grade*, *avg_assignments_nonzero*)
- **Κοινωνική δικτύωση (SNA):** αποτυπώνει την κοινωνική διάσταση της μάθησης μέσω του λειτουργικού ρόλου του φοιτητή στο προβαλλόμενο δίκτυο (*network_role_el*), ο οποίος προέκυψε από την ανάλυση της Ενότητας 4.2.3.5 και συμπυκνώνει τις επιμέρους μετρικές κεντρικότητας (*degree*, *weighted degree*, *betweenness*, *closeness*, *PageRank*)
- **Απογοήτευση (frustration):** αποτυπώνει τη συναισθηματική διάσταση (*frustration_signal_count*, *frustration_episode_count*)

4.3.2 Μεθοδολογία υπολογισμού σύνθετου δείκτη

Κωδικοποίηση κοινωνικού ρόλου

Για την ενσωμάτωση της κοινωνικής διάστασης στον σύνθετο δείκτη, ο ποιοτικός ρόλος κάθε φοιτητή (*network_role_el*) μετατράπηκε σε αριθμητική κλίμακα με ιεραρχική κωδικοποίηση (ordinal encoding): Ανεπαρκής Αλληλεπίδραση = 0, Περιφερειακός Συμμετέχων = 1, Δραστήριος Συμμετέχων = 2, Γέφυρα = 3, Ηγέτης = 4. Η κωδικοποίηση αυτή είναι λειτουργική και όχι ποσοτική, αποτυπώνει απλώς μια

Διπλωματική Εργασία

ιεραρχική διάταξη των ρόλων, από τον λιγότερο προς τον περισσότερο κοινωνικά ενεργό ή κεντρικό, ρόλο, χωρίς να υπονοεί ισοδύναμες αριθμητικές διαφορές μεταξύ τους.

Κανονικοποίηση και αντίστροφη κωδικοποίηση

Για τον συνδυασμό μεταβλητών διαφορετικής κλίμακας, εφαρμόστηκε τυποποίηση z-score (Field, 2013), η οποία μετασχηματίζει τις τιμές κάθε μεταβλητής ώστε να έχουν μέση τιμή 0 και τυπική απόκλιση 1, επιτρέποντας τον συνδυασμό μεταβλητών διαφορετικής κλίμακας σε έναν ενιαίο δείκτη. Καθώς τα δεδομένα αποτελούν το σύνολο του πληθυσμού και όχι δείγμα, η τυπική απόκλιση υπολογίστηκε με $ddof=0$.

Οι δείκτες απογοήτευσης (*frustration_signal_count* και *frustration_episode_count*) έχουν αντίστροφη σημασιολογική κατεύθυνση σε σχέση με τις υπόλοιπες μεταβλητές του προφίλ. Ενώ οι περισσότερες μεταβλητές αντιστοιχούν σε θετικές μορφές συμμετοχής ή επίδοσης, οι δείκτες απογοήτευσης αποτελούν αρνητικούς παράγοντες μάθησης. Για το λόγο αυτό, μετά την τυποποίηση των δεδομένων οι τιμές των συγκεκριμένων μεταβλητών αντιστράφηκαν (πολλαπλασιασμός με -1), έτσι ώστε υψηλότερα επίπεδα απογοήτευσης να μειώνουν τον συνολικό δείκτη προφίλ του φοιτητή (D'Mello & Graesser, 2012).

Στάθμιση χαρακτηριστικών ανά διάσταση

Για κάθε διάσταση υπολογίστηκε επιμέρους δείκτης ως σταθμισμένος γραμμικός συνδυασμός των αντιστοιχών τυποποιημένων χαρακτηριστικών. Τα βάρη των χαρακτηριστικών (Πίνακας 4.13) δεν προέκυψαν από βελτιστοποίηση, αλλά ορίστηκαν ευρετικά (heuristically), με γνώμονα τη σχετική σημασία κάθε χαρακτηριστικού για την περιγραφή της αντίστοιχης διάστασης του μαθησιακού προφίλ.

Ειδικότερα:

- Στη Συμμετοχή, μεγαλύτερη βαρύτητα (0,20) δόθηκε στις ενεργές ημέρες (*active_days*), τις αναρτήσεις (*forum_posts*) και τις ολοκληρωμένες εργασίες (*completed_assignments*), καθώς τα χαρακτηριστικά αυτά αποτυπώνουν την ουσιαστική εμπλοκή του φοιτητή. Αντίθετα, οι απλές προβολές (*forum_views*) έλαβαν μικρότερη βαρύτητα (0,10), καθώς αντιστοιχούν σε παθητική συμπεριφορά.
- Στην Επίδοση, η τελική εξέταση (*final_exam_grade*) έλαβε μεγαλύτερη βαρύτητα (0,60) έναντι του μέσου όρου των εργασιών (0,40), καθώς αποτελεί πιο ολοκληρωμένη και συγκρίσιμη αξιολόγηση των γνώσεων.

- Η Κοινωνική θέση εκπροσωπείται από τον κωδικοποιημένο ρόλο (*social_role_score*) με βάρος 1,0, καθώς η διάσταση αυτή αποτελείται από ένα μόνο χαρακτηριστικό (τον ρόλο), το οποίο ήδη συμπυκνώνει τις επιμέρους μετρικές κεντρικότητας..
- Η Απογοήτευση σταθμίστηκε εξίσου στους δύο δείκτες της (0,50 έκαστος), καθώς και οι δύο συμβάλλουν στην αποτύπωση της συναισθηματικής επιβάρυνσης.

Διάσταση	Χαρακτηριστικό	Βάρος	Αιτιολόγηση
Συμμετοχή	<i>active_days</i>	0,20	Αντιπροσωπεύει τη διάρκεια ενεργής παρουσίας
	<i>avg_events_per_active_day</i>	0,15	Μετρά την ένταση της δραστηριότητας
	<i>forum_posts</i>	0,20	Άμεση μέτρηση ενεργητικής συμβολής
	<i>forum_views</i>	0,10	Παθητική συμπεριφορά, μικρότερη βαρύτητα
	<i>quiz_attempts</i>	0,15	Αντανακλά προσπάθεια αυτοαξιολόγησης
	<i>completed_assignments</i>	0,20	Δείκτης ολοκλήρωσης, σημαντικός για συνεργασία
Επίδοση	<i>final_exam_grade</i>	0,60	Ολοκληρωμένη αξιολόγηση γνώσεων
	<i>avg_assignments_nonzero</i>	0,40	Μέση επίδοση σε εργασίες, συμπληρωματική
Κοινωνική θέση	<i>social_role_score</i> (0–4)	1,00	Ιεραρχική κωδικοποίηση ρόλου (ordinal encoding)
Απογοήτευση	<i>frustration_signal_count</i>	0,50	Συχνότητα σημάτων απογοήτευσης (μετά από αντιστροφή)
	<i>frustration_episode_count</i>	0,50	Πλήθος διακριτών επεισοδίων απογοήτευσης (μετά από αντιστροφή)

Πίνακας 4-13 Βάρη χαρακτηριστικών ανά διάσταση

Η ευρετική επιλογή των βαρών είναι συνεπής με τη λογική του σύνθετου δείκτη, ο οποίος δεν φιλοδοξεί να αποτελέσει βελτιστοποιημένο εργαλείο πρόβλεψης, αλλά μια ερμηνεύσιμη συμπύκνωση των διαστάσεων του μαθησιακού προφίλ για την υποστήριξη του σχηματισμού ομάδων.

Στάθμιση διαστάσεων

Για κάθε διάσταση υπολογίστηκε επιμέρους δείκτης ως σταθμισμένος γραμμικός συνδυασμός των αντίστοιχων μεταβλητών. Τα βάρη των διαστάσεων (Πίνακας 4.14) επιλέχθηκαν ευρετικά (heuristically), ώστε να αποτυπώνουν τη σχετική σημασία των διαφορετικών πτυχών της μαθησιακής δραστηριότητας, και στη συνέχεια υποβλήθηκαν σε έλεγχο ευαισθησίας. Η συμμετοχή λαμβάνει το υψηλότερο βάρος, καθώς η ενεργή εμπλοκή θεωρείται βασική προϋπόθεση της μαθησιακής διαδικασίας (Redmond et al., 2018), ενώ η επίδοση και η κοινωνική αλληλεπίδραση λειτουργούν ως συμπληρωματικοί δείκτες της μαθησιακής επιτυχίας (York et al., 2015· Siemens, 2005). Ο δείκτης απογοήτευσης ενσωματώνεται ως διορθωτικός παράγοντας με μικρότερο βάρος (Πίνακας 4-14).

Διάσταση	Βάρος	Θεωρητική τεκμηρίωση
Συμμετοχή (engagement)	40%	Η ενεργή αλληλεπίδραση αποτελεί προϋπόθεση μάθησης (Redmond et al., 2018)
Επίδοση (performance)	25%	Τα μαθησιακά αποτελέσματα ως τελικό κριτήριο (York et al., 2015)
Κοινωνική δικτύωση (SNA)	25%	Ο λειτουργικός ρόλος στο προβαλλόμενο δίκτυο – η κοινωνική διάσταση στη μάθηση (Siemens, 2005)
Απογοήτευση (frustration)	10%	Διορθωτικός παράγοντας για μαθησιακές δυσκολίες (D'Mello & Graesser, 2012)

Πίνακας 4-14 Βάρη διαστάσεων σύνθετου δείκτη

Ο τελικός σύνθετος δείκτης προφίλ υπολογίζεται ως γραμμικός συνδυασμός των τεσσάρων διαστάσεων:

$$profile_score = 0,40 \times engagement_score + 0,25 \times performance_score + 0,25 \times social_score + 0,10 \times frustration_score$$

όπου κάθε επιμέρους δείκτης αντιστοιχεί στο σταθμισμένο μέσο των κανονικοποιημένων μεταβλητών της αντίστοιχης διάστασης.

Έλεγχος ευαισθησίας των βαρών

Για να διαπιστωθεί αν η στρωματοποίηση που προέκυψε από τον σύνθετο δείκτη προφίλ (Ενότητα 4.3) είναι σταθερή, πραγματοποιήθηκε τοπικός έλεγχος ευαισθησίας των βαρών (local weight sensitivity analysis). Στόχος ήταν να εξεταστεί αν η ταξινόμηση των φοιτητών στα πέντε στρώματα αλλάζει όταν τα βάρη των χαρακτηριστικών και των διαστάσεων μεταβάλλονται εντός ενός ελεγχόμενου εύρους $\pm 20\%$ επί της αρχικής τους τιμής, πριν από την ανακανονικοποίησή τους. Το εύρος $\pm 20\%$ επιλέχθηκε

Διπλωματική Εργασία

ως επαρκής μεταβολή για τον έλεγχο της ευαισθησίας της στρωματοποίησης ως προς τα βάρη, σύμφωνα με τις γενικές κατευθύνσεις για σύνθετους δείκτες (OECD/European Union/EC-JRC, 2008). Η εξέταση αυτή είναι σημαντική, καθώς ένας δείκτης που χρησιμοποιείται για την κατηγοριοποίηση των φοιτητών θα πρέπει να παραμένει συνεπής ακόμη και όταν τα βάρη του μεταβάλλονται σε περιορισμένο βαθμό.

Ο έλεγχος υλοποιήθηκε σε Python, ακολουθώντας πιστά τη λογική υπολογισμού του σύνθετου δείκτη και της στρωματοποίησης. Αρχικά, επιβεβαιώθηκε ότι το βασικό σενάριο αναπαρήγαγε πλήρως την αρχική ταξινόμηση, ώστε να διασφαλιστεί η συγκρισιμότητα των αποτελεσμάτων. Στη συνέχεια, εξετάστηκαν 28 εναλλακτικά σενάρια. Σε κάθε σενάριο, μεταβλήθηκε ένα μόνο βάρος κατά $\pm 20\%$ επί της αρχικής του τιμής και το αντίστοιχο σύνολο βαρών ανακανονικοποιήθηκε ώστε το άθροισμά τους να παραμένει ίσο με 1. Πρόκειται, επομένως, για τοπικό έλεγχο ευαισθησίας, καθώς δεν εξετάστηκαν ταυτόχρονα μεταβολές πολλαπλών βαρών.

Τα σενάρια χωρίστηκαν σε δύο κατηγορίες: (α) μεταβολή των βαρών επιμέρους χαρακτηριστικών εντός των διαστάσεων της Συμμετοχής, της Επίδοσης και της Απογοήτευσης (20 σενάρια), και (β) μεταβολή των βαρών των τεσσάρων διαστάσεων (8 σενάρια). Η κοινωνική διάσταση δεν εξετάστηκε στο επίπεδο των επιμέρους χαρακτηριστικών, καθώς αποτελείται από ένα μόνο χαρακτηριστικό (*social_role_score*) με βάρος 1,00. Οποιαδήποτε μεταβολή του βάρους του, μετά την ανακανονικοποίηση, θα κατέληγε πάλι στην τιμή 1,00. Η επίδρασή της ελέγχθηκε μέσω της μεταβολής του βάρους ολόκληρης της διάστασης.

Για κάθε σενάριο, καταγράφηκαν: το ποσοστό των φοιτητών που παρέμειναν στο ίδιο στρώμα, η συμφωνία στη σειρά κατάταξης (συντελεστής Spearman), το μέγεθος των μετακινήσεων μεταξύ των στρωμάτων και η απόσταση των φοιτητών που μετακινήθηκαν από τα όρια των στρωμάτων.

Τα αποτελέσματα δείχνουν ότι η στρωματοποίηση είναι σταθερή. Στο χειρότερο σενάριο ως προς το ποσοστό παραμονής στο ίδιο στρώμα (μείωση του βάρους της Επίδοσης κατά 20% επί της αρχικής τιμής), 345 από τους 383 φοιτητές (90,08%) παρέμειναν στο ίδιο στρώμα, οι υπόλοιποι 38 (9,92%) μετακινήθηκαν μόνο σε γειτονικό στρώμα, ενώ δεν παρατηρήθηκε μετακίνηση κατά δύο ή περισσότερα στρώματα. Από αυτούς, 37 (97,37%) βρίσκονταν σε απόσταση μικρότερη του 0,10 (σε κλίμακα z-score) από το πλησιέστερο όριο στρωματοποίησης, γεγονός που υποδηλώνει ότι η παρατηρούμενη ευαισθησία περιορίζεται σε φοιτητές που βρίσκονταν ήδη κοντά στα όρια ταξινόμησης. Ο συντελεστής Spearman για το σενάριο αυτό ήταν 0,9955, που δείχνει σχεδόν πλήρη συμφωνία στη σειρά κατάταξης.

Η χαμηλότερη τιμή του συντελεστή Spearman καταγράφηκε στο σενάριο μείωσης του βάρους της Συμμετοχής κατά 20% ($\rho = 0,9947$), όπου το 91,91% των φοιτητών παρέμεινε στο ίδιο στρώμα. Τα σενάρια που αφορούσαν την Απογοήτευση εμφάνισαν τη μεγαλύτερη σταθερότητα (99,48% παραμονή) γεγονός που είναι συμβατό με το μικρό βάρος της συγκεκριμένης διάστασης στον τελικό σύνθετο δείκτη (10%).

Τα συγκεντρωτικά αποτελέσματα ανά κατηγορία σεναρίων παρουσιάζονται στον Πίνακα 4.15.

Κατηγορία σεναρίων	Εύρος παραμονής στο ίδιο στρώμα (%)	Μέσο ποσοστό παραμονής στο ίδιο στρώμα (%)	Μέγιστη μετακίνηση σε αριθμό στρωμάτων
Βάρη χαρακτηριστικών εντός της διάστασης της Συμμετοχής	96,08 – 98,43	97,52	1
Βάρη χαρακτηριστικών εντός της διάστασης της Επίδοσης	96,34 – 97,65	97,08	1
Βάρη χαρακτηριστικών εντός της διάστασης της Απογοήτευσης	99,48	99,48	1
Βάρη των τεσσάρων διαστάσεων	90,08 – 96,87	93,43	1

Πίνακας 4-15 Συνοπτικά αποτελέσματα τοπικού ελέγχου ευαισθησίας βαρών

Συνολικά, ο τοπικός έλεγχος ευαισθησίας δείχνει ότι η στρωματοποίηση δεν είναι εύθραυστη. Οι αλλαγές στην ταξινόμηση είναι περιορισμένες, μικρής έκτασης (μόνο γειτονικές μετακινήσεις) και εντοπίζονται σχεδόν αποκλειστικά σε οριακές περιπτώσεις. Η εικόνα αυτή ενισχύει την εμπιστοσύνη ότι η επιλογή των βαρών, αν και ευρετική, οδηγεί σε μια σταθερή κατηγοριοποίηση των φοιτητών εντός του εξεταζόμενου εύρους μεταβολής.

4.3.3 Δημιουργία στρωμάτων μαθησιακού προφίλ

Για την κατηγοριοποίηση χρησιμοποιήθηκε η τυποποιημένη τιμή του δείκτη (*profile_score_z*) με κατώφλια βασισμένα στην τυπική απόκλιση, σύμφωνα με καθιερωμένες πρακτικές στην πολυμεταβλητή ανάλυση (Hair, Black, Babin & Anderson, 2010), ώστε να αποτυπωθούν οι σχετικές διαφοροποιήσεις των φοιτητών σε σχέση με τον μέσο όρο του πληθυσμού του μαθήματος (Πίνακας 4-16).

Στρώμα	Κριτήριο z-score
Πολύ Χαμηλό	$z < -1.0$
Χαμηλό	$-1.0 \leq z < -0.3$

Μεσαίο	$-0.3 \leq z \leq 0.3$
Υψηλό	$0.3 < z \leq 1.0$
Πολύ Υψηλό	$z > 1.0$

Πίνακας 4-16 Όρια στρωματοποίησης

Η επιλογή συμμετρικών ορίων γύρω από το μηδέν διασφαλίζει ότι η στρωματοποίηση δεν εισάγει μεροληψία, ενώ η ζώνη ± 0.3 δημιουργεί ένα ευρύτερο μεσαίο στρώμα αποφεύγοντας υπερβολική λεπτομέρεια κοντά στο μέσο όρο.

4.3.4 Αποτελέσματα και ερμηνεία στρωματοποίησης

Στο σύνολο των 383 φοιτητών προέκυψε η ακόλουθη κατανομή (Πίνακας 4-17):

Στρώμα	N	Ποσοστό	Αθροιστικό %
Πολύ Χαμηλό	50	13,05%	13,05%
Χαμηλό	108	28,20%	41,25%
Μεσαίο	117	30,55%	71,80%
Υψηλό	56	14,62%	86,42%
Πολύ Υψηλό	52	13,58%	100,00%

Πίνακας 4-17 Κατανομή φοιτητών ανά στρώμα

Η κατανομή παρουσιάζει μεγαλύτερη συγκέντρωση φοιτητών στα μεσαία επίπεδα συμμετοχής και μικρότερα ποσοστά στα άκρα, γεγονός που αντανακλά την ετερογένεια συμμετοχής που παρατηρείται συχνά σε περιβάλλοντα εξ αποστάσεως εκπαίδευσης (Siemens, 2013).

Η ανάλυση μέσων τιμών ανά στρώμα (Πίνακας 4-18) επιβεβαιώνει τη διαφοροποίηση:

Στρώμα	active_days	forum_posts	social_role_score	final_exam_grade	profile_score
Πολύ Χαμηλό	86,52	1,02	0,02	4,87	-0,57
Χαμηλό	99,56	0,56	0,03	7,06	-0,27
Μεσαίο	128,74	1,37	0,23	7,96	-0,01
Υψηλό	163,48	3,68	0,70	8,66	0,26

Πολύ Υψηλό	191,35	16,13	2,56	8,26	0,86
------------	--------	-------	------	------	------

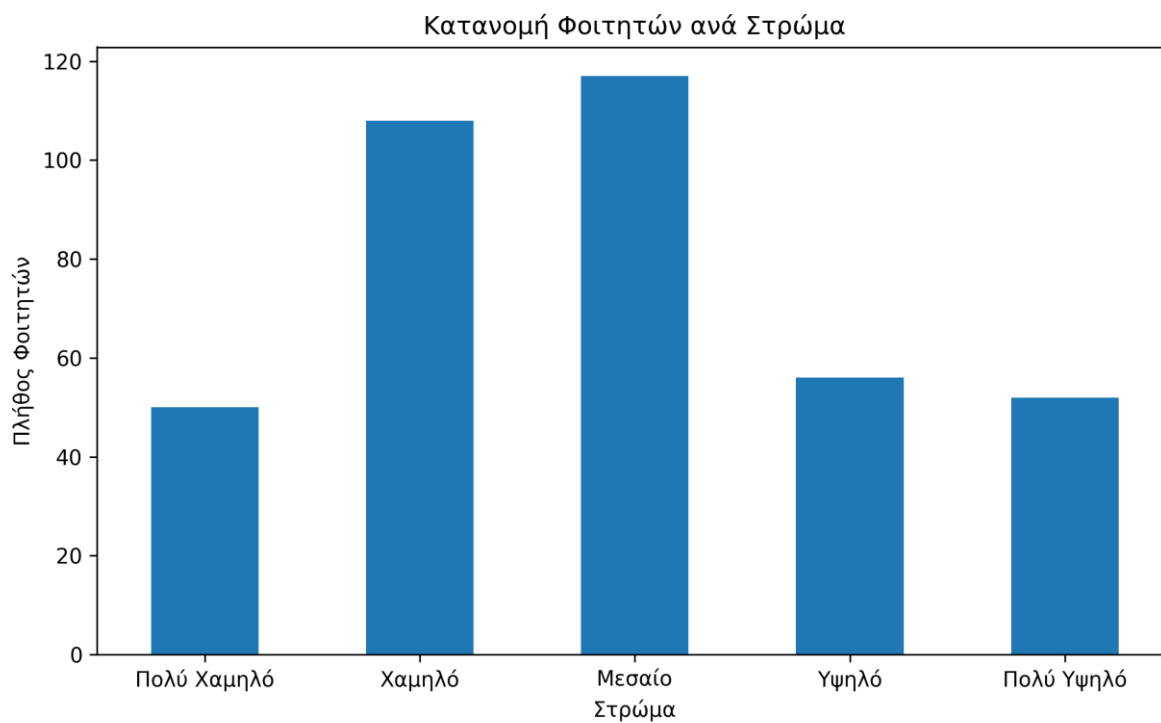
Πίνακας 4-18 Μέσες τιμές βασικών μεταβλητών ανά στρώμα

Οι μέσες τιμές δείχνουν σαφή διαφοροποίηση μεταξύ των στρωμάτων. Το `profile_score` αυξάνεται μονοτονικά, όπως αναμένεται από τον τρόπο ορισμού των στρωμάτων. Παράλληλα, οι μεταβλητές `active_days`, `forum_posts` και `social_role_score` εμφανίζουν συνολικά αυξητική τάση προς τα υψηλότερα στρώματα, με ιδιαίτερα έντονη διαφοροποίηση στο «Πολύ Υψηλό» στρώμα, υποδηλώνοντας ότι η υψηλή συμμετοχή συνοδεύεται από ποιοτική αλλαγή στην κοινωνική αλληλεπίδραση.

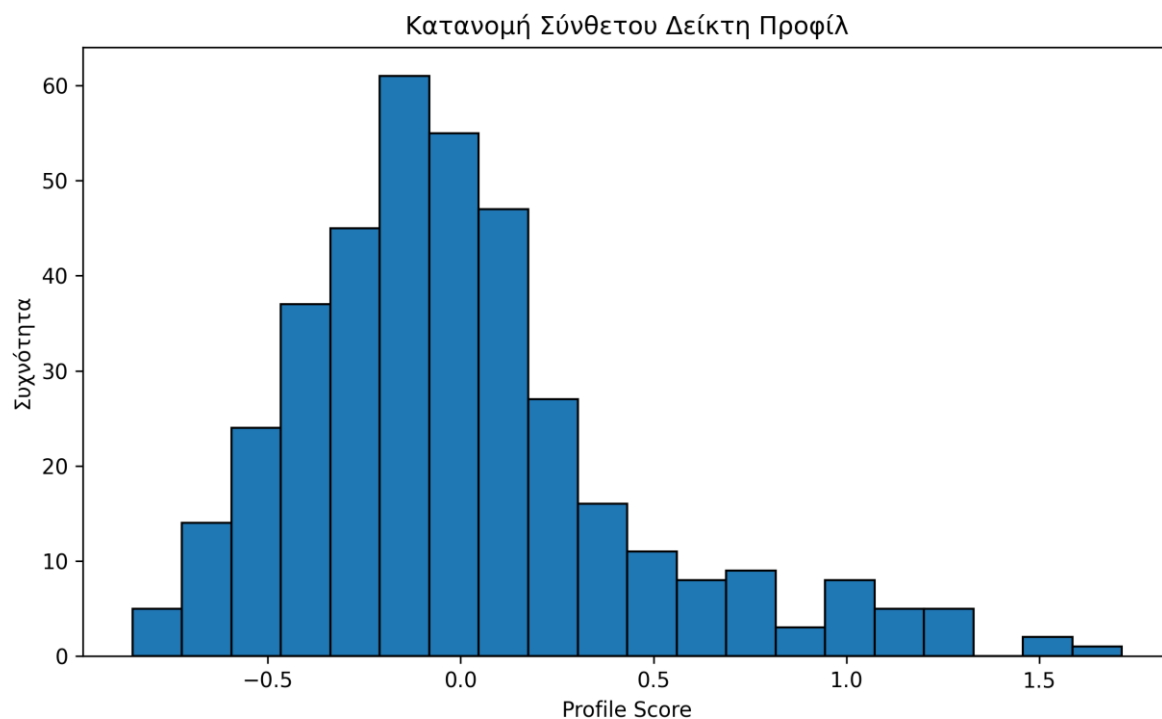
Αξίζει να σημειωθεί ότι το «Πολύ Υψηλό» στρώμα δεν εμφανίζει τη μέγιστη μέση τιμή στην τελική εξέταση (8,66 στο «Υψηλό» έναντι 8,26 στο «Πολύ Υψηλό»). Το εύρημα αυτό είναι αναμενόμενο, καθώς ο σύνθετος δείκτης δεν ταυτίζεται με την ακαδημαϊκή επίδοση, αλλά αποτυπώνει συνολικά τη συμμετοχή, την κοινωνική θέση, την επίδοση και τη συναισθηματική διάσταση. Συνεπώς, φοιτητές με ιδιαίτερα υψηλή συμμετοχή και ισχυρή κοινωνική παρουσία (υψηλό `social_role_score`) μπορούν να καταταγούν στο ανώτερο στρώμα ακόμη και αν η μέση επίδοσή τους δεν είναι η υψηλότερη.

Συνολικά, τα αποτελέσματα της στρωματοποίησης δείχνουν ότι οι φοιτητές δεν διαφοροποιούνται μόνο ως προς την επίδοση, αλλά και ως προς τη συμμετοχή και την κοινωνική τους θέση στο προβαλλόμενο δίκτυο. Τα στρώματα «Πολύ Χαμηλό» και «Χαμηλό» συγκεντρώνουν το 41,25% των φοιτητών και αντιπροσωπεύουν προφίλ χαμηλότερης συνολικής εμπλοκής, με περιορισμένη κοινωνική παρουσία. Το «Μεσαίο» στρώμα, με ποσοστό 30,55%, αντιστοιχεί στον βασικό κορμό του μαθήματος, ενώ τα στρώματα «Υψηλό» και «Πολύ Υψηλό» συγκεντρώνουν το 28,20% των φοιτητών και χαρακτηρίζονται από αυξημένη δραστηριότητα και ισχυρότερη κοινωνική παρουσία. Η διαφοροποίηση αυτή είναι ιδιαίτερα χρήσιμη για τον επόμενο στόχο της εργασίας, δηλαδή τον σχηματισμό ισορροπημένων συνεργατικών ομάδων.

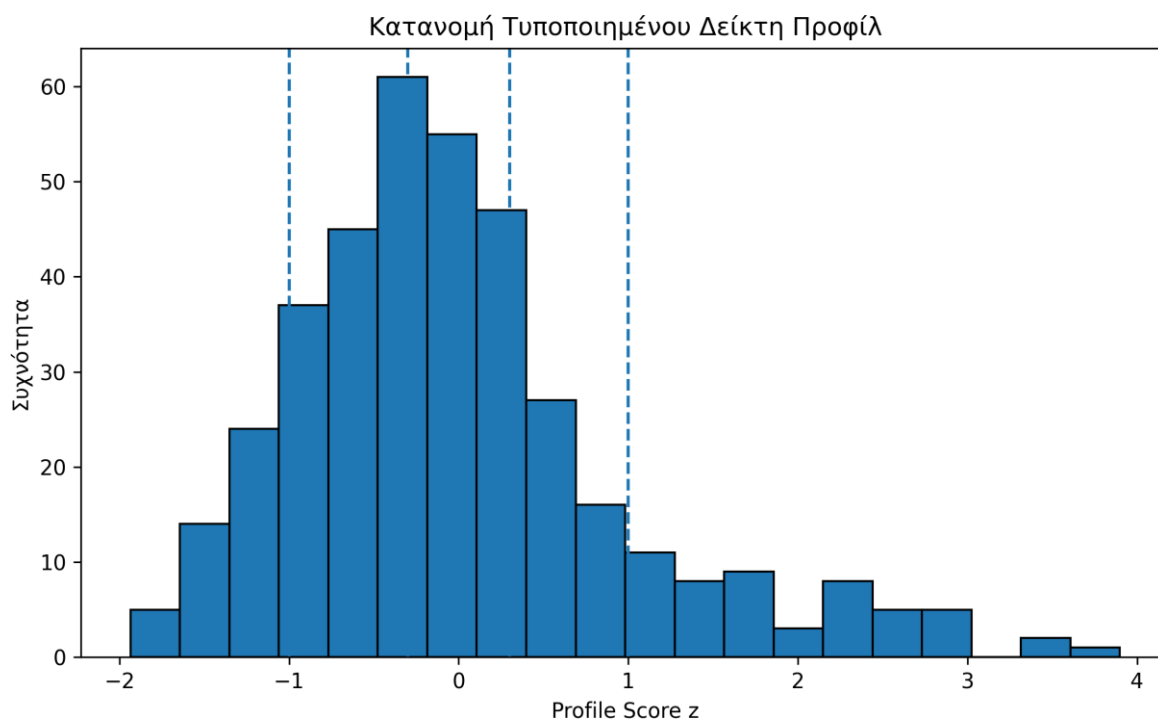
Η κατανομή των φοιτητών ανά στρώμα, η μορφή του σύνθετου δείκτη και οι διαφοροποιήσεις των δεικτών ανά στρώμα παρουσιάζονται στα Σχήματα 4-11 έως 4-19.



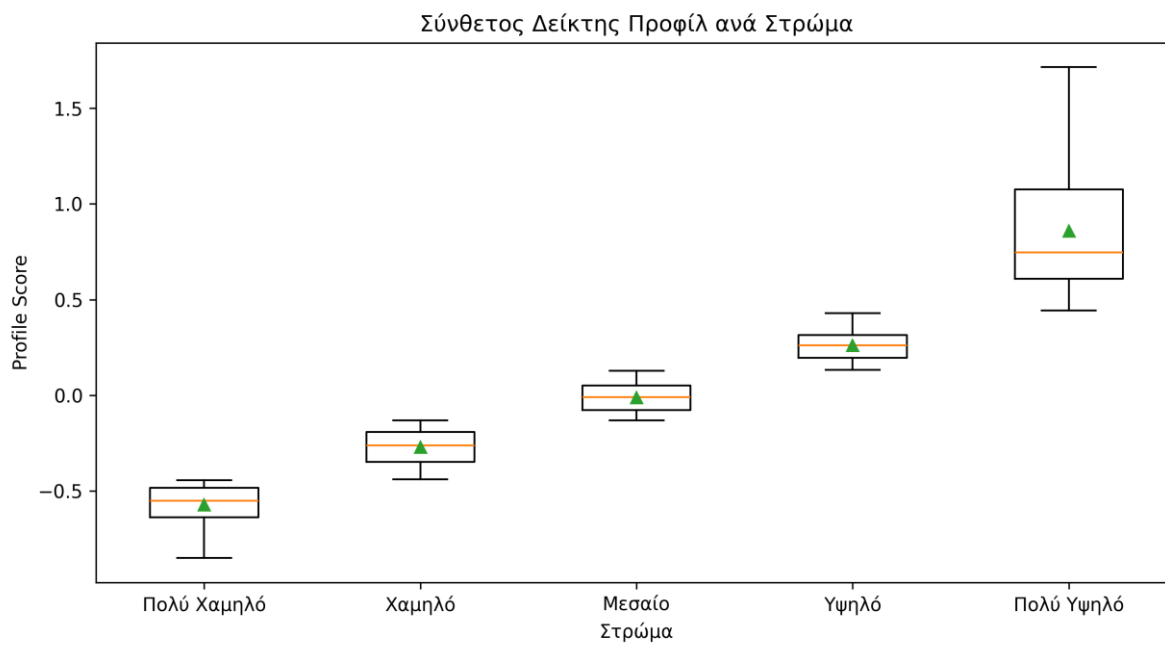
Σχήμα 4-11 Φοιτητές ανά στρώμα



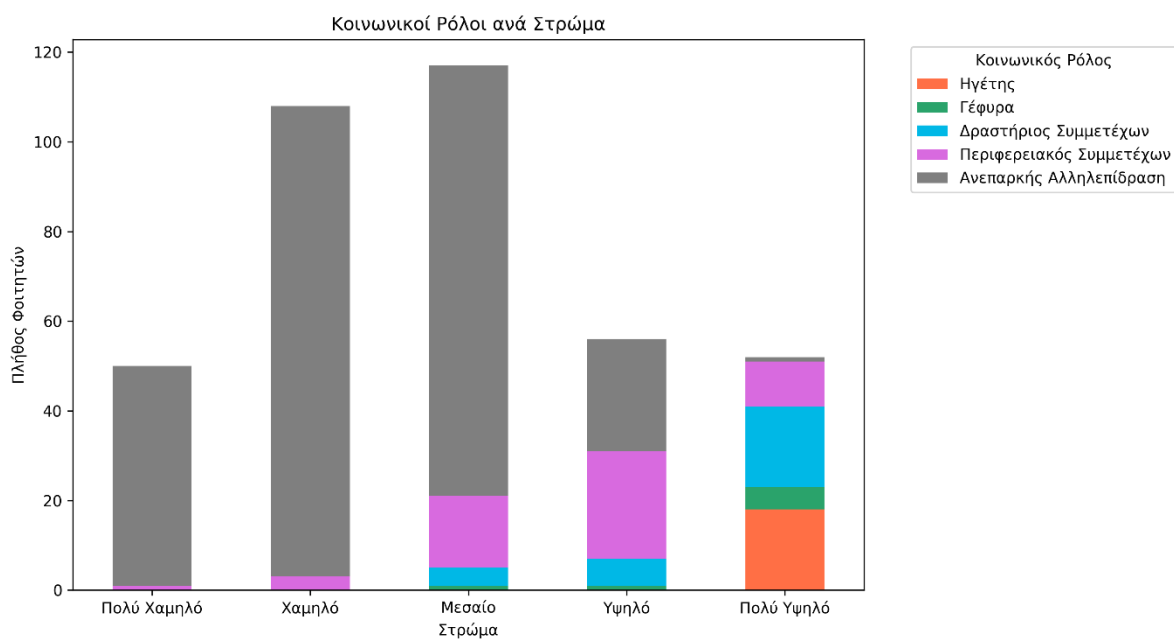
Σχήμα 4-12 Κατανομή σύνθετου δείκτη προφίλ



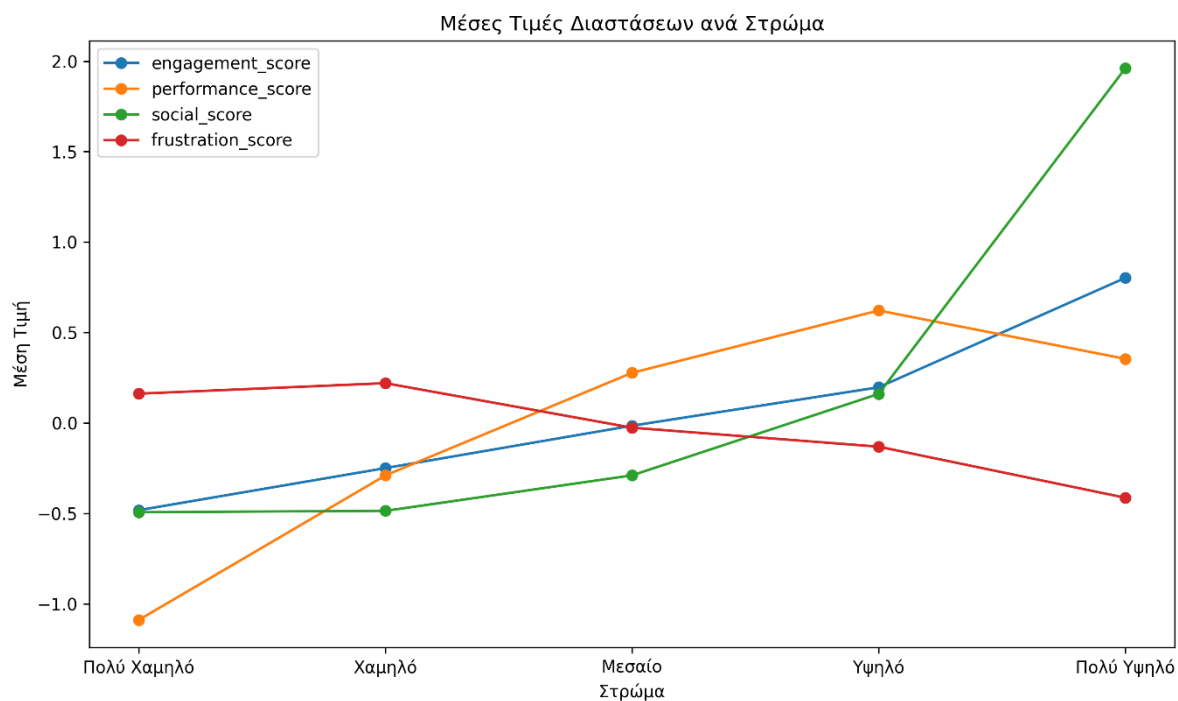
Σχήμα 4-13 Κατανομή τυποποιημένου δείκτη προφίλ



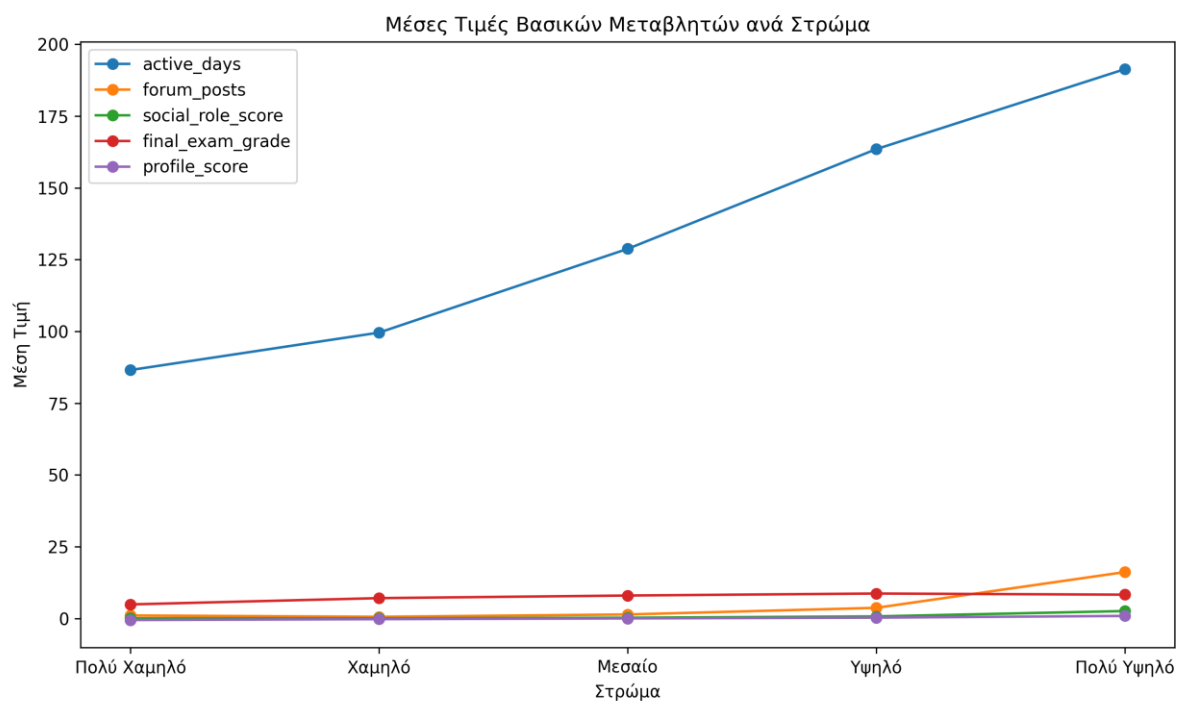
Σχήμα 4-14 Σύνθετος δείκτης προφίλ ανά στρώμα



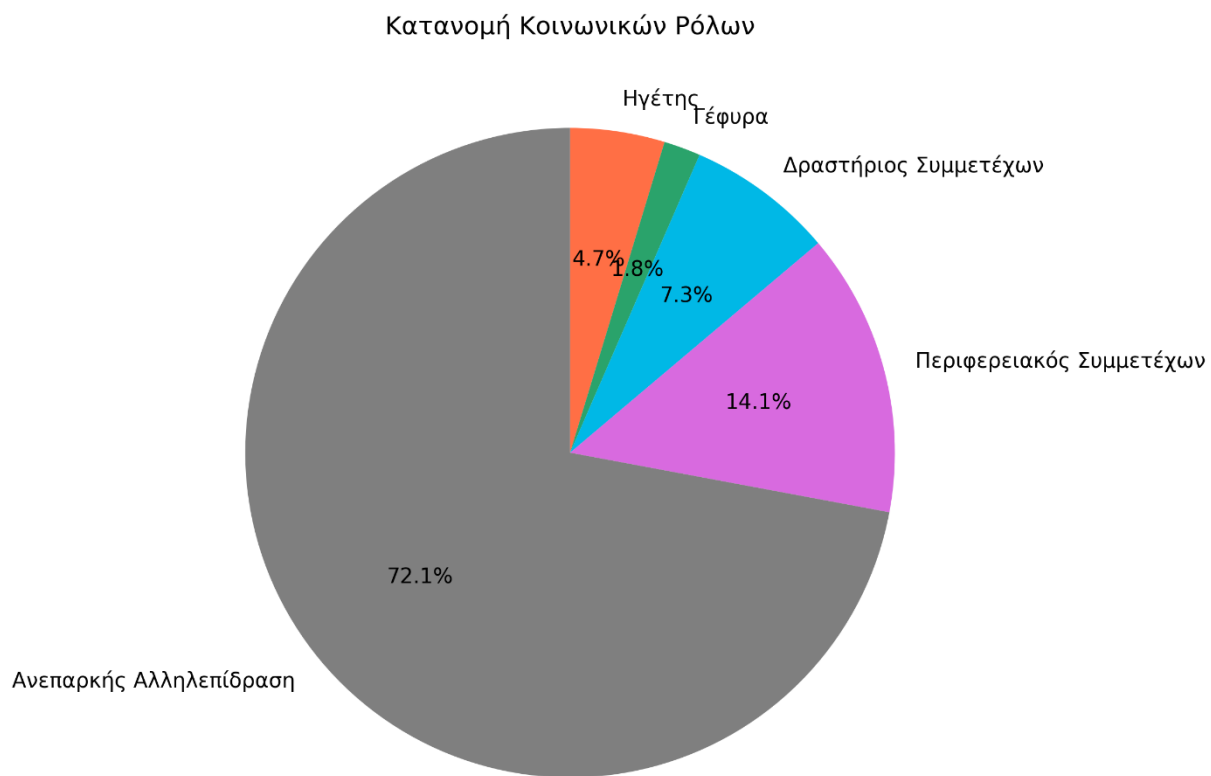
Σχήμα 4-15 Κοινωνικοί ρόλοι ανά στρώμα



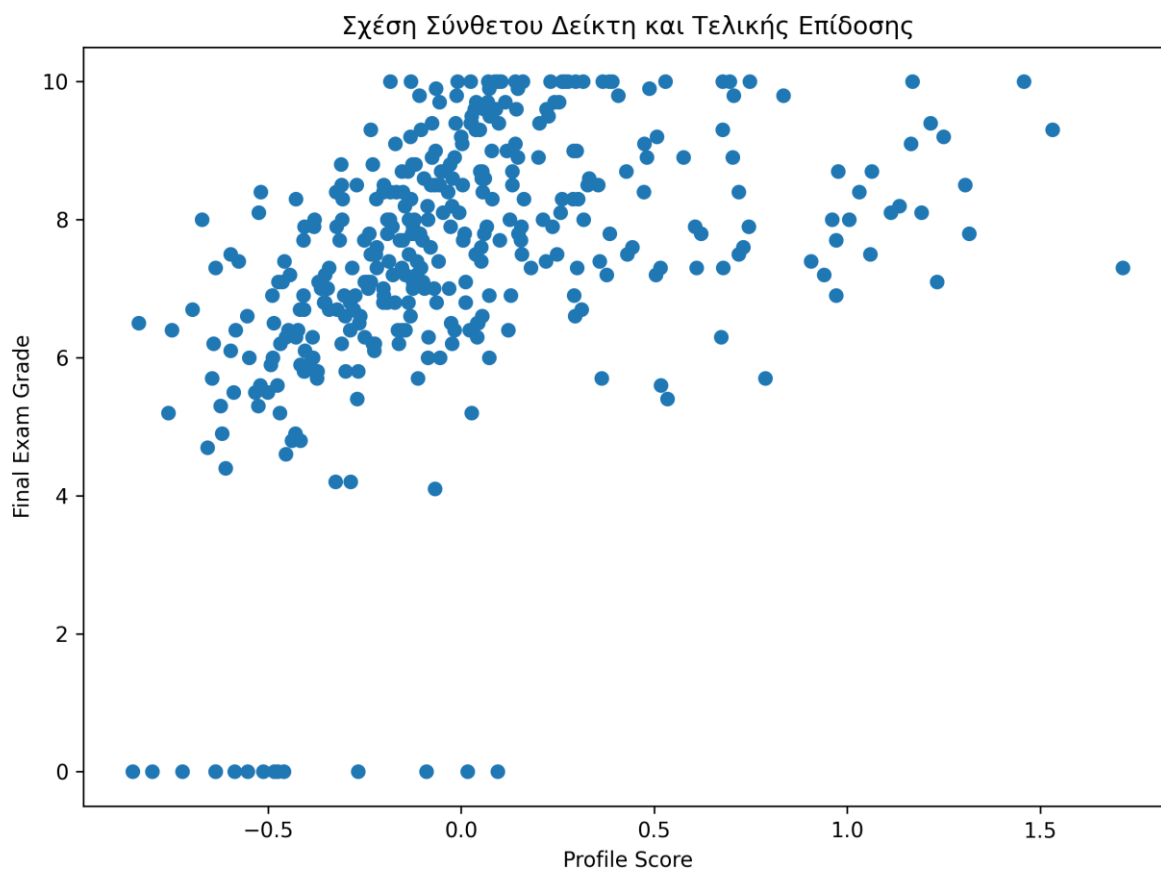
Σχήμα 4-16 Μέσες τιμές διαστάσεων ανά στρώμα



Σχήμα 4-17 Μέσες τιμές βασικών μεταβλητών ανά στρώμα



Σχήμα 4-18 Κατανομή κοινωνικών ρόλων



Σχήμα 4-19 Σχέση σύνθετου δείκτη και τελικής επίδοσης

Λεπτομέρειες υλοποίησης παρουσιάζονται στο Παράρτημα Α.

5 Σχεδιασμός και υλοποίηση του συστήματος συστάσεων

5.1 Διατύπωση του προβλήματος ως πολυκριτηριακή σύσταση

Η δημιουργία αποτελεσματικών ομάδων συνεργατικής μάθησης αποτελεί ένα σύνθετο πρόβλημα, καθώς η ποιότητα της συνεργασίας δεν εξαρτάται από ένα μόνο χαρακτηριστικό των φοιτητών, αλλά από τον συνδυασμό πολλαπλών παραγόντων που σχετίζονται με τη συμμετοχή, την επίδοση και την κοινωνική τους αλληλεπίδραση. Στο πλαίσιο των συστημάτων σύστασης, το πρόβλημα αυτό μπορεί να διατυπωθεί ως πρόβλημα πολυκριτηριακής σύστασης ομάδων (group recommendation), όπου το «αντικείμενο» που προτείνεται δεν είναι ένας πόρος ή ένα προϊόν, αλλά μια σύνθετη ομάδα ατόμων (Jameson & Smyth, 2007).

Στην παρούσα εργασία, η σύσταση ομάδων βασίζεται στο πολυδιάστατο προφίλ φοιτητή που δημιουργήθηκε στο Κεφάλαιο 4. Το προφίλ αυτό περιλαμβάνει μεταβλητές που αποτυπώνουν διαφορετικές διαστάσεις της μαθησιακής συμπεριφοράς: συμμετοχής (4.2.1), ακαδημαϊκής επίδοσης (4.2.2), κοινωνικής αλληλεπίδρασης μέσω μετρικών δικτύου (4.2.3), καθώς και δείκτες απογοήτευσης (4.2.4). Κάθε φοιτητής αναπαρίσταται ως διάνυσμα χαρακτηριστικών σε πολυδιάστατο χώρο.

Βάσει της θεωρίας Κοινωνικής Αλληλεξάρτησης (Butera & Buchs, 2019), οι ομάδες λειτουργούν αποτελεσματικότερα όταν παρουσιάζουν ταυτόχρονα ομοιογένεια ως προς τα κίνητρα (για κοινή βάση συνεργασίας) και ετερογένεια ως προς τις δεξιότητες και τους ρόλους (για λειτουργική συμπληρωματικότητα). Με βάση αυτές τις αρχές, το πρόβλημα της σύστασης ομάδων διατυπώθηκε ως διαδικασία λήψης απόφασης πολλαπλών κριτηρίων.

Η ανισοκατανομή των τιμών κεντρικότητας που αναδείχθηκε στο Κεφάλαιο 4 έχει άμεσες παιδαγωγικές συνέπειες για τον σχηματισμό των ομάδων. Οι φοιτητές με υψηλότερη κοινωνική δραστηριότητα μπορούν να λειτουργήσουν ως πυρήνες ενεργοποίησης της συνεργασίας, ενώ οι περιφερειακοί φοιτητές ή όσοι χαρακτηρίζονται από Ανεπαρκή Αλληλεπίδραση είναι σκόπιμο να κατανέμονται σε ομάδες με ενεργότερα μέλη. Επιδιώκεται, επομένως, μια μορφή ελεγχόμενης ετερογένειας, η οποία αξιοποιεί τη συμπληρωματικότητα και μπορεί να υποστηρίξει την αλληλοδιδασκαλία, χωρίς να δημιουργεί ακραίες ανισορροπίες όπου η συμμετοχή και η ευθύνη της συνεργασίας συγκεντρώνονται σε ένα μόνο μέλος.

Τα βασικά κριτήρια που χρησιμοποιούνται στη διαδικασία σύστασης είναι τα ακόλουθα:

1. Ισορροπία συμμετοχής και λειτουργική ετερογένειάς.

Για την αποφυγή δημιουργίας ομάδων με πολύ χαμηλή ή πολύ υψηλή συνολική δραστηριότητα, επιδιώκεται η κατανομή των φοιτητών από διαφορετικά επίπεδα συμμετοχής σε κάθε ομάδα. Αξιοποιείται η στρωματοποίηση βάσει σύνθετου δείκτη προφίλ (Ενότητα 4.3), όπου οι φοιτητές κατανέμονται σε πέντε στρώματα (Πολύ Χαμηλό έως Πολύ Υψηλό). Η στρωματοποίηση χρησιμοποιείται ως μηχανισμός

αρχικής εξισορρόπησης του πληθυσμού, ώστε να αποφεύγεται η συγκέντρωση φοιτητών με παρόμοια επίπεδα εμπλοκής στην ίδια ομάδα. Στόχος είναι κάθε ομάδα να περιλαμβάνει ισορροπημένο μείγμα από διαφορετικά στρώματα.

2. Ετερογένεια κοινωνικών ρόλων

Οι κοινωνικοί ρόλοι των φοιτητών εκτιμώνται μέσω μετρικών ανάλυσης κοινωνικών δικτύων, (degree centrality, betweenness centrality, PageRank) (4.2.3). Οι μετρικές αυτές επιτρέπουν τον εντοπισμό φοιτητών που λειτουργούν ως διαμεσολαβητές (Γέφυρες), συνδετικοί κόμβοι ή άτομα επιρροής (Ηγέτες) στο δίκτυο αλληλεπίδρασης. Η σύσταση των ομάδων στοχεύει στη δημιουργία ομάδων με ετερογενή σύνθεση ρόλων, ώστε κάθε ομάδα να περιλαμβάνει φοιτητές με διαφορετικά επίπεδα κοινωνικής επιρροής και διασύνδεσης. Ιδιαίτερη βαρύτητα δίνεται στην παρουσία Ηγέτη ή Γέφυρας, καθώς οι κόμβοι με υψηλή ενδιάμεση κεντρικότητα ή επιρροή λειτουργούν ως καταλύτες αλληλεπίδρασης και διευκολύνουν τη ροή πληροφορίας μέσα στην ομάδα.

3. Ισορροπία ακαδημαϊκής επίδοσης μεταξύ ομάδων.

Λαμβάνονται υπόψη ο μέσος όρος βαθμολογίας των εργασιών (*avg_assignments_nonzero*) και ο βαθμός τελικής εξέτασης (*final_exam_grade*) (4.2.2). Στόχος είναι οι ομάδες να παρουσιάζουν παρόμοιο επίπεδο συνολικής επίδοσης, ώστε να περιορίζεται η δημιουργία ομάδων με ακραία υψηλή ή χαμηλή ακαδημαϊκή δυναμική και να ενισχύεται η συγκρισιμότητα μεταξύ των ομάδων.

4. Αποφυγή συσσώρευσης υψηλών επιπέδων απογοήτευσης.

Τα σήματα απογοήτευσης ανιχνεύονται μέσω της ανάλυσης επαναλαμβανόμενων και ασυνήθιστα γρήγορων αλληλεπιδράσεων στο περιβάλλον της πλατφόρμας (4.2.4). Για την αποφυγή δημιουργίας ομάδων με υψηλό επίπεδο συνολικής απογοήτευσης, επιδιώκεται η ομοιόμορφη κατανομή των σχετικών δεικτών (σύνολο σημάτων και αριθμός επεισοδίων) μεταξύ των ομάδων.

Η προσέγγιση εντάσσεται στη λογική των Συστημάτων Συστάσεων με Επίγνωση Πλαισίου (CARS) (Adomavicius & Tuzhilin, 2011), όπου το πλαίσιο ορίζεται από τις παιδαγωγικές απαιτήσεις της συνεργατικής μάθησης και τα χαρακτηριστικά των φοιτητών.

5.2 Ομοφιλία και ετερογένεια στις προτεινόμενες ομάδες

Η έννοια της ομοφιλίας (homophily) —η τάση σύνδεσης ατόμων με παρόμοια χαρακτηριστικά— αποτελεί θεμελιώδη αρχή της κοινωνικής δικτύωσης με σημαντικές προεκτάσεις στην εκπαιδευτική διαδικασία (McPherson et al., 2001). Σε μαθησιακά περιβάλλοντα, η ομοφιλία μπορεί να οδηγήσει στον περιορισμό των αλληλεπιδράσεων εντός ομοιογενών υποσυνόλων, με αποτέλεσμα τη μειωμένη ανταλλαγή διαφορετικών απόψεων και την πιθανή ενίσχυση υφιστάμενων αντιλήψεων — ένα φαινόμενο που

στη βιβλιογραφία περιγράφεται ως «θάλαμοι αντήχησης» (echo chambers) όπου η επικοινωνία λαμβάνει χώρα κυρίως μεταξύ ατόμων με παρόμοιες απόψεις ή συμπεριφορές (Sunstein, 2001 · Cinelli et al., 2021).

Στο πλαίσιο της παρούσας μεθοδολογίας, η ανάλυση της ομοφιλίας εντάσσεται ως ενδιάμεσο διαγνωστικό στάδιο μεταξύ της σύνθεσης του μαθησιακού προφίλ (Κεφάλαιο 4) και του σχεδιασμού του αλγορίθμου σύστασης ομάδων (Ενότητες 5.3–5.4). Ο σκοπός της είναι διττός: αφενός, να αναδείξει τυχόν τάσεις ομοφιλίας ως προς επιλεγμένα χαρακτηριστικά του πολυδιάστατου προφίλ που αξιοποιούνται στη διαδικασία σύστασης ομάδων ενισχύοντας την τεκμηρίωση για την αποφυγή υπερβολικής ομοιογένειας, και αφετέρου, να παράσχει μια συνοπτική περιγραφή του δικτύου που αξιοποιείται για την ερμηνεία των αποτελεσμάτων του αλγορίθμου.

5.2.1 Μεθοδολογία υπολογισμού

Η ανάλυση ομοφιλίας εφαρμόζεται στο προβαλλόμενο δίκτυο φοιτητών (Ενότητα 4.2.3). Για κάθε επιλεγμένο χαρακτηριστικό του πολυδιάστατου προφίλ (Ενότητα 4.2.5) που αξιοποιείται στη διαδικασία σύστασης ομάδων, υπολογίζονται δείκτες που αποτυπώνουν τον βαθμό ομοιότητας μεταξύ των συνδεδεμένων φοιτητών.

Για τα κατηγορικά χαρακτηριστικά (profile_stratum, network_role_el), ο δείκτης ομοφιλίας (homophily_ratio) ορίζεται ως το ποσοστό των ακμών που συνδέουν φοιτητές με την ίδια τιμή του χαρακτηριστικού:

$$homophily_ratio_{attr} = \frac{|\{(u, v) \in E : attr(u) = attr(v)\}|}{|E|}$$

όπου E είναι το σύνολο των ακμών του προβαλλόμενου δικτύου.

Για τα αριθμητικά χαρακτηριστικά (profile_score_z, social_role_score, performance_score, engagement_score, frustration_score), υπολογίζεται η μέση απόλυτη διαφορά μεταξύ των συνδεδεμένων φοιτητών:

$$mean_abs_diff_{attr} = \frac{\sum_{(u,v) \in E} |attr(u) - attr(v)|}{|E|}$$

Στην περίπτωση των αριθμητικών χαρακτηριστικών, μικρότερη μέση απόλυτη διαφορά υποδηλώνει μεγαλύτερη ομοιότητα μεταξύ των συνδεδεμένων φοιτητών.

Για την αξιολόγηση της στατιστικής σημαντικότητας, εφαρμόζεται έλεγχος τυχαιοποίησης (permutation test) με 1.000 επαναλήψεις. Στον έλεγχο αυτό, η δομή του δικτύου διατηρείται σταθερή,

ενώ οι τιμές του εξεταζόμενου χαρακτηριστικού ανακατανέμονται τυχαία στους κόμβους. Έτσι, η παρατηρούμενη ομοφιλία συγκρίνεται με την αντίστοιχη τιμή που θα αναμενόταν αν τα χαρακτηριστικά των φοιτητών ήταν τυχαία κατανομημένα στο ίδιο δίκτυο (Hanneman & Riddle, 2005).

5.2.2 Εφαρμογή στο πειραματικό σύνολο δεδομένων

Η εφαρμογή της μεθοδολογίας στο πειραματικό σύνολο δεδομένων (294 ακμές, 107 μη απομονωμένοι φοιτητές) παρήγαγε τα αποτελέσματα των Πινάκων 5.1 και 5.2. Για την αποφυγή πλεονασμού, στον πίνακα παρουσιάζεται η τυποποιημένη εκδοχή του σύνθετου δείκτη προφίλ (profile_score_z).

Χαρακτηριστικό	Ομοφιλία	Τυχαία μέση τιμή	p-value
profile_stratum	57,14%	34,44%	< 0,001
network_role_el	30,27%	34,93%	0,902

Πίνακας 5-1 Δείκτες ομοφιλίας για κατηγορικά χαρακτηριστικά

Χαρακτηριστικό	Μέση απόλυτη διαφορά	Τυχαία μέση απόλυτη διαφορά	p-value
profile_score_z	1,306	1,122	0,982
social_role_score	1,337	1,153	0,964
performance_score	0,909	0,897	0,570
engagement_score	0,687	0,692	0,499
frustration_score	1,432	1,218	0,974

Πίνακας 5-2 Δείκτες ομοφιλίας για αριθμητικά χαρακτηριστικά

5.2.3 Συμπεράσματα για το συγκεκριμένο δίκτυο

Στο υπό εξέταση δίκτυο, παρατηρείται ισχυρή ομοφιλία ως προς το στρώμα του πολυδιάστατου προφίλ (profile_stratum). Συγκεκριμένα, το 57,14% των ακμών συνδέει φοιτητές που ανήκουν στο ίδιο στρώμα προφίλ, ενώ η αντίστοιχη τυχαία μέση τιμή είναι 34,44%. Το γεγονός που υποδηλώνει ότι, στο συγκεκριμένο δίκτυο αλληλεπίδρασης, οι φοιτητές τείνουν να συνδέονται με φοιτητές παρόμοιου μαθησιακού

προφίλ. Η παρατήρηση αυτή ενισχύει την τεκμηρίωση μιας στρατηγικής ομαδοποίησης που παρεμβαίνει ενεργά για τη διασφάλιση ετερογένειας ως προς αυτή τη διάσταση.

Αντίθετα, το δίκτυο δεν εμφανίζει στατιστικά σημαντική ομοφιλία ως προς τους κοινωνικούς ρόλους (`network_role_el`), αποτέλεσμα συμβατό με την απουσία συστηματικής τάσης σύνδεσης μεταξύ φοιτητών ίδιου ρόλου. Η ρητή προώθηση ετερογένειας ρόλων από τον αλγόριθμο, επομένως, δεν αντιστρέφει κάποια ισχυρή προϋπάρχουσα τάση ομοιοφιλικής σύνδεσης, αλλά αποτελεί παιδαγωγικά σκόπιμη επιλογή. Επιπλέον, η ανάλυση των αριθμητικών χαρακτηριστικών δεν αναδεικνύει στατιστικά σημαντικές τάσεις ομοφιλίας (όλα τα $p > 0,05$), γεγονός που υποδεικνύει ότι δεν παρατηρείται ισχυρή προϋπάρχουσα ομοιοφιλική δομή ως προς αυτές τις διαστάσεις, η οποία θα έπρεπε να ληφθεί υπόψη ως περιορισμός κατά την εξισορρόπηση των ομάδων.

Είναι σημαντικό να τονιστεί ότι τα συμπεράσματα αυτά αφορούν αποκλειστικά το συγκεκριμένο σύνολο δεδομένων. Η μεθοδολογία, ωστόσο, είναι γενική και μπορεί να εφαρμοστεί σε οποιοδήποτε προβλλόμενο δίκτυο φοιτητών που προκύπτει από δεδομένα Moodle, στο πλαίσιο της ίδιας αυτοματοποιημένης ροής επεξεργασίας. Σε κάθε νέο dataset, το στάδιο αυτό εκτελείται εκ νέου και παρέχει εξειδικευμένη περιγραφή των τάσεων ομοφιλίας του εκάστοτε δικτύου.

5.3 Ο αλγόριθμος σύστασης ομάδων

Σχεδιάστηκε και υλοποιήθηκε ένας επαναληπτικός αλγόριθμος σύστασης ομάδων, ο οποίος ακολουθεί greedy στρατηγική κατανομής.

Το πρόβλημα του σχηματισμού ομάδων είναι εκ φύσεως συνδυαστικό, καθώς ο αριθμός των πιθανών κατανομών αυξάνεται εκθετικά με το πλήθος των φοιτητών. Για τον λόγο αυτό, η εξαντλητική αναζήτηση της βέλτιστης λύσης καθίσταται υπολογιστικά μη ρεαλιστική για πραγματικά εκπαιδευτικά περιβάλλοντα. Η επιλογή μίας greedy προσέγγισης επιτρέπει τη λήψη διαδοχικών αποφάσεων τοποθέτησης με βάση σαφή και ερμηνεύσιμα κριτήρια, διατηρώντας παράλληλα υψηλή υπολογιστική αποδοτικότητα (Cormen et al., 2009).

Ο αλγόριθμος στηρίζεται σε δύο αρχές, οι οποίες εμπλουτίζονται από τα αποτελέσματα της ανάλυσης ομοφιλίας. Η πρώτη είναι η μέγιστη δυνατή διασπορά των ρόλων κοινωνικής αλληλεπίδρασης στις ομάδες, ώστε να αποφεύγεται η συγκέντρωση των κοινωνικά ενεργών φοιτητών σε λίγες ομάδες. Η δεύτερη είναι η διατήρηση ισορροπίας ως προς τα επίπεδα του πολυδιάστατου προφίλ (`strata`), έτσι ώστε να περιορίζεται η δημιουργία ομάδων με υπερβολικά ομοιογενείς ή άνισο μαθησιακό προφίλ. Για τον σκοπό αυτό, ο αλγόριθμος συνδυάζει πληροφορία από τον ρόλο κάθε φοιτητή (`network_role_el`), από το επίπεδο στρωματοποίησης (`profile_stratum`), καθώς και από τους αριθμητικούς δείκτες του πο-

λυδιάστατου προφίλ: τον συνολικό τυποποιημένο δείκτη προφίλ (`profile_score_z`), τον δείκτη κοινωνικού ρόλου (`social_role_score`), τον δείκτη επίδοσης (`performance_score`), τον δείκτη συμμετοχής (`engagement_score`) και τον δείκτη απογοήτευσης (`frustration_score`).

Επιπλέον, ο αλγόριθμος ενσωματώνει τα αποτελέσματα της ανάλυσης ομοφιλίας (Ενότητα 5.2) ως μηχανισμό προσαρμογής της παρέμβασης στις ιδιαιτερότητες του εκάστοτε δικτύου. Όταν σε μια διάσταση του προφίλ ανιχνεύεται στατιστικά σημαντική τάση ομοφιλίας (όπως συμβαίνει με το `profile_stratum` στο εξεταζόμενο dataset), ο αλγόριθμος ενεργοποιεί αυξημένες ποινές υπερσυγκέντρωσης. Με τον τρόπο αυτό, η παρέμβαση εντείνεται ακριβώς εκεί όπου η φυσική τάση του δικτύου οδηγεί σε ομοιογένεια, περιορίζοντας την πιθανότητα αναπαραγωγής των υφιστάμενων ομοιογενών δομών.

Η λειτουργία του αλγορίθμου οργανώνεται σε τρία βασικά στάδια, τα οποία συνοψίζονται στον παρακάτω πίνακα (Πίνακας 5-3).

Στάδιο	Περιγραφή	Έξοδος
1	Υπολογισμός αριθμού και μεγέθους ομάδων	Πλήθος ομάδων και χωρητικότητα
2	Υπολογισμός στόχων στρωματοποίησης	Στόχοι κατανομής ανά στρώμα
3	Ιεραρχική άπληστη (greedy) τοποθέτηση ανά ρόλο	Τελικές ομάδες

Πίνακας 5-3 Βασικά στάδια αλγορίθμου ομαδοποίησης

5.3.1 Βασική λογική του αλγορίθμου

Ο αλγόριθμος σχεδιάστηκε με γνώμονα την παιδαγωγική λειτουργικότητα των ομάδων και όχι απλώς τη μαθηματική εξισορρόπηση μέσων όρων. Το πρωταρχικό μέλημα είναι η διασπορά των φοιτητών που εμφανίζουν ενεργό ρόλο (Ηγέτης, Γέφυρα, Δραστήριος Συμμετέχων) στο περιβάλλον μάθησης, ώστε να μεγιστοποιηθεί η πιθανότητα κάθε ομάδα να διαθέτει τουλάχιστον ένα μέλος που μπορεί να λειτουργήσει ως πυρήνας ενεργοποίησης της συνεργασίας.

Η προσέγγιση αυτή διαφοροποιείται ουσιαστικά από μία απλή κατανομή των φοιτητών κατά φθίνουσα τιμή ενός συνολικού δείκτη επίδοσης ή συμμετοχής. Στην πράξη, η κατανομή των ρόλων στο dataset δεν είναι ισομερής, ορισμένοι ρόλοι, όπως αυτός του Ηγέτη και της Γέφυρας, είναι σχετικά σπάνιοι.

Επομένως, η δημιουργία ισορροπημένων ομάδων απαιτεί πρώτα τη διασπορά των σπανιότερων και πιο κρίσιμων ρόλων και στη συνέχεια τη συμπλήρωση των ομάδων με τα υπόλοιπα μέλη.

Κριτήριο επιτυχίας του αλγορίθμου δεν είναι η πλήρης συμμετρική αναπαραγωγή όλων των ρόλων σε κάθε ομάδα —κάτι που θα ήταν αδύνατο λόγω της φυσικής κατανομής τους— αλλά η μέγιστη δυνατή διασπορά των σπανιότερων ρόλων και η αποφυγή δημιουργίας ομάδων αποτελούμενων αποκλειστικά από φοιτητές με Ανεπαρκή Αλληλεπίδραση.

5.3.2 Είσοδος δεδομένων του αλγορίθμου

Ο αλγόριθμος εφαρμόζεται στο τελικό επεξεργασμένο dataset που έχει ήδη παραχθεί (Κεφάλαιο 4).

Ως ελάχιστες απαιτούμενες μεταβλητές θεωρούνται οι στήλες:

- *user_id*: για την ταυτοποίηση κάθε φοιτητή
- *network_role_el*: για την κατηγοριοποίηση ως προς τον ρόλο στο δίκτυο (Ηγέτης, Γέφυρα, Δραστήριος Συμμετέχων, Περιφερειακός Συμμετέχων, Ανεπαρκής Αλληλεπίδραση)
- *profile_stratum*: για την ένταξη στο αντίστοιχο επίπεδο του πολυδιάστατου προφίλ (Πολύ Υψηλό, Υψηλό, Μεσαίο, Χαμηλό, Πολύ Χαμηλό)

Επιπλέον αριθμητικές μεταβλητές — όπως ο συνολικός τυποποιημένος δείκτης προφίλ (*profile_score_z*), ο δείκτης κοινωνικού ρόλου (*social_role_score*), ο δείκτης επίδοσης (*performance_score*), ο δείκτης συμμετοχής (*engagement_score*) και ο δείκτης απογοήτευσης (*frustration_score*)— χρησιμοποιούνται για την αξιολόγηση της καταλληλότητας κάθε τοποθέτησης. Στη διαδικασία αυτή αξιοποιούνται επίσης τα αποτελέσματα της ανάλυσης ομοφιλίας (Ενότητα 5.2), τα οποία προσδιορίζουν τις διαστάσεις του προφίλ (*profile_stratum*, *network_role_el* ή επιμέρους αριθμητικοί δείκτες) όπου η ομοφιλία είναι στατιστικά σημαντική. Σε περιπτώσεις όπου κάποια από αυτές τις μεταβλητές δεν είναι διαθέσιμη στο dataset, η υλοποίηση του αλγορίθμου επιτρέπει την αντικατάσταση των ελλειπόν ή μη έγκυρων αριθμητικών τιμών με ουδέτερη τιμή (μηδέν), διατηρώντας έτσι τη δυνατότητα εφαρμογής του σε διαφορετικά σύνολα δεδομένων Moodle.

5.3.3 Υπολογισμός πλήθους και μεγέθους ομάδων

Το μέγεθος των ομάδων αποτελεί παραμετροποιήσιμη είσοδο, επιτρέποντας στον εκπαιδευτικό να προσαρμόζει τη σύσταση ανάλογα με τις ανάγκες της συνεργατικής δραστηριότητας. Στην παρούσα εφαρμογή εξετάζονται ενδεικτικά δύο σενάρια, με ομάδες των 5 και των 7 φοιτητών, ενώ η προεπιλεγμένη τιμή της υλοποίησης είναι το μέγεθος 5 σε περίπτωση που δεν εισαχθεί κάποια διαφορετική.

Καθώς το πλήθος των φοιτητών δεν είναι πάντα ακριβές πολλαπλάσιο του επιλεγμένου μεγέθους, επιτρέπεται μικρή απόκλιση: οι πλεονάζουσες θέσεις αφαιρούνται σταδιακά από τις τελευταίες ομάδες. Έτσι, οι ομάδες διαφέρουν το πολύ κατά ένα μέλος, παραμένοντας όσο το δυνατόν πιο ισομεγέθεις.

5.3.4 Υπολογισμός στόχων στρωματοποίησης

Πριν από τη διαδικασία τοποθέτησης, ο αλγόριθμος υπολογίζει στόχους κατανομής για κάθε επίπεδο του πολυδιάστατου προφίλ (`profile_stratum`). Οι στόχοι αυτοί δεν αποτελούν ξεχωριστή φάση ανάθεσης φοιτητών, αλλά χρησιμοποιούνται στη συνάρτηση κόστους κατά τη greedy τοποθέτηση, ώστε να αποθαρρύνεται η υπερσυγκέντρωση φοιτητών του ίδιου `stratum` στην ίδια ομάδα.

Για κάθε επίπεδο στρωματοποίησης υπολογίζεται η αναλογική παρουσία του στο συνολικό πληθυσμό και προσδιορίζεται αντίστοιχος στόχος για κάθε ομάδα.

Η βασική τιμή του στόχου προκύπτει από την ακέραιη διαίρεση του πλήθους των φοιτητών κάθε στρώματος με τον αριθμό των ομάδων. Όταν προκύπτει υπόλοιπο, οι επιπλέον θέσεις κατανέμονται διαδοχικά στις πρώτες ομάδες. Για παράδειγμα, αν υπάρχουν 10 φοιτητές υψηλού επιπέδου και 8 ομάδες, τότε 2 ομάδες θα λάβουν 2 φοιτητές από το συγκεκριμένο στρώμα και οι υπόλοιπες 6 από 1. Με τον τρόπο αυτό επιδιώκεται όσο το δυνατόν πιο ομοιόμορφη κατανομή των επιπέδων του προφίλ, αποφεύγοντας τη δημιουργία ομάδων με υπερσυγκέντρωση φοιτητών υψηλού ή χαμηλού επιπέδου.

5.3.5 Ιεραρχική greedy τοποθέτηση βάσει ρόλων

Αφού υπολογιστούν οι χωρητικότητες των ομάδων και οι στόχοι στρωματοποίησης, ο αλγόριθμος προχωρά στην ιεραρχική, ρόλο-ανά-ρόλο greedy τοποθέτηση των φοιτητών. Οι φοιτητές είναι ήδη ταξινομημένοι στις πέντε κατηγορίες ρόλων: Ηγέτης, Γέφυρα, Δραστήριος Συμμετέχων, Περιφερειακός Συμμετέχων και Ανεπαρκής Αλληλεπίδραση. Ο σχηματισμός των ομάδων ξεκινά από τους Ηγέτες, συνεχίζεται με τις Γέφυρες, έπειτα με τους Δραστήριους Συμμετέχοντες, με τους Περιφερειακούς Συμμετέχοντες και ολοκληρώνεται με την τοποθέτηση των φοιτητών με Ανεπαρκή Αλληλεπίδραση.

Πριν από κάθε φάση τοποθέτησης, εφαρμόζεται ταξινόμηση προτεραιότητας εντός της αντίστοιχης κατηγορίας. Οι φοιτητές με υψηλότερη τιμή ενός συνολικού δείκτη ισχύος (`_strength`) τοποθετούνται πρώτοι, ώστε οι φοιτητές με μεγαλύτερη επίδραση στη σύνθεση των ομάδων να κατανεμηθούν σε κατάλληλες διαθέσιμες ομάδες, επιτρέποντας ακριβέστερη αριθμητική εξισορρόπηση.

Για τους φοιτητές της κατηγορίας Ανεπαρκής Αλληλεπίδραση, η ταξινόμηση γίνεται με διαφορετικά κριτήρια: προτεραιότητα δίνεται σε όσους εμφανίζουν υψηλότερο δείκτη απογοήτευσης

(frustration_score), έτσι ώστε οι φοιτητές με υψηλότερη ένδειξη απογοήτευσης να έχουν μεγαλύτερη πιθανότητα να ενταχθούν σε ομάδες που διαθέτουν ήδη ενεργό πυρήνα συνεργασίας. Η ακολουθία των φάσεων και τα κριτήρια ταξινόμησης κάθε κατηγορίας απεικονίζονται στην αριστερή στήλη του Διαγράμματος 5.1.

5.3.6 Συνάρτηση κόστους: σκληροί περιορισμοί και ποινές

Για κάθε φοιτητή που πρόκειται να τοποθετηθεί σε ομάδα, ο αλγόριθμος εξετάζει όλες τις διαθέσιμες ομάδες και επιλέγει εκείνη που οδηγεί στη μικρότερη συνολική επιβάρυνση. Η αξιολόγηση οργανώνεται σε τρία επίπεδα. Πρώτον, ελέγχεται ο σκληρός περιορισμός χωρητικότητας (hard constraint), ο οποίος αποκλείει την τοποθέτηση σε πλήρη ομάδα. Δεύτερον, εφαρμόζονται **κανόνες λεξικογραφικής προτεραιότητας** που επιδιώκουν τη μέγιστη δυνατή κάλυψη των ομάδων με κοινωνικά ενεργά μέλη. Οι κανόνες αυτοί υλοποιούνται με ποινές κατά πολλές τάξεις μεγέθους μεγαλύτερες από τις υπόλοιπες, χωρίς όμως να αποκλείουν απολύτως μια τοποθέτηση. Τρίτον, εφαρμόζονται **ήπιες ποινές** (soft penalties) για τη συγκριτική αξιολόγηση των διαθέσιμων ομάδων ως προς την ισορροπία μεγέθους, στρωμάτων, κοινωνικών ρόλων, αριθμητικών χαρακτηριστικών και ομοφιλίας. Χαμηλότερη συνολική βαθμολογία σημαίνει καταλληλότερη ομάδα.

Οι βασικές συνιστώσες κόστους είναι:

- **Ποινή μεγέθους (size_penalty):** όταν η ομάδα είναι ήδη πλήρης, η ποινή λαμβάνει απαγορευτική τιμή (10^{12}), λειτουργώντας στην πράξη ως **σκληρός περιορισμός** (hard constraint): καμία πλήρης ομάδα δεν μπορεί να δεχτεί επιπλέον φοιτητή. Για τις μη πλήρεις ομάδες, η ποινή μεγέθους αυξάνεται αναλογικά όσο το τρέχον μέγεθος της ομάδας πλησιάζει τη χωρητικότητά της:

$$size_penalty(g) = \frac{size_g}{capacity_g}$$

- **Ποινή στρώματος (stratum penalty):** μηδενική όσο ο τρέχων αριθμός μελών του συγκεκριμένου στρώματος βρίσκεται κάτω από τον στόχο της ομάδας. Όταν η υποψήφια τοποθέτηση οδηγεί σε υπέρβαση του στόχου, η ποινή αυξάνεται γραμμικά:

$$stratum_penalty(g,s)=2.5+(current_{g,s}-target_{g,s}+1)\times 3.5$$

Όταν το profile_stratum έχει ανιχνευθεί ως στατιστικά σημαντική διάσταση ομοφιλίας, η ποινή αυτή πολλαπλασιάζεται επί 2.

- **Ποινή προβαλλόμενης ισορροπίας (projected balance penalty):** μετρά πόσο θα αποκλίνουν οι μέσοι όροι της ομάδας από τους καθολικούς μέσους όρους του πληθυσμού μετά την υποψήφια τοποθέτηση του φοιτητή. Σε αντίθεση με μια εναλλακτική mean-fit λογική (mean_fit_penalty), η οποία μετρά την απόσταση του φοιτητή από τους τρέχοντες μέσους όρους της ομάδας και

μπορεί να ευνοήσει την εσωτερική ομοιογένεια, η προβαλλόμενη ισορροπία εστιάζει στη διατήρηση συγκρίσιμων μέσων όρων μεταξύ των ομάδων:

$$balance_penalty(g, i) = \sum_c w_c \cdot \left| \frac{S_{g,c} + x_{i,c}}{size_g + 1} - \mu_c \right|$$

Όπου c είναι τα αριθμητικά χαρακτηριστικά (profile_score_z, social_role_score, performance_score, engagement_score, frustration_score), w_c τα αντίστοιχα βάρη (1,0 για τα περισσότερα, 0,6 για το social_role_score), $S_{\{g,c\}}$ το άθροισμα τιμών της ομάδας, $x_{\{i,c\}}$ η τιμή του φοιτητή και μ_c ο καθολικός μέσος όρος. Στόχος είναι η ισορροπία μεταξύ ομάδων, όχι η εσωτερική ομοιογένεια.

- **Προσαρμοστική ποινή ομοφιλίας για αριθμητικά χαρακτηριστικά (numeric homophily penalty):** όταν σε μια αριθμητική διάσταση έχει ανιχνευθεί στατιστικά σημαντική ομοφιλία, ενεργοποιείται πρόσθετη ποινή που αποθαρρύνει την τοποθέτηση φοιτητή σε ομάδα όπου ο μέσος όρος του χαρακτηριστικού είναι ήδη κοντά στην τιμή του. Στο εξεταζόμενο dataset δεν ενεργοποιείται καμία αριθμητική ποινή ομοφιλίας, αλλά ο μηχανισμός παραμένει διαθέσιμος για μελλοντικές εφαρμογές.
- **Ποινή ρόλων (role duplicate penalty):** αποτρέπει τη συγκέντρωση πολλαπλών φοιτητών του ίδιου ρόλου στην ίδια ομάδα. Οι ποινές είναι ιεραρχικά διαβαθμισμένες, λαμβάνοντας υπόψη τη σχετική σπανιότητα και την παιδαγωγική προτεραιότητα κάθε ρόλου: 120.000 για δεύτερο Ηγέτη, 100.000 για δεύτερη Γέφυρα, 80.000 για δεύτερο Δραστήριο Συμμετέχοντα, 800 για δεύτερο Περιφερειακό Συμμετέχοντα και 2.500 ανά πρόσθετο μέλος Ανεπαρκούς Αλληλεπίδρασης. Οι τιμές αυτές δεν αποτελούν εμπειρικά εκτιμημένες παραμέτρους, αλλά ιεραρχικά βάρη προτεραιότητας, τα οποία αποσκοπούν στη διασπορά των σπανιότερων ρόλων σε όσο το δυνατόν περισσότερες ομάδες. Το κρίσιμο στοιχείο δεν είναι η απόλυτη αριθμητική τιμή κάθε ποινής, αλλά η σχετική τάξη μεγέθους μεταξύ των κατηγοριών: οι ποινές των σπανιότερων ρόλων πρέπει να υπερσχύουν των δευτερευουσών ποινών ισορροπίας, χωρίς να αποκλείουν απολύτως την επανάληψη ενός ρόλου όταν η διαθέσιμη κατανομή των φοιτητών δεν επιτρέπει την πλήρη διασπορά του. Οι αναφερόμενες τιμές αντιστοιχούν στην προσθήκη δεύτερου μέλους του ίδιου ρόλου. Για κάθε επιπλέον επανάληψη, η ποινή αυξάνεται ανάλογα με τον αριθμό των μελών του συγκεκριμένου ρόλου που υπάρχουν ήδη στην ομάδα. Για παράδειγμα, η προσθήκη δεύτερου Ηγέτη επιβαρύνεται με 120.000, ενώ η προσθήκη τρίτου Ηγέτη με 240.000.
- **Κανόνας λεξικογραφικής κάλυψης (coverage priority):** πριν από οποιαδήποτε δευτερεύουσα βελτιστοποίηση, ο αλγόριθμος εφαρμόζει δύο πρακτικά σκληρούς κανόνες λεξικογραφικής προτεραιότητας: (α) αν υπάρχουν ομάδες χωρίς ενεργό πυρήνα, αποθαρρύνεται πρακτικά μέσω πολύ μεγάλης ποινής (10^{10}), η τοποθέτηση φοιτητή ενεργού πυρήνα σε ομάδα που ήδη διαθέτει

ενεργό πυρήνα. Με τον τρόπο αυτό, οι σπάνιοι ρόλοι διασπείρονται στις ακάλυπτες ομάδες πριν δημιουργηθούν ομάδες με περισσότερα από ένα μέλη ενεργού πυρήνα, (β) αν υπάρχουν ομάδες χωρίς κανένα μέλος επαρκούς κοινωνικής αλληλεπίδρασης, αποθαρρύνεται πρακτικά μέσω πολύ μεγάλης ποινής (5×10^9) η τοποθέτηση φοιτητή που δεν ανήκει στην κατηγορία Ανεπαρκής Αλληλεπίδραση σε ομάδα που ήδη διαθέτει μέλος επαρκούς κοινωνικής αλληλεπίδρασης ελέγχοντας την τρέχουσα σύνθεση κάθε ομάδας τη στιγμή της τοποθέτησης.

Ιδιαίτερη αντιμετώπιση εφαρμόζεται στην κατηγορία Ανεπαρκής Αλληλεπίδραση: οι φοιτητές αυτοί τοποθετούνται τελευταίοι, αφού έχει προηγηθεί η διασπορά των υπόλοιπων ρόλων, και αξιολογούνται με πρόσθετες ποινές στη συνάρτηση κόστους. Συγκεκριμένα, αποθαρρύνεται η τοποθέτησή τους σε ομάδες χωρίς ενεργό πυρήνα (ποινή 5.000) ή χωρίς μέλος επαρκούς κοινωνικής αλληλεπίδρασης (ποινή 80.000), ενώ παράλληλα αποθαρρύνεται η υπερβολική συγκέντρωση πολλών φοιτητών της ίδιας κατηγορίας στην ίδια ομάδα. Η λογική αυτής της επιλογής είναι να αυξηθεί η πιθανότητα ένταξής τους σε ομάδες όπου υπάρχει ήδη κοινωνική δραστηριότητα, χωρίς να δημιουργούνται ομάδες αποτελούμενες αποκλειστικά από φοιτητές με περιορισμένη ή μηδενική αλληλεπίδραση.

Στις μη πλήρεις ομάδες, οι παραπάνω συνιστώσες συνδυάζονται στη βασική συνάρτηση κόστους, η οποία αποτιμά το συνολικό κόστος της υποψήφιας τοποθέτησης του φοιτητή i στην ομάδα g :

$$C_{\text{base}}(i, g) = C_{\text{coverage}}(i, g) + 10C_{\text{size}}(g) + 25C_{\text{stratum}}(i, g) + 8C_{\text{balance}}(i, g) + 10C_{\text{num_hom}}(i, g) + C_{\text{role}}(i, g).$$

όπου C_{coverage} αντιστοιχεί στους κανόνες λεξικογραφικής κάλυψης, C_{size} στην ποινή μεγέθους, C_{stratum} στην ποινή στρώματος, C_{balance} στην ποινή προβαλλόμενης ισορροπίας, $C_{\text{num_hom}}$ στην προσαρμοστική ποινή αριθμητικής ομοφιλίας και C_{role} στην ποινή επανάληψης ρόλου. Οι συντελεστές 10, 25, 8 και 10 δεν αποτελούν εμπειρικά εκτιμημένες παραμέτρους, αλλά προκαθορισμένα βάρη σχεδιασμού για τη σχετική κλιμάκωση των επιμέρους συνιστωσών, ώστε να διατηρείται η επιθυμητή ιεράρχηση των κριτηρίων: οι κανόνες λεξικογραφικής κάλυψης υπερσχύουν των δευτερευουσών ποινών ισορροπίας. Στη βασική συνάρτηση προστίθενται, ανάλογα με τη φάση τοποθέτησης, οι ειδικές ποινές που αντιστοιχούν στην εκάστοτε κατηγορία κοινωνικού ρόλου, όπως ορίζονται στην υλοποίηση.

Οι παράμετροι της συνάρτησης κόστους συνοψίζονται στον Πίνακα 5-4

Συνιστώσα	Τιμή
-----------	------

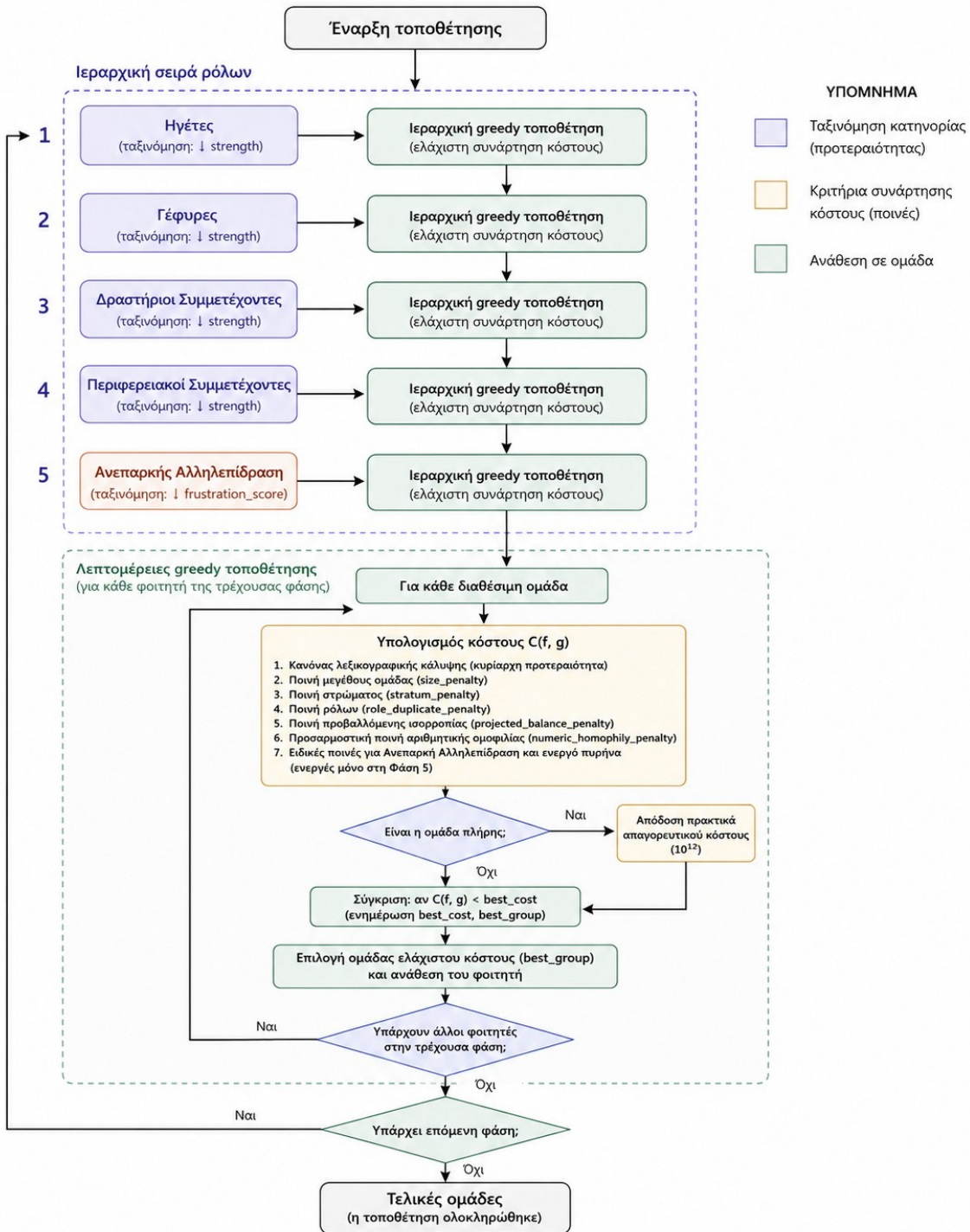
Ποινή πλήρους ομάδας	10^{12}
Ποινή δεύτερου Ηγέτη	120.000
Ποινή δεύτερης Γέφυρας	100.000
Ποινή δεύτερου Δραστήριου Συμμετέχοντα	80.000
Ποινή δεύτερου Περιφερειακού Συμμετέχοντα	800
Γενική ποινή επανάληψης Ανεπαρκούς Αλληλεπίδρασης	2.500
Ειδική ποινή συσσώρευσης Ανεπαρκούς Αλληλεπίδρασης	3.000
Ποινή ομάδας χωρίς μέλος επαρκούς αλληλεπίδρασης	80.000
Ποινή ομάδας χωρίς ενεργό πυρήνα (για φοιτητή Ανεπαρκούς Αλληλεπίδρασης)	5.000
Κάλυψη χωρίς ενεργό πυρήνα	10^{10}
Κάλυψη ομάδων χωρίς μέλος επαρκούς αλληλεπίδρασης	5×10^9
Μείωση όταν υπάρχει ενεργός πυρήνας	-100
Βάρος profile_score_z	1,0
Βάρος performance_score	1,0
Βάρος engagement_score	1,0
Βάρος frustration_score	1,0
Βάρος social_role_score	0,6
Συντελεστής ποινής μεγέθους	10
Συντελεστής ποινής στρώματος	25
Συντελεστής προβαλλόμενης ισορροπίας	8
Συντελεστής αριθμητικής ομοφιλίας	10

Πίνακας 5-4 Σύνοψη παραμέτρων της συνάρτησης κόστους

Σημείωση: Η τιμή 10^{12} υλοποιεί τον σκληρό περιορισμό χωρητικότητας και αποκλείει την τοποθέτηση σε πλήρη ομάδα. Οι τιμές 10^{10} και 5×10^9 λειτουργούν ως πρακτικά σκληροί περιορισμοί, επιβάλλοντας κυρίαρχη προτεραιότητα στους κανόνες λεξικογραφικής κάλυψης και καθιστώντας την αντίστοιχη τοποθέτηση πρακτικά απαγορευτική, εκτός εάν η επιθυμητή κάλυψη δεν μπορεί να

επιτευχθεί διαφορετικά. Οι υπόλοιπες, χαμηλότερες τιμές αποτελούν ήπιες ποινές, οι οποίες επιτρέπουν τη συγκριτική αξιολόγηση μεταξύ εναλλακτικών τοποθετήσεων. Οι τιμές προϋποθέτουν ότι οι αριθμητικοί δείκτες του προφίλ παραμένουν τυποποιημένοι, όπως ορίστηκαν στο Κεφάλαιο 4.

Η συνολική λογική του αλγορίθμου, από την ιεραρχική σειρά των φάσεων έως την αναλυτική αξιολόγηση κάθε υποψήφιας τοποθέτησης μέσω της συνάρτησης κόστους, απεικονίζεται συνοπτικά στο Σχήμα 5-1.



Σχήμα 5-1 Συνοπτική αναπαράσταση του αλγορίθμου ιεραρχικής greedy τοποθέτησης

5.4 Πειραματική εφαρμογή και αξιολόγηση

Για την αξιολόγηση της αποτελεσματικότητας του προτεινόμενου αλγορίθμου σύστασης ομάδων σχεδιάστηκε και υλοποιήθηκε μια πειραματική διαδικασία συγκριτικής αξιολόγησης. Στόχος της διαδικασίας ήταν να εξεταστεί κατά πόσο ο αλγόριθμος της Ενότητας 5.3 επιτυγχάνει τους παιδαγωγικούς στόχους που τέθηκαν κατά τον σχεδιασμό του, σε σύγκριση με απλούστερες και ευρέως χρησιμοποιούμενες στρατηγικές ομαδοποίησης.

5.4.1 Δεδομένα και παραμετροποίηση

Η πειραματική εφαρμογή πραγματοποιήθηκε στο σύνολο των 383 φοιτητών, για τους οποίους ήταν διαθέσιμο το πολυδιάστατο προφίλ που προέκυψε από τα προηγούμενα στάδια της ανάλυσης.

Για να διερευνηθεί η συμπεριφορά του αλγορίθμου υπό διαφορετικές συνθήκες συνεργατικής οργάνωσης, η διαδικασία εφαρμόστηκε για δύο επιθυμητά μεγέθη ομάδας: ομάδες μεγέθους 5 ατόμων, από τις οποίες προέκυψαν 77 ομάδες (75 πενταμελείς και 2 τετραμελείς), και ομάδες μεγέθους 7 ατόμων, από τις οποίες προέκυψαν 55 ομάδες (53 επταμελείς και 2 εξαμελείς). Η μικρή απόκλιση από το ακριβές πολλαπλάσιο οφείλεται στον μηχανισμό διαχείρισης overflow που περιγράφηκε στην Ενότητα 5.3.3.

Η επιλογή των δύο μεγεθών βασίζεται σε ευρήματα της βιβλιογραφίας για τη συνεργατική μάθηση: οι ομάδες 4–6 ατόμων θεωρούνται βέλτιστες για εντατική συνεργασία, ενώ οι ομάδες 6–8 ατόμων επιτρέπουν μεγαλύτερη ποικιλότητα ρόλων αλλά απαιτούν καλύτερο συντονισμό (Butera & Buchs, 2019). Η σύγκριση των δύο σεναρίων μεγέθους επιτρέπει, επιπλέον, την ανάλυση της κλιμάκωσης του αλγορίθμου και της επίδρασης του μεγέθους στην ποιότητα των αποτελεσμάτων.

5.4.2 Σενάρια σύγκρισης

Για κάθε μέγεθος ομάδας, η πειραματική διαδικασία συνέκρινε τρία διαφορετικά σενάρια κατανομής των φοιτητών:

Σενάριο A: Τυχαίος σχηματισμός (Random Baseline). Οι φοιτητές ανακατεύονται τυχαία και κατανέμονται διαδοχικά στις ομάδες, χωρίς να λαμβάνεται υπόψη κανένα χαρακτηριστικό τους. Η τυχαία κατανομή αποτελεί το πλέον διαδεδομένο σημείο αναφοράς στη βιβλιογραφία του αυτοματοποιημένου σχηματισμού ομάδων, αντιπροσωπεύοντας την «απουσία στρατηγικής». Για λόγους αναπαραγωγιμότητας χρησιμοποιήθηκε σταθερή τιμή σπόρου τυχαίας ακολουθίας (random seed = 42) έτσι ώστε να διασφαλιστεί ότι οποιαδήποτε επανεκτέλεση του πειράματος με την ίδια τιμή seed θα παράγει ακριβώς την ίδια κατανομή φοιτητών, επιτρέποντας τον έλεγχο και την επαλήθευση των αποτελεσμάτων.

Σενάριο Β: Σχηματισμός βάσει επίδοσης (Performance-Only Baseline). Οι φοιτητές ταξινομούνται κατά φθίνουσα τιμή του δείκτη ακαδημαϊκής επίδοσης (*performance_score*) και στη συνέχεια τοποθετούνται διαδοχικά στις ομάδες. Η προσέγγιση αυτή αντιστοιχεί σε μονοκριτηριακό σχηματισμό βάσει επίδοσης και χρησιμοποιείται ως σημείο αναφοράς για την αξιολόγηση της περίπτωσης κατά την οποία η κατανομή βασίζεται αποκλειστικά σε ένα κριτήριο, χωρίς να λαμβάνονται υπόψη οι υπόλοιπες διαστάσεις του μαθησιακού προφίλ.

Σενάριο Γ: Προτεινόμενη μέθοδος (Proposed). Εφαρμόζεται ο αλγόριθμος που αναλύθηκε στην Ενότητα 5.3.

5.4.3 Μετρικές αξιολόγησης

Για την αντικειμενική σύγκριση των τριών σεναρίων, ορίστηκε ένα κοινό σύνολο μετρικών που καλύπτει δύο θεμελιώδεις διαστάσεις. Η πρώτη διάσταση αφορά την παιδαγωγική λειτουργικότητα των ομάδων, δηλαδή την ικανότητά τους να λειτουργήσουν ως συνεργατικές μονάδες με εσωτερική ετερογένεια και ενεργό πυρήνα. Η δεύτερη διάσταση αφορά τη δομική ισορροπία μεταξύ των ομάδων, δηλαδή τον βαθμό συγκρισιμότητάς τους ως προς βασικά χαρακτηριστικά του πληθυσμού.

Μετρικές παιδαγωγικής λειτουργικότητας. Οι ακόλουθες μετρικές αποτιμούν την παρουσία και κατανομή των κοινωνικών ρόλων εντός των ομάδων:

- *groups_with_leader*: αριθμός ομάδων με τουλάχιστον έναν Ηγέτη.
- *groups_with_bridge*: αριθμός ομάδων με τουλάχιστον μία Γέφυρα.
- *groups_with_active*: αριθμός ομάδων με τουλάχιστον έναν Δραστήριο Συμμετέχοντα.
- *groups_with_peripheral*: αριθμός ομάδων με τουλάχιστον έναν Περιφερειακό Συμμετέχοντα.
- *groups_with_active_core*: αριθμός ομάδων με ενεργό πυρήνα (Ηγέτη, Γέφυρα ή Δραστήριο Συμμετέχοντα).
- *groups_with_non_insufficient*: αριθμός ομάδων με τουλάχιστον ένα μέλος που δεν ανήκει στην κατηγορία Ανεπαρκής Αλληλεπίδραση.
- *mean_role_diversity*: μέση ποικιλία ρόλων ανά ομάδα.
- *max_insufficient_interaction_in_group*: μέγιστος αριθμός φοιτητών με Ανεπαρκή Αλληλεπίδραση σε μία ομάδα.

Στις πρώτες επτά μετρικές, μεγαλύτερη τιμή υποδηλώνει ευρύτερη κάλυψη ή μεγαλύτερη ποικιλία ρόλων. Αντίθετα, στη μετρική *max_insufficient_interaction_in_group*, μικρότερη τιμή θεωρείται προτιμότερη.

Μετρικές δομικής ισορροπίας. Οι ακόλουθες μετρικές αποτιμούν την ισορροπία και τη συγκρισιμότητα μεταξύ των ομάδων. Σε όλες, μικρότερη τιμή υποδηλώνει μικρότερες διαφορές μεταξύ των ομάδων και, επομένως, καλύτερη συνολική ισορροπία.

- *mean_strata_deviation*: μέση απόκλιση της κατανομής στρωμάτων κάθε ομάδας από την καθολική κατανομή του πληθυσμού.
- *max_strata_deviation*: μέγιστη απόκλιση μιας ομάδας από την καθολική κατανομή στρωμάτων.
- *profile_mean_std*: τυπική απόκλιση του μέσου όρου του συνολικού προφίλ (*profile_score_z*) μεταξύ των ομάδων.
- *social_role_mean_std*: τυπική απόκλιση του μέσου *social_role_score* μεταξύ των ομάδων.
- *performance_mean_std*: τυπική απόκλιση του μέσου όρου ακαδημαϊκής επίδοσης μεταξύ των ομάδων.
- *engagement_mean_std*: τυπική απόκλιση της μέσης συμμετοχής μεταξύ των ομάδων.
- *frustration_mean_std*: τυπική απόκλιση της μέσης απογοήτευσης μεταξύ των ομάδων.

5.4.4 Αποτελέσματα για ομάδες 5 ατόμων

Στην πρώτη παραμετροποίηση, με στόχο τη δημιουργία ομάδων των 5 ατόμων, προέκυψαν 77 ομάδες. Ο παρακάτω πίνακας (Πίνακας 5-5) παρουσιάζει συγκεντρωτικά τα αποτελέσματα για τις τρεις υπό σύγκριση μεθόδους, με έντονη γραφή να υποδηλώνει την καλύτερη τιμή ανά γραμμή.

Μετρική	Τυχαίος	Επίδοση-Μόνο	Προτεινόμενος
groups_with_leader	16	16	18
groups_with_bridge	7	7	7
groups_with_active	24	25	28
groups_with_peripheral	42	43	54
groups_with_active_core	41	42	53
groups_with_non_insufficient	64	62	77
mean_role_diversity	2,1558	2,1818	2,3896
max_insufficient_interaction_in_group	5	5	4
mean_strata_deviation	0,7663	0,9145	0,4936
max_strata_deviation	1,7389	1,7389	0,8721

profile_mean_std	0,4325	0,6603	0,1985
social_role_mean_std	0,4355	0,4800	0,2458
performance_mean_std	0,3707	0,8389	0,2183
engagement_mean_std	0,2481	0,3055	0,1327
frustration_mean_std	0,4133	0,4666	0,4155

Πίνακας 5-5 Συγκριτικά αποτελέσματα για ομάδες 5 ατόμων

Τα αποτελέσματα αναδεικνύουν σαφή υπεροχή της προτεινόμενης μεθόδου στο σύνολο σχεδόν των εξεταζόμενων μετρικών. Ως προς την παιδαγωγική λειτουργικότητα, η προτεινόμενη μέθοδος είναι η μόνη που εξασφαλίζει ότι **όλες** οι ομάδες περιλαμβάνουν τουλάχιστον ένα μέλος που δεν ανήκει στην κατηγορία Ανεπαρκής Αλληλεπίδραση, επιτυγχάνοντας 77 ομάδες έναντι 62–64 των άλλων σεναρίων. Η διαφορά είναι ακόμη εντονότερη στην ύπαρξη ενεργού πυρήνα (Ηγέτη, Γέφυρα ή Δραστήριο Συμμετέχοντα): η προτεινόμενη μέθοδος επιτυγχάνει 53 ομάδες (68,8%) έναντι 41 (53,2%) του τυχαίου και 42 (54,5%) του performance-only. Αξίζει να σημειωθεί ότι η προτεινόμενη μέθοδος επιτυγχάνει τον **μαθηματικά μέγιστο δυνατό αριθμό ομάδων με ενεργό πυρήνα**, δεδομένου ότι στο σύνολο των 383 φοιτητών υπάρχουν μόνο 53 φοιτητές με ρόλο ενεργού πυρήνα (18 Ηγέτες + 7 Γέφυρες + 28 Δραστήριοι). Ιδιαίτερη σημασία έχει ότι η προτεινόμενη μέθοδος εξαλείφει πλήρως τις ομάδες αποτελούμενες αποκλειστικά από φοιτητές με Ανεπαρκή Αλληλεπίδραση — ένα φαινόμενο που εμφανίζεται σε 13 και 15 ομάδες αντίστοιχα στα δύο baseline σενάρια — ενώ παράλληλα περιορίζει το μέγιστο πλήθος τέτοιων φοιτητών ανά ομάδα σε 4.

Ως προς τη δομική ισορροπία, η προτεινόμενη μέθοδος παρουσιάζει σημαντικά χαμηλότερη απόκλιση από την πληθυσμιακή κατανομή στρωμάτων, τόσο στη μέση (mean_strata_deviation = 0,4936) όσο και στη μέγιστη τιμή (max_strata_deviation = 0,8721), έναντι 0,7663 και 1,7389 του τυχαίου σχηματισμού. Υπερτερεί επίσης σημαντικά στην ισορροπία του συνολικού προφίλ (profile_mean_std = 0,1985 έναντι 0,4325 του τυχαίου) και της συμμετοχής (engagement_mean_std = 0,1327 έναντι 0,2481 του τυχαίου). Η προτεινόμενη μέθοδος υπερτερεί επίσης στην ισορροπία της επίδοσης (performance_mean_std =

0,2183 έναντι 0,3707 του τυχαίου), γεγονός που αναδεικνύει την ικανότητά της να εξισορροπεί ταυτόχρονα πολλαπλές διαστάσεις του μαθησιακού προφίλ.

5.4.5 Αποτελέσματα για ομάδες 7 Ατόμων

Στη δεύτερη παραμετροποίηση, με στόχο τη δημιουργία ομάδων των 7 ατόμων, προέκυψαν 55 ομάδες. Ο παρακάτω πίνακας (Πίνακας 5-6) συνοψίζει τα αποτελέσματα.

Μετρική	Τυχαίος σχηματισμός	Επίδοση-Μόνο	Προτεινόμενη μέθοδος
groups_with_leader	16	16	18
groups_with_bridge	6	7	7
groups_with_active	21	21	28
groups_with_peripheral	38	37	54
groups_with_active_core	33	35	53
groups_with_non_insufficient	50	48	55
mean_role_diversity	2,4727	2,4727	2,9455
max_insufficient_interaction_in_group	7	7	6
mean_strata_deviation	0,5578	0,7918	0,1371
max_strata_deviation	1,1175	1,4360	0,5707
profile_mean_std	0,3649	0,6024	0,1057
social_role_mean_std	0,3941	0,3879	0,1442
performance_mean_std	0,2942	0,8363	0,1432
engagement_mean_std	0,2017	0,2464	0,1020
frustration_mean_std	0,3195	0,4005	0,2488

Πίνακας 5-6 Συγκριτικά αποτελέσματα για ομάδες 7 ατόμων

Σημείωση: Με έντονη γραφή επισημαίνεται η καλύτερη τιμή ανά μετρική. Στις μετρικές κάλυψης και ποικιλίας υψηλότερη τιμή θεωρείται καλύτερη, ενώ στις μετρικές απόκλισης και διασποράς χαμηλότερη τιμή θεωρείται καλύτερη.

Η εικόνα για τις επταμελείς ομάδες είναι ακόμη εντονότερα υπέρ της προτεινόμενης μεθόδου. Στην παιδαγωγική λειτουργικότητα, η προτεινόμενη μέθοδος εξασφαλίζει ότι όλες οι ομάδες διαθέτουν τουλάχιστον ένα μέλος εκτός της κατηγορίας «Ανεπαρκής Αλληλεπίδραση», ενώ 53 από τις 55 ομάδες περιλαμβάνουν ενεργό πυρήνα, δηλαδή τουλάχιστον έναν Ηγέτη, μία Γέφυρα ή Δραστήριο Συμμετέχοντα. Το εύρημα αυτό είναι ιδιαίτερα σημαντικό, καθώς σε μεγαλύτερες ομάδες αυξάνεται ο κίνδυνος συγκέντρωσης πολλών φοιτητών με ανεπαρκή αλληλεπίδραση. Η ποικιλία ρόλων είναι η υψηλότερη από όλα τα σενάρια (*mean_role_diversity* = 2,9455), υποδηλώνοντας ότι η προτεινόμενη μέθοδος κατανέμει αποτελεσματικότερα τους διαθέσιμους κοινωνικούς ρόλους μέσα στις ομάδες.

Ως προς τη δομική ισορροπία, η προτεινόμενη μέθοδος εμφανίζει εντυπωσιακά χαμηλότερες τιμές απόκλισης στρωμάτων: *mean_strata_deviation* = 0,1371, έναντι 0,5578 στο τυχαίο σχηματισμό και 0,7918 στο performance-only. Υπερτερεί επίσης στην ισορροπία του συνολικού προφίλ, του κοινωνικού ρόλου, της επίδοσης, της συμμετοχής και της απογοήτευσης. Ειδικότερα, ως προς την επίδοση, η προτεινόμενη μέθοδος εμφανίζει χαμηλότερη τυπική απόκλιση (*performance_mean_std* = 0,1432) από τον τυχαίο σχηματισμό (0,2942) και το performance-only (0,8363).

Λεπτομέρειες υλοποίησης παρουσιάζονται στο Παράρτημα Α.

5.4.6 Ανάλυση ευστάθειας και εμπειρική αξιολόγηση του τυχαίου σχηματισμού

Για την εμπειρική αξιολόγηση της ευστάθειας των συγκρίσεων και τον έλεγχο του κατά πόσο τα αποτελέσματα θα μπορούσαν να προκύψουν από τυχαία διακύμανση, το σενάριο του τυχαίου σχηματισμού εκτελέστηκε 100 φορές με διαφορετικές τιμές σπόρου τυχαίας ακολουθίας (*seeds* 0–99). Η επιλογή αυτή προσφέρει επαρκή αποτύπωση της μεταβλητότητας του τυχαίου σχηματισμού με περιορισμένο υπολογιστικό κόστος. Για κάθε μετρική υπολογίστηκαν η μέση τιμή, η τυπική απόκλιση και το κεντρικό 95% της εμπειρικής κατανομής — δηλαδή το διάστημα μεταξύ του 2,5ού και του 97,5ού εκατοστημορίου, το οποίο αποτυπώνει το εύρος μέσα στο οποίο κινείται το 95% των τιμών που παρήγαγε ο τυχαίος σχηματισμός (χωρίς αυτό να αποτελεί διάστημα εμπιστοσύνης). Το μέτρο αυτό συμπληρώνει την πληροφορία των ακραίων τιμών (ελάχιστο–μέγιστο), προσφέροντας μια λιγότερο ευαίσθητη σε τυχαίες α-

κράιες περιπτώσεις εικόνα του αναμενόμενου εύρους της τυχαίας κατανομής. Για κάθε μετρική υπολογίστηκε επίσης εμπειρικό p -value ως $p = (b + 1)/(N + 1)$, όπου $N = 100$ είναι το πλήθος των τυχαίων εκτελέσεων και b το πλήθος των εκτελέσεων που παρήγαγαν αποτέλεσμα ίσο ή καλύτερο από εκείνο της προτεινόμενης μεθόδου, λαμβάνοντας υπόψη την κατεύθυνση βελτιστοποίησης της εκάστοτε μετρικής. Επομένως, όταν καμία από τις 100 τυχαίες εκτελέσεις δεν παρήγαγε ίσο ή καλύτερο αποτέλεσμα, προέκυψε $p = 1/101 \approx 0,0099$. Οι βασικές μετρικές παιδαγωγικής λειτουργικότητας και δομικής ισορροπίας παρουσιάζονται στους Πίνακες 5-7 και 5-8 για ομάδες 5 και 7 ατόμων αντίστοιχα. Με έντονη γραφή σημειώνεται η καλύτερη τιμή ανά μετρική μεταξύ της μέσης τιμής του Random baseline, του Performance-only baseline και της προτεινόμενης μεθόδου. Σε περίπτωση ισοβαθμίας επισημαίνονται όλες οι ισοδύναμες τιμές.

Μετρική	Random: μέση τιμή [κεντρικό 95%]	Επίδοση- Μόνο	Προτεινό- μενος
Ομάδες με ενεργό πυρήνα ↑ (groups_with_active_core)	39,92 [35,00–43,53]	42	53
Ομάδες με τουλάχιστον ένα μέλος επαρκούς αλληλεπίδρασης ↑ (groups_with_non_insufficient)	62,20 [58,00–66,00]	62	77
Μέση ποικιλία κοινωνικών ρόλων ↑ (mean_role_diversity)	2,1484 [2,0649– 2,2146]	2,1818	2,3896
Μέση απόκλιση κατανομής στρωμάτων ↓ (mean_strata_deviation)	0,7338 [0,6929– 0,7708]	0,9145	0,4936
Τυπική απόκλιση μέσου συνολικού προφίλ ↓ (profile_mean_std)	0,4483 [0,3871– 0,5058]	0,6603	0,1985
Τυπική απόκλιση μέσης επίδοσης ↓ (performance_mean_std)	0,3659 [0,3268– 0,4088]	0,8389	0,2183

Πίνακας 5-7 Στατιστικά πολλαπλών τυχαίων εκτελέσεων και συγκριτικά αποτελέσματα — ομάδες 5 ατόμων

Μετρική	Random: μέση τιμή [κεντρικό 95%]	Επίδοση- Μόνο	Προτεινό- μενος
Ομάδες με ενεργό πυρήνα ↑ (groups_with_active_core)	35,39 [31,00–39,53]	35	53
Ομάδες με τουλάχιστον ένα μέλος επαρκούς αλληλεπίδρασης ↑ (groups_with_non_insufficient)	49,23 [46,00–52,53]	48	55
Μέση ποικιλία κοινωνικών ρόλων ↑ (mean_role_diversity)	2,4755 [2,3723– 2,5455]	2,4727	2,9455
Μέση απόκλιση κατανομής στρωμάτων ↓ (mean_strata_deviation)	0,5633 [0,5150– 0,6139]	0,7918	0,1371
Τυπική απόκλιση μέσου συνολικού προφίλ ↓ (profile_mean_std)	0,3802 [0,3216– 0,4494]	0,6024	0,1057
Τυπική απόκλιση μέσης επίδοσης ↓ (performance_mean_std)	0,3065 [0,2585– 0,3602]	0,8363	0,1432

Πίνακας 5-8 Στατιστικά πολλαπλών τυχαίων εκτελέσεων και συγκριτικά αποτελέσματα — ομάδες 7 ατόμων

Σημείωση: Το σύμβολο ↑ υποδηλώνει ότι μεγαλύτερη τιμή είναι προτιμότερη, ενώ το ↓ ότι μικρότερη τιμή είναι προτιμότερη. Για το Random baseline παρουσιάζονται η μέση τιμή και το κεντρικό 95% της εμπειρικής κατανομής των 100 επαναλήψεων.

Οι πλήρεις πίνακες όλων των μετρικών παρατίθενται στο **Παράρτημα Β**.

Οι αναφορές που ακολουθούν σε ελάχιστες ή μέγιστες τιμές αφορούν τα πραγματικά άκρα που καταγράφηκαν στις 100 τυχαίες εκτελέσεις και όχι τα όρια του κεντρικού 95% που παρουσιάζεται στους συνοπτικούς πίνακες.

Η ανάλυση ευστάθειας επιβεβαιώνει ότι τα πλεονεκτήματα της προτεινόμενης μεθόδου δεν εξηγούνται από την τυχαία διακύμανση.

Πρώτον, η προτεινόμενη μέθοδος υπερτερεί στις βασικές μετρικές παιδαγωγικής λειτουργικότητας. Στις πενταμελείς ομάδες, είναι η μόνη που εξασφαλίζει ότι όλες οι ομάδες έχουν τουλάχιστον ένα μέλος

που δεν ανήκει στην κατηγορία Ανεπαρκής Αλληλεπίδρασης — κάτι που δεν παρατηρήθηκε σε καμία από τις 100 τυχαίες κατανομές (όπου το μέγιστο ήταν 68 ομάδες). Επίσης, ο αριθμός των ομάδων με ενεργό πυρήνα (53) ήταν μεγαλύτερος από κάθε τυχαία εκτέλεση (μέγιστο 46). Στις επταμελείς ομάδες, η προτεινόμενη μέθοδος καταφέρνει να δώσει ενεργό πυρήνα σχεδόν σε όλες τις ομάδες (53 από τις 55), υπερβαίνοντας το μέγιστο που παρατηρήθηκε στις 100 τυχαίες εκτελέσεις (41).

Δεύτερον, η προτεινόμενη μέθοδος δημιουργεί ομάδες που είναι πολύ πιο ισορροπημένες μεταξύ τους. Η κατανομή των στρωμάτων του μαθησιακού προφίλ είναι σαφώς πιο κοντά στη συνολική κατανομή του πληθυσμού, σε σύγκριση με τις τυχαίες κατανομές — για τις επταμελείς ομάδες, η μέση απόκλιση στρωμάτων της προτεινόμενης μεθόδου (0,14) ήταν χαμηλότερη από κάθε τιμή που παρατηρήθηκε στις 100 τυχαίες εκτελέσεις (ελάχιστο 0,48). Το ίδιο ισχύει για την ισορροπία του συνολικού προφίλ (0,11 έναντι ελάχιστου 0,30) και του κοινωνικού δείκτη (0,14 έναντι ελάχιστου 0,33), ενώ στις πενταμελείς ομάδες η βελτίωση είναι εξίσου σημαντική, όπως προκύπτει από τα πλήρη αποτελέσματα του Παραρτήματος Β.

Τρίτον, η προτεινόμενη μέθοδος πετυχαίνει καλύτερη ισορροπία και στην ακαδημαϊκή επίδοση μεταξύ των ομάδων, με τιμές (0,22 για τις πενταμελείς και 0,14 για τις επταμελείς) που ήταν χαμηλότερες από κάθε τιμή που παρατηρήθηκε στις 100 τυχαίες κατανομές. Στη διάσταση της απογοήτευσης, σύμφωνα με τα πλήρη αποτελέσματα του Παραρτήματος Β, για τις πενταμελείς ομάδες δεν παρατηρείται σαφής εμπειρική υπεροχή της προτεινόμενης μεθόδου έναντι των τυχαίων κατανομών. Αντίθετα, στις επταμελείς ομάδες η προτεινόμενη μέθοδος υπερτερεί και σε αυτή τη διάσταση (0,25 έναντι ελάχιστου τυχαίου 0,30).

Συνολικά, η ανάλυση ευστάθειας επιβεβαιώνει ότι η υπεροχή της προτεινόμενης μεθόδου είναι ουσιαστική και όχι αποτέλεσμα τύχης. Για την πλειονότητα των κρίσιμων μετρικών, καμία από τις 100 τυχαίες εκτελέσεις δεν πέτυχε ίσο ή καλύτερο αποτέλεσμα (εμπειρική τιμή $p < 0,01$). Η προτεινόμενη μέθοδος δημιουργεί σταθερά καλύτερες ομάδες από εκείνες που προέκυψαν στις 100 εξεταζόμενες τυχαίες κατανομές, τόσο ως προς την παιδαγωγική τους λειτουργικότητα όσο και ως προς την ισορροπία τους, ενώ παρουσιάζει συνολικά καλύτερη επίδοση και από το Performance-only baseline.

Λεπτομέρειες υλοποίησης παρουσιάζονται στο Παράρτημα Α.

5.4.7 Συζήτηση και συμπεράσματα

Η συγκριτική ανάλυση των τριών σεναρίων για δύο διαφορετικά μεγέθη ομάδας, σε συνδυασμό με την ανάλυση ευστάθειας των 100 τυχαίων εκτελέσεων, οδηγεί σε συνεπή συμπεράσματα, τα οποία υποστηρίζουν ισχυρά την αποτελεσματικότητα της προτεινόμενης μεθόδου ως προς τα κριτήρια της παρούσας εσωτερικής αξιολόγησης.

Η προτεινόμενη μέθοδος υπερτερεί σαφώς στη διασφάλιση κάλυψης ρόλων, επιτυγχάνοντας 100% κάλυψη των ομάδων με τουλάχιστον ένα μέλος που δεν ανήκει στην κατηγορία Ανεπαρκής Αλληλεπίδραση, τόσο για μέγεθος 5 όσο και για μέγεθος 7, γεγονός που αποτελεί ποιοτική διαφορά έναντι των στρατηγικών αναφοράς (baseline). Το εύρημα αυτό συνάδει με τη θεωρία Κοινωνικής Αλληλεξάρτησης (Butera & Buchs, 2019), σύμφωνα με την οποία η παρουσία μελών με προηγούμενη κοινωνική δραστηριότητα μπορεί να υποστηρίξει την αλληλεπίδραση και τη συνεργασία στο εσωτερικό της ομάδας. Σε ένα σύνολο όπου μεγάλο μέρος των φοιτητών ανήκει στην κατηγορία Ανεπαρκής Αλληλεπίδραση, η δυνατότητα του αλγορίθμου να διασπείρει αποτελεσματικά τους λίγους διαθέσιμους ενεργούς φοιτητές σε όλες τις ομάδες είναι κρίσιμη για τον περιορισμό του κινδύνου σχηματισμού ομάδων χωρίς κοινωνικά ενεργό μέλος και αναδεικνύει την αξία της ρητής στόχευσης στη διασπορά των ρόλων. Η επίτευξη υψηλότερης ποικιλίας ρόλων υποδηλώνει ότι η προτεινόμενη μέθοδος δημιουργεί συνθήκες λειτουργικής συμπληρωματικότητας μεταξύ των μελών. Η ανάλυση ευστάθειας με 100 επαναλήψεις τυχαίου σχηματισμού επιβεβαίωσε ότι τα πλεονεκτήματα αυτά δεν οφείλονται σε τυχαία διακύμανση.

Παράλληλα, η προτεινόμενη μέθοδος επιτυγχάνει την καλύτερη αναλογική εκπροσώπηση στρωμάτων και στις δύο παραμετροποιήσεις, με τιμές απόκλισης σαφώς χαμηλότερες από τα δύο σενάρια αναφοράς. Η ιδιότητα αυτή ενισχύει τη συγκρισιμότητα των ομάδων ως προς τη σύνθεση του μαθησιακού προφίλ των μελών τους, ανταποκρινόμενη στον σχεδιαστικό στόχο της ισότιμης και συγκρίσιμης σύνθεσης των ομάδων. Η ενσωμάτωση των ευρημάτων της ανάλυσης ομοφιλίας, ιδίως ως προς τη στατιστικά σημαντική ομοφιλία του στρώματος προφίλ, ενίσχυσε την ικανότητα του αλγορίθμου να αποφεύγει την αναπαραγωγή υφιστάμενων ομοιογενών δομών του δικτύου.

Τα αποτελέσματα αναδεικνύουν επίσης τους περιορισμούς της μονοκριτηριακής προσέγγισης. Το performance-only baseline εμφανίζει τη χαμηλότερη συνολική επίδοση ως προς τον πολυκριτηριακό στόχο της εργασίας. Παρά την εύλογη εκ πρώτης όψεως λογική της ομαδοποίησης βάσει ακαδημαϊκής επίδοσης, η συγκεκριμένη στρατηγική οδηγεί σε περιορισμένη κάλυψη ορισμένων ρόλων και στις μεγαλύτερες αποκλίσεις μεταξύ ομάδων ως προς την επίδοση και τα στρώματα προφίλ. Το εύρημα αυτό καταδεικνύει ότι, ως προς τον πολυκριτηριακό στόχο της παρούσας εργασίας, η αποκλειστική χρήση

της ακαδημαϊκής επίδοσης δεν επαρκεί για την ταυτόχρονη εξισορρόπηση των διαφορετικών διαστάσεων της σύνθεσης των ομάδων.

Ο τυχαίος σχηματισμός, παρότι εμφανίζει συγκριτικά καλές επιδόσεις σε ορισμένες μετρικές, αδυνατεί να διασφαλίσει συστηματικά κάλυψη ρόλων και αναλογική κατανομή στρωμάτων. Η τυχειότητα δεν συνοδεύεται από καμία εγγύηση για τις υπόλοιπες διαστάσεις που κρίνονται παιδαγωγικά σημαντικές.

Η σύγκριση των δύο μεγεθών ομάδας αναδεικνύει μια αντιστάθμιση που συζητείται στη βιβλιογραφία (Butera & Buchs, 2019): οι μεγαλύτερες ομάδες (7 ατόμων) επιτυγχάνουν καλύτερη ισορροπία στρωμάτων, αλλά αναπόφευκτα συγκεντρώνουν περισσότερους φοιτητές της κατηγορίας Ανεπαρκής Αλληλεπίδραση ανά ομάδα. Η επιλογή μεγέθους εναπόκειται στον εκπαιδευτικό ανάλογα με τους στόχους της δραστηριότητας: μικρότερες ομάδες προσφέρουν εντατικότερη αλληλεπίδραση ανά μέλος, ενώ μεγαλύτερες επιτρέπουν μεγαλύτερη ποικιλότητα και καλύτερη αναλογική εκπροσώπηση.

Παρά τα θετικά αυτά ευρήματα, η παρούσα μελέτη παρουσιάζει ορισμένους περιορισμούς που θα πρέπει να ληφθούν υπόψη κατά την ερμηνεία των αποτελεσμάτων. Πρώτον, η ανάλυση βασίζεται σε συνθετικό dataset, το οποίο, παρά τη ρεαλιστική του δομή, δεν αποτυπώνει την πλήρη πολυπλοκότητα ενός πραγματικού ακαδημαϊκού περιβάλλοντος. Δεύτερον, η αξιολόγηση του αλγορίθμου περιορίστηκε σε προσομοίωση της διαδικασίας σχηματισμού ομάδων, ενώ η επίδρασή του στην ποιότητα της συνεργασίας και της αλληλεπίδρασης παραμένει προς διερεύνηση, καθώς δεν πραγματοποιήθηκε πραγματική εκπαιδευτική παρέμβαση. Επιπλέον, δεν μετρήθηκαν τα μαθησιακά αποτελέσματα μετά την ομαδοποίηση, γεγονός που περιορίζει την άμεση σύνδεση του αλγορίθμου με την ακαδημαϊκή επίδοση των φοιτητών. Τρίτον, η γενικευσιμότητα των ευρημάτων σε διαφορετικά εκπαιδευτικά πλαίσια, μαθήματα ή πλατφόρμες παραμένει αδιερεύνητη. Οι περιορισμοί αυτοί αποτελούν ταυτόχρονα κατευθύνσεις για μελλοντική έρευνα, η οποία θα μπορούσε να επεκταθεί στην εφαρμογή της μεθόδου σε πραγματικές συνθήκες και στη συσχέτιση των αποτελεσμάτων της με μετρήσιμους δείκτες μάθησης.

Συνολικά, η προτεινόμενη μέθοδος αναδεικνύεται ως η πλέον αποτελεσματική μεταξύ των τριών εξεταζόμενων στρατηγικών, επιτυγχάνοντας πολυκριτηριακή εξισορρόπηση μεταξύ λειτουργικής κάλυψης ρόλων, παιδαγωγικής ετερογένειας, ισορροπίας στρωμάτων και ομοιόμορφης κατανομής των ποσοτικών χαρακτηριστικών του προφίλ. Τα ευρήματα επιβεβαιώνουν την υπόθεση ότι ο συνδυασμός EDM, SNA και πολυκριτηριακής σύστασης, στο πλαίσιο των CARS (Adomavicius & Tuzhilin, 2011), μπορεί να οδηγήσει σε σχηματισμό ομάδων που ανταποκρίνεται καλύτερα στα παιδαγωγικά κριτήρια της λειτουργικής κάλυψης ρόλων, της ετερογένειας και της ισορροπίας μεταξύ των ομάδων, σε σύγκριση με απλούστερες προσεγγίσεις. Σε πρακτικό επίπεδο, η προτεινόμενη μέθοδος μπορεί να λειτουργήσει ως εργαλείο υποστήριξης αποφάσεων για τον εκπαιδευτικό, διευκολύνοντας τη δημιουργία συνεργατικών ομάδων με περισσότερο ισορροπημένη, ετερογενή και παιδαγωγικά τεκμηριωμένη σύνθεση.

6 Συμπεράσματα και Μελλοντικές Επεκτάσεις

6.1 Σύνοψη και απάντηση στα ερευνητικά ερωτήματα

Η παρούσα εργασία ολοκλήρωσε τη διερεύνηση του κεντρικού της αντικειμένου, απαντώντας στα πέντε ερευνητικά ερωτήματα που τέθηκαν στο Κεφάλαιο 1. Μέσω της ανάπτυξης και πειραματικής εφαρμογής μιας υβριδικής μεθοδολογίας, η οποία συνδυάζει τεχνικές Εξόρυξης Εκπαιδευτικών Δεδομένων (EDM), Ανάλυσης Κοινωνικών Δικτύων (SNA) και πολυκριτηριακής σύστασης, τεκμηριώθηκε η δυνατότητα αυτοματοποιημένου σχηματισμού παιδαγωγικά ισορροπημένων ομάδων. Η μεθοδολογία εκκινεί από δύο μόνο αρχεία CSV που εξάγονται από το Moodle —το αρχείο καταγραφής δραστηριότητας (logs) και το αρχείο βαθμολογιών (grades)— και, μέσω μιας σύνθετης αλυσίδας επεξεργασίας, καταλήγει στην παραγωγή συνεργατικών ομάδων βελτιωμένων ως προς τα καθορισμένα παιδαγωγικά και συμπεριφορικά κριτήρια, αξιοποιώντας αποκλειστικά τα δεδομένα αυτά σε συνθετικό περιβάλλον.

EE1: Ποια δεδομένα αλληλεπίδρασης του Moodle είναι καταλληλότερα για την εξαγωγή αξιόπιστων δεικτών μαθησιακής και κοινωνικής συμπεριφοράς φοιτητών;

Η ανάλυση έδειξε ότι τα δεδομένα που περιέχονται στα δύο βασικά αρχεία εξαγωγής της πλατφόρμας —το αρχείο καταγραφής δραστηριότητας (log file) και το αρχείο βαθμολογιών (grades)— επαρκούν για την εφαρμογή της προτεινομένης μεθοδολογίας στο συγκεκριμένο πλαίσιο. Από το πρώτο αντλούνται πληροφορίες συμμετοχής (π.χ., πρόσβαση σε πόρους, συμμετοχή σε φόρουμ, προσπάθειες σε κουίζ), κοινωνικής αλληλεπίδρασης και απογοήτευσης, ενώ από το δεύτερο αντλούνται οι δείκτες ακαδημαϊκής επίδοσης. Η βασική διαπίστωση είναι ότι τα συγκεκριμένα αρχεία, υπό την προϋπόθεση κατάλληλης προεπεξεργασίας, παρέχουν όλη την απαραίτητη πληροφορία για την εφαρμογή της μεθοδολογίας, χωρίς να απαιτούνται πρόσθετες πηγές δεδομένων.

EE2: Πώς μπορούν οι δείκτες αυτοί να συνδυαστούν σε ένα ενιαίο, πολυδιάστατο προφίλ φοιτητή που να υποστηρίζει τον σχηματισμό συνεργατικών ομάδων;

Η διαδικασία σύνθεσης των τεσσάρων διαστάσεων του προφίλ (Ενότητες 4.2.1–4.2.5) έδειξε ότι δείκτες συμμετοχής, επίδοσης, κοινωνικής δικτύωσης και απογοήτευσης μπορούν να ενσωματωθούν σε έναν ενιαίο σταθμισμένο δείκτη, βάσει του οποίου πραγματοποιείται η στρωματοποίηση του πληθυσμού (Ενότητα 4.3). Το πολυδιάστατο προφίλ που προέκυψε παρέχει μια ερμηνεύσιμη αναπαράσταση της ετερογένειας του πληθυσμού, επιτρέποντας τη συγκριτική αξιολόγηση των προφίλ και την ισορροπημένη, πολυκριτηριακή κατανομή των φοιτητών στις ομάδες.

EE3: Πώς μπορεί η Ανάλυση Κοινωνικών Δικτύων να συμβάλει στον εντοπισμό λειτουργικών ρόλων και στην αποτύπωση του τρόπου διασύνδεσης και συμμετοχής των φοιτητών στη μαθησιακή κοινότητα;

Η κατασκευή διμερούς γράφου φοιτητών-συζητήσεων (Ενότητα 4.2.3.1) και η προβολή του σε μονότροπο δίκτυο φοιτητών (Ενότητα 4.2.3.3) επέτρεψαν τον υπολογισμό μετρικών κεντρικότητας (Ενότητα 4.2.3.4) και την αντιστοίχισή τους σε κοινωνικούς ρόλους (Ενότητα 4.2.3.5). Η ανάλυση ανέδειξε σαφή διαφοροποίηση μεταξύ Ηγετών, Γεφυρών, Δραστήριων Συμμετεχόντων, Περιφερειακών Συμμετεχόντων και φοιτητών με Ανεπαρκή Αλληλεπίδραση. Η στατιστική διαφοροποίηση μέσω του ελέγχου Kruskal-Wallis επιβεβαίωσε στατιστικά σημαντικές διαφοροποιήσεις μεταξύ των κατηγοριών κοινωνικών ρόλων ως προς όλες τις εξεταζόμενες μετρικές ($p < 0,001$). Η SNA αποτύπωσε τον τρόπο διασύνδεσης και συμμετοχής των φοιτητών, αναδεικνύοντας πρότυπα αλληλεπίδρασης που αξιοποιήθηκαν για την παιδαγωγικά τεκμηριωμένη συγκρότηση των ομάδων.

EE4: Πώς μπορεί να διαμορφωθεί ένα αλγοριθμικό μοντέλο που αντιμετωπίζει τον σχηματισμό ομάδων ως πολυκριτηριακό πρόβλημα σύστασης, λαμβάνοντας υπόψη παιδαγωγικούς και συμπεριφορικούς παράγοντες;

Ο σχεδιασμός του greedy αλγορίθμου (Ενότητα 5.3), βασισμένου σε πολυκριτηριακή προσέγγιση (Ενότητα 5.1), κατέδειξε ότι είναι εφικτή η μετάφραση παιδαγωγικών αρχών σε υπολογιστικά αποδοτική διαδικασία. Ο αλγόριθμος ενσωματώνει: (α) διασπορά των κοινωνικά ενεργών και διαμεσολαβητικών ρόλων μεταξύ των ομάδων, (β) ισορροπία στρωμάτων του πολυδιάστατου προφίλ, (γ) greedy επιλογή με συνάρτηση κόστους που κωδικοποιεί παιδαγωγικές προτεραιότητες, και (δ) ειδική διαχείριση φοιτητών με Ανεπαρκή Αλληλεπίδραση. Παράλληλα, ο αλγόριθμος ενσωματώνει τα ευρήματα της ανάλυσης ομοφιλίας (Ενότητα 5.2), η οποία υπολογίστηκε στο προβαλλόμενο δίκτυο φοιτητών. Με βάση τα ευρήματα αυτά, ο αλγόριθμος ενεργοποιεί ισχυρότερες ποινές στις διαστάσεις όπου το δίκτυο εμφανίζει στατιστικά σημαντική τάση ομοιογένειας (π.χ., στο στρώμα του πολυδιάστατου προφίλ - profile_stratum).

EE5: Με ποιες πολυδιάστατες μετρικές μπορεί να αξιολογηθεί η αποτελεσματικότητα του προτεινόμενου συστήματος συστάσεων;

Το προτεινόμενο πλαίσιο αξιολόγησης (Ενότητα 5.4.3) περιλαμβάνει μετρικές παιδαγωγικής λειτουργικότητας (κάλυψη ρόλων, ποικιλία ρόλων, συσσώρευση παθητικών) και μετρικές δομικής ισορροπίας (αποκλίσεις στρωμάτων, τυπικές αποκλίσεις μέσων όρων). Η συγκριτική εφαρμογή του (Ενότητες Διπλωματική Εργασία

5.4.4, 5.4.5) ανέδειξε σαφή υπεροχή της προτεινόμενης μεθόδου έναντι απλούστερων στρατηγικών (τυχαίας ή μονοκριτηριακής). Για παράδειγμα, στις επταμελείς ομάδες, η προτεινόμενη μέθοδος πέτυχε 100% κάλυψη ομάδων με τουλάχιστον ένα μέλος εκτός της κατηγορίας Ανεπαρκής Αλληλεπίδραση και σχεδόν πλήρη κάλυψη με ενεργό πυρήνα (53 από τις 55 ομάδες, 96,4%), έναντι κατά πολύ χαμηλότερων ποσοστών των εναλλακτικών μεθόδων, ενώ παρουσίασε κατά πολύ χαμηλότερη απόκλιση στρωμάτων ($\text{mean_strata_deviation} = 0,1371$) σε σύγκριση με τη μέση τιμή των 100 εκτελέσεων του τυχαίου σχηματισμού (0,5633). Η ανάλυση ευστάθειας με 100 ανεξάρτητες εκτελέσεις του τυχαίου σχηματισμού (Ενότητα 5.4.6) επιβεβαίωσε ότι τα πλεονεκτήματα αυτά δεν οφείλονται σε τυχαία διακύμανση.

6.2 Συμβολή της εργασίας

Η παρούσα εργασία συμβάλλει στη βιβλιογραφία των Learning Analytics σε τέσσερα βασικά σημεία.

Πρώτον, προτείνεται ένα ολοκληρωμένο πλαίσιο εξαγωγής πολυδιάστατων προφίλ φοιτητών που συνδυάζει, σε ενιαίο μεθοδολογικό πλαίσιο, δείκτες από τέσσερις διακριτές διαστάσεις: συμμετοχή, ακαδημαϊκή επίδοση, κοινωνική αλληλεπίδραση μέσω Ανάλυσης Κοινωνικών Δικτύων (SNA) και απογοήτευση. Η προσέγγιση αυτή αποτυπώνει την πολυπλοκότητα της διαδικτυακής μαθησιακής συμπεριφοράς χωρίς απαίτηση εξωτερικών ερωτηματολογίων ή παρεμβατικών μεθόδων συλλογής δεδομένων, αξιοποιώντας αποκλειστικά τα δεδομένα δραστηριότητας και επίδοσης που είναι διαθέσιμα στην πλατφόρμα. Η στρωματοποίηση των προφίλ σε πέντε επίπεδα παρέχει μια ερμηνεύσιμη αναπαράσταση της ετερογένειας του πληθυσμού, διευκολύνοντας τη λήψη παιδαγωγικών αποφάσεων.

Δεύτερον, σχεδιάστηκε ένα ερμηνεύσιμο σύστημα συστάσεων που μεταφράζει παιδαγωγικές αρχές της Θεωρίας Κοινωνικής Αλληλεξάρτησης σε υπολογιστικά κριτήρια. Ο greedy αλγόριθμος ενσωματώνει παιδαγωγικές έννοιες —διασπορά των κοινωνικά ενεργών και διαμεσολαβητικών ρόλων (Ηγετών και Γεφυρών), ισορροπία συμμετοχής μέσω στρωματοποίησης, αποφυγή συσσώρευσης υψηλών επιπέδων απογοήτευσης, και ειδική διαχείριση φοιτητών με Ανεπαρκή Αλληλεπίδραση— προσφέροντας διαφάνεια και δυνατότητα παιδαγωγικής παρέμβασης. Η καινοτομία έγκειται στην ενσωμάτωση των ευρημάτων της ανάλυσης ομοφιλίας: ο αλγόριθμος ενεργοποιεί ισχυρότερες ποινές στις διαστάσεις όπου το δίκτυο εμφανίζει στατιστικά σημαντική τάση ομοιογένειας, καθιστώντας τον προσαρμοστικό και ικανό να ανταποκρίνεται στις ιδιαιτερότητες του εκάστοτε δικτύου αλληλεπιδράσεων.

Τρίτον, ορίζεται ένα συστηματικό πλαίσιο συγκριτικής αξιολόγησης μεθόδων σχηματισμού ομάδων, το οποίο περιλαμβάνει δύο κατηγορίες μετρικών: μετρικές παιδαγωγικής λειτουργικότητας που αποτιμούν την εσωτερική σύνθεση των ομάδων (κάλυψη και ποικιλία ρόλων, συσσώρευση παθητικών μελών) και μετρικές δομικής ισορροπίας που αποτιμούν τη συγκρισιμότητα μεταξύ των ομάδων (αποκλίσεις στρωμάτων, τυπικές αποκλίσεις μέσων όρων). Το πλαίσιο αυτό προσφέρει μια πολυδιάστατη, αντικειμενική

βάση σύγκρισης διαφορετικών στρατηγικών ομαδοποίησης, καλύπτοντας τόσο την παιδαγωγική όσο και τη στατιστική διάσταση της αξιολόγησης και αντιμετωπίζοντας έναν περιορισμό που αναδείχθηκε στην ανασκόπηση της σχετικής βιβλιογραφίας.

Τέταρτον, αναπτύχθηκε μια πλήρως τεκμηριωμένη μεθοδολογία παραγωγής συνθετικών δεδομένων υψηλής δομικής πιστότητας με προδιαγραφές που αντλήθηκαν από την επίσημη τεκμηρίωση του Moodle. Η προσέγγιση αυτή αντιμετωπίζει αποτελεσματικά τους περιορισμούς που θέτει ο Γενικός Κανονισμός Προστασίας Δεδομένων (GDPR) και επιτρέπει την πειραματική διερεύνηση μεθοδολογιών σε ρεαλιστικό περιβάλλον, χωρίς άμεση πρόσβαση σε πραγματικά δεδομένα. Η συστηματική τεκμηρίωση των προδιαγραφών παραγωγής (δομική πιστότητα, χρονική ακρίβεια, ανωνυμοποίηση, προσομοίωση ρόλων) μπορεί να λειτουργήσει ως χρήσιμη μεθοδολογική βάση για ανάλογες έρευνες.

Συνολικά, η εργασία καλύπτει ένα σημαντικό ερευνητικό κενό, αναδεικνύοντας ότι ο συνδυασμός δεδομένων δραστηριότητας και επίδοσης από το Moodle, ανάλυσης κοινωνικών δικτύων και ερμηνεύσιμων, παιδαγωγικά τεκμηριωμένων αλγορίθμων μπορεί να υποστηρίξει αποτελεσματικά τον σχεδιασμό και τον σχηματισμό συνεργατικών ομάδων σε συστήματα διαχείρισης μάθησης. Η προτεινόμενη προσέγγιση προσφέρει μια λειτουργική υπολογιστική ροή που μπορεί να αποτελέσει τη βάση για την ανάπτυξη εργαλείου υποστήριξης εκπαιδευτικών και ερευνητών, μειώνοντας την υποκειμενικότητα στον σχηματισμό ομάδων και αξιοποιώντας πλήρως τα δεδομένα που ήδη συλλέγονται από τα LMS. Η μεθοδολογία είναι, κατ' αρχήν, μεταφέρσιμη σε διαφορετικές εγκαταστάσεις Moodle, υπό την προϋπόθεση προσαρμογής στη δομή και στα διαθέσιμα πεδία των εκάστοτε αρχείων εξαγωγής, γεγονός που καθιστά τη συμβολή της ιδιαίτερα πρακτική και επεκτάσιμη.

6.3 Περιορισμοί της έρευνας

Η παρούσα μελέτη υπόκειται σε ορισμένους περιορισμούς, οι οποίοι πρέπει να ληφθούν υπόψη κατά την ερμηνεία και γενίκευση των ευρημάτων.

Πρώτον, η πειραματική εφαρμογή της μεθοδολογίας βασίστηκε σε συνθετικά δεδομένα υψηλής πιστότητας και όχι σε πραγματικά δεδομένα αλληλεπίδρασης φοιτητών. Η επιλογή αυτή υπαγορεύθηκε από περιορισμούς πρόσβασης σε πραγματικά εκπαιδευτικά δεδομένα λόγω ζητημάτων προστασίας προσωπικών δεδομένων (GDPR) και θεσμικών περιορισμών. Τα συνθετικά δεδομένα δεν μπορούν να αποτυπώσουν πλήρως την πολυπλοκότητα της πραγματικής ανθρώπινης συμπεριφοράς σε ένα εκπαιδευτικό περιβάλλον. Συγκεκριμένα, η χρονική κατανομή της δραστηριότητας, τα πρότυπα αλληλεπίδρασης και η συναισθηματική διάσταση της μάθησης προσεγγίζονται μέσω στατιστικών κανόνων και όχι μέσω πραγματικών συμπεριφορικών δεδομένων.

Δεύτερον, η ανάλυση της κοινωνικής αλληλεπίδρασης βασίστηκε αποκλειστικά στις αλληλεπιδράσεις που καταγράφονται στο φόρουμ του Moodle. Σε πραγματικά μαθησιακά περιβάλλοντα, σημαντικό μέρος της επικοινωνίας μεταξύ φοιτητών πραγματοποιείται εκτός της πλατφόρμας, μέσω άλλων ψηφιακών μέσων (π.χ., εφαρμογές ανταλλαγής μηνυμάτων, κοινωνικά δίκτυα, ομαδικές συναντήσεις). Κατά συνέπεια, το κοινωνικό δίκτυο που αναλύθηκε αποτυπώνει μόνο ένα τμήμα της συνολικής κοινωνικής δυναμικής, ενδεχομένως παραλείποντας σημαντικές αλληλεπιδράσεις που διαμορφώνουν τους λειτουργικούς ρόλους των φοιτητών.

Τρίτον, η αντιστοίχιση μετρικών SNA σε κοινωνικούς ρόλους (Ενότητα 4.2.3) είναι εν μέρει ερμηνευτική και εξαρτάται από τη φύση του εκάστοτε δικτύου. Στο προβαλλόμενο δίκτυο που αναλύθηκε, η παρουσία 28 απομονωμένων κόμβων κατέστησε αναγκαίο τον χειρισμό των κόμβων αυτών με μηδενικές τιμές σε ορισμένες μετρικές.

Τέταρτον, ο προτεινόμενος αλγόριθμος σύστασης ομάδων βασίζεται σε μια greedy ευρετική προσέγγιση, η οποία επιλέχθηκε λόγω της υπολογιστικής αποδοτικότητας και της ερμηνευσιμότητάς της. Αν και η προσέγγιση αυτή είναι υπολογιστικά αποδοτική και εφαρμόστηκε επιτυχώς στο εξεταζόμενο σύνολο φοιτητών, δεν εγγυάται τον εντοπισμό της βέλτιστης δυνατής κατανομής ομάδων σε παγκόσμιο επίπεδο.

Πέμπτον, η μεθοδολογία περιλαμβάνει παραμετρικές επιλογές (βάρη διαστάσεων, κατώφλια φιλτραρίσματος, όρια z-score, βάρη συνάρτησης κόστους). Παρότι πραγματοποιήθηκε τοπικός έλεγχος ευαισθησίας των βαρών του σύνθετου δείκτη προφίλ, δεν διενεργήθηκε συνολική και συστηματική ανάλυση ευαισθησίας για το σύνολο των παραμέτρων, ούτε εξετάστηκε η επίδραση εναλλακτικών τιμών τους στις τελικές ομάδες. Επομένως, η διερεύνηση της ευστάθειας της μεθόδου ως προς τις υπόλοιπες παραμετρικές επιλογές αποτελεί κατεύθυνση για μελλοντική έρευνα.

Έκτον, η αξιολόγηση του προτεινόμενου συστήματος είναι αποκλειστικά εκ των προτέρων (a priori). Οι μετρικές που χρησιμοποιήθηκαν (Ενότητα 5.4.3) αξιολογούν τη σύνθεση των ομάδων πριν από την έναρξη της συνεργασίας, εστιάζοντας στην κατανομή ρόλων, στην ισορροπία στρωμάτων και στην ομοιογένεια των μέσων όρων. Δεν κατέστη δυνατή η παρακολούθηση της πραγματικής συνεργατικής απόδοσης των ομάδων σε ένα πραγματικό μάθημα, ούτε η συλλογή ανατροφοδότησης από τους φοιτητές και τους διδάσκοντες σχετικά με την εμπειρία τους.

6.4 Προτάσεις για μελλοντική έρευνα

Η παρούσα εργασία θέτει τις βάσεις για ένα ολοκληρωμένο πλαίσιο σχηματισμού ομάδων, ανοίγοντας ταυτόχρονα πολλαπλές κατευθύνσεις για μελλοντική διερεύνηση και επέκταση. Οι προτάσεις που ακολουθούν οργανώνονται σε τρεις θεματικούς άξονες.

Εφαρμογή σε πραγματικά δεδομένα και παιδαγωγική αξιολόγηση. Η άμεση επόμενη κατεύθυνση είναι η εφαρμογή της μεθοδολογίας σε πραγματικά εκπαιδευτικά δεδομένα από μαθήματα που φιλοξενούνται στο Moodle. Αυτό θα επέτρεπε την εκ των υστέρων (a posteriori) αξιολόγηση της παιδαγωγικής αποτελεσματικότητας του αλγορίθμου, μέσω της σύγκρισης της συνεργατικής απόδοσης των ομάδων που προέκυψαν από την προτεινόμενη μέθοδο με εκείνες που προέκυψαν από εναλλακτικές στρατηγικές (π.χ., τυχαίο σχηματισμό, σχηματισμό βάσει επίδοσης), καθώς και μέσω της ποιοτικής αξιολόγησης της συνεργασίας και της συλλογής ανατροφοδότησης από φοιτητές και διδάσκοντες. Μια τέτοια μελέτη θα μπορούσε να αξιοποιήσει ανεξάρτητες εξωτερικές μετρικές, όπως η ποιότητα των παραγόμενων έργων, ο βαθμός συμμετοχής, ή η ικανοποίηση των μελών, προσφέροντας έτσι ισχυρότερη τεκμηρίωση της παιδαγωγικής αξίας της μεθόδου.

Διερεύνηση εναλλακτικών αλγοριθμικών προσεγγίσεων. Μελλοντική έρευνα θα μπορούσε να διερευνήσει εναλλακτικές ή υβριδικές προσεγγίσεις, όπως εξελικτικούς αλγορίθμους ή τεχνικές πολυκριτηριακής βελτιστοποίησης, αξιολογώντας παράλληλα την ικανότητά τους να διατηρούν την ερμηνευσιμότητα που απαιτεί ένα εκπαιδευτικό εργαλείο. Ιδιαίτερο ενδιαφέρον παρουσιάζει η διερεύνηση της ευαισθησίας του αλγορίθμου σε διαφορετικές παραμετρικές διαμορφώσεις (βάρη, κατώφλια, όρια z-score), ώστε να προσδιοριστούν οι βέλτιστες τιμές για διαφορετικά εκπαιδευτικά περιβάλλοντα.

Ενσωμάτωση πρόσθετων πηγών δεδομένων και ανάλυση περιεχομένου. Μελλοντική έρευνα θα μπορούσε να ενσωματώσει τεχνικές επεξεργασίας φυσικής γλώσσας (Natural Language Processing - NLP) για την ανάλυση του περιεχομένου των συζητήσεων, επιτρέποντας τον εντοπισμό θεματικών ομοιογενειών ή την ανίχνευση συναισθηματικών τάσεων πέρα από τα σήματα απογοήτευσης. Επιπλέον, η αξιοποίηση δεδομένων από άλλες ψηφιακές πλατφόρμες που χρησιμοποιούνται από τους φοιτητές θα μπορούσε να προσφέρει μια πληρέστερη αποτύπωση της κοινωνικής δυναμικής, συμπληρώνοντας τα δεδομένα του φόρουμ.

Οι ανωτέρω κατευθύνσεις αναδεικνύουν ότι η παρούσα εργασία αποτελεί αφετηρία για περαιτέρω έρευνα και όχι τελικό στάδιο διερεύνησης του προβλήματος. Η συνδυαστική αξιοποίηση δεδομένων αλληλεπίδρασης, τεχνικών ανάλυσης δικτύων και ερμηνεύσιμων αλγοριθμικών μεθόδων μπορεί να εξελιχθεί σε ένα ισχυρό εργαλείο υποστήριξης της συνεργατικής μάθησης. Παράλληλα, η μεθοδολογία είναι σχεδιασμένη με γνώμονα τη μελλοντική ενσωμάτωση στο περιβάλλον του Moodle, μια προοπτική που εξετάζεται λεπτομερώς στη συνέχεια (Ενότητα 6.5).

6.5 Σχεδιασμός για μελλοντική ενσωμάτωση στο Moodle

Η μεθοδολογία που αναπτύχθηκε στα προηγούμενα κεφάλαια βασίζεται σε δεδομένα που κατ' αρχήν είναι διαθέσιμα σε εγκαταστάσεις Moodle: αρχεία καταγραφής αλληλεπίδρασης και βαθμολογίες. Σε μια πραγματική εγκατάσταση, τα αρχεία αυτά (κύριο αρχείο καταγραφής και αρχείο βαθμολογιών) είναι προσβάσιμα σε μορφή κατάλληλη για επεξεργασία, όπως xlsx ή csv, εφόσον ο χρήστης διαθέτει ρόλο με κατάλληλα δικαιώματα πρόσβασης σε αναφορές και εξαγωγή δεδομένων (π.χ., Tutor, Εκπαιδευτής). Το γεγονός αυτό καθιστά την προτεινόμενη μεθοδολογία εφαρμόσιμη υπό τις κατάλληλες προϋποθέσεις σε πραγματικά εκπαιδευτικά περιβάλλοντα, χωρίς να απαιτείται παρέμβαση στον πυρήνα του Moodle, αλλά με την προϋπόθεση ύπαρξης κατάλληλης υποδομής για τη φιλοξενία της εξωτερικής υπηρεσίας ή την εγκατάσταση σχετικού plugin.

Στο πλαίσιο της παρούσας εργασίας, και λαμβάνοντας υπόψη την αδυναμία πρόσβασης σε πραγματικά δεδομένα, δεν κατέστη εφικτή η υλοποίηση ενός λειτουργικού πρωτοτύπου ενσωματωμένου σε πραγματική εγκατάσταση Moodle. Ωστόσο, είναι σκόπιμο να τεκμηριωθεί ο τρόπος με τον οποίο η προτεινόμενη λύση θα μπορούσε μελλοντικά να ενσωματωθεί στο Moodle, είτε ως εξωτερική υπηρεσία ανάλυσης είτε ως επέκταση (plugin) της πλατφόρμας.

6.5.1 Προϋποθέσεις ενσωμάτωσης

Για την πραγματική εφαρμογή της προτεινόμενης μεθοδολογίας σε ένα μάθημα που φιλοξενείται στο Moodle, απαιτούνται ορισμένες προϋποθέσεις:

- Πρόσβαση σε πραγματικά δεδομένα: Η ανάπτυξη και δοκιμή ενός ολοκληρωμένου συστήματος προϋποθέτει πρόσβαση σε πραγματικά αρχεία καταγραφής και βαθμολογίες φοιτητών, με όλες τις απαραίτητες άδειες και εγκρίσεις.
- Διαχειριστικά δικαιώματα στην πλατφόρμα: Η ενσωμάτωση μιας εξωτερικής υπηρεσίας ή η εγκατάσταση ενός plugin απαιτεί πρόσβαση σε επίπεδο διαχειριστή του Moodle, κάτι που δεν είναι διαθέσιμο σε ερευνητικό πλαίσιο χωρίς τη συνεργασία του τμήματος Πληροφορικής του Ιδρύματος.
- Συμμόρφωση με το GDPR: Η επεξεργασία δεδομένων φοιτητών, ακόμη και ψευδωνυμοποιημένων, υπόκειται σε αυστηρές προϋποθέσεις. Απαιτούνται κατάλληλη νομική βάση επεξεργασίας, ενημέρωση των φοιτητών, εξασφάλιση των αναγκαίων θεσμικών και δεοντολογικών εγκρίσεων και τήρηση των αρχών προστασίας προσωπικών δεδομένων.

- **Υποδομή φιλοξενίας:** Η εξωτερική υπηρεσία επεξεργασίας (π.χ. σε Python) απαιτεί ένα διακομιστή (server) με δυνατότητα επικοινωνίας με το Moodle, είτε εντός του ίδιου δικτύου είτε μέσω ασφαλών κλήσεων API.

6.5.2 Ενδεικτική ροή λειτουργίας

Υπό τις παραπάνω προϋποθέσεις, η ροή λειτουργίας θα μπορούσε να έχει ως εξής:

- **Εξαγωγή δεδομένων:** Τα απαραίτητα δεδομένα (logs, βαθμολογίες) αντλούνται από το Moodle, είτε μέσω των ενσωματωμένων reports (π.χ. εξαγωγή σε CSV) είτε, ιδανικά, μέσω του Moodle API για αυτοματοποιημένη πρόσβαση.
- **Επεξεργασία - Υπολογισμός προφίλ και ομάδων:** Τα δεδομένα τροφοδοτούνται σε ένα εξωτερικό σύστημα (π.χ. μια υπηρεσία υλοποιημένη σε Python), το οποίο εκτελεί τον αλγόριθμο υπολογισμού των μετρικών (ανάλυση κοινωνικών δικτύων, δείκτες συμμετοχής, σημεία απογοήτευσης) και στη συνέχεια τον αλγόριθμο σχηματισμού ομάδων που περιεγράφηκε στην Ενότητα 5.3. Η πλήρης υλοποίηση της μεθοδολογίας σε Python, με αρθρωτή δομή pipeline που οργανώνει κάθε στάδιο επεξεργασίας ως διακριτό αυτόνομο βήμα, διευκολύνει σημαντικά τη μελλοντική ενσωμάτωση της μεθόδου, είτε μέσω αυτοματοποιημένων κλήσεων API είτε μέσω της ανάπτυξης ενός plugin για το Moodle, καθώς απαιτεί ελάχιστες τροποποιήσεις για τη μετατροπή της πειραματικής ροής σε λειτουργική υπηρεσία.
- **Παρουσίαση αποτελεσμάτων:** Τα αποτελέσματα (αντιστοίχιση φοιτητών σε ομάδες) επιστρέφονται στο Moodle και εμφανίζονται στον εκπαιδευτικό μέσω μιας κατάλληλης διεπαφής, δίνοντάς του τη δυνατότητα να τα αποδεχτεί, να τα τροποποιήσει ή να τα αποθηκεύσει.

Η παρούσα εργασία δεν φιλοδοξεί να δώσει οριστικές απαντήσεις σε ένα πρόβλημα τόσο πολύπλοκο όσο η υποστήριξη της συνεργατικής μάθησης. Επιχειρεί, ωστόσο, να αναδείξει ότι η συστηματική αξιοποίηση δεδομένων αλληλεπίδρασης μπορεί να μετατραπεί σε χρήσιμο εργαλείο για τον εκπαιδευτικό — ένα εργαλείο που δεν αντικαθιστά την κρίση του, αλλά την εμπλουτίζει. Η συνολική αξία της έγκειται ακριβώς σε αυτή τη σύνδεση: τα Learning Analytics δεν χρειάζεται να περιορίζονται στην εκ των υστέρων περιγραφή της δραστηριότητας των φοιτητών, αλλά μπορούν να υποστηρίξουν ενεργά τον σχεδιασμό της εκπαιδευτικής διαδικασίας. Απομένουν, βέβαια, τεχνικές και ερευνητικές προκλήσεις — η εφαρμογή σε πραγματικά δεδομένα, η εκ των υστέρων παιδαγωγική αξιολόγηση και η ενσωμάτωση στην πλατφόρμα. Η μελέτη αυτή, ωστόσο, αναδεικνύει μια προοπτική που υπερβαίνει την τεχνική διάσταση της μεθοδολογίας, θέτοντας ένα ευρύτερο ερώτημα: κατά πόσο η εκπαιδευτική κοινότητα είναι έτοιμη να αξιοποιήσει κριτικά τέτοια εργαλεία, αναγνωρίζοντας τόσο τις δυνατότητες όσο και τα όριά τους, χωρίς να τα εκλαμβάνει ούτε ως πανάκεια ούτε ως απειλή;

Βιβλιογραφία

- Adomavicius, G., & Tuzhilin, A. (2005). Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17(6), 734–749. <https://doi.org/10.1109/TKDE.2005.99>
- Adomavicius, G., & Tuzhilin, A. (2011). Context-aware recommender systems. In F. Ricci, L. Rokach, B. Shapira, & P. B. Kantor (Eds.), *Recommender systems handbook* (pp. 217–253). Springer. https://doi.org/10.1007/978-0-387-85820-3_7
- Baker, R. S., & Inventado, P. S. (2014). Educational data mining and learning analytics. In J. A. Larusson & B. White (Eds.), *Learning analytics: From research to practice* (pp. 61–75). Springer. https://doi.org/10.1007/978-1-4614-3305-7_4
- Bastian, M., Heymann, S., & Jacomy, M. (2009). Gephi: An open source software for exploring and manipulating networks. *Proceedings of the International AAAI Conference on Web and Social Media*, 3(1), 361–362. <https://doi.org/10.1609/icwsm.v3i1.13937>
- Bellogín, A., Castells, P., & Cantador, I. (2014). Neighbor selection and weighting in user-based collaborative filtering: A performance prediction approach. *ACM Transactions on the Web*, 8(2), Article 12, 1–30. <https://doi.org/10.1145/2579993>
- Bonafini, F. C. (2018). Characterizing super-posters in a MOOC for teachers’ professional development. *Online Learning*, 22(4), 89–108. <https://doi.org/10.24059/olj.v22i4.1503>
- Borgatti, S. P. (2005). Centrality and network flow. *Social Networks*, 27(1), 55–71. <https://doi.org/10.1016/j.socnet.2004.11.008>
- Borgatti, S. P., & Everett, M. G. (1997). Network analysis of 2-mode data. *Social Networks*, 19(3), 243–269. [https://doi.org/10.1016/S0378-8733\(96\)00301-2](https://doi.org/10.1016/S0378-8733(96)00301-2)
- Brin, S., & Page, L. (1998). The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1–7), 107–117. [https://doi.org/10.1016/S0169-7552\(98\)00110-X](https://doi.org/10.1016/S0169-7552(98)00110-X)

X

- Burke, R. (2000). Knowledge-based recommender systems. In A. Kent (Ed.), *Encyclopedia of library and information science* (Vol. 69, Suppl. 32, pp. 180–200). Marcel Dekker.
- Burke, R. (2002). Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12(4), 331–370. <https://doi.org/10.1023/A:1021240730564>
- Burt, R. S. (1992). *Structural holes: The social structure of competition*. Harvard University Press. <https://doi.org/10.4159/9780674029095>
- Butera, F., & Buchs, C. (2019). Social interdependence and the promotion of cooperative learning. In K. Sassenberg & M. L. W. Vliek (Eds.), *Social psychology in action: Evidence-based interventions from theory to practice* (pp. 111–127). Springer. https://doi.org/10.1007/978-3-030-13788-5_8
- Cinelli, M., De Francisci Morales, G., Galeazzi, A., Quattrociocchi, W., & Starnini, M. (2021). The echo chamber effect on social media. *Proceedings of the National Academy of Sciences*, 118(9), e2023301118. <https://doi.org/10.1073/pnas.2023301118>
- Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). *Introduction to algorithms* (3rd ed.). MIT Press.
- de Laat, M., Lally, V., Lipponen, L., & Simons, R.-J. (2007). Investigating patterns of interaction in networked learning and computer-supported collaborative learning: A role for social network analysis. *International Journal of Computer-Supported Collaborative Learning*, 2(1), 87–103. <https://doi.org/10.1007/s11412-007-9006-4>
- de Nooy, W., Mrvar, A., & Batagelj, V. (2018). *Exploratory social network analysis with Pajek: Revised and expanded edition for updated software* (3rd ed.). Cambridge University Press. <https://doi.org/10.1017/9781108565691>
- Deutsch, M. (1949). A theory of co-operation and competition. *Human Relations*, 2(2), 129–152. <https://doi.org/10.1177/001872674900200204>
- Deutsch, M. (1962). Cooperation and trust: Some theoretical notes. In M. R. Jones (Ed.), *Nebraska symposium on motivation* (Vol. 10, pp. 275–319). University of Nebraska Press.

- Deutsch, M. (1968). Field theory in social psychology. In G. Lindzey & E. Aronson (Eds.), *The handbook of social psychology* (2nd ed., Vol. 1, pp. 412–487). Addison-Wesley.
- D’Mello, S. K., & Graesser, A. C. (2012). Dynamics of affective states during complex learning. *Learning and Instruction*, 22(2), 145–157. <https://doi.org/10.1016/j.learninstruc.2011.10.001>
- Dougiamas, M., & Taylor, P. C. (2003). Moodle: Using learning communities to create an open source course management system. In D. Lassner & C. McNaught (Eds.), *Proceedings of ED-MEDIA 2003—World Conference on Educational Multimedia, Hypermedia & Telecommunications* (pp. 171–178). Association for the Advancement of Computing in Education.
- Drachsler, H., Verbert, K., Santos, O. C., & Manouselis, N. (2015). Panorama of recommender systems to support learning. In F. Ricci, L. Rokach, & B. Shapira (Eds.), *Recommender systems handbook* (2nd ed., pp. 421–451). Springer. https://doi.org/10.1007/978-1-4899-7637-6_12
- Dunn, O. J. (1964). Multiple comparisons using rank sums. *Technometrics*, 6(3), 241–252. <https://doi.org/10.1080/00401706.1964.10490181>
- Ester, M., Kriegel, H.-P., Sander, J., & Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In E. Simoudis, J. Han, & U. Fayyad (Eds.), *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96)* (pp. 226–231). AAAI Press.
- Felfernig, A., & Burke, R. (2008). Constraint-based recommender systems: Technologies and research issues. In *Proceedings of the 10th International Conference on Electronic Commerce* (pp. 17–26). Association for Computing Machinery. <https://doi.org/10.1145/1409540.1409544>
- Field, A. (2013). *Discovering statistics using IBM SPSS statistics* (4th ed.). SAGE Publications.
- Fredricks, J. A., Blumenfeld, P. C., & Paris, A. H. (2004). School engagement: Potential of the concept, state of the evidence. *Review of Educational Research*, 74(1), 59–109. <https://doi.org/10.3102/00346543074001059>
- Freeman, L. C. (1977). A set of measures of centrality based on betweenness. *Sociometry*, 40(1), 35–41. <https://doi.org/10.2307/3033543>

- Freeman, L. C. (1979). Centrality in social networks: Conceptual clarification. *Social Networks*, 1(3), 215–239. [https://doi.org/10.1016/0378-8733\(78\)90021-7](https://doi.org/10.1016/0378-8733(78)90021-7)
- Grunspan, D. Z., Wiggins, B. L., & Goodreau, S. M. (2014). Understanding classrooms through social network analysis: A primer for social network analysis in education research. *CBE—Life Sciences Education*, 13(2), 167–178. <https://doi.org/10.1187/cbe.13-08-0162>
- Hair, J. F., Jr., Black, W. C., Babin, B. J., & Anderson, R. E. (2010). *Multivariate data analysis* (7th ed.). Pearson Prentice Hall.
- Hanneman, R. A., & Riddle, M. (2005). *Introduction to social network methods*. University of California, Riverside. <https://faculty.ucr.edu/~hanneman/nettext/>
- Hertzum, M., & Hornbæk, K. (2023). Frustration: Still a common user experience. *ACM Transactions on Computer-Human Interaction*, 30(3), Article 42. <https://doi.org/10.1145/3582432>
- Jacomy, M., Venturini, T., Heymann, S., & Bastian, M. (2014). ForceAtlas2, a continuous graph layout algorithm for handy network visualization designed for the Gephi software. *PLOS ONE*, 9(6), e98679. <https://doi.org/10.1371/journal.pone.0098679>
- Jain, A. K. (2010). Data clustering: 50 years beyond K-means. *Pattern Recognition Letters*, 31(8), 651–666. <https://doi.org/10.1016/j.patrec.2009.09.011>
- Jameson, A., & Smyth, B. (2007). Recommendation to groups. In P. Brusilovsky, A. Kobsa, & W. Nejdl (Eds.), *The adaptive web: Methods and strategies of web personalization* (pp. 596–627). Springer. https://doi.org/10.1007/978-3-540-72079-9_20
- Johnson, D. W., & Johnson, R. T. (1989). *Cooperation and competition: Theory and research*. Interaction Book Company.
- Johnson, D. W., & Johnson, R. T. (2009). An educational psychology success story: Social interdependence theory and cooperative learning. *Educational Researcher*, 38(5), 365–379. <https://doi.org/10.3102/0013189X09339057>
- Klein, J. T. (1998). *Computer response to user frustration* [Master's thesis, Massachusetts Institute of Technology]. DSpace@MIT. <https://hdl.handle.net/1721.1/61838>

- Kuhn, H. W. (1955). The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1–2), 83–97. <https://doi.org/10.1002/nav.3800020109>
- Lewin, K. (1948). *Resolving social conflicts: Selected papers on group dynamics* (G. W. Lewin, Ed.). Harper.
- Loganbill, J. (2025, December 3). *We're celebrating 500 million users on registered Moodle sites!* Moodle. <https://moodle.com/news/500-million-users-on-registered-moodle-sites/>
- Long, P., & Siemens, G. (2011). Penetrating the fog: Analytics in learning and education. *EDUCAUSE Review*, 46(5), 31–40. <https://er.educause.edu/articles/2011/9/penetrating-the-fog-analytics-in-learning-and-education>
- Manouselis, N., Drachsler, H., Vuorikari, R., Hummel, H., & Koper, R. (2011). Recommender systems in technology enhanced learning. In F. Ricci, L. Rokach, B. Shapira, & P. B. Kantor (Eds.), *Recommender systems handbook* (pp. 387–415). Springer. https://doi.org/10.1007/978-0-387-85820-3_12
- Maqtary, N., Mohsen, A., & Bechkoum, K. (2019). Group formation techniques in computer-supported collaborative learning: A systematic literature review. *Technology, Knowledge and Learning*, 24(2), 169–190. <https://doi.org/10.1007/s10758-017-9332-1>
- Mateos, P., & Bellogín, A. (2025). A systematic literature review of recent advances on context-aware recommender systems. *Artificial Intelligence Review*, 58, Article 20. <https://doi.org/10.1007/s10462-024-10939-4>
- McPherson, M., Smith-Lovin, L., & Cook, J. M. (2001). Birds of a feather: Homophily in social networks. *Annual Review of Sociology*, 27, 415–444. <https://doi.org/10.1146/annurev.soc.27.1.415>
- Montgomery, D. C., & Runger, G. C. (2011). *Applied statistics and probability for engineers* (5th ed.). Wiley.
- Moodle. (χ.χ.-a). *Grade export*. Ανακτήθηκε στις 29 Ιουνίου 2026 από https://docs.moodle.org/en/Grade_export
- Moodle. (χ.χ.-b). *Logs*. Ανακτήθηκε στις 29 Ιουνίου 2026 από <https://docs.moodle.org/en/Logs>

- Moodle. (χ.χ.-c). *Moodle architecture*. Ανακτήθηκε στις 29 Ιουνίου 2026 από https://docs.moodle.org/dev/Moodle_architecture
- Moodle. (χ.χ.-d). *The 'open source' way*. Ανακτήθηκε στις 29 Ιουνίου 2026 από <https://moodle.com/about/open-source/>
- Moodle. (χ.χ.-e). *Statistics*. Ανακτήθηκε στις 29 Ιουνίου 2026 από <https://stats.moodle.org/>
- Moreno, J., Ovalle, D. A., & Vicari, R. M. (2012). A genetic algorithm approach for group formation in collaborative learning considering multiple student characteristics. *Computers & Education*, 58(1), 560–569. <https://doi.org/10.1016/j.compedu.2011.09.011>
- Murtagh, F., & Contreras, P. (2012). Algorithms for hierarchical clustering: An overview. *WIREs Data Mining and Knowledge Discovery*, 2(1), 86–97. <https://doi.org/10.1002/widm.53>
- Newman, M. E. J. (2004). Analysis of weighted networks. *Physical Review E*, 70(5), 056131. <https://doi.org/10.1103/PhysRevE.70.056131>
- OECD/European Union/EC-JRC. (2008). *Handbook on constructing composite indicators: Methodology and user guide*. OECD Publishing. <https://doi.org/10.1787/9789264043466-en>
- Opsahl, T. (2013). Triadic closure in two-mode networks: Redefining the global and local clustering coefficients. *Social Networks*, 35(2), 159–167. <https://doi.org/10.1016/j.socnet.2011.07.001>
- Papert, S., & Harel, I. (1991). Situating constructionism. In I. Harel & S. Papert (Eds.), *Constructionism: Research reports and essays, 1985–1990* (pp. 1–11). Ablex Publishing Corporation.
- Pazzani, M. J., & Billsus, D. (2007). Content-based recommendation systems. In P. Brusilovsky, A. Kobsa, & W. Nejdl (Eds.), *The adaptive web: Methods and strategies of web personalization* (pp. 325–341). Springer. https://doi.org/10.1007/978-3-540-72079-9_10
- Pekrun, R. (2006). The control-value theory of achievement emotions: Assumptions, corollaries, and implications for educational research and practice. *Educational Psychology Review*, 18(4), 315–341. <https://doi.org/10.1007/s10648-006-9029-9>

- Rabbany, R., ElAtia, S., Takaffoli, M., & Zaïane, O. R. (2014). Collaborative learning of students in online discussion forums: A social network analysis perspective. In A. Peña-Ayala (Ed.), *Educational data mining: Applications and trends* (pp. 441–466). Springer. https://doi.org/10.1007/978-3-319-02738-8_16
- Rajendran, R., Iyer, S., Murthy, S., Wilson, C., & Sheard, J. (2013). A theory-driven approach to predict frustration in an ITS. *IEEE Transactions on Learning Technologies*, 6(4), 378–388. <https://doi.org/10.1109/TLT.2013.31>
- Redmond, P., Heffernan, A., Abawi, L., Brown, A., & Henderson, R. (2018). An online engagement framework for higher education. *Online Learning*, 22(1), 183–204. <https://doi.org/10.24059/olj.v22i1.1175>
- Ricci, F., Rokach, L., & Shapira, B. (2015). Recommender systems: Introduction and challenges. In F. Ricci, L. Rokach, & B. Shapira (Eds.), *Recommender systems handbook* (2nd ed., pp. 1–34). Springer. https://doi.org/10.1007/978-1-4899-7637-6_1
- Romero, C., & Ventura, S. (2010). Educational data mining: A review of the state of the art. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 40(6), 601–618. <https://doi.org/10.1109/TSMCC.2010.2053532>
- Romero, C., & Ventura, S. (2013). Data mining in education. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 3(1), 12–27. <https://doi.org/10.1002/widm.1075>
- Romero, C., & Ventura, S. (2020). Educational data mining and learning analytics: An updated survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 10(3), e1355. <https://doi.org/10.1002/widm.1355>
- Saqr, M. (2024). Group-level analysis of engagement poorly reflects individual students’ processes: Why we need idiographic learning analytics. *Computers in Human Behavior*, 150, 107991. <https://doi.org/10.1016/j.chb.2023.107991>
- Siemens, G. (2005). Connectivism: A learning theory for the digital age. *International Journal of Instructional Technology and Distance Learning*, 2(1), 3–10. https://www.itdl.org/Journal/Jan_05/article01.htm

- Siemens, G. (2013). Learning analytics: The emergence of a discipline. *American Behavioral Scientist*, 57(10), 1380–1400. <https://doi.org/10.1177/0002764213498851>
- Strauß, S., & Rummel, N. (2021). Promoting regulation of equal participation in online collaboration by combining a group awareness tool and adaptive prompts. But does it even matter? *International Journal of Computer-Supported Collaborative Learning*, 16(1), 67–104. <https://doi.org/10.1007/s11412-021-09340-y>
- Sunstein, C. R. (2001). *Echo chambers: Bush v. Gore, impeachment, and beyond*. Princeton University Press.
- Su, X., & Khoshgoftaar, T. M. (2009). A survey of collaborative filtering techniques. *Advances in Artificial Intelligence*, 2009, Article 421425. <https://doi.org/10.1155/2009/421425>
- Tsoni, R., Koutouzis, M., & Verykios, V. S. (2026). When clicks reveal student frustration: A sensor-free analytics approach with adaptive thresholds. *Διεθνές Συνέδριο για την Ανοικτή & εξ Αποστάσεως Εκπαίδευση*, 13(1), 214–229. <https://doi.org/10.12681/icodl.8654>
- Tsoni, R., Panagiotakopoulos, C. T., & Verykios, V. S. (2022). Revealing latent traits in the social behavior of distance learning students. *Education and Information Technologies*, 27(3), 3529–3565. <https://doi.org/10.1007/s10639-021-10742-6>
- Tsoni, R., Sakkopoulos, E., & Verykios, V. S. (2022). Revealing latent student traits in distance learning through SNA and PCA. In M. Ivanović, A. Klačnja-Milićević, & L. C. Jain (Eds.), *Handbook on intelligent techniques in the educational process* (pp. 185–209). Springer. https://doi.org/10.1007/978-3-031-04662-9_10
- Tsoni, R., Samaras, C. S., Paxinou, E., Panagiotakopoulos, C. T., & Verykios, V. S. (2019). From analytics to cognition: Expanding the reach of data in learning. In *Proceedings of the 11th International Conference on Computer Supported Education (CSEDU 2019)* (Vol. 2, pp. 458–465). SCITEPRESS. <https://doi.org/10.5220/0007751904580465>
- Verbert, K., Manouselis, N., Ochoa, X., Wolpers, M., Drachsler, H., Bosnić, I., & Duval, E. (2012). Context-aware recommender systems for learning: A survey and future challenges. *IEEE Transactions on Learning Technologies*, 5(4), 318–335. <https://doi.org/10.1109/TLT.2012.11>

- Vignery, K., & Laurier, W. (2020). A methodology and theoretical taxonomy for centrality measures: What are the best centrality indicators for student networks? *PLOS ONE*, 15(12), e0244377. <https://doi.org/10.1371/journal.pone.0244377>
- Wasserman, S., & Faust, K. (1994). *Social network analysis: Methods and applications*. Cambridge University Press. <https://doi.org/10.1017/CBO9780511815478>
- Wenger, E. (1998). *Communities of practice: Learning, meaning, and identity*. Cambridge University Press. <https://doi.org/10.1017/CBO9780511803932>
- York, T. T., Gibson, C., & Rankin, S. (2015). Defining and measuring academic success. *Practical Assessment, Research, and Evaluation*, 20(1), Article 5. <https://doi.org/10.7275/hz5x-tx03>
- Zambrano, A. F., Baker, R. S., Baral, S., Heffernan, N. T., & Lan, A. (2024). From reaction to anticipation: Predicting future affect. In B. Paaßen & C. D. Epp (Eds.), *Proceedings of the 17th International Conference on Educational Data Mining* (pp. 566–574). International Educational Data Mining Society. <https://doi.org/10.5281/zenodo.12729886>

Παράρτημα Α

Πηγαίος Κώδικας Υλοποίησης (Python Scripts)

Στο παρόν παράρτημα παρατίθεται ο πλήρης πηγαίος κώδικας που αναπτύχθηκε για την υλοποίηση της μεθοδολογίας της διπλωματικής εργασίας. Περιλαμβάνονται 22 αρχεία Python, οργανωμένα σύμφωνα με τη λογική σειρά του pipeline: προεπεξεργασία, εξαγωγή χαρακτηριστικών, Ανάλυση Κοινωνικών Δικτύων, ανίχνευση σημάτων απογοήτευσης, σύνθεση και στρωματοποίηση του προφίλ, ανάλυση ομοφιλίας, σύσταση ομάδων και πειραματική αξιολόγηση.

Οι ενότητες χαρακτηρίζονται ως Κύριο, Βοηθητικό ή Έλεγχος. Τα κύρια scripts αποτελούν την αναγκαία υπολογιστική αλυσίδα, ενώ τα βοηθητικά scripts παράγουν οπτικοποιήσεις ή αρχεία Gephi και τα scripts ελέγχου τεκμηριώνουν την ποιότητα, τη συνέπεια και την ευστάθεια των αποτελεσμάτων.

Περιβάλλον εκτέλεσης και οργάνωση αρχείων

Έκδοση Python: Python 3.10 ή νεότερη έκδοση.

Βιβλιοθήκες: pandas, numpy, networkx, matplotlib, scipy, scikit-posthocs και openpyxl.

Δομή φακέλου: Τα scripts και τα πρωτογενή αρχεία εισόδου τοποθετούνται στον ίδιο βασικό φάκελο. Τα scripts δημιουργούν αυτόματα τους επιμέρους φακέλους αποτελεσμάτων όπου αυτό προβλέπεται.

Συνοπτική σειρά εκτέλεσης

A/A	Script	Στάδιο	Χαρακτήρας
A.1	00_xlsx_to_csv.py	Προετοιμασία δεδομένων	Κύριο
A.2	01_filtering_quality_checks.py	Προεπεξεργασία	Κύριο
A.3	02_04_2_1_extract_participation_features.py	Δείκτες συμμετοχής	Κύριο
A.4	03_04_2_2_extract_academic_performance_features.py	Δείκτες επίδοσης	Κύριο
A.5	04_sna_423_postcreated_edges_roles.py	SNA - διμερές δίκτυο	Κύριο
A.6	04b_sna_423_PREPARING_NODES_EDGES_FOR_GEPHI.py	SNA - οπτικοποίηση διμερούς δικτύου	Βοηθητικό
A.7	04c_4_2_3_student_student_weighted_pro.py	SNA - προβολή και μετρικές	Κύριο
A.8	04d_4_2_3_visualize_sna_metrics.py	SNA - περιγραφική οπτικοποίηση	Βοηθητικό
A.9	04e_4_2_3_assign_social_roles.py	SNA - κοινωνικοί ρόλοι	Κύριο
A.10	04f_4_2_3_validate_social_roles_kruskal_dunn.py	SNA - στατιστική επικύρωση	Έλεγχος
A.11	04g_4_2_3_prepare_gephi_projected_roles_visualization.py	SNA - οπτικοποίηση προβαλλόμενου δικτύου	Βοηθητικό

A.12	06_frustration_signals.py	Σήματα απογοήτευσης	Κύριο
A.13	07_04_2_5_final_student_profile.py	Σύνθεση τελικού προφίλ	Κύριο
A.14	07b_04_2_5_profile_visualization_REVISED.py	Τελικό προφίλ - οπτικοποίηση	Βοηθητικό
A.15	07c_04_2_5_numerical_validation_report.py	Τελικό προφίλ - αριθμητικός έλεγχος	Έλεγχος
A.16	08_04_2_6_student_profile_stratification.py	Στρωματοποίηση	Κύριο
A.17	08b_04_3_strata_visualization_REVISED.py	Στρωματοποίηση - οπτικοποίηση	Βοηθητικό
A.18	08c_04_3_2_weight_sensitivity_analysis.py	Στρωματοποίηση - ανάλυση ευαισθησίας	Έλεγχος
A.19	09_HOMOPHILY_analysis.py	Ανάλυση ομοφιλίας	Κύριο
A.20	10_05_3_grouping_ALGORITHM_v2.py	Σύσταση ομάδων	Κύριο
A.21	11_05_4_experimental_evaluation.py	Συγκριτική αξιολόγηση	Κύριο
A.22	12_05_4_repeated_random_evaluation.py	Ανάλυση ευστάθειας αξιολόγησης	Κύριο

Ενδεικτικές εντολές για τα τελικά στάδια:

```
python 10_05_3_grouping_ALGORITHM_v2.py --all-sizes
python 11_05_4_experimental_evaluation.py --all-sizes
python 12_05_4_repeated_random_evaluation.py --all-sizes --random-runs 100
```

A.1. 00_xlsx_to_csv.py

Χαρακτήρας: Κύριο script - Προετοιμασία δεδομένων

Περιγραφή: Μετατρέπει το συνθετικό αρχείο καταγραφής δραστηριοτήτων από μορφή Excel σε CSV, ώστε να χρησιμοποιηθεί από τα επόμενα στάδια του pipeline.

Είσοδοι: Logfile_synthetic_userids.xlsx

Έξοδοι: Logfile_synthetic_userids.csv

Πηγαίος κώδικας

```
from pathlib import Path
import pandas as pd

# φάκελος όπου βρίσκεται αυτό το script
CURRENT_DIR = Path(__file__).resolve().parent

input_file = CURRENT_DIR / "Logfile_synthetic_userids.xlsx"
output_file = CURRENT_DIR / (input_file.stem + ".csv")

print("Διαβάζω:", input_file)

if not input_file.exists():
    raise FileNotFoundError(f"Δεν βρέθηκε το αρχείο: {input_file}")

df = pd.read_excel(input_file, engine="openpyxl")

print("Γραμμές / στήλες:", df.shape)

df.to_csv(output_file, index=False, encoding="utf-8-sig")

print("Αποθηκεύτηκε:", output_file)
```

A.2. 01_filtering_quality_checks.py

Χαρακτήρας: Κύριο script - Προεπεξεργασία

Περιγραφή: Φιλτράρει τις εγγραφές του log file, διατηρώντας ενέργειες με Origin=web και αναγνωριστικά φοιτητών της μορφής user###. Ελέγχει την εγκυρότητα των timestamps, τη συνέπεια Description–User και τη μοναδικότητα της αντιστοίχισης χρηστών.

Είσοδοι: Logfile_synthetic_userid.csv

Έξοδοι: Logfile_filtered_OK.csv · Logfile_filtered_ISSUES.csv · Report_4_1_3.csv · Report_4_1_3.xlsx

Πηγαίος κώδικας

```
import pandas as pd
import re
from pathlib import Path

CURRENT_DIR = Path(__file__).resolve().parent
print("Running from:", CURRENT_DIR)
# =====
# PYΘΙΣΕΙΣ
# =====
input_file = CURRENT_DIR / "Logfile_synthetic_userid.csv"
print("Reading:", input_file)

COL_TIME = "Time"
COL_USER = "User full name"
COL_ORIGIN = "Origin"
COL_DESC = "Description"

# Φοιτητές: user###
student_pattern = re.compile(r"^user\d{3}$", re.IGNORECASE)

# Actor μέσα στο Description: π.χ. "The user with id 'user107' ..."
actor_pattern = re.compile(r"user\s-with\s+id\s*['\"]?(user\d{3})['\"]?", re.IGNORECASE)

# Μορφή του Time στο αρχείο (π.χ. 5/09/24, 23:53)
TIME_FORMAT = "%d/%m/%y, %H:%M"

# =====
# 1) ΔΙΑΒΑΣΜΑ ΟΛΟΥ ΤΟΥ ΑΡΧΕΙΟΥ
# =====
df_full = pd.read_csv(input_file, encoding="utf-8-sig")
print("Γραμμές (όπως διαβάστηκαν):", len(df_full))

# =====
# 2) ΠΙΝΑΚΑΣ ΓΙΑ ΕΛΕΓΧΟΥΣ/ΦΙΛΤΡΑ
# =====
df = df_full[[COL_TIME, COL_USER, COL_ORIGIN, COL_DESC]].copy()

# =====
# 3) ΦΙΛΤΡΑΡΙΣΜΑ Origin = web (πρώτα)
# =====
df[COL_ORIGIN] = df[COL_ORIGIN].fillna("").astype(str)
mask_web = df[COL_ORIGIN].str.lower().eq("web")
df_web = df[mask_web].copy()
print("Μετά το origin=web:", len(df_web))

# =====
# 4) ΦΙΛΤΡΑΡΙΣΜΑ ΦΟΙΤΗΤΩΝ user###
# =====
df_web[COL_USER] = df_web[COL_USER].fillna("").astype(str)
mask_students = df_web[COL_USER].str.match(student_pattern)
df_students = df_web[mask_students].copy()
print("Μετά το φιλτράρισμα φοιτητών:", len(df_students))
print("Μοναδικοί φοιτητές:", df_students[COL_USER].nunique())

# =====
# 5) ΕΛΕΓΧΟΣ TIMESTAMP (parse -> valid)
# =====
df_students["parsed_time"] = pd.to_datetime(
    df_students[COL_TIME],
    format=TIME_FORMAT,
    errors="coerce"
)
```

Διπλωματική Εργασία

```

invalid_time = df_students["parsed_time"].isna()

# =====
# 6) CONSISTENCY Description (actor == User full name)
# =====
df_students[COL_DESC] = df_students[COL_DESC].fillna("").astype(str)
df_students["actor_user"] = df_students[COL_DESC].str.extract(actor_pattern, expand=False)

missing_actor = df_students["actor_user"].isna()
mismatch_actor = (~missing_actor) & (
    df_students["actor_user"].str.lower() != df_students[COL_USER].str.lower()
)

# =====
# 7) UNIQUENESS (κάθε user -> 1 actor_user)
# =====
map_counts = df_students.groupby(COL_USER)["actor_user"].nunique(dropna=True)
bad_users = map_counts[map_counts > 1].index
uniqueness_problem = df_students[COL_USER].isin(bad_users)

# =====
# 8) ISSUE MASK + ΕΠΙΛΟΓΗ ΓΡΑΜΜΩΝ (σε επίπεδο index)
# =====
issue_mask = invalid_time | missing_actor | mismatch_actor | uniqueness_problem

idx_ok = df_students.index[~issue_mask]
idx_issues = df_students.index[issue_mask]

# =====
# 9) ΤΕΛΙΚΑ DATAFRAMES
# =====
df_ok_full = df_full.loc[idx_ok].copy()
df_issues_full = df_full.loc[idx_issues].copy()

issue_reason = pd.Series("", index=df_students.index)
issue_reason.loc[invalid_time] += "invalid_timestamp;"
issue_reason.loc[missing_actor] += "missing_actor_in_description;"
issue_reason.loc[mismatch_actor] += "actor_mismatch_user_full_name;"
issue_reason.loc[uniqueness_problem] += "uniqueness_mapping_problem;"
df_issues_full["issue"] = issue_reason.loc[idx_issues].values

# =====
# 10) ΑΠΟΘΗΚΕΥΣΗ
# =====
output_ok = CURRENT_DIR / "Logfile_filtered_OK.csv"
output_issues = CURRENT_DIR / "Logfile_filtered_ISSUES.csv"
output_report = CURRENT_DIR / "Report_4_1_3.csv"
output_report_xlsx = CURRENT_DIR / "Report_4_1_3.xlsx"

df_ok_full.to_csv(output_ok, index=False, encoding="utf-8-sig")
df_issues_full.to_csv(output_issues, index=False, encoding="utf-8-sig")

print("\nΑποθηκεύτηκαν:")
print(output_ok)
print(output_issues)
print(output_report)

report = pd.DataFrame([
    {"Stage": "Raw (full schema)", "Rows": len(df_full)},
    {"Stage": "Origin=web", "Rows": int(mask_web.sum())},
    {"Stage": "Students user###", "Rows": len(df_students), "Unique students": df_students[COL_USER].nunique()},
    {"Stage": "Final OK (full schema)", "Rows": len(df_ok_full), "Unique students": df_ok_full[COL_USER].nunique()},
    {"Stage": "Issues (full schema)", "Rows": len(df_issues_full)}
])
report.to_excel(output_report_xlsx, index=False)
report.to_csv(output_report, index=False, encoding="utf-8-sig")

print("\nΤΕΛΙΚΑ ΑΠΟΤΕΛΕΣΜΑΤΑ")
print("-----")
print("OK εγγραφές:", len(df_ok_full))
print("Issues εγγραφές:", len(df_issues_full))
print("\nΔημιουργήθηκαν αρχεία:")
print("- Logfile_filtered_OK.csv (ίδια δομή στηλών με το αρχικό)")
print("- Logfile_filtered_ISSUES.csv (ίδια δομή στηλών + στήλη issue)")
print("- Report_4_1_3.csv")
print("- Report_4_1_3.xlsx")

```

A.3. 02_04_2_1_extract_participation_features.py

Χαρακτήρας: Κύριο script - Δείκτες συμμετοχής

Περιγραφή: Εξάγει ανά φοιτητή τους δείκτες συμμετοχής που χρησιμοποιούνται στο πολυδιάστατο προφίλ: συνολικά events, δημοσιεύσεις και προβολές forum, προσπάθειες quiz, υποβολές εργασιών, προβολές πόρων και μαθήματος, ενεργές ημέρες, μέση δραστηριότητα ανά ενεργή ημέρα και λόγο εμπλοκής στο forum.

Είσοδοι: Logfile_filtered_OK.csv

Έξοδοι: 04.2.1_participation_features.csv · 04.2.1_participation_quality_report.xlsx

Πηγαίος κώδικας

```
"""
04_2_1_extract_participation_features.py
Algorithm 4.2.1 - Εξαγωγή Δεικτών Συμμετοχής (Participation Features)

Είσοδος:
- Logfile_filtered_OK.csv (φιλτραρισμένο, μόνο φοιτητές)

Έξοδος:
- 04.2.1_participation_features.csv
- 04.2.1_participation_quality_report.xlsx (quality control report)

Παραγόμενοι δείκτες (ανά φοιτητή):
- user_id
- total_events
- forum_posts
- forum_views
- quiz_attempts
- assignment_submissions
- resource_views
- course_views
- active_days
- avg_events_per_active_day
- forum_engagement_ratio
"""

import sys
import numpy as np
import pandas as pd

# -----
# 0) Ρυθμίσεις / Παράμετροι
# -----
INPUT_PATH = "Logfile_filtered_OK.csv"
OUTPUT_PATH = "04.2.1_participation_features.csv"
QC_REPORT_PATH = "04.2.1_participation_quality_report.xlsx"

USER_COL = "User full name"
TIME_COL = "Time"
EVENT_COL = "Event name"
COMP_COL = "Component"

TIME_FORMAT = "%d/%m/%y, %H:%M"

# Expanded "content modules" (you can edit this set if needed)
RESOURCE_COMPONENTS = {"File", "Folder", "URL", "Glossary", "Page", "Book"}

QUIZ_ATTEMPT_EVENTS = {"Quiz attempt started", "Quiz attempt submitted"}

def require_columns(df: pd.DataFrame, cols: list[str]) -> None:
    missing = [c for c in cols if c not in df.columns]
    if missing:
        raise ValueError(
            f"Λείπουν στήλες από το αρχείο: {missing}\n"
            f"Υπάρχουσες στήλες: {df.columns.tolist()}"
        )

def load_logs(path: str) -> pd.DataFrame:
    df = pd.read_csv(path, encoding="utf-8-sig")
```

Διπλωματική Εργασία

```

require_columns(df, [USER_COL, TIME_COL, EVENT_COL, COMP_COL])
return df

def add_datetime_columns(df: pd.DataFrame) -> pd.DataFrame:
    out = df.copy()
    out["datetime"] = pd.to_datetime(out[TIME_COL], format=TIME_FORMAT, errors="coerce")
    out = out.dropna(subset=["datetime"]).copy()
    out["date"] = out["datetime"].dt.date
    return out

def compute_participation_features(df: pd.DataFrame) -> pd.DataFrame:
    event = df[EVENT_COL].astype(str)
    comp = df[COMP_COL].astype(str)

    # Forum
    mask_forum = comp.eq("Forum")
    mask_forum_posts = mask_forum & (
        event.str.contains(r"\bposted\b", case=False, na=False)
        | event.str.contains(r"\bcreated\b", case=False, na=False)
    )
    mask_forum_views = mask_forum & event.str.contains(r"\bviewed\b", case=False, na=False)

    # Quiz
    mask_quiz_attempts = comp.eq("Quiz") & event.isin(QUIZ_ATTEMPT_EVENTS)

    # Assignment submissions (submitted only)
    mask_assignment_submissions = comp.eq("Assignment") & event.str.contains(
        r"\bsubmitted\b", case=False, na=False
    )

    # Resource views (content access)
    mask_resource_views = event.eq("Course module viewed") & comp.isin(RESOURCE_COMPONENTS)

    # Course views (navigation)
    mask_course_views = event.eq("Course viewed")

    # 0/1 columns
    work = df[[USER_COL, EVENT_COL, "date"]].copy()
    work["forum_posts"] = mask_forum_posts.astype(np.int32)
    work["forum_views"] = mask_forum_views.astype(np.int32)
    work["quiz_attempts"] = mask_quiz_attempts.astype(np.int32)
    work["assignment_submissions"] = mask_assignment_submissions.astype(np.int32)
    work["resource_views"] = mask_resource_views.astype(np.int32)
    work["course_views"] = mask_course_views.astype(np.int32)

    agg = (
        work.groupby(USER_COL, dropna=False)
        .agg(
            total_events=(EVENT_COL, "count"),
            forum_posts=("forum_posts", "sum"),
            forum_views=("forum_views", "sum"),
            quiz_attempts=("quiz_attempts", "sum"),
            assignment_submissions=("assignment_submissions", "sum"),
            resource_views=("resource_views", "sum"),
            course_views=("course_views", "sum"),
            active_days=("date", "nunique"),
        )
        .reset_index()
        .rename(columns={USER_COL: "user_id"})
    )

    denom = agg["active_days"].replace(0, np.nan)
    agg["avg_events_per_active_day"] = (agg["total_events"] / denom).round(2).fillna(0)
    agg["forum_engagement_ratio"] = (agg["forum_posts"] / (agg["forum_views"] + 1)).round(3)

    # Optional consistency check: if someone has events, they should have >=1 active day
    # (kept as warning, not fatal)
    inconsistent = agg.query("total_events > 0 and active_days == 0")
    if len(inconsistent) > 0:
        print("WARNING: Found students with total_events>0 but active_days==0:")
        print(inconsistent[["user_id", "total_events", "active_days"]].head(10))

    return agg

def quality_checks_and_report(participation: pd.DataFrame) -> None:
    print("Students:", len(participation))

    cols = ["total_events", "forum_posts", "forum_views", "quiz_attempts", "assignment_submissions"]
    desc = participation[cols].describe()
    print(desc)

    top_assign = (
        participation.sort_values("assignment_submissions", ascending=False)
        [["user_id", "assignment_submissions", "total_events"]]
    )

```

```

        .head(10)
    )

    submissions_no_quiz = (
        participation.query("assignment_submissions > 0 and quiz_attempts == 0")
        [["user_id", "assignment_submissions", "quiz_attempts", "total_events"]]
        .head(50)
    )

    low_activity = (
        participation.query("total_events <= 5")
        [["user_id", "total_events", "active_days", "course_views", "resource_views"]]
        .sort_values("total_events")
        .head(50)
    )

    # Save QC report
    with pd.ExcelWriter(QC_REPORT_PATH, engine="openpyxl") as writer:
        desc.to_excel(writer, sheet_name="summary_stats")
        top_assign.to_excel(writer, sheet_name="top_assignment_subs", index=False)
        submissions_no_quiz.to_excel(writer, sheet_name="subs_no_quiz", index=False)
        low_activity.to_excel(writer, sheet_name="low_activity", index=False)

    print("Saved QC report:", QC_REPORT_PATH)

def main() -> int:
    try:
        df = load_logs(INPUT_PATH)
        df = add_datetime_columns(df)

        participation = compute_participation_features(df)

        # QC + report
        quality_checks_and_report(participation)

        # Save main features
        participation.to_csv(OUTPUT_PATH, index=False, encoding="utf-8-sig")
        print("Saved:", OUTPUT_PATH)
        return 0

    except Exception as e:
        print("ERROR:", e, file=sys.stderr)
        return 1

if __name__ == "__main__":
    raise SystemExit(main())

```

A.4. 03_04_2_2_extract_academic_performance_features.py

Χαρακτήρας: Κύριο script - Δείκτες επίδοσης

Περιγραφή: Υπολογίζει τον βαθμό τελικής εξέτασης, το πλήθος ολοκληρωμένων εργασιών και τον μέσο βαθμό των παραδοθεισών εργασιών. Παράλληλα ελέγχει τη συμβατότητα των αναγνωριστικών με το αρχείο συμμετοχής.

Είσοδοι: Grades_synthetic_same_students.xlsx · 04.2.1_participation_features.csv

Έξοδοι: 04.2.2_academic_performance_features_MIN.csv · 04.2.2_performance_id_check.xlsx

Πηγαίος κώδικας

```
"""
04_2_2_extract_academic_performance_features.py
Algorithm 4.2.2 - Εξαγωγή Δεικτών Ακαδημαϊκής Επίδοσης
Στόχος:
    Δημιουργία ενός απλού πίνακα επίδοσης (performance features) ανά φοιτητή,
    με μόνο τα απαραίτητα στοιχεία που θα ενσωματωθούν στο πολυδιάστατο προφίλ.

Είσοδοι:
    - Grades_synthetic_same_students.xlsx
    - 04.2.1_participation_features.csv (μόνο για έλεγχο συμβατότητας IDs)

Έξοδοι:
    - 04.2.2_academic_performance_features_MIN.csv
    - 04.2.2_performance_id_check.xlsx

Περιεχόμενο MIN αρχείου (ανά φοιτητή):
    - user_id
    - final_exam
    - completed_assignments
    - avg_assignments_nonzero
"""

import sys
import numpy as np
import pandas as pd

# -----
# 0) Paths / Columns
# -----
GRADES_INPUT = "Grades_synthetic_same_students.xlsx"
PARTICIPATION_INPUT = "04.2.1_participation_features.csv"

OUTPUT_MIN = "04.2.2_academic_performance_features_MIN.csv"
OUTPUT_IDCHECK = "04.2.2_performance_id_check.xlsx"

USER_COL_IN = "Student ID"
FINAL_COL_IN = "Final Exam"
ASSIGNMENT_COLS_IN = ["WA1", "WA2", "WA3", "WA4", "WA5"]

# -----
# 1) Helpers
# -----
def require_columns(df: pd.DataFrame, cols: list[str]) -> None:
    missing = [c for c in cols if c not in df.columns]
    if missing:
        raise ValueError(
            f"Λείπουν στήλες: {missing}\n"
            f"Υπάρχουσες στήλες: {df.columns.tolist()}"
        )

def safe_to_numeric(df: pd.DataFrame, cols: list[str]) -> pd.DataFrame:
    out = df.copy()
    for c in cols:
        out[c] = pd.to_numeric(out[c], errors="coerce")
    return out

# -----
# 2) Load grades
# -----
def load_grades(path: str) -> pd.DataFrame:
    df = pd.read_excel(path, engine="openpyxl")
```

Διπλωματική Εργασία

```

require_columns(df, [USER_COL_IN, FINAL_COL_IN] + ASSIGNMENT_COLS_IN)

df = df.rename(columns={USER_COL_IN: "user_id", FINAL_COL_IN: "final_exam"})
df["user_id"] = df["user_id"].astype(str).str.strip()

df = safe_to_numeric(df, ASSIGNMENT_COLS_IN + ["final_exam"])
return df

# -----
# 3) Compute minimal performance features
# -----
def compute_min_features(grades: pd.DataFrame) -> pd.DataFrame:
    assignment = grades[ASSIGNMENT_COLS_IN].copy()

    # completed assignments: πόσες εργασίες έχουν βαθμό > 0
    completed = (assignment > 0).sum(axis=1)

    # avg assignments (non-zero): μέσος όρος μόνο για παραδοθείσες (0 ως "μη παράδοση")
    avg_nonzero = assignment.replace(0, np.nan).mean(axis=1).round(2).fillna(0)

    out = pd.DataFrame({
        "user_id": grades["user_id"],
        "final_exam": grades["final_exam"],
        "completed_assignments": completed.astype(int),
        "avg_assignments_nonzero": avg_nonzero
    })

    return out

# -----
# 4) ID compatibility check
# -----
def id_check(participation_path: str, min_features: pd.DataFrame) -> pd.DataFrame:
    p = pd.read_csv(participation_path, encoding="utf-8-sig")
    if "user_id" not in p.columns:
        raise ValueError("To participation αρχείο δεν έχει στήλη 'user_id'.")

    participation_users = set(p["user_id"].astype(str).str.strip())
    grades_users = set(min_features["user_id"].astype(str).str.strip())

    summary = pd.DataFrame({
        "metric": ["grades_users", "participation_users", "common_users",
                  "missing_in_grades", "missing_in_participation"],
        "count": [
            len(grades_users),
            len(participation_users),
            len(grades_users & participation_users),
            len(participation_users - grades_users),
            len(grades_users - participation_users),
        ]
    })

    missing_in_grades = sorted(list(participation_users - grades_users))
    missing_in_participation = sorted(list(grades_users - participation_users))

    # Save details in one file (two sheets)
    with pd.ExcelWriter(OUTPUT_IDCHECK, engine="openpyxl") as writer:
        summary.to_excel(writer, sheet_name="id_check_summary", index=False)
        pd.DataFrame({"user_id": missing_in_grades}).to_excel(writer, sheet_name="missing_in_grades", index=False)
        pd.DataFrame({"user_id": missing_in_participation}).to_excel(writer, sheet_name="missing_in_participation", index=False)

    print("Saved:", OUTPUT_IDCHECK)
    print(summary.to_string(index=False))

    return summary

# -----
# 5) Main
# -----
def main() -> int:
    try:
        grades = load_grades(GRADES_INPUT)
        min_features = compute_min_features(grades)

        # Save minimal features
        min_features.to_csv(OUTPUT_MIN, index=False, encoding="utf-8-sig")
        print("Saved:", OUTPUT_MIN)

        # ID compatibility (keeps the pipeline defensible)
        id_check(PARTICIPATION_INPUT, min_features)

    return 0

```

```
except Exception as e:
    print("ERROR:", e, file=sys.stderr)
    return 1

if __name__ == "__main__":
    raise SystemExit(main())
```

A.5. 04_sna_423_postcreated_edges_roles.py

Χαρακτήρας: Κύριο script - SNA - διμερές δίκτυο

Περιγραφή: Εντοπίζει μόνο ενεργητικές συνεισφορές στο forum, αφαιρεί διπλές καταμετρήσεις του ίδιου post και κατασκευάζει τις σταθμισμένες ακμές φοιτητή-συζήτησης του διμερούς δικτύου.

Είσοδοι: Logfile_filtered_OK.csv

Έξοδοι: edges_gephi_active_forum.csv · active_forum_audit.csv · unparsed_active_forum_samples.csv (μόνο εφόσον υπάρχουν μη αναγνωρισμένες εγγραφές)

Πηγαίος κώδικας

```
# 04_sna_423_postcreated_edges_roles.py
# -----
# Εξάγει ΜΟΝΟ ενεργητική συμμετοχή στο forum:
#   - post created
#   - discussion created
#   - some content has been posted.
#
# Αγνοεί:
#   - views
#   - subscriptions
#   - updates
#   - deletes
#   - οποιαδήποτε άλλη μη παραγωγική ενέργεια
#
# Output:
#   edges_gephi_active_forum.csv
#
# Στήλες:
#   Source, Target, Weight_active, W_post_created, W_discussion_created, W_some_content_posted
#
# To Weight_active = συνολικές ενεργητικές συνεισφορές στο thread
# -----

import re
import csv
from collections import defaultdict, Counter
from pathlib import Path
import pandas as pd

# =====
# CONFIG
# =====
INPUT_CSV = "Logfile_filtered_OK.csv"
OUTPUT_EDGES = "edges_gephi_active_forum.csv"
OUTPUT_AUDIT = "active_forum_audit.csv"
OUTPUT_UNPARSED = "unparsed_active_forum_samples.csv"

FORUM_COMPONENT = "Forum"

ACCEPT_EVENTS = {
    "post created",
    "discussion created",
    "some content has been posted.",
}

# =====
# REGEX
# =====
USER_RE = re.compile(r"user with id 'user(\d+)'", re.IGNORECASE)
DISCUSSION_RE = re.compile(r"discussion(?: with id)? '(\d+)'", re.IGNORECASE)
POST_ID_RE = re.compile(r"(?:forum\s+)?post with id '(\d+)'", re.IGNORECASE)

def extract_ids(description: str):
    """
    Επιστρέφει (user_id, discussion_id, post_id_or_None)
    """
    if description is None:
        return None, None, None

    text = str(description)

    um = USER_RE.search(text)
    dm = DISCUSSION_RE.search(text)
    pm = POST_ID_RE.search(text)
```

```

user_id = um.group(1) if um else None
discussion_id = dm.group(1) if dm else None
post_id = pm.group(1) if pm else None

return user_id, discussion_id, post_id

def classify_action(event_name: str):
    """
    Κατηγοριοποίηση μόνο των ενεργητικών forum actions που θέλουμε.
    """
    ev = (event_name or "").strip().lower()

    if ev == "post created":
        return "post_created"
    if ev == "discussion created":
        return "discussion_created"
    if ev == "some content has been posted.":
        return "some_content_posted"

    return None

def main():
    script_dir = Path(__file__).parent
    input_path = script_dir / INPUT_CSV
    edges_path = script_dir / OUTPUT_EDGES
    audit_path = script_dir / OUTPUT_AUDIT
    unparsed_path = script_dir / OUTPUT_UNPARSED

    # weights ανά (user, discussion)
    edge_action_counts = defaultdict(lambda: {
        "post_created": 0,
        "discussion_created": 0,
        "some_content_posted": 0,
    })

    # dedupe:
    # για posts: (user_id, post_id) ώστε να μην μετρηθεί 2 φορές το ίδιο post
    # όταν εμφανίζεται και ως "post created" και ως "some content has been posted."
    seen_posts = set() # (user_id, post_id)

    # για discussion created: 1 φορά ανά (user, discussion)
    seen_discussions = set() # (user_id, discussion_id)

    # audit
    total_rows = 0
    forum_rows = 0
    accepted_rows = 0
    parsed_ok = 0
    parsed_fail = 0
    dedup_skips = 0

    per_event_detected = Counter()
    per_action_counted = Counter()
    per_event_fail = Counter()
    per_event_dedup = Counter()

    unparsed_samples = []

    with open(input_path, "r", encoding="utf-8-sig", newline="") as f:
        reader = csv.DictReader(f)

        required = ["Component", "Event name", "Description"]
        missing = [c for c in required if c not in reader.fieldnames]
        if missing:
            raise ValueError(f"Missing required columns: {missing}\nFound columns: {reader.fieldnames}")

        for rownum, row in enumerate(reader, start=2):
            total_rows += 1

            comp = row["Component"]
            if comp is None or str(comp).strip() != FORUM_COMPONENT:
                continue
            forum_rows += 1

            ev_raw = row["Event name"]
            ev_text = "" if ev_raw is None else str(ev_raw)
            ev_norm = ev_text.strip().lower()

            action = classify_action(ev_text)
            if action is None:
                continue

            accepted_rows += 1
            per_event_detected[ev_norm] += 1

```

```

desc = row["Description"]
desc_text = "" if desc is None else str(desc)

user_id, discussion_id, post_id = extract_ids(desc_text)

if user_id is None or discussion_id is None:
    parsed_fail += 1
    per_event_fail[ev_norm] += 1
    if len(unparsed_samples) < 300:
        unparsed_samples.append({
            "rownum": rownum,
            "event_name": ev_text,
            "description": desc_text
        })
    continue

# discussion created
if action == "discussion_created":
    key = (user_id, discussion_id)
    if key in seen_discussions:
        dedup_skips += 1
        per_event_dedup[ev_norm] += 1
        continue

    seen_discussions.add(key)
    edge_action_counts[(user_id, discussion_id)]["discussion_created"] += 1
    per_action_counted["discussion_created"] += 1
    parsed_ok += 1
    continue

# post created / some content has been posted.
# εδώ απαιτούμε post_id για ασφαλές dedupe
if post_id is None:
    parsed_fail += 1
    per_event_fail[ev_norm] += 1
    if len(unparsed_samples) < 300:
        unparsed_samples.append({
            "rownum": rownum,
            "event_name": ev_text,
            "description": desc_text
        })
    continue

post_key = (user_id, post_id)
if post_key in seen_posts:
    dedup_skips += 1
    per_event_dedup[ev_norm] += 1
    continue

seen_posts.add(post_key)
edge_action_counts[(user_id, discussion_id)][action] += 1
per_action_counted[action] += 1
parsed_ok += 1

# write edges csv
with open(edges_path, "w", newline="", encoding="utf-8-sig") as f:
    w = csv.writer(f)
    w.writerow([
        "Source",
        "Target",
        "Weight_active",
        "W_post_created",
        "W_discussion_created",
        "W_some_content_posted",
    ])

for (user_id, discussion_id), d in sorted(edge_action_counts.items(), key=lambda x: (x[0][0], x[0][1])):
    weight_active = (
        d["post_created"] +
        d["discussion_created"] +
        d["some_content_posted"]
    )
    w.writerow([
        user_id,
        discussion_id,
        weight_active,
        d["post_created"],
        d["discussion_created"],
        d["some_content_posted"],
    ])

# write audit csv
all_events = set(per_event_detected) | set(per_event_fail) | set(per_event_dedup)
with open(audit_path, "w", newline="", encoding="utf-8-sig") as f:
    w = csv.writer(f)
    w.writerow([

```

```

        "event_name",
        "detected_rows",
        "parse_fail_rows",
        "dedup_skips_rows"
    ])
    for ev in sorted(all_events):
        w.writerow([
            ev,
            per_event_detected.get(ev, 0),
            per_event_fail.get(ev, 0),
            per_event_dedup.get(ev, 0),
        ])

# write unparsed samples
if unparsed_samples:
    with open(unparsed_path, "w", newline="", encoding="utf-8-sig") as f:
        w = csv.DictWriter(f, fieldnames=["rownum", "event_name", "description"])
        w.writeheader()
        w.writerows(unparsed_samples)

# summary
n_edges = len(edge_action_counts)
n_users = len({u for (u, _t) in edge_action_counts.keys()}) if n_edges else 0
n_threads = len({t for (_, t) in edge_action_counts.keys()}) if n_edges else 0

sum_active = 0
for d in edge_action_counts.values():
    sum_active += d["post_created"] + d["discussion_created"] + d["some_content_posted"]

print("Done.")
print(f"Total rows in sheet: {total_rows}")
print(f"Forum rows: {forum_rows}")
print(f"Accepted active rows detected: {accepted_rows}")
print(f"Parsed OK rows counted: {parsed_ok}")
print(f"Parse FAIL rows: {parsed_fail}")
print(f"Dedup skips: {dedup_skips}")
print("--- Counted actions (after dedupe) ---")
print(f"  post_created: {per_action_counted.get('post_created', 0)}")
print(f"  discussion_created: {per_action_counted.get('discussion_created', 0)}")
print(f"  some_content_posted: {per_action_counted.get('some_content_posted', 0)}")
print("--- Output stats ---")
print(f"Edges: {n_edges} | Users: {n_users} | Threads: {n_threads}")
print(f"Sum(Weight_active): {sum_active}")
print(f"Edges file: {edges_path}")
print(f"Audit file: {audit_path}")
if unparsed_samples:
    print(f"Unparsed samples file: {unparsed_path}")

if per_event_detected:
    print("\nEvent names contributing to ACTIVE participation:")
    for ev, c in per_event_detected.most_common():
        print(f"  {ev}: {c}")

if __name__ == "__main__":
    main()

```

A.6. 04b_sna_423_PREPARING_NODES_EDGES_FOR_GEPHI.py

Χαρακτήρας: Βοηθητικό script - SNA - οπτικοποίηση διμερούς δικτύου

Περιγραφή: Μετατρέπει τα αναγνωριστικά φοιτητών και συζητήσεων σε σαφή IDs συμβατά με το Gephi και δημιουργεί χωριστά αρχεία κόμβων και ακμών για το διμερές δίκτυο.

Είσοδοι: edges_gephi_active_forum.csv

Έξοδοι: Gephi_nodes.csv · Gephi_edges.csv

Πηγαίος κώδικας

```
import pandas as pd
import re

def format_student_id(x):
    """
    Μετατρέπει student id σε μορφή S_xxx
    π.χ. 7 -> S_007, 45 -> S_045, 123 -> S_123
    """
    s = str(x).strip()
    digits = re.sub(r"\D", "", s)

    if not digits:
        raise ValueError(f"Μη έγκυρο student id: {x}")

    return f"S_{digits.zfill(3)}"

def format_discussion_id(x):
    """
    Μετατρέπει discussion id σε μορφή D_xxxxx
    π.χ. 7 -> D_00007, 45 -> D_00045, 67890 -> D_67890
    """
    s = str(x).strip()
    digits = re.sub(r"\D", "", s)

    if not digits:
        raise ValueError(f"Μη έγκυρο discussion id: {x}")

    return f"D_{digits.zfill(5)}"

def main():
    input_file = "edges_gephi_active_forum.csv"
    nodes_output = "Gephi_nodes.csv"
    edges_output = "Gephi_edges.csv"

    # Διάβασμα αρχείου
    df = pd.read_csv(input_file)

    required_cols = {"Source", "Target", "Weight_active"}
    missing = required_cols - set(df.columns)
    if missing:
        raise ValueError(f"Λείπουν οι στήλες: {missing}")

    # Κρατάμε τα αρχικά ids για έλεγχο
    df["Source_original"] = df["Source"].astype(str).str.strip()
    df["Target_original"] = df["Target"].astype(str).str.strip()

    # Μετατροπή ids σε σαφή string ids για Gephi
    df["Source"] = df["Source"].apply(format_student_id)
    df["Target"] = df["Target"].apply(format_discussion_id)

    # -----
    # Edges
    # -----
    edges = df[["Source", "Target", "Weight_active"]].copy()
    edges = edges.rename(columns={"Weight_active": "Weight"})

    # Προαιρετικά: αν υπάρχουν από λάθος διπλές εγγραφές για ίδιο Source-Target,
    # τις αθροίζουμε ώστε το αρχείο να είναι "καθαρό"
    edges = (
        edges.groupby(["Source", "Target"], as_index=False)["Weight"]
        .sum()
    )

    edges.to_csv(edges_output, index=False, encoding="utf-8-sig")
```

```

# -----
# Nodes
# -----
source_nodes = (
    df[["Source", "Source_original"]]
    .drop_duplicates()
    .rename(columns={
        "Source": "Id",
        "Source_original": "OriginalId"
    })
)
source_nodes["Label"] = source_nodes["Id"]
source_nodes["NodeType"] = "student"

target_nodes = (
    df[["Target", "Target_original"]]
    .drop_duplicates()
    .rename(columns={
        "Target": "Id",
        "Target_original": "OriginalId"
    })
)
target_nodes["Label"] = target_nodes["Id"]
target_nodes["NodeType"] = "discussion"

nodes = pd.concat([source_nodes, target_nodes], ignore_index=True)
nodes = nodes[["Id", "Label", "NodeType", "OriginalId"]]
nodes = nodes.drop_duplicates(subset=["Id"])
nodes = nodes.sort_values(by=["NodeType", "Id"]).reset_index(drop=True)

nodes.to_csv(nodes_output, index=False, encoding="utf-8-sig")

print("Ετοιμο.")
print(f"Παρήχθη: {nodes_output}")
print(f"Παρήχθη: {edges_output}")
print(f"Edges: {len(edges)}")
print(f"Nodes: {len(nodes)}")
print(f"Students: {(nodes['NodeType'] == 'student').sum()}")
print(f"Discussions: {(nodes['NodeType'] == 'discussion').sum()}")

if __name__ == "__main__":
    main()

```

A.7. 04c_4_2_3_student_student_weighted_pro.py

Χαρακτήρας: Κύριο script - SNA - προβολή και μετρικές

Περιγραφή: Προβάλλει το διμερές δίκτυο σε μη κατευθυνόμενο δίκτυο φοιτητή-φοιτητή, υπολογίζει σταθμισμένες ακμές κοινής ενεργητικής συμμετοχής, εφαρμόζει φίλτρο 75ου εκατοστημορίου και υπολογίζει τις μετρικές degree, weighted degree, degree centrality, betweenness, closeness και PageRank.

Είσοδοι: edges_gephi_active_forum.csv

Έξοδοι: 04c_4_2_3_student_student_projection_full.csv · 04c_4_2_3_student_student_projection_filtered_p75.csv · 04c_4_2_3_student_student_nodes_metrics.csv · 04c_4_2_3_student_student_projection_report.txt

Πηγαίος κώδικας

```
# 04c_4_2_3_student_student_weighted_pro.py
# -----
# Student-student weighted projection + centrality metrics
#
# Input:
#   edges_gephi_active_forum.csv
#
# Expected columns:
#   Source, Target, Weight_active
#
# Outputs:
#   04c_4_2_3_student_student_projection_full.csv
#   04c_4_2_3_student_student_projection_filtered_p75.csv
#   04c_4_2_3_student_student_nodes_metrics.csv
#   04c_4_2_3_student_student_projection_report.txt
# -----

import itertools
from pathlib import Path

import pandas as pd
import networkx as nx

# =====
# CONFIG
# =====

INPUT_FILE = "edges_gephi_active_forum.csv"

OUTPUT_FULL = "04c_4_2_3_student_student_projection_full.csv"
OUTPUT_FILTERED = "04c_4_2_3_student_student_projection_filtered_p75.csv"
OUTPUT_NODES_METRICS = "04c_4_2_3_student_student_nodes_metrics.csv"
OUTPUT_REPORT = "04c_4_2_3_student_student_projection_report.txt"

PERCENTILE_THRESHOLD = 75

# Συνεπής επιλογή:
# Weight = συνολική ένταση κοινής ενεργητικής συμμετοχής.
GEPHI_WEIGHT_COLUMN = "Weight_active_sum"

def build_graph(edges_df, all_students):
    G = nx.Graph()

    for student in all_students:
        G.add_node(student)

    for row in edges_df.itertuples(index=False):
        G.add_edge(
            row.Source,
            row.Target,
            weight=float(row.Weight)
        )

    return G

def compute_centralities(G):
    degree = dict(G.degree())
    weighted_degree = dict(G.degree(weight="weight"))
    degree_centrality = nx.degree_centrality(G)
```

```

# Μετατροπή strength -> distance.
# Μεγάλο weight = ισχυρός δεσμός = μικρότερη απόσταση.
G_distance = G.copy()

for _u, _v, data in G_distance.edges(data=True):
    w = data.get("weight", 1)
    data["distance"] = 1 / w if w > 0 else 1

betweenness = nx.betweenness_centrality(
    G_distance,
    weight="distance",
    normalized=True
)

closeness = nx.closeness_centrality(
    G_distance,
    distance="distance"
)

pagerank = nx.pagerank(
    G,
    weight="weight"
)

metrics = pd.DataFrame({
    "Id": list(G.nodes()),
    "sna_degree": [degree.get(n, 0) for n in G.nodes()],
    "sna_weighted_degree": [weighted_degree.get(n, 0) for n in G.nodes()],
    "sna_degree_centrality": [degree_centrality.get(n, 0) for n in G.nodes()],
    "sna_betweenness": [betweenness.get(n, 0) for n in G.nodes()],
    "sna_closeness": [closeness.get(n, 0) for n in G.nodes()],
    "sna_pagerank": [pagerank.get(n, 0) for n in G.nodes()],
})

return metrics

def main():
    script_dir = Path(__file__).parent
    input_path = script_dir / INPUT_FILE

    full_path = script_dir / OUTPUT_FULL
    filtered_path = script_dir / OUTPUT_FILTERED
    nodes_metrics_path = script_dir / OUTPUT_NODES_METRICS
    report_path = script_dir / OUTPUT_REPORT

    if not input_path.exists():
        raise FileNotFoundError(f"Δεν βρέθηκε το αρχείο εισόδου: {input_path}")

    df = pd.read_csv(input_path, encoding="utf-8-sig")

    required_cols = {"Source", "Target", "Weight_active"}
    missing = required_cols - set(df.columns)
    if missing:
        raise ValueError(f"Λείπουν οι στήλες: {missing}")

    df = df[["Source", "Target", "Weight_active"]].copy()

    df["Source"] = df["Source"].astype(str).str.strip()
    df["Target"] = df["Target"].astype(str).str.strip()
    df["Weight_active"] = pd.to_numeric(df["Weight_active"], errors="coerce").fillna(0)

    df = df[(df["Source"] != "") & (df["Target"] != "")]
    df = df[df["Weight_active"] > 0]

    df = (
        df.groupby(["Source", "Target"], as_index=False)["Weight_active"]
        .sum()
    )

    all_students = sorted(df["Source"].unique())

    # =====
    # STUDENT-STUDENT PROJECTION
    # =====

    projected_edges = {}

    for discussion_id, group in df.groupby("Target"):
        students = group[["Source", "Weight_active"]].copy()

        if len(students) < 2:
            continue

        for row_a, row_b in itertools.combinations(students.itertuples(index=False), 2):
            s1 = row_a.Source
            s2 = row_b.Source

```

```

w1 = float(row_a.Weight_active)
w2 = float(row_b.Weight_active)

source, target = sorted([s1, s2])
key = (source, target)

if key not in projected_edges:
    projected_edges[key] = {
        "Source": source,
        "Target": target,
        "Weight_shared_threads": 0,
        "Weight_active_min": 0.0,
        "Weight_active_sum": 0.0,
        "Common_threads": [],
    }

    projected_edges[key]["Weight_shared_threads"] += 1
    projected_edges[key]["Weight_active_min"] += min(w1, w2)
    projected_edges[key]["Weight_active_sum"] += (w1 + w2)
    projected_edges[key]["Common_threads"].append(str(discussion_id))

projection = pd.DataFrame(projected_edges.values())

if projection.empty:
    raise ValueError(
        "Δεν δημιουργήθηκαν student-student ακμές. "
        "Έλεγχτε αν υπάρχουν discussions με τουλάχιστον 2 φοιτητές."
    )

projection["Common_threads"] = projection["Common_threads"].apply(lambda x: ";".join(x))
projection["Common_thread_count"] = projection["Weight_shared_threads"]

if GEPHI_WEIGHT_COLUMN not in projection.columns:
    raise ValueError(f"Μη έγκυρη επιλογή GEPHI_WEIGHT_COLUMN: {GEPHI_WEIGHT_COLUMN}")

projection["Weight"] = projection[GEPHI_WEIGHT_COLUMN]

projection = projection[
    [
        "Source",
        "Target",
        "Weight",
        "Weight_shared_threads",
        "Weight_active_min",
        "Weight_active_sum",
        "Common_thread_count",
        "Common_threads",
    ]
].sort_values(by=["Weight", "Source", "Target"], ascending=[False, True, True])

projection.to_csv(full_path, index=False, encoding="utf-8-sig")

# =====
# FILTERED PROJECTION
# =====

threshold_value = projection["Weight"].quantile(PERCENTILE_THRESHOLD / 100)

filtered = projection[projection["Weight"] >= threshold_value].copy()
filtered.to_csv(filtered_path, index=False, encoding="utf-8-sig")

# =====
# CENTRALITY METRICS
# =====

G = build_graph(filtered, all_students)
metrics = compute_centralities(G)

initial_threads_per_student = df.groupby("Source")["Target"].nunique()
initial_active_weight_per_student = df.groupby("Source")["Weight_active"].sum()

metrics["Label"] = metrics["Id"]
metrics["NodeType"] = "student"
metrics["Initial_active_threads"] = metrics["Id"].map(initial_threads_per_student).fillna(0).astype(int)
metrics["Initial_active_weight"] = metrics["Id"].map(initial_active_weight_per_student).fillna(0)

metrics = metrics[
    [
        "Id",
        "Label",
        "NodeType",
        "Initial_active_threads",
        "Initial_active_weight",
        "sna_degree",
        "sna_weighted_degree",
        "sna_degree_centrality",
    ]

```

```

        "sna_betweenness",
        "sna_closeness",
        "sna_pagerank",
    ]
].sort_values(by="sna_weighted_degree", ascending=False)

metrics.to_csv(nodes_metrics_path, index=False, encoding="utf-8-sig")

# =====
# REPORT
# =====

n_student_thread_edges = len(df)
n_students = df["Source"].nunique()
n_threads = df["Target"].nunique()
n_projected_edges = len(projection)
n_filtered_edges = len(filtered)
isolated_students = metrics[metrics["sna_degree"] == 0]["Id"].nunique()

report_lines = [
    "04c - Student-student weighted projection + centrality metrics",
    "=====",
    "",
    f"Input file: {input_path.name}",
    f"Student-thread rows after cleaning: {n_student_thread_edges}",
    f"Unique students: {n_students}",
    f"Unique discussions/threads: {n_threads}",
    "",
    "Projection rule:",
    "Two students are connected if they participated actively in the same discussion/thread.",
    "",
    f"Weight column used: {GEPHI_WEIGHT_COLUMN}",
    "",
    "Weight definitions:",
    "Weight_shared_threads = number of common active discussion threads.",
    "Weight_active_min = sum of min(active contribution of student A, active contribution of student B) per common",
    "thread.",
    "Weight_active_sum = sum of active contributions of both students per common thread.",
    "",
    "Centrality interpretation:",
    "For Degree and PageRank, Weight is used as tie strength.",
    "For Betweenness and Closeness, distance = 1 / Weight is used.",
    "",
    f"Projected student-student edges full graph: {n_projected_edges}",
    f"Percentile threshold: {PERCENTILE_THRESHOLD}%",
    f"Threshold value: {threshold_value}",
    f"Filtered projected edges: {n_filtered_edges}",
    f"Students with degree 0 after filtering: {isolated_students}",
    "",
    "Output files:",
    f"- {full_path.name}",
    f"- {filtered_path.name}",
    f"- {nodes_metrics_path.name}",
    f"- {report_path.name}",
]

report_path.write_text("\n".join(report_lines), encoding="utf-8-sig")

print("Done.")
print(f"Student-thread rows after cleaning: {n_student_thread_edges}")
print(f"Unique students: {n_students}")
print(f"Unique discussions/threads: {n_threads}")
print(f"Projected student-student edges full graph: {n_projected_edges}")
print(f"Weight column used: {GEPHI_WEIGHT_COLUMN}")
print(f"Percentile threshold: {PERCENTILE_THRESHOLD}%",)
print(f"Threshold value: {threshold_value}")
print(f"Filtered edges: {n_filtered_edges}")
print(f"Students with degree 0 after filtering: {isolated_students}")
print(f"Saved: {full_path.name}")
print(f"Saved: {filtered_path.name}")
print(f"Saved: {nodes_metrics_path.name}")
print(f"Saved: {report_path.name}")

if __name__ == "__main__":
    main()

```

A.8. 04d_4_2_3_visualize_sna_metrics.py

Χαρακτήρας: Βοηθητικό script - SNA - περιγραφική οπτικοποίηση

Περιγραφή: Παράγει περιγραφικά στατιστικά, ιστογράμματα και boxplots για τις έξι μετρικές κεντρικότητας του προβαλλόμενου δικτύου.

Είσοδοι: 04c_4_2_3_student_student_nodes_metrics.csv

Έξοδοι: 04d_4_2_3_sna_metrics_histograms.png · 04d_4_2_3_sna_metrics_boxplots_grid.png · 04d_4_2_3_sna_metrics_summary.csv

Πηγαίος κώδικας

```
# 04d_4_2_3_visualize_sna_metrics.py

from pathlib import Path
import pandas as pd
import matplotlib.pyplot as plt

INPUT_FILE = "04c_4_2_3_student_student_nodes_metrics.csv"

OUTPUT_HISTOGRAMS = "04d_4_2_3_sna_metrics_histograms.png"
OUTPUT_BOXPLOTS = "04d_4_2_3_sna_metrics_boxplots_grid.png"
OUTPUT_SUMMARY = "04d_4_2_3_sna_metrics_summary.csv"

METRICS = [
    "sna_degree",
    "sna_weighted_degree",
    "sna_degree_centrality",
    "sna_betweenness",
    "sna_closeness",
    "sna_pagerank",
]

TITLES = {
    "sna_degree": "Degree",
    "sna_weighted_degree": "Weighted Degree / Strength",
    "sna_degree_centrality": "Degree Centrality",
    "sna_betweenness": "Betweenness Centrality",
    "sna_closeness": "Closeness Centrality",
    "sna_pagerank": "PageRank",
}

def main():
    script_dir = Path(__file__).parent
    input_path = script_dir / INPUT_FILE

    if not input_path.exists():
        raise FileNotFoundError(f"Δεν βρέθηκε το αρχείο: {input_path}")

    df = pd.read_csv(input_path, encoding="utf-8-sig")

    missing = [col for col in METRICS if col not in df.columns]
    if missing:
        raise ValueError(f"Λείπουν οι στήλες: {missing}")

    for col in METRICS:
        df[col] = pd.to_numeric(df[col], errors="coerce").fillna(0)

    summary = df[METRICS].describe().T
    summary["zero_values"] = [(df[col] == 0).sum() for col in METRICS]
    summary["non_zero_values"] = [(df[col] != 0).sum() for col in METRICS]
    summary.to_csv(script_dir / OUTPUT_SUMMARY, encoding="utf-8-sig")

    # -----
    # Histograms 2x3
    # -----
    fig, axes = plt.subplots(2, 3, figsize=(18, 10))
    axes = axes.flatten()

    for ax, metric in zip(axes, METRICS):
        ax.hist(df[metric], bins=25)
        ax.set_title(TITLES[metric])
        ax.set_xlabel("Τιμή μετρικής")
        ax.set_ylabel("Πλήθος φοιτητών")
        ax.grid(True, alpha=0.3)
```

```

fig.suptitle(
    "Κατανομές μετρικών κεντρικότητας στο προβαλλόμενο student-student δίκτυο",
    fontsize=16
)
plt.tight_layout()
plt.savefig(script_dir / OUTPUT_HISTOGRAMS, dpi=300, bbox_inches="tight")
plt.close()

# -----
# Separate boxplots 2x3
# -----
fig, axes = plt.subplots(2, 3, figsize=(18, 10))
axes = axes.flatten()

for ax, metric in zip(axes, METRICS):
    ax.boxplot(
        df[metric],
        vert=True,
        showmeans=True
    )
    ax.set_title(TITLES[metric])
    ax.set_ylabel("Τιμή μετρικής")
    ax.set_xticks([1])
    ax.set_xticklabels([""])
    ax.grid(True, alpha=0.3)

fig.suptitle(
    "Boxplots μετρικών κεντρικότητας στο προβαλλόμενο student-student δίκτυο",
    fontsize=16
)
plt.tight_layout()
plt.savefig(script_dir / OUTPUT_BOXPLOTS, dpi=300, bbox_inches="tight")
plt.close()

print("Done.")
print(f"Saved: {OUTPUT_HISTOGRAMS}")
print(f"Saved: {OUTPUT_BOXPLOTS}")
print(f"Saved: {OUTPUT_SUMMARY}")

if __name__ == "__main__":
    main()

```

A.9. 04e_4_2_3_assign_social_roles.py

Χαρακτήρας: Κύριο script - SNA - κοινωνικοί ρόλοι

Περιγραφή: Τυποποιεί επιλεγμένες μετρικές SNA και αντιστοιχίζει τους φοιτητές στους πέντε τελικούς κοινωνικούς ρόλους: Ηγέτης, Γέφυρα, Δραστήριος Συμμετέχων, Περιφερειακός Συμμετέχων και Ανεπαρκής Αλληλεπίδραση.

Είσοδοι: 04c_4_2_3_student_student_nodes_metrics.csv

Έξοδοι: 04e_4_2_3_student_social_roles.csv · 04e_4_2_3_social_role_distribution.csv · 04e_4_2_3_social_role_assignment_report.txt

Πηγαίος κώδικας

```
# 04e_4_2_3_assign_social_roles.py
# -----
# Αντιστοίχιση φοιτητών σε κοινωνικούς ρόλους με βάση SNA metrics
#
# Input:
#   04c_4_2_3_student_student_nodes_metrics.csv
#
# Outputs:
#   04e_4_2_3_student_social_roles.csv
#   04e_4_2_3_social_role_distribution.csv
#   04e_4_2_3_social_role_assignment_report.txt
# -----

from pathlib import Path
import pandas as pd

INPUT_FILE = "04c_4_2_3_student_student_nodes_metrics.csv"

OUTPUT_ROLES = "04e_4_2_3_student_social_roles.csv"
OUTPUT_DISTRIBUTION = "04e_4_2_3_social_role_distribution.csv"
OUTPUT_REPORT = "04e_4_2_3_social_role_assignment_report.txt"

HIGH_Z = 1.0
ACTIVE_Z = 0.0

def zscore(series: pd.Series) -> pd.Series:
    mean = series.mean()
    std = series.std(ddof=0)

    if std == 0 or pd.isna(std):
        return pd.Series([0.0] * len(series), index=series.index)

    return (series - mean) / std

def zscore_nonzero(series: pd.Series) -> pd.Series:
    result = pd.Series([0.0] * len(series), index=series.index)

    nonzero = series[series > 0]

    if len(nonzero) == 0:
        return result

    mean = nonzero.mean()
    std = nonzero.std(ddof=0)

    if std == 0 or pd.isna(std):
        return result

    result.loc[nonzero.index] = (nonzero - mean) / std
    return result

def assign_role(row):
    """
    Priority:
    1. No Interaction
    2. Leader
    3. Bridge
    4. Active Participant
    5. Peripheral Participant
    """
```

```

"""
if row["sna_degree"] == 0:
    return "No Interaction"

if (
    row["sna_weighted_degree_z"] > HIGH_Z
    and row["sna_pagerank_z"] > HIGH_Z
    and row["sna_closeness_z"] >= ACTIVE_Z
):
    return "Leader"

# Bridge:
# όχι z > 1, γιατί η betweenness είναι πολύ skewed.
# Εδώ Bridge = φοιτητής με μη μηδενική betweenness
# και πάνω από τον μέσο όρο των μη μηδενικών betweenness τιμών.
if row["sna_betweenness"] > 0 and row["sna_betweenness_z"] > 0:
    return "Bridge"

if row["sna_weighted_degree_z"] > ACTIVE_Z:
    return "Active Participant"

return "Peripheral Participant"

def main():
    script_dir = Path(__file__).parent
    input_path = script_dir / INPUT_FILE

    if not input_path.exists():
        raise FileNotFoundError(f"Δεν βρέθηκε το αρχείο: {input_path}")

    df = pd.read_csv(input_path, encoding="utf-8-sig")

    required_cols = [
        "Id",
        "sna_degree",
        "sna_weighted_degree",
        "sna_betweenness",
        "sna_closeness",
        "sna_pagerank",
    ]

    missing = [c for c in required_cols if c not in df.columns]
    if missing:
        raise ValueError(f"Λείπουν στήλες: {missing}")

    metric_cols = [
        "sna_degree",
        "sna_weighted_degree",
        "sna_betweenness",
        "sna_closeness",
        "sna_pagerank",
    ]

    for col in metric_cols:
        df[col] = pd.to_numeric(df[col], errors="coerce").fillna(0)

    df["sna_weighted_degree_z"] = zscore(df["sna_weighted_degree"])
    df["sna_closeness_z"] = zscore(df["sna_closeness"])
    df["sna_pagerank_z"] = zscore(df["sna_pagerank"])

    # Για τη betweenness χρησιμοποιούμε z-score μόνο στις μη μηδενικές τιμές
    df["sna_betweenness_z"] = zscore_nonzero(df["sna_betweenness"])

    df["network_role"] = df.apply(assign_role, axis=1)

    role_labels_el = {
        "Leader": "Ηγέτης",
        "Bridge": "Γέφυρα",
        "Active Participant": "Δραστήριος Συμμετέχων",
        "Peripheral Participant": "Περιφερειακός Συμμετέχων",
        "No Interaction": "Ανεπαρκής Αλληλεπίδραση",
    }

    df["network_role_el"] = df["network_role"].map(role_labels_el)

    output_cols = [
        "Id",
        "Label",
        "NodeType",
        "Initial_active_threads",
        "Initial_active_weight",
        "sna_degree",
        "sna_weighted_degree",
        "sna_betweenness",
        "sna_closeness",

```

```

        "sna_pagerank",
        "sna_weighted_degree_z",
        "sna_betweenness_z",
        "sna_closeness_z",
        "sna_pagerank_z",
        "network_role",
        "network_role_el",
    ]

    output_cols = [c for c in output_cols if c in df.columns]

    roles_path = script_dir / OUTPUT_ROLES
    df[output_cols].to_csv(roles_path, index=False, encoding="utf-8-sig")

    distribution = (
        df["network_role_el"]
        .value_counts()
        .rename_axis("network_role_el")
        .reset_index(name="count")
    )

    distribution["percentage"] = (
        distribution["count"] / len(df) * 100
    ).round(2)

    role_order = {
        "Ηγέτης": 1,
        "Γέφυρα": 2,
        "Δραστήριος Συμμετέχων": 3,
        "Περιφερειακός Συμμετέχων": 4,
        "Ανεπαρκής Αλληλεπίδραση": 5,
    }

    distribution["role_order"] = distribution["network_role_el"].map(role_order)

    distribution = (
        distribution
        .sort_values("role_order")
        .drop(columns="role_order")
    )

    distribution_path = script_dir / OUTPUT_DISTRIBUTION
    distribution.to_csv(distribution_path, index=False, encoding="utf-8-sig")

    report_lines = [
        "04e - Social Role Assignment Report",
        "=" * 50,
        "",
        f"Input file: {INPUT_FILE}",
        f"Total students: {len(df)}",
        "",
        "Role assignment method:",
        "Z-score based role assignment using SNA metrics.",
        "",
        "Priority order:",
        "1. No Interaction: degree = 0",
        "2. Leader",
        "3. Bridge",
        "4. Active Participant",
        "5. Peripheral Participant",
        "",
        "Criteria:",
        f"- Leader: weighted_degree_z > {HIGH_Z} AND pagerank_z > {HIGH_Z} AND closeness_z >= {ACTIVE_Z}",
        "- Bridge: betweenness > 0 AND betweenness_z > 0",
        f"- Active Participant: weighted_degree_z > {ACTIVE_Z}",
        "- Peripheral Participant: all remaining students",
        "",
        "Normalization:",
        "- weighted_degree, closeness and pagerank use standard z-score normalization.",
        "- betweenness uses z-score normalization only on non-zero values because of its highly skewed distribution.",
        "",
        "Role distribution:",
    ]

    for row in distribution.itertuples(index=False):
        report_lines.append(
            f"- {row.network_role_el}: {row.count} students ({row.percentage}%)"
        )

    report_lines.extend([
        "",
        "Output files:",
        f"- {OUTPUT_ROLES}",
        f"- {OUTPUT_DISTRIBUTION}",
        f"- {OUTPUT_REPORT}",
    ])

```

```

report_path = script_dir / OUTPUT_REPORT
report_path.write_text("\n".join(report_lines), encoding="utf-8-sig")

print("Done.")
print(f"Total students: {len(df)}")
print()
print("Role distribution:")
print(distribution.to_string(index=False))
print()
print(f"Saved: {OUTPUT_ROLES}")
print(f"Saved: {OUTPUT_DISTRIBUTION}")
print(f"Saved: {OUTPUT_REPORT}")

if __name__ == "__main__":
    main()

```

A.10. 04f_4_2_3_validate_social_roles_kruskal_dunn.py

Χαρακτήρας: Έλεγχος script - SNA - στατιστική επικύρωση

Περιγραφή: Ελέγχει αν οι κατανομές των μετρικών SNA διαφέρουν μεταξύ των κοινωνικών ρόλων με Kruskal–Wallis και, όπου υπάρχει συνολική στατιστική σημαντικότητα, εκτελεί Dunn post-hoc με διόρθωση Bonferroni.

Είσοδοι: 04e_4_2_3_student_social_roles.csv

Έξοδοι: 04f_4_2_3_kruskal_wallis_results.csv · 04f_4_2_3_dunn_posthoc_<metric>.csv · 04f_4_2_3_role_metric_boxplots.png · 04f_4_2_3_statistical_validation_report.txt

Πηγαίος κώδικας

```
# 04f_4_2_3_validate_social_roles_kruskal_dunn.py
# -----
# Στατιστική επικύρωση κοινωνικών ρόλων:
# - Kruskal-Wallis test
# - Dunn post-hoc test με Bonferroni correction
#
# Input:
# 04e_4_2_3_student_social_roles.csv
#
# Outputs:
# 04f_4_2_3_kruskal_wallis_results.csv
# 04f_4_2_3_dunn_posthoc_<metric>.csv
# 04f_4_2_3_role_metric_boxplots.png
# 04f_4_2_3_statistical_validation_report.txt
# -----

from pathlib import Path

import pandas as pd
import matplotlib.pyplot as plt
from scipy.stats import kruskal
import scikit_posthocs as sp

# =====
# CONFIG
# =====

INPUT_FILE = "04e_4_2_3_student_social_roles.csv"

OUTPUT_KRUSKAL = "04f_4_2_3_kruskal_wallis_results.csv"
OUTPUT_BOXPLOTS = "04f_4_2_3_role_metric_boxplots.png"
OUTPUT_REPORT = "04f_4_2_3_statistical_validation_report.txt"

ROLE_COL = "network_role_el"

METRICS = [
    "sna_degree",
    "sna_weighted_degree",
    "sna_betweenness",
    "sna_closeness",
    "sna_pagerank",
]

METRIC_TITLES = {
    "sna_degree": "Degree",
    "sna_weighted_degree": "Weighted Degree",
    "sna_betweenness": "Betweenness",
    "sna_closeness": "Closeness",
    "sna_pagerank": "PageRank",
}

ROLE_ORDER = [
    "Ηγέτης",
    "Γέφυρα",
    "Δραστήριος Συμμετέχων",
    "Περιφερειακός Συμμετέχων",
    "Ανεπαρκής Αλληλεπίδραση",
]

ALPHA = 0.05
P_ADJUST = "bonferroni"
```

```

def significance_label(p_value):
    if pd.isna(p_value):
        return "NA"
    if p_value < 0.001:
        return "***"
    if p_value < 0.01:
        return "**"
    if p_value < 0.05:
        return "*"
    return "ns"

def main():
    script_dir = Path(__file__).parent
    input_path = script_dir / INPUT_FILE

    if not input_path.exists():
        raise FileNotFoundError(f"Δεν βρέθηκε το αρχείο: {input_path}")

    df = pd.read_csv(input_path, encoding="utf-8-sig")

    required_cols = [ROLE_COL] + METRICS
    missing = [col for col in required_cols if col not in df.columns]

    if missing:
        raise ValueError(f"Λείπουν οι στήλες: {missing}")

    for metric in METRICS:
        df[metric] = pd.to_numeric(df[metric], errors="coerce").fillna(0)

    roles_present = [role for role in ROLE_ORDER if role in df[ROLE_COL].unique()]

    # Κρατάμε μόνο γραμμές με έγκυρο ρόλο
    df = df[df[ROLE_COL].isin(roles_present)].copy()

    # =====
    # KRUSKAL-WALLIS
    # =====

    kruskal_results = []
    dunn_files = []

    for metric in METRICS:
        groups = [
            df.loc[df[ROLE_COL] == role, metric].dropna()
            for role in roles_present
        ]

        groups = [g for g in groups if len(g) > 0]

        if len(groups) < 2:
            h_stat = None
            p_value = None
            sig = "not_applicable"
            dunn_created = False
        else:
            h_stat, p_value = kruskal(*groups)
            sig = significance_label(p_value)

            dunn_created = False

            if p_value < ALPHA:
                dunn = sp.posthoc_dunn(
                    df,
                    val_col=metric,
                    group_col=ROLE_COL,
                    p_adjust=P_ADJUST
                )

                # Τακτοποίηση σειράς γραμμών/στηλών
                dunn = dunn.loc[roles_present, roles_present]

                dunn_output = f"04f_4_2_3_dunn_posthoc_{metric}.csv"
                dunn_path = script_dir / dunn_output
                dunn.to_csv(dunn_path, encoding="utf-8-sig")

                dunn_files.append(dunn_output)
                dunn_created = True

        kruskal_results.append({
            "metric": metric,
            "metric_label": METRIC_TITLES[metric],
            "kruskal_H": h_stat,
            "p_value": p_value,
            "significance": sig,
            "dunn_posthoc_created": dunn_created,

```

```

        "p_adjust": P_ADJUST if dunn_created else "",
    })

kruskal_df = pd.DataFrame(kruskal_results)

kruskal_path = script_dir / OUTPUT_KRUSKAL
kruskal_df.to_csv(kruskal_path, index=False, encoding="utf-8-sig")

# =====
# BOXPLOTS BY ROLE
# =====

fig, axes = plt.subplots(2, 3, figsize=(21, 11))
axes = axes.flatten()

for ax, metric in zip(axes, METRICS):
    data = [
        df.loc[df[ROLE_COL] == role, metric].dropna()
        for role in roles_present
    ]

    ax.boxplot(
        data,
        tick_labels=roles_present,
        showmeans=True
    )

    ax.set_title(METRIC_TITLES[metric])
    ax.set_ylabel("Τιμή μετρικής")
    ax.tick_params(axis="x", rotation=30)
    ax.grid(True, alpha=0.3)

# 6o subplot: συνοπτικός πίνακας Kruskal-Wallis
summary_ax = axes[-1]
summary_ax.axis("off")

summary_lines = ["Kruskal-Wallis"]
for row in kruskal_df.itertuples(index=False):
    if row.p_value is None or pd.isna(row.p_value):
        summary_lines.append(f"{row.metric_label}: NA")
    else:
        summary_lines.append(
            f"{row.metric_label}: p={row.p_value:.4g} {row.significance}"
        )

summary_ax.text(
    0.02,
    0.95,
    "\n".join(summary_lines),
    va="top",
    ha="left",
    fontsize=11
)

fig.suptitle(
    "Κατανομή μετρικών κεντρικότητας ανά κοινωνικό ρόλο",
    fontsize=16
)

plt.tight_layout()
plt.savefig(script_dir / OUTPUT_BOXPLOTS, dpi=300, bbox_inches="tight")
plt.close()

# =====
# REPORT
# =====

report_lines = [
    "04f - Statistical validation of social roles",
    "=====",
    "",
    f"Input file: {INPUT_FILE}",
    f"Total students: {len(df)}",
    "",
    "Roles included:",
]

role_counts = df[ROLE_COL].value_counts()

for role in roles_present:
    report_lines.append(f"- {role}: {role_counts.get(role, 0)} students")

report_lines.extend([
    "",
    "Statistical procedure:",
    "1. Kruskal-Wallis non-parametric test across social roles.",
    f"2. Dunn post-hoc test with {P_ADJUST} correction for metrics with p < {ALPHA}.",
])

```

```

    "",
    "Null hypothesis (Kruskal-Wallis):",
    "The distributions of the metric are the same across social roles.",
    "",
    "Alternative hypothesis:",
    "At least one social role differs in the distribution of the metric.",
    "",
    "Kruskal-Wallis results:",
])

for row in kruskal_df.itertuples(index=False):
    if row.p_value is None or pd.isna(row.p_value):
        report_lines.append(f"- {row.metric_label}: not applicable")
    else:
        report_lines.append(
            f"- {row.metric_label}: H = {row.kruskal_H:.4f}, "
            f"p = {row.p_value:.6f}, significance = {row.significance}"
        )

report_lines.extend([
    "",
    "Dunn post-hoc output files:",
])

if dunn_files:
    for filename in dunn_files:
        report_lines.append(f"- {filename}")
else:
    report_lines.append("- No Dunn post-hoc tests were created because no metric was significant.")

report_lines.extend([
    "",
    "Significance codes:",
    "**** p < 0.001",
    "*** p < 0.01",
    "** p < 0.05",
    "ns not significant",
    "",
    "Output files:",
    f"- {OUTPUT_KRUSKAL}",
    f"- {OUTPUT_BOXPLOTS}",
    f"- {OUTPUT_REPORT}",
])

report_path = script_dir / OUTPUT_REPORT
report_path.write_text("\n".join(report_lines), encoding="utf-8-sig")

# =====
# CONSOLE OUTPUT
# =====

print("Done.")
print(f"Total students: {len(df)}")
print()
print("Kruskal-Wallis results:")
print(kruskal_df.to_string(index=False))
print()

if dunn_files:
    print("Dunn post-hoc files:")
    for filename in dunn_files:
        print(f"- {filename}")
    print()

print(f"Saved: {OUTPUT_KRUSKAL}")
print(f"Saved: {OUTPUT_BOXPLOTS}")
print(f"Saved: {OUTPUT_REPORT}")

if __name__ == "__main__":
    main()

```

A.11. 04g_4_2_3_prepare_gephi_projected_roles_visualization.py

Χαρακτήρας: Βοηθητικό script - SNA - οπτικοποίηση προβαλλόμενου δικτύου

Περιγραφή: Προετοιμάζει τα αρχεία κόμβων και ακμών του προβαλλόμενου δικτύου για το Gephi, διατηρώντας όλους τους κόμβους, συμπεριλαμβανομένων των απομονωμένων φοιτητών με degree=0.

Είσοδοι: 04e_4_2_3_student_social_roles.csv · 04c_4_2_3_student_student_projection_filtered_p75.csv

Έξοδοι: Gephi_projected_NODES_roles_ALL_135.csv · Gephi_projected_EDGES_filtered_p75.csv

Πηγαίος κώδικας

```
from pathlib import Path
import pandas as pd

folder = Path(__file__).parent

# =====
# INPUT FILES
# =====

NODES_INPUT = folder / "04e_4_2_3_student_social_roles.csv"
EDGES_INPUT = folder / "04c_4_2_3_student_student_projection_filtered_p75.csv"

# =====
# OUTPUT FILES
# =====

NODES_OUTPUT = folder / "Gephi_projected_NODES_roles_ALL_135.csv"
EDGES_OUTPUT = folder / "Gephi_projected_EDGES_filtered_p75.csv"

# =====
# LOAD
# =====

nodes = pd.read_csv(NODES_INPUT, encoding="utf-8-sig")
edges = pd.read_csv(EDGES_INPUT, encoding="utf-8-sig")

print("Nodes columns:")
print(nodes.columns.tolist())

print("\nEdges columns:")
print(edges.columns.tolist())

# =====
# NODES
# =====

# Βρες το αναγνωριστικό φοιτητή
if "Id" in nodes.columns:
    id_col = "Id"
elif "user_id" in nodes.columns:
    id_col = "user_id"
elif "student_id" in nodes.columns:
    id_col = "student_id"
else:
    raise ValueError("Δεν βρέθηκε στήλη Id / user_id / student_id στο nodes file.")

required_node_cols = [
    "network_role_el",
    "sna_weighted_degree"
]

for col in required_node_cols:
    if col not in nodes.columns:
        raise ValueError(f"Λείπει η στήλη από το nodes file: {col}")

optional_node_cols = [
    "sna_degree",
    "sna_weighted_degree",
    "sna_betweenness",
    "sna_closeness",
    "sna_pagerank",
    "network_role_el"
]

existing_node_cols = [
    col for col in optional_node_cols
    if col in nodes.columns
```

```

]

gephi_nodes = nodes[[id_col] + existing_node_cols].copy()

gephi_nodes = geph_nodes.rename(columns={
    id_col: "Id"
})

gephi_nodes["Label"] = geph_nodes["Id"]

ordered_cols = ["Id", "Label"] + [
    col for col in geph_nodes.columns
    if col not in ["Id", "Label"]
]

gephi_nodes = geph_nodes[ordered_cols]

# Σημαντικό:
# Δεν κρατάμε μόνο τους κόμβους που εμφανίζονται στις ακμές.
# Θέλουμε να διατηρηθούν και οι απομονωμένοι κόμβοι degree = 0.

# =====
# EDGES
# =====

rename_edges = {}

for col in edges.columns:
    lower = col.lower()
    if lower == "source":
        rename_edges[col] = "Source"
    elif lower == "target":
        rename_edges[col] = "Target"
    elif lower == "weight":
        rename_edges[col] = "Weight"

edges = edges.rename(columns=rename_edges)

required_edge_cols = ["Source", "Target", "Weight"]

for col in required_edge_cols:
    if col not in edges.columns:
        raise ValueError(f"Λείπει η στήλη από το edges file: {col}")

gephi_edges = edges[["Source", "Target", "Weight"]].copy()

# =====
# CHECKS
# =====

edge_nodes = set(gephi_edges["Source"]).union(set(gephi_edges["Target"]))
all_nodes = set(gephi_nodes["Id"])

missing_from_nodes = edge_nodes - all_nodes
isolated_nodes = all_nodes - edge_nodes

if missing_from_nodes:
    raise ValueError(
        f"Υπάρχουν κόμβοι στις ακμές που δεν υπάρχουν στο nodes file: {missing_from_nodes}"
    )

print("\nCreated Gephi data:")
print("Nodes:", len(gephi_nodes))
print("Edges:", len(gephi_edges))
print("Nodes appearing in edges:", len(edge_nodes))
print("Isolated nodes:", len(isolated_nodes))

print("\nRole distribution:")
print(gephi_nodes["network_role_el"].value_counts(dropna=False))

print("\nIsolated role distribution:")
print(
    geph_nodes.loc[
        geph_nodes["Id"].isin(isolated_nodes),
        "network_role_el"
    ].value_counts(dropna=False)
)

# =====
# SAVE
# =====

gephi_nodes.to_csv(NODES_OUTPUT, index=False, encoding="utf-8-sig")
gephi_edges.to_csv(EDGES_OUTPUT, index=False, encoding="utf-8-sig")

print("\nSaved:")
print(NODES_OUTPUT.name)

```

```
print(EDGES_OUTPUT.name)
```

A.12. 06_frustration_signals.py

Χαρακτήρας: Κύριο script - Σήματα απογοήτευσης

Περιγραφή: Κατηγοριοποιεί τα events, τα συγκεντρώνει ανά λεπτό και ανιχνεύει ασυνήθιστα έντονες ριπές δραστηριότητας με εξατομικευμένα προσαρμοστικά κατώφλια, φίλτρο επιμονής και συγχώνευση διαδοχικών λεπτών σε επεισόδια.

Είσοδοι: Logfile_filtered_OK.csv

Έξοδοι: 06_04_2_4_frustration_detailed_signals.csv · 06_04_2_4_frustration_episodes.csv · 06_04_2_4_frustration_summary.csv · 06_04_2_4_frustration_thresholds.csv

Πηγαίος κώδικας

```
import pandas as pd
from pathlib import Path

# =====
# PARAMETERS
# =====

SCRIPT_DIR = Path(__file__).resolve().parent
INPUT_FILE = SCRIPT_DIR / "Logfile_filtered_OK.csv"

TIME_COLUMN = "Time"
TIME_FORMAT = "%d/%m/%y, %H:%M"

USER_COLUMN = "User full name"
EVENT_NAME_COLUMN = "Event name"
EVENT_CONTEXT_COLUMN = "Event context"

DISPERSION_FACTOR = 2.0

CATEGORY_MIN_THRESHOLDS = {
    "login": 2,
    "quiz": 3,
    "submission": 3,
    "forum": 3,
    "navigation": 4,
    "other": 3,
}

PERSISTENCE_WINDOW_MINUTES = 5
MIN_FLAGGED_MINUTES_IN_WINDOW = 2

MERGE_GAP_MINUTES = 2

# =====
# LOAD FILE
# =====

def load_file(input_file):
    ext = Path(input_file).suffix.lower()

    if ext == ".xlsx":
        return pd.read_excel(input_file)

    if ext == ".csv":
        encodings = ["utf-8", "utf-8-sig", "cp1252", "latin1"]

        for enc in encodings:
            try:
                return pd.read_csv(input_file, encoding=enc, low_memory=False)
            except Exception:
                pass

        raise ValueError("Δεν ήταν δυνατό να διαβαστεί το αρχείο.")

# =====
# PREPROCESS
# =====
```

```

def preprocess(df):

    df = df.copy()

    df[TIME_COLUMN] = pd.to_datetime(
        df[TIME_COLUMN],
        format=TIME_FORMAT,
        errors="coerce"
    )

    df = df.dropna(subset=[TIME_COLUMN])

    for col in [USER_COLUMN, EVENT_NAME_COLUMN, EVENT_CONTEXT_COLUMN]:
        df[col] = df[col].astype(str).str.strip()

    df["Minute"] = df[TIME_COLUMN].dt.floor("min")

    return df

# =====
# EVENT CATEGORIZATION
# =====

def categorize_event(event_name, event_context):

    en = str(event_name).lower()
    ec = str(event_context).lower()

    combined = f"{ec} | {en}"

    if "login" in combined or "logged in" in combined:
        return "login"

    if "quiz" in combined or "attempt" in combined:
        return "quiz"

    if "assign" in combined or "submission" in combined or "upload" in combined:
        return "submission"

    if "forum" in combined or "discussion" in combined or "post" in combined:
        return "forum"

    navigation_terms = [
        "course viewed",
        "course module viewed",
        "dashboard viewed",
        "page viewed",
        "resource viewed",
        "url viewed",
        "book viewed",
        "folder viewed",
    ]

    if any(term in en for term in navigation_terms):
        return "navigation"

    return "other"

def add_category(df):

    df = df.copy()

    df["Category"] = df.apply(
        lambda row: categorize_event(
            row[EVENT_NAME_COLUMN],
            row[EVENT_CONTEXT_COLUMN]
        ),
        axis=1
    )

    return df

# =====
# MINUTE SUMMARY
# =====

def build_minute_summary(df):

    minute_summary = (
        df.groupby([USER_COLUMN, "Minute", "Category"], as_index=False)
        .agg(
            event_count=(EVENT_NAME_COLUMN, "size")
        )
        .sort_values([USER_COLUMN, "Minute", "Category"])
    )

```

```

        .reset_index(drop=True)
    )

    return minute_summary

# =====
# ADAPTIVE THRESHOLDS
# =====

def compute_thresholds(minute_summary):

    thresholds = (
        minute_summary.groupby([USER_COLUMN, "Category"], as_index=False)["event_count"]
        .agg(
            mean_count="mean",
            std_count=lambda x: x.std(ddof=0)
        )
    )

    thresholds["std_count"] = thresholds["std_count"].fillna(0)

    thresholds["category_min_threshold"] = (
        thresholds["Category"].map(CATEGORY_MIN_THRESHOLDS)
    )

    thresholds["raw_threshold"] = (
        thresholds["mean_count"] + DISPERSION_FACTOR * thresholds["std_count"]
    )

    thresholds["final_threshold"] = thresholds[
        ["raw_threshold", "category_min_threshold"]
    ].max(axis=1)

    return thresholds

# =====
# FLAG MINUTES
# =====

def flag_minutes(minute_summary, thresholds):

    detailed = minute_summary.merge(
        thresholds,
        on=[USER_COLUMN, "Category"],
        how="left"
    )

    detailed["flagged"] = detailed["event_count"] > detailed["final_threshold"]

    return detailed

# =====
# PERSISTENCE FILTER
# =====

def apply_persistence_filter(detailed):

    detailed = detailed.copy()

    detailed["persistent_flag"] = False

    flagged = detailed[detailed["flagged"]].copy()

    if flagged.empty:
        return detailed

    rows = []

    for (student, category), g in flagged.groupby([USER_COLUMN, "Category"]):

        g = g.sort_values("Minute")

        minutes = g["Minute"].tolist()

        keep = []

        for t in minutes:

            start = t - pd.Timedelta(minutes=PERSISTENCE_WINDOW_MINUTES)
            end = t + pd.Timedelta(minutes=PERSISTENCE_WINDOW_MINUTES)

            count = sum(start <= x <= end for x in minutes)

            keep.append(count >= MIN_FLAGGED_MINUTES_IN_WINDOW)

```

```

g["persistent_flag"] = keep

rows.append(g[[USER_COLUMN, "Minute", "Category", "persistent_flag"]])

persist_df = pd.concat(rows)

detailed = detailed.merge(
    persist_df,
    on=[USER_COLUMN, "Minute", "Category"],
    how="left",
    suffixes=("", "_new")
)

detailed["persistent_flag"] = detailed["persistent_flag_new"]

detailed["persistent_flag"] = detailed["persistent_flag"].where(
    detailed["persistent_flag"].notna(),
    False
).astype(bool)

detailed = detailed.drop(columns=["persistent_flag_new"])

return detailed

# =====
# BUILD EPISODES
# =====

def build_episodes(detailed):

    persistent = detailed[detailed["persistent_flag"]].copy()

    if persistent.empty:
        return pd.DataFrame()

    episodes_all = []

    for (student, category), g in persistent.groupby([USER_COLUMN, "Category"]):

        g = g.sort_values("Minute")

        g["prev"] = g["Minute"].shift(1)

        g["gap"] = (
            (g["Minute"] - g["prev"]).dt.total_seconds() / 60
        )

        g["new_episode"] = g["gap"].isna() | (g["gap"] > MERGE_GAP_MINUTES)

        g["episode_id"] = g["new_episode"].cumsum()

        episodes = (
            g.groupby("episode_id", as_index=False)
            .agg(
                **{
                    USER_COLUMN: (USER_COLUMN, "first"),
                    "Category": ("Category", "first"),
                    "episode_start": ("Minute", "min"),
                    "episode_end": ("Minute", "max"),
                    "flagged_minutes": ("Minute", "size"),
                    "max_event_count": ("event_count", "max"),
                    "mean_event_count": ("event_count", "mean")
                }
            )
        )

        episodes_all.append(episodes)

    return pd.concat(episodes_all, ignore_index=True)

# =====
# MAIN
# =====

def main():

    print("Frustration detection")

    df = load_file(INPUT_FILE)

    df = preprocess(df)

    df = add_category(df)

```

```

minute_summary = build_minute_summary(df)

thresholds = compute_thresholds(minute_summary)

detailed = flag_minutes(minute_summary, thresholds)

detailed = apply_persistence_filter(detailed)

episodes = build_episodes(detailed)

print("Minute records:", len(minute_summary))
print("Flagged minutes:", detailed["flagged"].sum())
print("Persistent flagged minutes:", detailed["persistent_flag"].sum())
print("Episodes:", len(episodes))

if not episodes.empty:
    print("\n--- Episodes per category ---")
    print(episodes["Category"].value_counts())

    print("\n--- Total flagged minutes per category ---")
    print(episodes.groupby("Category")["flagged_minutes"].sum())

print("\n--- Unique events per category ---")
print(df.groupby("Category")[EVENT_NAME_COLUMN].nunique())

print("\n--- Sample events per category ---")

for cat in df["Category"].unique():

    print("\nCATEGORY:", cat)

    print(
        df[df["Category"] == cat][EVENT_NAME_COLUMN]
        .value_counts()
        .head(10)
    )

# =====
# SAVE OUTPUTS
# =====

detailed_output = SCRIPT_DIR / "06_04_2_4_frustration_detailed_signals.csv"
episodes_output = SCRIPT_DIR / "06_04_2_4_frustration_episodes.csv"
summary_output = SCRIPT_DIR / "06_04_2_4_frustration_summary.csv"
thresholds_output = SCRIPT_DIR / "06_04_2_4_frustration_thresholds.csv"

detailed.to_csv(
    detailed_output,
    index=False,
    encoding="utf-8-sig"
)

episodes.to_csv(
    episodes_output,
    index=False,
    encoding="utf-8-sig"
)

summary_by_student = (
    detailed[detailed["persistent_flag"]]
    .groupby(USER_COLUMN, as_index=False)
    .agg(
        frustration_signal_count=("persistent_flag", "size")
    )
)

episode_counts = (
    episodes.groupby(USER_COLUMN, as_index=False)
    .size()
    .rename(columns={"size": "frustration_episode_count"})
)

summary_by_student = summary_by_student.merge(
    episode_counts,
    on=USER_COLUMN,
    how="outer"
).fillna(0)

summary_by_student["frustration_signal_count"] = (
    summary_by_student["frustration_signal_count"].astype(int)
)

summary_by_student["frustration_episode_count"] = (
    summary_by_student["frustration_episode_count"].astype(int)
)

summary_by_student.to_csv(

```

```

        summary_output,
        index=False,
        encoding="utf-8-sig"
    )

    thresholds.to_csv(
        thresholds_output,
        index=False,
        encoding="utf-8-sig"
    )

    print("\nSaved:")
    print("-", detailed_output)
    print("-", episodes_output)
    print("-", summary_output)
    print("-", thresholds_output)

if __name__ == "__main__":
    main()

```

A.13. 07_04_2_5_final_student_profile.py

Χαρακτήρας: Κύριο script - Σύνθεση τελικού προφίλ

Περιγραφή: Κανονικοποιεί τα αναγνωριστικά και συνενώνει, με outer joins, τους δείκτες συμμετοχής, επίδοσης, SNA, κοινωνικών ρόλων και απογοήτευσης. Δημιουργεί τόσο το πλήρες μαθησιακό προφίλ όσο και το περιορισμένο σύνολο χαρακτηριστικών που χρησιμοποιείται στη στρωματοποίηση και την ομαδοποίηση.

Είσοδοι: 04.2.1_participation_features.csv · 04.2.2_academic_performance_features_MIN.csv · 04c_4_2_3_student_student_nodes_metrics.csv · 04e_4_2_3_student_social_roles.csv · 06_04_2_4_frustration_summary.csv

Έξοδοι: 07_04_2_5_full_learning_profile_all_features.csv · 07_04_2_5_grouping_profile_final_features.csv

Πηγαίος κώδικας

```
from pathlib import Path
import pandas as pd

# =====
# 1. PATHS
# =====

folder = Path(__file__).parent

participation_file = folder / "04.2.1_participation_features.csv"
performance_file = folder / "04.2.2_academic_performance_features_MIN.csv"
sna_file = folder / "04c_4_2_3_student_student_nodes_metrics.csv"
roles_file = folder / "04e_4_2_3_student_social_roles.csv"
frustration_file = folder / "06_04_2_4_frustration_summary.csv"

output_full_profile = (
    folder / "07_04_2_5_full_learning_profile_all_features.csv"
)

output_grouping_profile = (
    folder / "07_04_2_5_grouping_profile_final_features.csv"
)

# =====
# 2. HELPERS
# =====

def read_csv_flexible(path: Path):

    encodings = ["utf-8-sig", "utf-8", "cp1253", "latin1"]
    separators = [",", ";"]

    last_error = None

    for enc in encodings:
        for sep in separators:
            try:
                df = pd.read_csv(path, encoding=enc, sep=sep)

                if df.shape[1] > 1:
                    return df

            except Exception as e:
                last_error = e

    raise ValueError(
        f"Αποτυχία ανάγνωσης αρχείου {path}\n"
        f"Τελευταίο σφάλμα: {last_error}"
    )

def normalize_user_id(value):

    if pd.isna(value):
        return pd.NA

    s = str(value).strip()
```

```

    if s == "":
        return pd.NA

    s_lower = s.lower()

    if s_lower.startswith("user"):

        suffix = s_lower.replace("user", "").strip()

        if suffix.isdigit():
            return f"user{int(suffix):03d}"

        return s_lower

    if s.isdigit():
        return f"user{int(s):03d}"

    return s_lower

def ensure_user_id(df, source_col):

    df = df.copy()

    if source_col not in df.columns:
        raise ValueError(
            f"Δεν βρέθηκε η στήλη '{source_col}'"
        )

    df["user_id"] = df[source_col].apply(normalize_user_id)

    return df

def require_columns(df, cols, name):

    missing = [c for c in cols if c not in df.columns]

    if missing:
        raise ValueError(
            f"Στο αρχείο {name} λείπουν οι στήλες: {missing}"
        )

def check_duplicates(df, name):

    duplicates = (
        df["user_id"]
        .astype(str)
        .duplicated()
        .sum()
    )

    if duplicates > 0:

        raise ValueError(
            f"Το αρχείο {name} έχει "
            f"{duplicates} διπλότυπα user_id"
        )

# =====
# 3. LOAD FILES
# =====

df_part = read_csv_flexible(participation_file)
df_perf = read_csv_flexible(performance_file)
df_sna = read_csv_flexible(sna_file)
df_roles = read_csv_flexible(roles_file)
df_frust = read_csv_flexible(frustration_file)

# =====
# 4. USER ID NORMALIZATION
# =====

df_part = ensure_user_id(df_part, "user_id")
df_perf = ensure_user_id(df_perf, "user_id")
df_sna = ensure_user_id(df_sna, "Id")
df_roles = ensure_user_id(df_roles, "Id")

if "user_id" in df_frust.columns:
    df_frust = ensure_user_id(df_frust, "user_id")
else:
    df_frust = ensure_user_id(df_frust, "User full name")

```

```

# =====
# 5. PARTICIPATION FEATURES
# =====

participation_cols = [
    "user_id",
    "total_events",
    "forum_posts",
    "forum_views",
    "quiz_attempts",
    "assignment_submissions",
    "resource_views",
    "course_views",
    "active_days",
    "avg_events_per_active_day",
    "forum_engagement_ratio"
]

require_columns(df_part, participation_cols, "Participation")
df_part = df_part[participation_cols].copy()

# =====
# 6. PERFORMANCE FEATURES
# =====

performance_cols = [
    "user_id",
    "final_exam",
    "completed_assignments",
    "avg_assignments_nonzero"
]

require_columns(df_perf, performance_cols, "Performance")
df_perf = df_perf[performance_cols].copy()

# =====
# 7. SNA FEATURES
# =====

sna_cols = [
    "user_id",
    "Initial_active_threads",
    "Initial_active_weight",
    "sna_degree",
    "sna_weighted_degree",
    "sna_degree_centrality",
    "sna_betweenness",
    "sna_closeness",
    "sna_pagerank"
]

require_columns(df_sna, sna_cols, "SNA")
df_sna = df_sna[sna_cols].copy()

# =====
# 8. ROLES
# =====

role_cols = [
    "user_id",
    "network_role",
    "network_role_el"
]

require_columns(df_roles, role_cols, "Roles")
df_roles = df_roles[role_cols].copy()

# =====
# 9. FRUSTRATION
# =====

frustration_cols = [
    "user_id",
    "frustration_signal_count",
    "frustration_episode_count"
]

require_columns(df_frust, frustration_cols, "Frustration")
df_frust = df_frust[frustration_cols].copy()

# =====
# 10. DUPLICATE CHECKS

```

```

# =====

check_duplicates(df_part, "Participation")
check_duplicates(df_perf, "Performance")
check_duplicates(df_sna, "SNA")
check_duplicates(df_roles, "Roles")
check_duplicates(df_frust, "Frustration")

# =====
# 11. MERGE FULL PROFILE
# =====

df_full = df_part.merge(df_perf, on="user_id", how="outer")

df_full = df_full.merge(
    df_sna,
    on="user_id",
    how="outer"
)

df_full = df_full.merge(
    df_roles,
    on="user_id",
    how="outer"
)

df_full = df_full.merge(
    df_frust,
    on="user_id",
    how="outer"
)

# =====
# 12. HANDLE MISSING VALUES
# =====

numeric_cols = (
    df_full
    .select_dtypes(include=["number"])
    .columns
)

df_full[numeric_cols] = (
    df_full[numeric_cols]
    .fillna(0)
)

object_cols = (
    df_full
    .select_dtypes(include=["object", "string"])
    .columns
    .tolist()
)

object_cols = [
    c for c in object_cols
    if c != "user_id"
]

df_full[object_cols] = (
    df_full[object_cols]
    .fillna("unknown")
)

df_full["network_role_el"] = (
    df_full["network_role_el"]
    .replace(
        "unknown",
        "Ανεπαρκής Αλληλεπίδραση"
    )
)

# =====
# 13. FINAL EXAM RENAME
# =====

df_full["final_exam_grade"] = df_full["final_exam"]

# =====
# 14. GROUPING PROFILE
# =====

grouping_cols = [

```

```

"user_id",

# Participation
"active_days",
"avg_events_per_active_day",
"forum_posts",
"forum_views",
"quiz_attempts",

# Performance
"avg_assignments_nonzero",
"completed_assignments",
"final_exam_grade",

# Social abstraction
"network_role_el",

# Frustration
"frustration_signal_count",
"frustration_episode_count"
]

require_columns(
    df_full,
    grouping_cols,
    "Grouping profile"
)

df_grouping = df_full[grouping_cols].copy()

# =====
# 15. SORT
# =====

df_full = (
    df_full
    .sort_values("user_id")
    .reset_index(drop=True)
)

df_grouping = (
    df_grouping
    .sort_values("user_id")
    .reset_index(drop=True)
)

# =====
# 16. SAVE
# =====

df_full.to_csv(
    output_full_profile,
    index=False,
    encoding="utf-8-sig"
)

df_grouping.to_csv(
    output_grouping_profile,
    index=False,
    encoding="utf-8-sig"
)

# =====
# 17. REPORT
# =====

print("\nΤελικό πλήρες προφίλ")
print("-----")
print("Rows:", len(df_full))
print("Columns:", df_full.shape[1])
print("Unique user_id:", df_full["user_id"].nunique())

print("\nΤελικό grouping profile")
print("-----")
print("Rows:", len(df_grouping))
print("Columns:", df_grouping.shape[1])
print("Unique user_id:", df_grouping["user_id"].nunique())

print("\nΣτήλες grouping profile:")

for col in df_grouping.columns:
    print("-", col)

```

```
print("\nΑποθηκεύτηκαν:")  
print("-", output_full_profile.name)  
print("-", output_grouping_profile.name)
```

A.14. 07b_04_2_5_profile_visualization_REVISIED.py

Χαρακτήρας: Βοηθητικό script - Τελικό προφίλ - οπτικοποίηση

Περιγραφή: Ελέγχει την κατανομή των κοινωνικών ρόλων και δημιουργεί boxplots των χαρακτηριστικών συμμετοχής, επίδοσης και απογοήτευσης του τελικού προφίλ.

Είσοδοι: 07_04_2_5_grouping_profile_final_features.csv

Έξοδοι: 07b_04_2_5_network_roles_distribution.png · 07b_04_2_5_network_roles_distribution_summary.csv · 07b_04_2_5_participation_features_boxplots.png · 07b_04_2_5_performance_features_boxplots.png · 07b_04_2_5_frustration_features_boxplots.png

Πηγαίος κώδικας

```
from pathlib import Path
import pandas as pd
import matplotlib.pyplot as plt

# =====
# 1. PATHS
# =====

folder = Path(__file__).parent

input_file = (
    folder / "07_04_2_5_grouping_profile_final_features.csv"
)

role_plot_output = (
    folder / "07b_04_2_5_network_roles_distribution.png"
)

# Extra output for checking totals
role_summary_output = (
    folder / "07b_04_2_5_network_roles_distribution_summary.csv"
)

participation_plot_output = (
    folder / "07b_04_2_5_participation_features_boxplots.png"
)

performance_plot_output = (
    folder / "07b_04_2_5_performance_features_boxplots.png"
)

frustration_plot_output = (
    folder / "07b_04_2_5_frustration_features_boxplots.png"
)

# =====
# 2. LOAD
# =====

df = pd.read_csv(input_file, encoding="utf-8-sig")

# =====
# 3. COLUMN CHECK
# =====

required_cols = [
    "user_id",
    "active_days",
    "avg_events_per_active_day",
    "forum_posts",
    "forum_views",
    "quiz_attempts",
    "avg_assignments_nonzero",
    "completed_assignments",
    "final_exam_grade",
    "network_role_el",
    "frustration_signal_count",
    "frustration_episode_count",
]

missing = [
    col for col in required_cols
```

```

    if col not in df.columns
]

if missing:
    raise ValueError(
        f"Λείπουν στήλες: {missing}"
    )

# =====
# 4. NUMERIC CONVERSION
# =====

numeric_cols = [
    "active_days",
    "avg_events_per_active_day",
    "forum_posts",
    "forum_views",
    "quiz_attempts",
    "avg_assignments_nonzero",
    "completed_assignments",
    "final_exam_grade",
    "frustration_signal_count",
    "frustration_episode_count",
]

for col in numeric_cols:
    df[col] = (
        pd.to_numeric(df[col], errors="coerce")
        .fillna(0)
    )

# =====
# 5. ROLE NORMALIZATION AND DISTRIBUTION
# =====

ROLE_REPLACEMENTS = {
    "Χωρίς ισχυρή αλληλεπίδραση": "Ανεπαρκής Αλληλεπίδραση",
    "Χωρίς Ισχυρή Αλληλεπίδραση": "Ανεπαρκής Αλληλεπίδραση",
    "χωρίς ισχυρή αλληλεπίδραση": "Ανεπαρκής Αλληλεπίδραση",
    "Without strong interaction": "Ανεπαρκής Αλληλεπίδραση",
    "No strong interaction": "Ανεπαρκής Αλληλεπίδραση",
    "Insufficient interaction": "Ανεπαρκής Αλληλεπίδραση",
}

df["network_role_el"] = (
    df["network_role_el"]
    .fillna("Ανεπαρκής Αλληλεπίδραση")
    .replace(ROLE_REPLACEMENTS)
)

# Same order and colors as the Gephi role legend, from most central/important
# to weakest interaction category.
role_order = [
    "Ηγέτης",
    "Γέφυρα",
    "Δραστήριος Συμμετέχων",
    "Περιφερειακός Συμμετέχων",
    "Ανεπαρκής Αλληλεπίδραση",
]

role_colors = {
    "Ηγέτης": "#ff6f45",
    "Γέφυρα": "#2aa36b",
    "Δραστήριος Συμμετέχων": "#00b8e6",
    "Περιφερειακός Συμμετέχων": "#d86adf",
    "Ανεπαρκής Αλληλεπίδραση": "#7f7f7f",
}

unknown_roles = sorted(
    set(df["network_role_el"].dropna().unique())
    - set(role_order)
)

if unknown_roles:
    raise ValueError(
        "Βρέθηκαν άγνωστοι κοινωνικοί ρόλοι στο τελικό προφίλ: "
        f"{unknown_roles}"
    )

role_counts = (
    df["network_role_el"]
    .value_counts()
    .reindex(role_order, fill_value=0)
)

```

```

role_percentages = (role_counts / len(df) * 100).round(2)

role_summary = pd.DataFrame({
    "network_role_el": role_order,
    "count": role_counts.values,
    "percentage": role_percentages.values,
})

role_summary.to_csv(
    role_summary_output,
    index=False,
    encoding="utf-8-sig"
)

# -----
# Totals / consistency checks
# -----

total_rows = len(df)
unique_users = df["user_id"].nunique()
role_total = int(role_counts.sum())
duplicate_users = total_rows - unique_users

print("\n=====")
print("Έλεγχος συνόλων τελικού προφίλ")
print("=====")
print(f"Γραμμές τελικού προφίλ: {total_rows}")
print(f"Μοναδικοί φοιτητές user_id: {unique_users}")
print(f"Σύνολο από κατανομή ρόλων: {role_total}")
print(f"Διπλότυπα user_id: {duplicate_users}")

if role_total != total_rows:
    raise ValueError(
        "Το άθροισμα των ρόλων δεν συμφωνεί με το πλήθος γραμμών "
        f"({role_total} != {total_rows})."
    )

if duplicate_users != 0:
    print("ΠΡΟΕΙΔΟΠΟΙΗΣΗ: Υπάρχουν διπλότυπα user_id στο τελικό προφίλ.")

if total_rows not in [382, 383]:
    print(
        "ΠΡΟΕΙΔΟΠΟΙΗΣΗ: Το σύνολο φοιτητών δεν είναι 382 ή 383. "
        "Έλεγξε αν αυτό είναι αναμενόμενο."
    )

print("\nΚατανομή κοινωνικών ρόλων:")
print(role_summary.to_string(index=False))

# -----
# Plot: social roles distribution
# -----

plt.figure(figsize=(11, 6))

bars = plt.barh(
    role_summary["network_role_el"],
    role_summary["count"],
    color=[role_colors[role] for role in role_summary["network_role_el"]],
    edgecolor="black",
    linewidth=0.6,
)

plt.title(
    "Κατανομή κοινωνικών ρόλων στο τελικό προφίλ"
)

plt.xlabel("Πλήθος φοιτητών")
plt.ylabel("Κοινωνικός ρόλος")

# Put most important role at the top.
plt.gca().invert_yaxis()

# Add count and percentage labels next to bars.
max_count = role_summary["count"].max()
for bar, count, percentage in zip(
    bars,
    role_summary["count"],
    role_summary["percentage"]
):
    x = bar.get_width()
    y = bar.get_y() + bar.get_height() / 2
    plt.text(
        x + max_count * 0.015,
        y,
        f"{int(count)} ({percentage:.2f}%)",
        va="center",
    )

```

```

        fontsize=10,
    )

plt.xlim(0, max_count * 1.18)
plt.grid(axis="x", alpha=0.25)
plt.tight_layout()

plt.savefig(
    role_plot_output,
    dpi=300,
    bbox_inches="tight"
)

plt.close()

# =====
# 6. BOXPLOT HELPER
# =====

def save_boxplot(
    dataframe,
    columns,
    title,
    output_path
):
    data = [
        dataframe[col].dropna()
        for col in columns
    ]

    plt.figure(figsize=(12, 6))

    plt.boxplot(
        data,
        tick_labels=columns,
        showmeans=True
    )

    plt.title(title)

    plt.ylabel("Τιμή χαρακτηριστικού")

    plt.xticks(
        rotation=30,
        ha="right"
    )

    plt.grid(True, alpha=0.3)

    plt.tight_layout()

    plt.savefig(
        output_path,
        dpi=300
    )

    plt.close()

# =====
# 7. PARTICIPATION BOXPLOTS
# =====

participation_cols = [
    "active_days",
    "avg_events_per_active_day",
    "forum_posts",
    "forum_views",
    "quiz_attempts",
]

save_boxplot(
    df,
    participation_cols,
    "Κατανομή χαρακτηριστικών συμμετοχής",
    participation_plot_output
)

# =====
# 8. PERFORMANCE BOXPLOTS
# =====

performance_cols = [
    "avg_assignments_nonzero",

```

```

        "completed_assignments",
        "final_exam_grade",
    ]

    save_boxplot(
        df,
        performance_cols,
        "Κατανομή χαρακτηριστικών επίδοσης",
        performance_plot_output
    )

# =====
# 9. FRUSTRATION BOXPLOTS
# =====

frustration_cols = [
    "frustration_signal_count",
    "frustration_episode_count",
]

save_boxplot(
    df,
    frustration_cols,
    "Κατανομή χαρακτηριστικών απογοήτευσης",
    frustration_plot_output
)

# =====
# 10. DONE
# =====

print("\nΑποθηκεύτηκαν:")

print("-", role_plot_output.name)
print("-", role_summary_output.name)
print("-", participation_plot_output.name)
print("-", performance_plot_output.name)
print("-", frustration_plot_output.name)

```

A.15. 07c_04_2_5_numerical_validation_report.py

Χαρακτήρας: Έλεγχος script - Τελικό προφίλ - αριθμητικός έλεγχος

Περιγραφή: Ελέγχει το συνολικό πλήθος και τη μοναδικότητα των φοιτητών, τη συνέπεια μεταξύ degree και κοινωνικού ρόλου, τις κενές ή αρνητικές τιμές και τα βασικά περιγραφικά στατιστικά των δύο αρχείων τελικού προφίλ.

Είσοδοι: 07_04_2_5_full_learning_profile_all_features.csv · 07_04_2_5_grouping_profile_final_features.csv

Έξοδοι: 07c_04_2_5_numerical_validation_report.txt

Πηγαίος κώδικας

```
# 07c_04_2_5_numerical_validation_report.py
# -----
# Αριθμητικός έλεγχος συνέπειας του τελικού προφίλ
#
# Ελέγχει:
# - Συνολικό πλήθος φοιτητών
# - Ενεργούς φοιτητές στο forum / SNA
# - Κατανομή κοινωνικών ρόλων
# - Συνέπεια μεταξύ SNA και ρόλων
# - Κενές τιμές
# - Αρνητικές τιμές
# - Περιγραφικά στατιστικά
#
# Input:
# 07_04_2_5_full_learning_profile_all_features.csv
# 07_04_2_5_grouping_profile_final_features.csv
#
# Output:
# 07c_04_2_5_numerical_validation_report.txt
# -----

from pathlib import Path
import pandas as pd

# =====
# PATHS
# =====

folder = Path(__file__).parent

full_profile_file = (
    folder / "07_04_2_5_full_learning_profile_all_features.csv"
)

grouping_profile_file = (
    folder / "07_04_2_5_grouping_profile_final_features.csv"
)

output_report = (
    folder / "07c_04_2_5_numerical_validation_report.txt"
)

# =====
# HELPERS
# =====

def section(title):

    return [
        """
        """ * 60,
        title,
        """
        """ * 60,
    ]

def check(label, condition, ok_msg="", fail_msg=""):

    status = "OK" if condition else "ΠΡΟΒΛΗΜΑ"

    msg = ok_msg if condition else fail_msg

    return f" [{status}] {label}: {msg}"
```

```

# =====
# LOAD
# =====

df_full = pd.read_csv(
    full_profile_file,
    encoding="utf-8-sig"
)

df_group = pd.read_csv(
    grouping_profile_file,
    encoding="utf-8-sig"
)

report = []

report.append(
    "07c - Αριθμητικός έλεγχος τελικού προφίλ φοιτητών"
)

report.append("=" * 60)

report.append(
    f"Full profile:      {full_profile_file.name}"
)

report.append(
    f"Grouping profile: {grouping_profile_file.name}"
)

# =====
# 1. ΠΛΗΘΟΣ ΦΟΙΤΗΤΩΝ
# =====

report += section("1. ΠΛΗΘΟΣ ΦΟΙΤΗΤΩΝ")

total_full = df_full["user_id"].nunique()
total_group = df_group["user_id"].nunique()

report.append(
    f" Full profile - σύνολο φοιτητών: {total_full}"
)

report.append(
    f" Grouping profile - σύνολο φοιτητών: {total_group}"
)

report.append(
    check(
        "Ισότητα πλήθους",
        total_full == total_group,
        ok_msg=f"{total_full} φοιτητές και στα δύο αρχεία",
        fail_msg=f"Διαφορά: {abs(total_full - total_group)} φοιτητές"
    )
)

report.append(
    check(
        "Αναμενόμενο πλήθος (383)",
        total_full == 383,
        ok_msg="383 φοιτητές",
        fail_msg=f"Βρέθηκαν {total_full} αντί 383"
    )
)

# =====
# DUPLICATES
# =====

dup_full = df_full["user_id"].duplicated().sum()
dup_group = df_group["user_id"].duplicated().sum()

report.append(
    check(
        "Διπλότυπα full profile",
        dup_full == 0,
        ok_msg="Κανένα",
        fail_msg=f"{dup_full} διπλότυπα"
    )
)

report.append(
    check(

```

```

        "Διπλότυπα grouping profile",
        dup_group == 0,
        ok_msg="Κανένα",
        fail_msg=f"{dup_group} διπλότυπα"
    )
)

# =====
# 2. ΕΝΕΡΓΟΙ ΦΟΙΤΗΤΕΣ ΣΤΟ SNA
# =====

report += section(
    "2. ΕΝΕΡΓΟΙ ΦΟΙΤΗΤΕΣ ΣΤΟ FORUM / SNA"
)

if "sna_degree" in df_full.columns:

    students_with_sna = (
        df_full["sna_degree"] > 0
    ).sum()

    students_without_sna = (
        df_full["sna_degree"] == 0
    ).sum()

    pct_with = round(
        students_with_sna / total_full * 100,
        1
    )

    pct_without = round(
        students_without_sna / total_full * 100,
        1
    )

    report.append(
        f" Με sna_degree > 0: "
        f"{students_with_sna} ({pct_with}%)"
    )

    report.append(
        f" Με sna_degree = 0: "
        f"{students_without_sna} ({pct_without}%)"
    )

    report.append(
        check(
            "Άθροισμα = σύνολο",
            students_with_sna + students_without_sna == total_full,
            ok_msg="Σωστό",
            fail_msg="Δεν συμφωνεί"
        )
    )

else:

    report.append(
        " [ΠΑΡΑΤΗΡΗΣΗ] "
        "Δεν βρέθηκε στήλη sna_degree."
    )

# =====
# 3. ΚΑΤΑΝΟΜΗ ΚΟΙΝΩΝΙΚΩΝ ΡΟΛΩΝ
# =====

report += section(
    "3. ΚΑΤΑΝΟΜΗ ΚΟΙΝΩΝΙΚΩΝ ΡΟΛΩΝ"
)

for df_name, df in [
    ("Full profile", df_full),
    ("Grouping profile", df_group)
]:

    if "network_role_el" not in df.columns:

        report.append(
            f" [{df_name}] "
            "Δεν βρέθηκε στήλη network_role_el."
        )

        continue

```

```

role_counts = (
    df["network_role_el"]
    .value_counts(dropna=False)
)

total_roles = role_counts.sum()

report.append(
    f"\n {df_name} ({total_roles} φοιτητές):"
)

for role, count in role_counts.items():

    pct = round(
        count / total_roles * 100,
        1
    )

    report.append(
        f"    {role}: {count} ({pct}%)"
    )

report.append(
    check(
        f"{df_name} - άθροισμα ρόλων",
        total_roles == (
            total_full
            if df_name == "Full profile"
            else total_group
        ),
        ok_msg="Σωστό",
        fail_msg=(
            f"Αναμενόταν {total_full}, "
            f"βρέθηκαν {total_roles}"
        )
    )
)

# =====
# 4. ΣΥΝΕΠΕΙΑ SNA DEGREE VS ΡΟΛΟΣ
# =====

report += section(
    "4. ΣΥΝΕΠΕΙΑ SNA DEGREE = 0 VS ΡΟΛΟΣ"
)

if (
    "sna_degree" in df_full.columns
    and "network_role_el" in df_full.columns
):

    degree_zero = df_full[
        df_full["sna_degree"] == 0
    ]

    no_interaction = df_full[
        df_full["network_role_el"]
        .str.lower()
        .isin(["ανεπαρκής αλληλεπίδραση"])
    ]

    report.append(
        f"    sna_degree = 0: "
        f"{len(degree_zero)}"
    )

    report.append(
        f"    Ανεπαρκής Αλληλεπίδραση: "
        f"{len(no_interaction)}"
    )

    report.append(
        check(
            "Ισότητα",
            len(degree_zero) == len(no_interaction),
            ok_msg="Συνεπές",
            fail_msg=(
                f"Διαφορά: "
                f"{abs(len(degree_zero) - len(no_interaction))}"
            )
        )
    )

    inconsistent = df_full[
        (df_full["sna_degree"] == 0)

```

```

        &
        (
            ~df_full["network_role_e1"]
            .str.lower()
            .isin(["ανεπαρκής αλληλεπίδραση"])
        )
    ]

    report.append(
        check(
            "Degree=0 με άλλο ρόλο",
            len(inconsistent) == 0,
            ok_msg="Κανένας",
            fail_msg=(
                f"{len(inconsistent)} "
                "ασυνεπείς εγγραφές"
            )
        )
    )
)

else:

    report.append(
        " Δεν είναι δυνατός ο έλεγχος."
    )

# =====
# 5. KENEΣ TIMEΣ
# =====

report += section(
    "5. KENEΣ TIMEΣ ANA ΠΕΔΙΟ"
)

report.append("\n Grouping profile:")

for col in df_group.columns:

    null_count = df_group[col].isna().sum()

    if null_count > 0:

        report.append(
            f" [ΠΡΟΒΛΗΜΑ] "
            f"{col}: {null_count} κενές τιμές"
        )

    else:

        report.append(
            f" [OK] {col}: 0 κενές τιμές"
        )

# =====
# 6. APNHTIKEΣ TIMEΣ
# =====

report += section(
    "6. APIΘMHTIKA OPIA"
)

numeric_check_cols = [

    "active_days",
    "avg_events_per_active_day",
    "forum_posts",
    "forum_views",
    "quiz_attempts",
    "avg_assignments_nonzero",
    "completed_assignments",
    "final_exam_grade",
    "frustration_signal_count",
    "frustration_episode_count",
]

for col in numeric_check_cols:

    if col not in df_group.columns:

        report.append(
            f" [ΠΑΡΑΤΗΡΗΣΗ] "
            f"Δεν βρέθηκε στήλη: {col}"
        )

```

```

        continue

    neg_count = (
        df_group[col] < 0
    ).sum()

    report.append(
        check(
            f"{col} >= 0",
            neg_count == 0,
            ok_msg="Σωστό",
            fail_msg=f"{neg_count} αρνητικές τιμές"
        )
    )

# =====
# 7. ΠΕΡΙΓΡΑΦΙΚΑ ΣΤΑΤΙΣΤΙΚΑ
# =====

report += section(
    "7. ΠΕΡΙΓΡΑΦΙΚΑ ΣΤΑΤΙΣΤΙΚΑ"
)

numeric_cols_present = [
    c for c in numeric_check_cols
    if c in df_group.columns
]

desc = (
    df_group[numeric_cols_present]
    .describe()
    .T
)

desc = desc[
    ["count", "mean", "std", "min", "50%", "max"]
].round(3)

report.append("")

report.append(
    f"  {'Πεδίο':<35}"
    f"{'Count':>6}"
    f"{'Mean':>8}"
    f"{'Std':>8}"
    f"{'Min':>6}"
    f"{'Median':>8}"
    f"{'Max':>8}"
)

report.append(" " + "-" * 85)

for col in numeric_cols_present:

    row = desc.loc[col]

    report.append(
        f"  {'col':<35}"
        f"{'int(row['count']):>6}"
        f"{'row['mean']:>8.2f}"
        f"{'row['std']:>8.2f}"
        f"{'row['min']:>6.2f}"
        f"{'row['50%']:>8.2f}"
        f"{'row['max']:>8.2f}"
    )

# =====
# 8. SUMMARY
# =====

report += section("8. ΣΥΝΟΨΗ")

problems = [
    line for line in report
    if "[ΠΡΟΒΛΗΜΑ]" in line
]

if problems:

    report.append(
        f"  Βρέθηκαν {len(problems)} προβλήματα:"
    )

```

```

        for p in problems:
            report.append(f" {p.strip()}")

else:

    report.append(
        " Κανένα πρόβλημα – "
        "το προφίλ είναι συνεπές."
    )

# =====
# SAVE
# =====

report_text = "\n".join(report)

output_report.write_text(
    report_text,
    encoding="utf-8-sig"
)

print(report_text)

print(
    f"\nΑποθηκεύτηκε: "
    f"{output_report.name}"
)

```

A.16. 08_04_2_6_student_profile_stratification.py

Χαρακτήρας: Κύριο script - Στρωματοποίηση

Περιγραφή: Τυποποιεί τα χαρακτηριστικά, υπολογίζει τις τέσσερις επιμέρους διαστάσεις και τον σταθμισμένο σύνθετο δείκτη προφίλ και κατατάσσει τους φοιτητές σε πέντε στρώματα βάσει των ορίων του profile_score_z.

Είσοδοι: 07_04_2_5_grouping_profile_final_features.csv

Έξοδοι: 08_04_2_6_student_profile_strata.csv · 08_04_2_6_for_grouping.csv · 08_04_2_6_stratum_summary.csv · 08_04_2_6_means_by_stratum.csv · 08_04_2_6_std_by_stratum.csv · 08_04_2_6_stratification_report.txt

Πηγαίος κώδικας

```
# 08_04_2_6_student_profile_stratification.py
# =====
# STUDENT PROFILE STRATIFICATION
#
# =====

from pathlib import Path

import pandas as pd

# =====
# PATHS
# =====

folder = Path(__file__).parent

INPUT_FILE = folder / "07_04_2_5_grouping_profile_final_features.csv"

OUTPUT_FILE = folder / "08_04_2_6_student_profile_strata.csv"

GROUP_FILE = folder / "08_04_2_6_for_grouping.csv"

SUMMARY_FILE = folder / "08_04_2_6_stratum_summary.csv"

MEANS_FILE = folder / "08_04_2_6_means_by_stratum.csv"

STD_FILE = folder / "08_04_2_6_std_by_stratum.csv"

REPORT_FILE = folder / "08_04_2_6_stratification_report.txt"

# =====
# LOAD DATA
# =====

df = pd.read_csv(
    INPUT_FILE,
    encoding="utf-8-sig"
)

print("\n=====")
print("Student Profile Stratification")
print("=====")
print("Input file:", INPUT_FILE.name)
print("Students:", len(df))

# =====
# REQUIRED COLUMNS CHECK
# =====

required_columns = [

    "user_id",

    # Engagement
    "active_days",
    "avg_events_per_active_day",
    "forum_posts",
    "forum_views",
    "quiz_attempts",
    "completed_assignments",
```

```

# Performance
"final_exam_grade",
"avg_assignments_nonzero",

# Social role
"network_role_el",

# Frustration
"frustration_signal_count",
"frustration_episode_count"
]

missing_columns = [
    col for col in required_columns
    if col not in df.columns
]

if missing_columns:
    raise ValueError(
        "Missing required columns: "
        + ", ".join(missing_columns)
    )

# =====
# SOCIAL ROLE → ORDINAL SCORE
# =====

role_score_map = {

    "Ανεπαρκής Αλληλεπίδραση": 0,

    "Περιφερειακός Συμμετέχων": 1,

    "Δραστήριος Συμμετέχων": 2,

    "Γέφυρα": 3,

    "Ηγέτης": 4
}

df["social_role_score"] = (
    df["network_role_el"]
    .map(role_score_map)
)

unknown_roles = df[df["social_role_score"].isna()][df["network_role_el"].unique()]

if len(unknown_roles) > 0:
    print("\nWARNING: Unknown social role labels found:")
    for role in unknown_roles:
        print("-", role)

    print("Unknown roles will be coded as 0.")

    df["social_role_score"] = df["social_role_score"].fillna(0)

# =====
# FEATURE GROUPS
# =====

engagement_features = [

    "active_days",

    "avg_events_per_active_day",

    "forum_posts",

    "forum_views",

    "quiz_attempts",

    "completed_assignments"
]

performance_features = [

    "final_exam_grade",

    "avg_assignments_nonzero"
]

social_features = [

```

```

    "social_role_score"
]

frustration_features = [
    "frustration_signal_count",
    "frustration_episode_count"
]

all_features = (
    engagement_features
    + performance_features
    + social_features
    + frustration_features
)

# =====
# NUMERIC CONVERSION AND MISSING VALUES
# =====

for col in all_features:
    df[col] = pd.to_numeric(
        df[col],
        errors="coerce"
    )

missing_values = df[all_features].isna().sum()

if missing_values.sum() > 0:
    print("\nWARNING: Missing numeric values found.")
    print("They will be replaced with 0.\n")
    print(missing_values[missing_values > 0])

    df[all_features] = df[all_features].fillna(0)

# =====
# FEATURE WEIGHTS WITHIN DIMENSIONS
# =====

engagement_weights = {
    "active_days": 0.20,
    "avg_events_per_active_day": 0.15,
    "forum_posts": 0.20,
    "forum_views": 0.10,
    "quiz_attempts": 0.15,
    "completed_assignments": 0.20
}

performance_weights = {
    "final_exam_grade": 0.60,
    "avg_assignments_nonzero": 0.40
}

social_weights = {
    "social_role_score": 1.00
}

frustration_weights = {
    "frustration_signal_count": 0.50,
    "frustration_episode_count": 0.50
}

# =====
# DIMENSION WEIGHTS
# =====

dimension_weights = {
    "engagement": 0.40,

```

```

    "performance": 0.25,

    "social": 0.25,

    "frustration": 0.10
}

# =====
# CHECK THAT WEIGHTS SUM TO 1
# =====

def check_weights_sum(weights_dict, name):

    total = sum(weights_dict.values())

    if round(total, 6) != 1.0:
        raise ValueError(
            f"Weights for {name} do not sum to 1. "
            f"Current sum: {total}"
        )

check_weights_sum(engagement_weights, "engagement")
check_weights_sum(performance_weights, "performance")
check_weights_sum(social_weights, "social")
check_weights_sum(frustration_weights, "frustration")
check_weights_sum(dimension_weights, "dimensions")

# =====
# STANDARDIZATION: MANUAL Z-SCORE
#  $z = (x - \text{mean}) / \text{std}$ 
# ddof=0 because the full student population is used
# =====

scaled_df = pd.DataFrame(index=df.index)

zscore_statistics = {}

for col in all_features:

    mean_val = df[col].mean()

    std_val = df[col].std(ddof=0)

    zscore_statistics[col] = {
        "mean": mean_val,
        "std": std_val
    }

    if std_val == 0:

        scaled_df[f"{col}_z"] = 0

    else:

        scaled_df[f"{col}_z"] = (
            (df[col] - mean_val) / std_val
        )

# =====
# REVERSE FRUSTRATION DIRECTION
# Higher Frustration should reduce the final profile score
# =====

for feature in frustration_features:

    scaled_df[f"{feature}_z"] = (
        -1 * scaled_df[f"{feature}_z"]
    )

# =====
# COMPUTE DIMENSION SCORES
# =====

df["engagement_score"] = sum(

    scaled_df[f"{feature}_z"] * weight

    for feature, weight
    in engagement_weights.items()
)

```

```

df["performance_score"] = sum(
    scaled_df[f"{feature}_z"] * weight
    for feature, weight
    in performance_weights.items()
)

df["social_score"] = sum(
    scaled_df[f"{feature}_z"] * weight
    for feature, weight
    in social_weights.items()
)

df["frustration_score"] = sum(
    scaled_df[f"{feature}_z"] * weight
    for feature, weight
    in frustration_weights.items()
)

# =====
# COMPUTE FINAL PROFILE SCORE
# =====

df["profile_score"] = (
    df["engagement_score"]
    * dimension_weights["engagement"]
    +
    df["performance_score"]
    * dimension_weights["performance"]
    +
    df["social_score"]
    * dimension_weights["social"]
    +
    df["frustration_score"]
    * dimension_weights["frustration"]
)

# =====
# PROFILE PERCENTILE
# =====

df["profile_percentile"] = (
    df["profile_score"]
    .rank(pct=True)
)

# =====
# STANDARDIZE FINAL PROFILE SCORE
# =====

profile_mean = df["profile_score"].mean()
profile_std = df["profile_score"].std(ddof=0)
if profile_std == 0:
    df["profile_score_z"] = 0
else:
    df["profile_score_z"] = (
        (df["profile_score"] - profile_mean)
        / profile_std
    )

# =====
# ASSIGN STRATA
# =====

def assign_stratum(z):

```

```

    if z < -1.0:
        return "Πολύ Χαμηλό"

    elif z < -0.3:
        return "Χαμηλό"

    elif z <= 0.3:
        return "Μεσαίο"

    elif z <= 1.0:
        return "Υψηλό"

    else:
        return "Πολύ Υψηλό"

df["profile_stratum"] = (
    df["profile_score_z"]
    .apply(assign_stratum)
)

# =====
# STRATUM CODES
# =====

stratum_code_map = {

    "Πολύ Χαμηλό": 1,

    "Χαμηλό": 2,

    "Μεσαίο": 3,

    "Υψηλό": 4,

    "Πολύ Υψηλό": 5
}

df["profile_stratum_code"] = (
    df["profile_stratum"]
    .map(stratum_code_map)
)

stratum_order = [

    "Πολύ Χαμηλό",

    "Χαμηλό",

    "Μεσαίο",

    "Υψηλό",

    "Πολύ Υψηλό"
]

# =====
# APPEND Z-SCORE COLUMNS
# =====

df = pd.concat(
    [df, scaled_df],
    axis=1
)

# =====
# SAVE FULL STRATIFICATION OUTPUT
# =====

df.to_csv(
    OUTPUT_FILE,
    index=False,
    encoding="utf-8-sig"
)

# =====
# SAVE SIMPLIFIED FILE FOR GROUP FORMATION
# =====

grouping_columns = [

    "user_id",

```

```

    "profile_score",
    "profile_score_z",
    "profile_percentile",
    "profile_stratum",
    "profile_stratum_code",
    "engagement_score",
    "performance_score",
    "social_score",
    "frustration_score",
    "network_role_el",
    "social_role_score"
]

df[grouping_columns].to_csv(
    GROUP_FILE,
    index=False,
    encoding="utf-8-sig"
)

# =====
# STRATUM SUMMARY
# =====

counts = (
    df["profile_stratum"]
    .value_counts()
    .reindex(stratum_order)
)

percentages = (
    counts / len(df) * 100
).round(2)

cumulative_percentages = (
    percentages.cumsum()
).round(2)

summary_df = pd.DataFrame({
    "count": counts,
    "percentage": percentages,
    "cumulative_percentage": cumulative_percentages
})

summary_df.to_csv(
    SUMMARY_FILE,
    encoding="utf-8-sig"
)

# =====
# MEANS AND STANDARD DEVIATIONS BY STRATUM
# =====

selected_columns = [
    "active_days",
    "forum_posts",
    "social_role_score",
    "final_exam_grade",
    "profile_score"
]

means_by_stratum = (

```

```

df.groupby("profile_stratum")[selected_columns]

    .mean()

    .reindex(stratum_order)
)

std_by_stratum = (

    df.groupby("profile_stratum")[selected_columns]

    .std()

    .reindex(stratum_order)
)

means_by_stratum.to_csv(
    MEANS_FILE,
    encoding="utf-8-sig"
)

std_by_stratum.to_csv(
    STD_FILE,
    encoding="utf-8-sig"
)

# =====
# TEXT REPORT
# =====

with open(REPORT_FILE, "w", encoding="utf-8") as report:

    report.write("STUDENT PROFILE STRATIFICATION REPORT\n")
    report.write("=====\n\n")

    report.write(f"Input file: {INPUT_FILE.name}\n")
    report.write(f"Students: {len(df)}\n\n")

    report.write("Output files:\n")
    report.write(f"- {OUTPUT_FILE.name}\n")
    report.write(f"- {GROUP_FILE.name}\n")
    report.write(f"- {SUMMARY_FILE.name}\n")
    report.write(f"- {MEANS_FILE.name}\n")
    report.write(f"- {STD_FILE.name}\n\n")

    report.write("Dimension weights:\n")
    for key, value in dimension_weights.items():
        report.write(f"- {key}: {value}\n")

    report.write("\nRole score mapping:\n")
    for key, value in role_score_map.items():
        report.write(f"- {key}: {value}\n")

    report.write("\nStratification thresholds:\n")
    report.write("z < -1.0          -> Πολύ Χαμηλό\n")
    report.write("-1.0 <= z < -0.3      -> Χαμηλό\n")
    report.write("-0.3 <= z <= 0.3     -> Μεσαίο\n")
    report.write("0.3 < z <= 1.0       -> Υψηλό\n")
    report.write("z > 1.0             -> Πολύ Υψηλό\n\n")

    report.write("Distribution across strata:\n")
    report.write(summary_df.to_string())
    report.write("\n\n")

    report.write("Mean values by stratum:\n")
    report.write(means_by_stratum.round(2).to_string())
    report.write("\n\n")

    report.write("Standard deviation by stratum:\n")
    report.write(std_by_stratum.round(2).to_string())
    report.write("\n\n")

    report.write("Z-score statistics for original features:\n")
    for feature, stats in zscore_statistics.items():
        report.write(
            f"- {feature}: "
            f"mean={stats['mean']:.4f}, "
            f"std={stats['std']:.4f}\n"
        )

# =====
# FINAL CONSOLE OUTPUT
# =====

```

```

print("\nOutput saved:", OUTPUT_FILE.name)
print("Grouping file saved:", GROUP_FILE.name)
print("Summary file saved:", SUMMARY_FILE.name)
print("Means file saved:", MEANS_FILE.name)
print("Std file saved:", STD_FILE.name)
print("Report file saved:", REPORT_FILE.name)

print("\nDistribution across strata:\n")
print(summary_df)

print("\nMean values by stratum:\n")
print(means_by_stratum.round(2))

print("\n=====")
print("Stratification completed successfully")
print("=====")

print("Students:", len(df))

print("\nThresholds used:")
print("z < -1.0          -> Πολύ Χαμηλό")
print("-1.0 <= z < -0.3   -> Χαμηλό")
print("-0.3 <= z <= 0.3    -> Μεσαίο")
print("0.3 < z <= 1.0      -> Υψηλό")
print("z > 1.0             -> Πολύ Υψηλό")

print("\nDone.")

```

A.17. 08b_04_3_strata_visualization_REVISIED.py

Χαρακτήρας: Βοηθητικό script - Στρωματοποίηση - οπτικοποίηση

Περιγραφή: Παράγει τις γραφικές απεικονίσεις και τους συνοδευτικούς πίνακες της στρωματοποίησης: κατανομή στρωμάτων, ιστογράμματα και boxplots του σύνθετου δείκτη, κοινωνικούς ρόλους ανά στρώμα και μέσες τιμές επιμέρους διαστάσεων.

Είσοδοι: 08_04_2_6_student_profile_strata.csv

Έξοδοι: Φάκελος 08b_04_3_strata_visualizations/ με αρχεία PNG και CSV

Πηγαίος κώδικας

```
# 08b_04_3_strata_visualization_REVISIED.py
# =====
# VISUALIZATION OF STUDENT STRATA
# =====

from pathlib import Path

import pandas as pd
import matplotlib.pyplot as plt

# =====
# PATHS
# =====

folder = Path(__file__).parent

INPUT_FILE = (
    folder /
    "08_04_2_6_student_profile_strata.csv"
)

OUTPUT_DIR = (
    folder /
    "08b_04_3_strata_visualizations"
)

OUTPUT_DIR.mkdir(exist_ok=True)

# =====
# LOAD DATA
# =====

df = pd.read_csv(
    INPUT_FILE,
    encoding="utf-8-sig"
)

print("\n=====")
print("Strata Visualization")
print("=====")
print("Students:", len(df))

# =====
# STRATUM ORDER
# =====

stratum_order = [
    "Πολύ Χαμηλό",
    "Χαμηλό",
    "Μεσαίο",
    "Υψηλό",
    "Πολύ Υψηλό"
]

df["profile_stratum"] = pd.Categorical(
    df["profile_stratum"],
```

```

        categories=stratum_order,
        ordered=True
    )

# =====
# SOCIAL ROLE ORDER AND COLORS – CONSISTENT WITH GEPHI
# =====

ROLE_INSUFFICIENT = "Ανεπαρκής Αλληλεπίδραση"

ROLE_REPLACEMENTS = {
    "Χωρίς ισχυρή αλληλεπίδραση": ROLE_INSUFFICIENT,
    "Χωρίς Ισχυρή Αλληλεπίδραση": ROLE_INSUFFICIENT,
    "χωρίς ισχυρή αλληλεπίδραση": ROLE_INSUFFICIENT,
    "Without strong interaction": ROLE_INSUFFICIENT,
    "No strong interaction": ROLE_INSUFFICIENT,
    "Insufficient interaction": ROLE_INSUFFICIENT,
}

# Σειρά σπουδαιότητας / ερμηνευτικής βαρύτητας
role_order = [
    "Ηγέτης",
    "Γέφυρα",
    "Δραστήριος Συμμετέχων",
    "Περιφερειακός Συμμετέχων",
    ROLE_INSUFFICIENT,
]

# Χρώματα ίδια με το Gephi legend
role_colors = {
    "Ηγέτης": "#ff6f45",
    "Γέφυρα": "#2aa36b",
    "Δραστήριος Συμμετέχων": "#00b8e6",
    "Περιφερειακός Συμμετέχων": "#d86adf",
    ROLE_INSUFFICIENT: "#7f7f7f",
}

# Κανονικοποίηση παλιών labels, ώστε να μη βγει ξανά παλιά ονομασία σε εικόνα/CSV
df["network_role_el"] = (
    df["network_role_el"]
    .fillna(ROLE_INSUFFICIENT)
    .replace(ROLE_REPLACEMENTS)
)

unknown_roles = sorted(
    set(df["network_role_el"].dropna().unique())
    - set(role_order)
)

if unknown_roles:
    raise ValueError(
        f"Άγνωστοι κοινωνικοί ρόλοι στο network_role_el: {unknown_roles}"
    )

df["network_role_el"] = pd.Categorical(
    df["network_role_el"],
    categories=role_order,
    ordered=True
)

# =====
# 1. STRATA DISTRIBUTION
# =====

counts = (
    df["profile_stratum"]
    .value_counts()
    .reindex(stratum_order)
)

percentages = (
    counts / len(df) * 100
).round(2)

summary_df = pd.DataFrame({
    "count": counts,
    "percentage": percentages
})

```

```

summary_df.to_csv(
    OUTPUT_DIR /
    "01_strata_distribution_summary.csv",
    encoding="utf-8-sig"
)

plt.figure(figsize=(8, 5))
counts.plot(kind="bar")
plt.title("Κατανομή Φοιτητών ανά Στρώμα")
plt.xlabel("Στρώμα")
plt.ylabel("Πλήθος Φοιτητών")
plt.xticks(rotation=0)
plt.tight_layout()
plt.savefig(
    OUTPUT_DIR /
    "01_strata_distribution.png",
    dpi=300
)
plt.close()

# =====
# 2. PROFILE SCORE HISTOGRAM
# =====

plt.figure(figsize=(8, 5))

plt.hist(
    df["profile_score"],
    bins=20,
    edgecolor="black"
)

plt.title("Κατανομή Σύνθετου Δείκτη Προφίλ")
plt.xlabel("Profile Score")
plt.ylabel("Συχνότητα")
plt.tight_layout()
plt.savefig(
    OUTPUT_DIR /
    "02_profile_score_histogram.png",
    dpi=300
)
plt.close()

# =====
# 3. PROFILE SCORE Z HISTOGRAM
# =====

plt.figure(figsize=(8, 5))

plt.hist(
    df["profile_score_z"],
    bins=20,
    edgecolor="black"
)

plt.axvline(-1.0, linestyle="--")
plt.axvline(-0.3, linestyle="--")
plt.axvline(0.3, linestyle="--")
plt.axvline(1.0, linestyle="--")

plt.title("Κατανομή Τυποποιημένου Δείκτη Προφίλ")

```

```

plt.xlabel("Profile Score z")
plt.ylabel("Συχνότητα")
plt.tight_layout()
plt.savefig(
    OUTPUT_DIR /
    "03_profile_score_z_histogram.png",
    dpi=300
)
plt.close()

# =====
# 4. PROFILE SCORE BOXPLOT
# =====

boxplot_data = [
    df.loc[
        df["profile_stratum"] == s,
        "profile_score"
    ].dropna()
    for s in stratum_order
]

plt.figure(figsize=(9, 5))
plt.boxplot(
    boxplot_data,
    tick_labels=stratum_order,
    showmeans=True
)

plt.title("Σύνθετος Δείκτης Προφίλ ανά Στρώμα")
plt.xlabel("Στρώμα")
plt.ylabel("Profile Score")
plt.tight_layout()
plt.savefig(
    OUTPUT_DIR /
    "04_profile_score_boxplot.png",
    dpi=300
)
plt.close()

# =====
# 5. SOCIAL ROLES BY STRATUM
# =====

role_distribution = pd.crosstab(
    df["profile_stratum"],
    df["network_role_el"]
).reindex(
    index=stratum_order,
    columns=role_order,
    fill_value=0
)

role_distribution.to_csv(
    OUTPUT_DIR /
    "05_social_roles_by_stratum.csv",
    encoding="utf-8-sig"
)

role_distribution.plot(

```

```

        kind="bar",

        stacked=True,

        figsize=(11, 6),

        color=[role_colors[role] for role in role_distribution.columns]
    )

plt.title("Κοινωνικοί Ρόλοι ανά Στρώμα")
plt.xlabel("Στρώμα")
plt.ylabel("Πλήθος Φοιτητών")
plt.xticks(rotation=0)
plt.legend(
    title="Κοινωνικός Ρόλος",
    bbox_to_anchor=(1.05, 1),
    loc="upper left"
)
plt.tight_layout()
plt.savefig(
    OUTPUT_DIR /
    "05_social_roles_by_stratum.png",
    dpi=300
)
plt.close()

```

```

# =====
# 6. MEAN DIMENSION SCORES
# =====

```

```

dimension_cols = [
    "engagement_score",
    "performance_score",
    "social_score",
    "frustration_score"
]

mean_dimensions = (
    df.groupby(
        "profile_stratum",
        observed=False
    )[dimension_cols]
    .mean()
    .reindex(stratum_order)
)

mean_dimensions.to_csv(
    OUTPUT_DIR /
    "06_mean_dimension_scores.csv",
    encoding="utf-8-sig"
)

plt.figure(figsize=(10, 6))
for col in dimension_cols:
    plt.plot(
        stratum_order,
        mean_dimensions[col],
        marker="o",
        label=col
    )

```

```

    )

plt.title("Μέσες Τιμές Διαστάσεων ανά Στρώμα")
plt.xlabel("Στρώμα")
plt.ylabel("Μέση Τιμή")
plt.legend()
plt.tight_layout()

plt.savefig(
    OUTPUT_DIR /
    "06_mean_dimension_scores.png",
    dpi=300
)

plt.close()

# =====
# 7. KEY VARIABLES BY STRATUM
# =====

key_cols = [
    "active_days",
    "forum_posts",
    "social_role_score",
    "final_exam_grade",
    "profile_score"
]

means_key = (
    df.groupby(
        "profile_stratum",
        observed=False
    )[key_cols]
    .mean()
    .reindex(stratum_order)
)

means_key.to_csv(
    OUTPUT_DIR /
    "07_mean_key_variables.csv",
    encoding="utf-8-sig"
)

plt.figure(figsize=(10, 6))

for col in key_cols:
    plt.plot(
        stratum_order,
        means_key[col],
        marker="o",
        label=col
    )

plt.title("Μέσες Τιμές Βασικών Μεταβλητών ανά Στρώμα")
plt.xlabel("Στρώμα")
plt.ylabel("Μέση Τιμή")
plt.legend()
plt.tight_layout()

plt.savefig(

```

```

    OUTPUT_DIR /
    "07_mean_key_variables.png",

    dpi=300
)

plt.close()

# =====
# 8. ROLE DISTRIBUTION PIE CHART
# =====

role_counts = (
    df["network_role_el"]
    .value_counts()
    .reindex(role_order, fill_value=0)
)

role_summary = pd.DataFrame({
    "count": role_counts,
    "percentage": (role_counts / len(df) * 100).round(2)
})

role_summary.to_csv(
    OUTPUT_DIR / "08_role_distribution_summary.csv",
    encoding="utf-8-sig"
)

# Αφαιρούμε μηδενικές κατηγορίες μόνο από το pie chart
role_counts_plot = role_counts[role_counts > 0]

plt.figure(figsize=(8, 8))

plt.pie(
    role_counts_plot.values,
    labels=role_counts_plot.index,
    autopct="%1.1f%%",
    startangle=90,
    counterclock=False,
    colors=[role_colors[role] for role in role_counts_plot.index]
)

plt.title("Κατανομή Κοινωνικών Ρόλων")

plt.tight_layout()

plt.savefig(
    OUTPUT_DIR /
    "08_role_distribution_pie.png",

    dpi=300
)

plt.close()

# =====
# 9. PROFILE SCORE VS FINAL EXAM
# =====

plt.figure(figsize=(8, 6))

plt.scatter(
    df["profile_score"],
    df["final_exam_grade"]
)

plt.title(
    "Σχέση Σύνθετου Δείκτη και Τελικής Επίδοσης"
)

plt.xlabel("Profile Score")

plt.ylabel("Final Exam Grade")

plt.tight_layout()

```

```

plt.savefig(
    OUTPUT_DIR /
    "09_profile_vs_exam_scatter.png",
    dpi=300
)

plt.close()

# =====
# FINAL REPORT
# =====

print("\n=====")
print("Visualization completed successfully")
print("=====")

print("Input file:", INPUT_FILE.name)

print("Output folder:", OUTPUT_DIR.name)

print("\nRole distribution check:")
print("Rows in profile:", len(df))
print("Unique user_id:", df["user_id"].nunique())
print("Sum of role counts:", int(role_counts.sum()))
print("Duplicate user_id:", int(df["user_id"].duplicated().sum()))

print("\nRole counts:")
print(role_counts)

print("\nSocial roles by stratum totals:")
print(role_distribution.sum(axis=1))

print("\nCreated files:")

for file in sorted(OUTPUT_DIR.iterdir()):
    print("-", file.name)

print("\nDone.")

```

A.18. 08c_04_3_2_weight_sensitivity_analysis.py

Χαρακτήρας: Έλεγχος script - Στρωματοποίηση - ανάλυση ευαισθησίας

Περιγραφή: Αναπαράγει το baseline της στρωματοποίησης και εξετάζει την ευστάθεια των αποτελεσμάτων μετά από σχετική μεταβολή $\pm 20\%$ των βαρών, με ανακανονικοποίηση των υπολοίπων βαρών.

Είσοδοι: 07_04_2_5_grouping_profile_final_features.csv · 08_04_2_6_student_profile_strata.csv

Έξοδοι: baseline_reproduction_diagnostics.csv · sensitivity_summary.csv · sensitivity_stratum_distributions.csv · sensitivity_stratum_changes.csv · sensitivity_scenario_weights.csv · sensitivity_report.txt (στον φάκελο 08c_04_3_2_weight_sensitivity_analysis/)

Πηγαίος κώδικας

```
# 08c_04_3_2_weight_sensitivity_analysis.py
# =====
# WEIGHT SENSITIVITY ANALYSIS
# Uses exactly the same profile-construction and stratification logic as:
# 08_04_2_6_student_profile_stratification.py
# =====

from pathlib import Path

import numpy as np
import pandas as pd

# =====
# PATHS AND SETTINGS
# =====

folder = Path(__file__).parent
script_stem = Path(__file__).stem

INPUT_FILE = folder / "07_04_2_5_grouping_profile_final_features.csv"
BASELINE_FILE = folder / "08_04_2_6_student_profile_strata.csv"

# All output files are stored in a folder with the same name as this script.
OUTPUT_DIR = folder / script_stem
OUTPUT_DIR.mkdir(parents=True, exist_ok=True)

DIAGNOSTICS_FILE = OUTPUT_DIR / "baseline_reproduction_diagnostics.csv"
SUMMARY_FILE = OUTPUT_DIR / "sensitivity_summary.csv"
DISTRIBUTIONS_FILE = OUTPUT_DIR / "sensitivity_stratum_distributions.csv"
CHANGES_FILE = OUTPUT_DIR / "sensitivity_stratum_changes.csv"
WEIGHTS_FILE = OUTPUT_DIR / "sensitivity_scenario_weights.csv"
REPORT_FILE = OUTPUT_DIR / "sensitivity_report.txt"

# Relative one-at-a-time change: -20% and +20%.
DELTA = 0.20

# =====
# FEATURE GROUPS AND BASELINE WEIGHTS
# Exact values from 08_04_2_6_student_profile_stratification.py
# =====

engagement_features = [
    "active_days",
    "avg_events_per_active_day",
    "forum_posts",
    "forum_views",
    "quiz_attempts",
    "completed_assignments",
]

performance_features = [
    "final_exam_grade",
    "avg_assignments_nonzero",
]

social_features = [
    "social_role_score",
]

frustration_features = [
    "frustration_signal_count",
```

```

        "frustration_episode_count",
    ]

    all_features = (
        engagement_features
        + performance_features
        + social_features
        + frustration_features
    )

    engagement_weights = {
        "active_days": 0.20,
        "avg_events_per_active_day": 0.15,
        "forum_posts": 0.20,
        "forum_views": 0.10,
        "quiz_attempts": 0.15,
        "completed_assignments": 0.20,
    }

    performance_weights = {
        "final_exam_grade": 0.60,
        "avg_assignments_nonzero": 0.40,
    }

    social_weights = {
        "social_role_score": 1.00,
    }

    frustration_weights = {
        "frustration_signal_count": 0.50,
        "frustration_episode_count": 0.50,
    }

    dimension_weights = {
        "engagement": 0.40,
        "performance": 0.25,
        "social": 0.25,
        "frustration": 0.10,
    }

    role_score_map = {
        "Ανεπαρκής Αλληλεπίδραση": 0,
        "Περιφερειακός Συμμετέχων": 1,
        "Δραστήριος Συμμετέχων": 2,
        "Γέφυρα": 3,
        "Ηγέτης": 4,
    }

    stratum_code_map = {
        "Πολύ Χαμηλό": 1,
        "Χαμηλό": 2,
        "Μεσαίο": 3,
        "Υψηλό": 4,
        "Πολύ Υψηλό": 5,
    }

    stratum_order = [
        "Πολύ Χαμηλό",
        "Χαμηλό",
        "Μεσαίο",
        "Υψηλό",
        "Πολύ Υψηλό",
    ]

    stratum_thresholds = np.array([-1.0, -0.3, 0.3, 1.0], dtype=float)

    # =====
    # HELPER FUNCTIONS
    # =====

    def check_weights_sum(weights_dict, name):
        total = sum(weights_dict.values())
        if round(total, 6) != 1.0:
            raise ValueError(
                f"Weights for {name} do not sum to 1. Current sum: {total}"
            )

    def assign_stratum(z):
        # Exact threshold logic from the original stratification script.
        if z < -1.0:
            return "Πολύ Χαμηλό"
        elif z < -0.3:
            return "Χαμηλό"
        elif z <= 0.3:
            return "Μεσαίο"

```

```

elif z <= 1.0:
    return "Υψηλό"
else:
    return "Πολύ Υψηλό"

def manual_zscore(series):
    # Exact population z-score logic (ddof=0) from the original script.
    mean_val = series.mean()
    std_val = series.std(ddof=0)
    if std_val == 0:
        return pd.Series(0.0, index=series.index)
    return (series - mean_val) / std_val

def weighted_sum(scaled_df, weights_dict):
    return sum(
        scaled_df[f"{feature}_z"] * weight
        for feature, weight in weights_dict.items()
    )

def renormalize_after_relative_change(weights_dict, target, factor):
    changed = weights_dict.copy()
    changed[target] = changed[target] * factor
    total = sum(changed.values())
    return {key: value / total for key, value in changed.items()}

def compute_profile(
    scaled_df,
    engagement_w,
    performance_w,
    social_w,
    frustration_w,
    dimension_w,
):
    """Reproduce the original dimension-score and profile-score calculations."""

    engagement_score = weighted_sum(scaled_df, engagement_w)
    performance_score = weighted_sum(scaled_df, performance_w)
    social_score = weighted_sum(scaled_df, social_w)
    frustration_score = weighted_sum(scaled_df, frustration_w)

    profile_score = (
        engagement_score * dimension_w["engagement"]
        + performance_score * dimension_w["performance"]
        + social_score * dimension_w["social"]
        + frustration_score * dimension_w["frustration"]
    )

    profile_percentile = profile_score.rank(pct=True)
    profile_score_z = manual_zscore(profile_score)
    profile_stratum = profile_score_z.apply(assign_stratum)
    profile_stratum_code = profile_stratum.map(stratum_code_map)

    return pd.DataFrame(
        {
            "engagement_score": engagement_score,
            "performance_score": performance_score,
            "social_score": social_score,
            "frustration_score": frustration_score,
            "profile_score": profile_score,
            "profile_percentile": profile_percentile,
            "profile_score_z": profile_score_z,
            "profile_stratum": profile_stratum,
            "profile_stratum_code": profile_stratum_code,
        }
    )

def nearest_threshold_distance(z_series):
    values = z_series.to_numpy(dtype=float)
    distances = np.min(
        np.abs(values[:, np.newaxis] - stratum_thresholds[np.newaxis, :]),
        axis=1,
    )
    return pd.Series(distances, index=z_series.index)

def spearman_correlation(series_a, series_b):
    # Spearman correlation without requiring scipy.
    return float(series_a.rank(method="average").corr(series_b.rank(method="average")))

def flatten_weights(level, group_name, scenario_name, weights_dict):
    rows = []

```

```

    for item, weight in weights_dict.items():
        rows.append(
            {
                "scenario": scenario_name,
                "level": level,
                "weight_group": group_name,
                "item": item,
                "weight": weight,
            }
        )
    return rows

# =====
# VALIDATE BASELINE WEIGHTS
# =====

check_weights_sum(engagement_weights, "engagement")
check_weights_sum(performance_weights, "performance")
check_weights_sum(social_weights, "social")
check_weights_sum(frustration_weights, "frustration")
check_weights_sum(dimension_weights, "dimensions")

# =====
# LOAD AND PREPARE DATA
# Exact preprocessing logic from the original script
# =====

if not INPUT_FILE.exists():
    raise FileNotFoundError(
        f"Input file not found: {INPUT_FILE}\n"
        "Place the script in the same folder as "
        "07_04_2_5_grouping_profile_final_features.csv."
    )

df = pd.read_csv(INPUT_FILE, encoding="utf-8-sig")

print("\n=====")
print("Weight Sensitivity Analysis")
print("=====")
print("Input file:", INPUT_FILE.name)
print("Students:", len(df))
print("Relative weight change: ±{:.0f}%".format(DELTA * 100))
print("Output folder:", OUTPUT_DIR.name)

required_columns = [
    "user_id",
    *engagement_features,
    *performance_features,
    "network_role_el",
    *frustration_features,
]

missing_columns = [col for col in required_columns if col not in df.columns]
if missing_columns:
    raise ValueError("Missing required columns: " + ", ".join(missing_columns))

# Exact social-role mapping logic.
df["social_role_score"] = df["network_role_el"].map(role_score_map)
unknown_roles = df[df["social_role_score"].isna()][ "network_role_el"].unique()

if len(unknown_roles) > 0:
    print("\nWARNING: Unknown social role labels found:")
    for role in unknown_roles:
        print("-", role)
    print("Unknown roles will be coded as 0.")
    df["social_role_score"] = df["social_role_score"].fillna(0)

# Exact numeric conversion and zero-imputation logic.
for col in all_features:
    df[col] = pd.to_numeric(df[col], errors="coerce")

missing_values = df[all_features].isna().sum()
if missing_values.sum() > 0:
    print("\nWARNING: Missing numeric values found.")
    print("They will be replaced with 0.\n")
    print(missing_values[missing_values > 0])
    df[all_features] = df[all_features].fillna(0)

# Exact z-score standardization and frustration reversal.
scaled_df = pd.DataFrame(index=df.index)
for col in all_features:
    scaled_df[f"{col}_z"] = manual_zscore(df[col])

for feature in frustration_features:
    scaled_df[f"{feature}_z"] = -1 * scaled_df[f"{feature}_z"]

```

```

# =====
# BASELINE REPRODUCTION
# =====

baseline = compute_profile(
    scaled_df=scaled_df,
    engagement_w=engagement_weights,
    performance_w=performance_weights,
    social_w=social_weights,
    frustration_w=frustration_weights,
    dimension_w=dimension_weights,
)

baseline["user_id"] = df["user_id"].values
baseline_distance = nearest_threshold_distance(baseline["profile_score_z"])

# Compare with the original output file, when available.
diagnostic_rows = [
    {
        "diagnostic": "input_rows",
        "value": len(df),
        "status": "INFO",
        "detail": "Number of records in the input file.",
    },
    {
        "diagnostic": "unknown_role_labels",
        "value": len(unknown_roles),
        "status": "OK" if len(unknown_roles) == 0 else "WARNING",
        "detail": "Unknown role labels are coded as 0, exactly as in the original script.",
    },
    {
        "diagnostic": "numeric_values_imputed_with_zero",
        "value": int(missing_values.sum()),
        "status": "OK" if missing_values.sum() == 0 else "WARNING",
        "detail": "Missing/non-numeric values are replaced with 0, exactly as in the original script.",
    },
]

baseline_reproduction_ok = True

if BASELINE_FILE.exists():
    original = pd.read_csv(BASELINE_FILE, encoding="utf-8-sig")

    if "user_id" not in original.columns:
        raise ValueError(
            f"The baseline file {BASELINE_FILE.name} does not contain user_id."
        )

    required_baseline_columns = [
        "profile_score",
        "profile_score_z",
        "profile_stratum",
        "engagement_score",
        "performance_score",
        "social_score",
        "frustration_score",
    ]
    absent = [col for col in required_baseline_columns if col not in original.columns]
    if absent:
        raise ValueError(
            f"The baseline file {BASELINE_FILE.name} is missing: "
            + ", ".join(absent)
        )

    if df["user_id"].duplicated().any() or original["user_id"].duplicated().any():
        raise ValueError("Duplicate user_id values found; baseline comparison is ambiguous.")

    comparison = baseline.merge(
        original[["user_id", *required_baseline_columns]],
        on="user_id",
        how="outer",
        suffixes=("_recomputed", "_original"),
        indicator=True,
    )

    unmatched = int((comparison["_merge"] != "both").sum())
    diagnostic_rows.append(
        {
            "diagnostic": "unmatched_user_ids",
            "value": unmatched,
            "status": "OK" if unmatched == 0 else "ERROR",
            "detail": "User IDs present in only one of the two files.",
        }
    )
)

```

```

numeric_compare_columns = [
    "profile_score",
    "profile_score_z",
    "engagement_score",
    "performance_score",
    "social_score",
    "frustration_score",
]

both = comparison[comparison["_merge"] == "both"].copy()
for col in numeric_compare_columns:
    difference = (
        pd.to_numeric(both[f"{col}_recomputed"], errors="coerce")
        - pd.to_numeric(both[f"{col}_original"], errors="coerce")
    ).abs()
    max_diff = float(difference.max()) if len(difference) else np.nan
    diagnostic_rows.append(
        {
            "diagnostic": f"max_abs_difference_{col}",
            "value": max_diff,
            "status": "OK" if pd.notna(max_diff) and max_diff <= 1e-10 else "ERROR",
            "detail": "Maximum absolute difference between recomputed and original values.",
        }
    )

stratum_mismatches = int(
    (
        both["profile_stratum_recomputed"].astype(str)
        != both["profile_stratum_original"].astype(str)
    ).sum()
)
diagnostic_rows.append(
    {
        "diagnostic": "profile_stratum_mismatches",
        "value": stratum_mismatches,
        "status": "OK" if stratum_mismatches == 0 else "ERROR",
        "detail": "Different stratum assignments between recomputed and original output.",
    }
)

baseline_reproduction_ok = (
    unmatched == 0
    and stratum_mismatches == 0
    and all(
        row["status"] != "ERROR"
        for row in diagnostic_rows
        if row["diagnostic"].startswith("max_abs_difference__")
    )
)
else:
    diagnostic_rows.append(
        {
            "diagnostic": "baseline_file_available",
            "value": 0,
            "status": "WARNING",
            "detail": (
                f"{BASELINE_FILE.name} was not found. The internal baseline was computed "
                "with the exact original logic, but external reproduction was not verified."
            ),
        }
    )

pd.DataFrame(diagnostic_rows).to_csv(
    DIAGNOSTICS_FILE,
    index=False,
    encoding="utf-8-sig",
    float_format="%.12f",
)

if BASELINE_FILE.exists() and not baseline_reproduction_ok:
    raise RuntimeError(
        "Baseline reproduction failed. Check "
        f"{DIAGNOSTICS_FILE.name} before interpreting sensitivity results."
    )

# =====
# SENSITIVITY SCENARIOS
# =====

summary_rows = []
distribution_rows = []
change_rows = []
weight_rows = []

baseline_codes = baseline["profile_stratum"].map(stratum_code_map)

```

```

def record_scenario(
    scenario_name,
    level,
    target,
    direction,
    engagement_w,
    performance_w,
    social_w,
    frustration_w,
    dimension_w,
):
    result = compute_profile(
        scaled_df=scaled_df,
        engagement_w=engagement_w,
        performance_w=performance_w,
        social_w=social_w,
        frustration_w=frustration_w,
        dimension_w=dimension_w,
    )

    scenario_codes = result["profile_stratum"].map(stratum_code_map)
    signed_shift = scenario_codes - baseline_codes
    absolute_shift = signed_shift.abs()
    changed = absolute_shift > 0
    changed_n = int(changed.sum())

    changed_within_005_n = int((changed & (baseline_distance <= 0.05)).sum())
    changed_within_010_n = int((changed & (baseline_distance <= 0.10)).sum())

    summary_rows.append(
        {
            "scenario": scenario_name,
            "level": level,
            "target": target,
            "direction": direction,
            "relative_change": 0.0 if level == "baseline" else DELTA,
            "spearman_rho": spearman_correlation(
                baseline["profile_score_z"], result["profile_score_z"]
            ),
            "same_stratum_n": int((absolute_shift == 0).sum()),
            "same_stratum_pct": float((absolute_shift == 0).mean() * 100),
            "adjacent_move_n": int((absolute_shift == 1).sum()),
            "adjacent_move_pct": float((absolute_shift == 1).mean() * 100),
            "move_ge_2_n": int((absolute_shift >= 2).sum()),
            "move_ge_2_pct": float((absolute_shift >= 2).mean() * 100),
            "max_stratum_shift": int(absolute_shift.max()),
            "mean_abs_profile_z_change": float(
                (result["profile_score_z"] - baseline["profile_score_z"]).abs().mean()
            ),
            "max_abs_profile_z_change": float(
                (result["profile_score_z"] - baseline["profile_score_z"]).abs().max()
            ),
            "changed_within_0_05_of_threshold_n": changed_within_005_n,
            "changed_within_0_05_of_threshold_pct_of_changed": (
                float(changed_within_005_n / changed_n * 100) if changed_n else 0.0
            ),
            "changed_within_0_10_of_threshold_n": changed_within_010_n,
            "changed_within_0_10_of_threshold_pct_of_changed": (
                float(changed_within_010_n / changed_n * 100) if changed_n else 0.0
            ),
        }
    )

    counts = result["profile_stratum"].value_counts().reindex(stratum_order, fill_value=0)
    for stratum in stratum_order:
        distribution_rows.append(
            {
                "scenario": scenario_name,
                "stratum": stratum,
                "count": int(counts.loc[stratum]),
                "percentage": float(counts.loc[stratum] / len(df) * 100),
            }
        )

    for idx in df.index[changed]:
        change_rows.append(
            {
                "scenario": scenario_name,
                "user_id": df.loc[idx, "user_id"],
                "baseline_profile_score_z": float(baseline.loc[idx, "profile_score_z"]),
                "scenario_profile_score_z": float(result.loc[idx, "profile_score_z"]),
                "distance_to_nearest_baseline_threshold": float(baseline_distance.loc[idx]),
                "baseline_stratum": baseline.loc[idx, "profile_stratum"],
                "scenario_stratum": result.loc[idx, "profile_stratum"],
                "signed_stratum_shift": int(signed_shift.loc[idx]),
                "abs_stratum_shift": int(absolute_shift.loc[idx]),
            }
        )

```

```

    }
)

weight_rows.extend(flatten_weights(level, "engagement", scenario_name, engagement_w))
weight_rows.extend(flatten_weights(level, "performance", scenario_name, performance_w))
weight_rows.extend(flatten_weights(level, "social", scenario_name, social_w))
weight_rows.extend(flatten_weights(level, "frustration", scenario_name, frustration_w))
weight_rows.extend(flatten_weights(level, "dimensions", scenario_name, dimension_w))

# Baseline scenario.
record_scenario(
    scenario_name="baseline",
    level="baseline",
    target="",
    direction="",
    engagement_w=engagement_weights,
    performance_w=performance_weights,
    social_w=social_weights,
    frustration_w=frustration_weights,
    dimension_w=dimension_weights,
)

# One-at-a-time changes of weights within dimensions.
component_groups = {
    "engagement": engagement_weights,
    "performance": performance_weights,
    "frustration": frustration_weights,
}

for group_name, base_weights in component_groups.items():
    for feature in base_weights:
        for direction, factor in (
            ("minus_20_percent", 1.0 - DELTA),
            ("plus_20_percent", 1.0 + DELTA),
        ):
            modified = renormalize_after_relative_change(base_weights, feature, factor)

            current_engagement = engagement_weights
            current_performance = performance_weights
            current_frustration = frustration_weights

            if group_name == "engagement":
                current_engagement = modified
            elif group_name == "performance":
                current_performance = modified
            elif group_name == "frustration":
                current_frustration = modified

            record_scenario(
                scenario_name=f"component_{group_name}_{feature}_{direction}",
                level="component",
                target=f"{group_name}.{feature}",
                direction=direction,
                engagement_w=current_engagement,
                performance_w=current_performance,
                social_w=social_weights,
                frustration_w=current_frustration,
                dimension_w=dimension_weights,
            )

# One-at-a-time changes of the four dimension weights.
for dimension in dimension_weights:
    for direction, factor in (
        ("minus_20_percent", 1.0 - DELTA),
        ("plus_20_percent", 1.0 + DELTA),
    ):
        modified_dimensions = renormalize_after_relative_change(
            dimension_weights, dimension, factor
        )

        record_scenario(
            scenario_name=f"dimension_{dimension}_{direction}",
            level="dimension",
            target=dimension,
            direction=direction,
            engagement_w=engagement_weights,
            performance_w=performance_weights,
            social_w=social_weights,
            frustration_w=frustration_weights,
            dimension_w=modified_dimensions,
        )

# =====
# SAVE OUTPUT FILES
# =====

```

```

summary_df = pd.DataFrame(summary_rows)
distributions_df = pd.DataFrame(distribution_rows)
changes_df = pd.DataFrame(change_rows)
weights_df = pd.DataFrame(weight_rows)

summary_df.to_csv(
    SUMMARY_FILE,
    index=False,
    encoding="utf-8-sig",
    float_format="%.6f",
)

distributions_df.to_csv(
    DISTRIBUTIONS_FILE,
    index=False,
    encoding="utf-8-sig",
    float_format="%.6f",
)

changes_df.to_csv(
    CHANGES_FILE,
    index=False,
    encoding="utf-8-sig",
    float_format="%.6f",
)

weights_df.to_csv(
    WEIGHTS_FILE,
    index=False,
    encoding="utf-8-sig",
    float_format="%.8f",
)

# =====
# TEXT REPORT
# =====

nonbaseline = summary_df[summary_df["scenario"] != "baseline"].copy()
worst_same = nonbaseline.sort_values(
    ["same_stratum_pct", "spearman_rho"], ascending=[True, True]
).iloc[0]
worst_rho = nonbaseline.sort_values("spearman_rho", ascending=True).iloc[0]
worst_shift = nonbaseline.sort_values(
    ["move_ge_2_pct", "adjacent_move_pct"], ascending=[False, False]
).iloc[0]

component_results = nonbaseline[nonbaseline["level"] == "component"]
dimension_results = nonbaseline[nonbaseline["level"] == "dimension"]

report_lines = [
    "WEIGHT SENSITIVITY ANALYSIS REPORT",
    "=====",
    "",
    f"Input file: {INPUT_FILE.name}",
    f"Original stratification file: {BASELINE_FILE.name}",
    f"Students: {len(df)}",
    f"Relative one-at-a-time change: ±{DELTA * 100:.0f}%",
    f"Alternative scenarios: {len(nonbaseline)}",
    f"- Component-weight scenarios: {len(component_results)}",
    f"- Dimension-weight scenarios: {len(dimension_results)}",
    "",
    "Method:",
    "- One weight is changed at a time.",
    "- The affected set of weights is renormalized to sum to 1.",
    "- Feature z-scores, frustration reversal, dimension scores, final profile score,",
    "  final profile z-score and stratum assignment follow exactly the logic of",
    "  08_04_2_6_student_profile_stratification.py.",
    "- Stratification thresholds remain fixed at -1.0, -0.3, 0.3 and 1.0.",
    "",
    "Baseline reproduction:",
    (
        "- Successfully verified against the original stratification output."
        if BASELINE_FILE.exists() and baseline_reproduction_ok
        else "- External baseline file was not available; see diagnostics."
    ),
    "",
    "Worst scenario by same-stratum retention:",
    f"- Scenario: {worst_same['scenario']}",
    f"- Same stratum: {worst_same['same_stratum_n']} students "
    f"f'({worst_same['same_stratum_pct']:.2f}%)",
    f"- Adjacent-stratum movement: {worst_same['adjacent_move_n']} students "
    f"f'({worst_same['adjacent_move_pct']:.2f}%)",
    f"- Movement by two or more strata: {worst_same['move_ge_2_n']} students "
    f"f'({worst_same['move_ge_2_pct']:.2f}%)",
    f"- Spearman rho: {worst_same['spearman_rho']:.6f}",

```

```

f"- Changed students within 0.10 of a baseline threshold: "
f"{worst_same['changed_within_0_10_of_threshold_n']} "
f"({worst_same['changed_within_0_10_of_threshold_pct_of_changed']:.2f}% of changed)",
"",
"Lowest rank correlation:",
f"- Scenario: {worst_rho['scenario']}",
f"- Spearman rho: {worst_rho['spearman_rho']:.6f}",
f"- Same stratum: {worst_rho['same_stratum_pct']:.2f}%",
"",
"Largest non-adjacent movement:",
f"- Scenario: {worst_shift['scenario']}",
f"- Movement by two or more strata: {worst_shift['move_ge_2_n']} students "
f"({worst_shift['move_ge_2_pct']:.2f}%",
f"- Maximum shift: {int(worst_shift['max_stratum_shift'])} strata",
"",
"Output files:",
f"- {DIAGNOSTICS_FILE.name}",
f"- {SUMMARY_FILE.name}",
f"- {DISTRIBUTIONS_FILE.name}",
f"- {CHANGES_FILE.name}",
f"- {WEIGHTS_FILE.name}",
]

```

```
REPORT_FILE.write_text("\n".join(report_lines), encoding="utf-8")
```

```

# =====
# FINAL CONSOLE OUTPUT
# =====

```

```

print("\nBaseline reproduction:", "OK" if baseline_reproduction_ok else "NOT VERIFIED")
print("Alternative scenarios:", len(nonbaseline))
print("\nWorst scenario by same-stratum retention:")
print("Scenario:", worst_same["scenario"])
print("Same stratum: {:.2f}%".format(worst_same["same_stratum_pct"]))
print("Adjacent movement: {:.2f}%".format(worst_same["adjacent_move_pct"]))
print("Movement >= 2 strata: {:.2f}%".format(worst_same["move_ge_2_pct"]))
print("Spearman rho: {:.6f}".format(worst_same["spearman_rho"]))

```

```

print("\nOutput folder:", OUTPUT_DIR)
print("\n=====")
print("Sensitivity analysis completed successfully")
print("=====")

```

A.19. 09_HOMOPHILY_analysis.py

Χαρακτήρας: Κύριο script - Ανάλυση ομοφιλίας

Περιγραφή: Υπολογίζει για κατηγορικά χαρακτηριστικά το ποσοστό ακμών μεταξύ όμοιων κόμβων και για αριθμητικά χαρακτηριστικά τη μέση απόλυτη διαφορά των συνδεδεμένων φοιτητών. Η στατιστική σημαντικότητα εκτιμάται με permutation tests 1.000 επαναλήψεων, διατηρώντας σταθερή τη δομή του δικτύου.

Είσοδοι: 04c_4_2_3_student_student_projection_filtered_p75.csv · 04c_4_2_3_student_student_nodes_metrics.csv · 08_04_2_6_for_grouping.csv

Έξοδοι: 09_homophily_outputs/homophily_categorical_report.csv · 09_homophily_outputs/homophily_numeric_report.csv · 09_homophily_outputs/homophily_edge_details.csv · 09_homophily_outputs/homophily_run_report.txt

Πηγαίος κώδικας

```
# -*- coding: utf-8 -*-
"""
09_HOMOPHILY_analysis.py

Υπολογισμός ομοφιλίας στο προβαλλόμενο student-student δίκτυο.

Το script τρέχει απλά με:

    python 09_HOMOPHILY_analysis.py

Αναμενόμενη θέση:
Το script πρέπει να βρίσκεται στον ίδιο φάκελο με τα παρακάτω αρχεία:

    04c_4_2_3_student_student_projection_filtered_p75.csv
    04c_4_2_3_student_student_nodes_metrics.csv
    08_04_2_6_for_grouping.csv

Αρχεία εισόδου:
- EDGES_FILE: ακμές του φιλτραρισμένου student-student δικτύου
- NODES_FILE: κόμβοι/μετρικές του student-student δικτύου, χρησιμοποιείται για έλεγχο
- PROFILES_FILE: τελικό προφίλ για ομαδοποίηση, περιέχει strata και χαρακτηριστικά φοιτητών

Αρχεία εξόδου:
- 09_homophily_outputs/homophily_categorical_report.csv
- 09_homophily_outputs/homophily_numeric_report.csv
- 09_homophily_outputs/homophily_edge_details.csv
- 09_homophily_outputs/homophily_run_report.txt
"""

from pathlib import Path

import numpy as np
import pandas as pd

# =====
# 1. ΕΝΣΩΜΑΤΩΜΕΝΕΣ ΠΥΘΜΙΣΕΙΣ ΑΡΧΕΙΩΝ
# =====

BASE_DIR = Path(__file__).resolve().parent

# Ακμές του προβαλλόμενου και φιλτραρισμένου student-student δικτύου.
EDGES_FILE = BASE_DIR / "04c_4_2_3_student_student_projection_filtered_p75.csv"

# Κόμβοι του student-student δικτύου με SNA μετρικές.
# Χρησιμοποιείται για έλεγχο συμβατότητας κόμβων.
NODES_FILE = BASE_DIR / "04c_4_2_3_student_student_nodes_metrics.csv"

# Τελικό αρχείο για ομαδοποίηση μετά τη στρωματοποίηση.
PROFILES_FILE = BASE_DIR / "08_04_2_6_for_grouping.csv"

OUTPUT_DIR = BASE_DIR / "09_homophily_outputs"

N_PERM = 1000
SEED = 42

# =====
# 2. ΒΟΗΘΗΤΙΚΕΣ ΣΥΝΑΡΤΗΣΕΙΣ
# =====
```

```

def normalize_user_id(value):
    """
    Μετατρέπει τα user IDs σε συνεπή μορφή.

    Στόχος:
    user100 -> 100
    User100 -> 100
    100 -> 100
    100.0 -> 100
    """
    if pd.isna(value):
        return np.nan

    value = str(value).strip()

    # Αφαίρεση πιθανού prefix user/User/USER
    value = value.replace("User", "").replace("USER", "").replace("user", "")

    # Αν έχει διαβαστεί ως float, π.χ. 100.0, το κάνουμε 100
    try:
        number = float(value)
        if number.is_integer():
            return str(int(number))
    except ValueError:
        pass

    return value

def read_table(path):
    """Διαβάζει CSV ή Excel αρχείο εισόδου."""
    path = Path(path)

    if not path.exists():
        raise FileNotFoundError(f"Δεν βρέθηκε το αρχείο: {path}")

    if path.suffix.lower() in [".xlsx", ".xls"]:
        return pd.read_excel(path)

    return pd.read_csv(path, encoding="utf-8-sig")

def find_first_existing_column(df, candidates):
    """Βρίσκει την πρώτη στήλη που υπάρχει από μια λίστα πιθανών ονομάτων."""
    for col in candidates:
        if col in df.columns:
            return col
    return None

def load_edges(edge_path):
    """Φορτώνει και κανονικοποιεί τις ακμές student-student."""
    edges = read_table(edge_path)

    source_col = find_first_existing_column(edges, ["source", "Source", "SOURCE"])
    target_col = find_first_existing_column(edges, ["target", "Target", "TARGET"])
    weight_col = find_first_existing_column(
        edges,
        ["Weight", "weight", "W", "w", "edge_weight"]
    )

    if source_col is None or target_col is None:
        raise ValueError(
            "Το αρχείο ακμών πρέπει να έχει στήλες Source/Target ή source/target."
        )

    if weight_col is None:
        edges["Weight"] = 1.0
        weight_col = "Weight"

    edges = edges[[source_col, target_col, weight_col]].copy()
    edges.columns = ["source", "target", "Weight"]

    edges["source"] = edges["source"].apply(normalize_user_id)
    edges["target"] = edges["target"].apply(normalize_user_id)
    edges["Weight"] = pd.to_numeric(edges["Weight"], errors="coerce").fillna(1.0)

    edges = edges.dropna(subset=["source", "target"])
    edges = edges[edges["source"] != edges["target"]].copy()

    # Το student-student δίκτυο είναι μη κατευθυνόμενο.
    # Άρα ενοποιούμε πιθανές διπλές ακμές A-B και B-A.
    edges["u"] = edges[["source", "target"]].min(axis=1)
    edges["v"] = edges[["source", "target"]].max(axis=1)

    edges = (
        edges.groupby(["u", "v"], as_index=False)["Weight"]

```

```

        .sum()
        .rename(columns={"u": "source", "v": "target"})
    )

    return edges

def load_nodes(node_path):
    """Φορτώνει τους κόμβους student-student δικτύου."""
    nodes = read_table(node_path)

    node_col = find_first_existing_column(
        nodes,
        ["user_id", "Id", "ID", "id", "Label", "label", "node", "Node"]
    )

    if node_col is None:
        raise ValueError(
            "Το αρχείο κόμβων δεν έχει αναγνωρίσιμη στήλη ID. "
            "Πιθανές στήλες: user_id, Id, ID, Label."
        )

    nodes = nodes.copy()
    nodes["user_id"] = nodes[node_col].apply(normalize_user_id)

    return nodes

def load_profiles(profile_path):
    """Φορτώνει το τελικό προφίλ φοιτητών."""
    profiles = read_table(profile_path)

    user_col = find_first_existing_column(
        profiles,
        ["user_id", "User ID", "User full name", "user", "id", "ID"]
    )

    if user_col is None:
        raise ValueError(
            "Το αρχείο προφίλ πρέπει να περιέχει στήλη user_id ή αντίστοιχη στήλη."
        )

    profiles = profiles.copy()
    profiles["user_id"] = profiles[user_col].apply(normalize_user_id)

    return profiles

def attach_attributes_to_edges(edges, profiles, attributes):
    """Προσθέτει στις ακμές τα χαρακτηριστικά source και target."""
    missing = [c for c in attributes if c not in profiles.columns]

    if missing:
        raise ValueError("Λείπουν από το προφίλ οι στήλες: " + ", ".join(missing))

    node_attrs = profiles[["user_id"] + attributes].drop_duplicates("user_id")

    edge_details = edges.merge(
        node_attrs,
        left_on="source",
        right_on="user_id",
        how="left"
    ).drop(columns=["user_id"])

    edge_details = edge_details.merge(
        node_attrs,
        left_on="target",
        right_on="user_id",
        how="left",
        suffixes=("_source", "_target")
    ).drop(columns=["user_id"])

    return edge_details

# =====
# 3. ΟΜΟΦΙΛΙΑ ΚΑΤΗΓΟΡΙΚΩΝ ΧΑΡΑΚΤΗΡΙΣΤΙΚΩΝ
# =====

def categorical_homophily(edge_details, attr):
    """
    Υπολογίζει ομοφιλία για κατηγορικό χαρακτηριστικό.

    homophily_ratio:
    ποσοστό ακμών που ενώνουν φοιτητές με ίδια τιμή στο χαρακτηριστικό.
    """
    source_attr = f"{attr}_source"

```

```

target_attr = f"{attr}_target"

valid = edge_details.dropna(subset=[source_attr, target_attr]).copy()

if valid.empty:
    return {
        "attribute": attr,
        "valid_edges": 0,
        "same_edges": np.nan,
        "homophily_ratio": np.nan,
        "weighted_homophily_ratio": np.nan
    }

same = valid[source_attr].astype(str) == valid[target_attr].astype(str)

homophily_ratio = same.mean()

if valid["Weight"].sum() > 0:
    weighted_homophily_ratio = valid.loc[same, "Weight"].sum() / valid["Weight"].sum()
else:
    weighted_homophily_ratio = np.nan

return {
    "attribute": attr,
    "valid_edges": len(valid),
    "same_edges": int(same.sum()),
    "homophily_ratio": homophily_ratio,
    "weighted_homophily_ratio": weighted_homophily_ratio
}

def permutation_test_categorical(edges, profiles, attr, observed, n_perm=N_PERM, seed=SEED):
    """
    Permutation test για κατηγορικά χαρακτηριστικά.

    Κρατάμε σταθερό το δίκτυο και ανακατεύουμε τις τιμές του χαρακτηριστικού
    στους κόμβους.

    Για κατηγορικά χαρακτηριστικά:
    p-value = πιθανότητα η τυχαία ομοφιλία να είναι >= της παρατηρούμενης.
    """
    rng = np.random.default_rng(seed)

    nodes = pd.Index(pd.unique(edges[["source", "target"]].values.ravel()))
    attr_map = profiles.set_index("user_id")[attr].reindex(nodes)

    valid_nodes = attr_map.dropna().index
    attr_values = attr_map.dropna().values

    valid_edges = edges[
        edges["source"].isin(valid_nodes) & edges["target"].isin(valid_nodes)
    ].copy()

    if valid_edges.empty or pd.isna(observed):
        return np.nan, np.nan, np.nan

    null_values = []

    for _ in range(n_perm):
        shuffled_values = rng.permutation(attr_values)
        shuffled_map = dict(zip(valid_nodes, shuffled_values))

        source_values = valid_edges["source"].map(shuffled_map).astype(str)
        target_values = valid_edges["target"].map(shuffled_map).astype(str)

        null_values.append((source_values == target_values).mean())

    null_values = np.array(null_values)

    p_value = (np.sum(null_values >= observed) + 1) / (n_perm + 1)

    return null_values.mean(), null_values.std(ddof=1), p_value

# =====
# 4. ΟΜΟΦΙΛΙΑ ΑΡΙΘΜΗΤΙΚΩΝ ΧΑΡΑΚΤΗΡΙΣΤΙΚΩΝ
# =====

def numeric_homophily(edge_details, attr):
    """
    Υπολογίζει ομοφιλία για αριθμητικό χαρακτηριστικό.

    mean_abs_difference:
    μέση απόλυτη διαφορά μεταξύ συνδεδεμένων φοιτητών.

    Όσο μικρότερη είναι η διαφορά, τόσο μεγαλύτερη είναι η ομοιότητα.
    """

```

```

source_attr = f"{attr}_source"
target_attr = f"{attr}_target"

valid = edge_details.dropna(subset=[source_attr, target_attr]).copy()

valid[source_attr] = pd.to_numeric(valid[source_attr], errors="coerce")
valid[target_attr] = pd.to_numeric(valid[target_attr], errors="coerce")

valid = valid.dropna(subset=[source_attr, target_attr])

if valid.empty:
    return {
        "attribute": attr,
        "valid_edges": 0,
        "mean_abs_difference": np.nan,
        "weighted_mean_abs_difference": np.nan
    }

diff = (valid[source_attr] - valid[target_attr]).abs()

if valid["Weight"].sum() > 0:
    weighted_mean_abs_difference = np.average(diff, weights=valid["Weight"])
else:
    weighted_mean_abs_difference = np.nan

return {
    "attribute": attr,
    "valid_edges": len(valid),
    "mean_abs_difference": diff.mean(),
    "weighted_mean_abs_difference": weighted_mean_abs_difference
}

def permutation_test_numeric(edges, profiles, attr, observed, n_perm=N_PERM, seed=SEED):
    """
    Permutation test για αριθμητικά χαρακτηριστικά.

    Κρατάμε σταθερό το δίκτυο και ανακατεύουμε τις αριθμητικές τιμές στους κόμβους.

    Για αριθμητικά χαρακτηριστικά:
    p-value = πιθανότητα η τυχαία μέση απόσταση να είναι <= της παρατηρούμενης.

    Αυτό επειδή μικρότερη απόσταση σημαίνει μεγαλύτερη ομοφιλία.
    """
    rng = np.random.default_rng(seed)

    nodes = pd.Index(pd.unique(edges[["source", "target"]].values.ravel()))

    attr_map = pd.to_numeric(
        profiles.set_index("user_id")[attr].reindex(nodes),
        errors="coerce"
    )

    valid_nodes = attr_map.dropna().index
    attr_values = attr_map.dropna().values

    valid_edges = edges[
        edges["source"].isin(valid_nodes) & edges["target"].isin(valid_nodes)
    ].copy()

    if valid_edges.empty or pd.isna(observed):
        return np.nan, np.nan, np.nan

    null_values = []

    for _ in range(n_perm):
        shuffled_values = rng.permutation(attr_values)
        shuffled_map = dict(zip(valid_nodes, shuffled_values))

        source_values = valid_edges["source"].map(shuffled_map).astype(float)
        target_values = valid_edges["target"].map(shuffled_map).astype(float)

        null_values.append((source_values - target_values).abs().mean())

    null_values = np.array(null_values)

    p_value = (np.sum(null_values <= observed) + 1) / (n_perm + 1)

    return null_values.mean(), null_values.std(ddof=1), p_value

# =====
# 5. ΚΥΡΙΑ ΔΙΑΔΙΚΑΣΙΑ
# =====

def main():
    OUTPUT_DIR.mkdir(parents=True, exist_ok=True)

```

```

print("Ξεκινά ο υπολογισμός ομοφιλίας.")
print()
print("Αρχεία εισόδου:")
print(f"Ακμές: {EDGES_FILE.name}")
print(f"Κόμβοι: {NODES_FILE.name}")
print(f"Προφίλ: {PROFILES_FILE.name}")
print()

edges = load_edges(EDGES_FILE)
nodes = load_nodes(NODES_FILE)
profiles = load_profiles(PROFILES_FILE)

# Χαρακτηριστικά για ομοφιλία.
# Το script κρατά μόνο όσα υπάρχουν πραγματικά στο αρχείο προφίλ.
categorical_attrs_requested = [
    "profile_stratum",
    "network_role_el",
    "network_role",
    "stratum"
]

numeric_attrs_requested = [
    "profile_score",
    "profile_score_z",
    "social_role_score",
    "performance_score",
    "engagement_score",
    "frustration_score",
    "avg_assignments_nonzero",
    "completed_assignments",
    "final_exam_grade",
    "active_days",
    "avg_events_per_active_day",
    "forum_posts",
    "forum_views",
    "quiz_attempts",
    "frustration_signal_count",
    "frustration_episode_count"
]

categorical_attrs = [
    col for col in categorical_attrs_requested
    if col in profiles.columns
]

numeric_attrs = [
    col for col in numeric_attrs_requested
    if col in profiles.columns
]

if not categorical_attrs and not numeric_attrs:
    raise ValueError(
        "Δεν βρέθηκαν κατάλληλες στήλες ομοφιλίας στο αρχείο προφίλ."
    )

all_attrs = categorical_attrs + numeric_attrs
edge_details = attach_attributes_to_edges(edges, profiles, all_attrs)

# Έλεγχος συμβατότητας κόμβων
edge_nodes = set(pd.unique(edges[["source", "target"]].values.ravel()))
node_file_nodes = set(nodes["user_id"].dropna())
profile_nodes = set(profiles["user_id"].dropna())

nodes_in_edges_not_in_nodes_file = sorted(edge_nodes - node_file_nodes)
nodes_in_edges_not_in_profile = sorted(edge_nodes - profile_nodes)

# -----
# Κατηγορική ομοφιλία
# -----

categorical_rows = []

for attr in categorical_attrs:
    observed = categorical_homophily(edge_details, attr)

    null_mean, null_std, p_value = permutation_test_categorical(
        edges=edges,
        profiles=profiles,
        attr=attr,
        observed=observed["homophily_ratio"],
        n_perm=N_PERM,
        seed=SEED
    )

    observed["null_mean"] = null_mean
    observed["null_std"] = null_std

```

```

observed["p_value"] = p_value

if (
    pd.notna(p_value)
    and p_value < 0.05
    and observed["homophily_ratio"] > null_mean
):
    observed["interpretation"] = "ένδειξη ομοφιλίας"
else:
    observed["interpretation"] = "μη ισχυρή ένδειξη ομοφιλίας"

categorical_rows.append(observed)

edge_details[f"{attr}_same"] = (
    edge_details[f"{attr}_source"].astype(str)
    == edge_details[f"{attr}_target"].astype(str)
)

categorical_report = pd.DataFrame(categorical_rows)

# -----
# Αριθμητική ομοφιλία
# -----

numeric_rows = []

for attr in numeric_attrs:
    observed = numeric_homophily(edge_details, attr)

    null_mean, null_std, p_value = permutation_test_numeric(
        edges=edges,
        profiles=profiles,
        attr=attr,
        observed=observed["mean_abs_difference"],
        n_perm=N_PERM,
        seed=SEED
    )

    observed["null_mean_abs_difference"] = null_mean
    observed["null_std_abs_difference"] = null_std
    observed["p_value"] = p_value

    if (
        pd.notna(p_value)
        and p_value < 0.05
        and observed["mean_abs_difference"] < null_mean
    ):
        observed["interpretation"] = "ένδειξη ομοφιλίας"
    else:
        observed["interpretation"] = "μη ισχυρή ένδειξη ομοφιλίας"

    numeric_rows.append(observed)

    source_values = pd.to_numeric(
        edge_details[f"{attr}_source"],
        errors="coerce"
    )
    target_values = pd.to_numeric(
        edge_details[f"{attr}_target"],
        errors="coerce"
    )

    edge_details[f"{attr}_abs_diff"] = (source_values - target_values).abs()

numeric_report = pd.DataFrame(numeric_rows)

# =====
# 6. APXEIA ΕΞΟΔΟΥ - MONO CSV KAI TXT
# =====

categorical_path = OUTPUT_DIR / "homophily_categorical_report.csv"
numeric_path = OUTPUT_DIR / "homophily_numeric_report.csv"
details_path = OUTPUT_DIR / "homophily_edge_details.csv"
run_report_path = OUTPUT_DIR / "homophily_run_report.txt"

categorical_report.to_csv(
    categorical_path,
    index=False,
    encoding="utf-8-sig"
)

numeric_report.to_csv(
    numeric_path,
    index=False,
    encoding="utf-8-sig"
)

```

```

edge_details.to_csv(
    details_path,
    index=False,
    encoding="utf-8-sig"
)

# Αναφορά εκτέλεσης
report_lines = []

report_lines.append("ΑΝΑΦΟΡΑ ΥΠΟΛΟΓΙΣΜΟΥ ΟΜΟΦΙΛΙΑΣ")
report_lines.append("=" * 60)
report_lines.append("")
report_lines.append(f"Αρχείο ακμών: {EDGES_FILE.name}")
report_lines.append(f"Αρχείο κόμβων: {NODES_FILE.name}")
report_lines.append(f"Αρχείο προφίλ: {PROFILES_FILE.name}")
report_lines.append("")
report_lines.append(f"Αριθμός ακμών: {len(edges)}")
report_lines.append(f"Αριθμός κόμβων στο αρχείο κόμβων: {len(nodes)}")
report_lines.append(f"Αριθμός φοιτητών στο αρχείο προφίλ: {len(profiles)}")
report_lines.append(f"Μοναδικοί κόμβοι που εμφανίζονται στις ακμές: {len(edge_nodes)}")
report_lines.append("")
report_lines.append(f"Κατηγορικά χαρακτηριστικά που ελέγχθηκαν: {categorical_attrs}")
report_lines.append(f"Αριθμητικά χαρακτηριστικά που ελέγχθηκαν: {numeric_attrs}")
report_lines.append("")
report_lines.append(
    f"Κόμβοι σε ακμές που δεν βρέθηκαν στο αρχείο κόμβων: "
    f"{len(nodes_in_edges_not_in_nodes_file)}"
)
report_lines.append(
    f"Κόμβοι σε ακμές που δεν βρέθηκαν στο προφίλ: "
    f"{len(nodes_in_edges_not_in_profile)}"
)
report_lines.append("")
report_lines.append("Ερμηνεία:")
report_lines.append(
    "- Για κατηγορικά χαρακτηριστικά: "
    "homophily_ratio > null_mean και p_value < 0.05 "
    "σημαίνει ένδειξη ομοφιλίας."
)
report_lines.append(
    "- Για αριθμητικά χαρακτηριστικά: "
    "mean_abs_difference < null_mean_abs_difference και p_value < 0.05 "
    "σημαίνει ένδειξη ομοφιλίας."
)
report_lines.append("")
report_lines.append("Αρχεία εξόδου:")
report_lines.append(f"- {categorical_path.name}")
report_lines.append(f"- {numeric_path.name}")
report_lines.append(f"- {details_path.name}")
report_lines.append(f"- {run_report_path.name}")

run_report_path.write_text(
    "\n".join(report_lines),
    encoding="utf-8"
)

print("Ολοκληρώθηκε ο υπολογισμός ομοφιλίας.")
print()
print(f"Ακμές δικτύου: {len(edges)}")
print(f"Κόμβοι στις ακμές: {len(edge_nodes)}")
print(f"Κατηγορικά χαρακτηριστικά: {categorical_attrs}")
print(f"Αριθμητικά χαρακτηριστικά: {numeric_attrs}")
print()
print("Αρχεία εξόδου:")
print(f"- {categorical_path}")
print(f"- {numeric_path}")
print(f"- {details_path}")
print(f"- {run_report_path}")

if __name__ == "__main__":
    main()

```

A.20. 10_05_3_grouping_ALGORITHM_v2.py

Χαρακτήρας: Κύριο script - Σύσταση ομάδων

Περιγραφή: Υλοποιεί τον προσαρμοστικό role-aware και homophily-aware greedy αλγόριθμο. Ελέγχει και κανονικοποιεί τις εισόδους, διαβάζει τις ενεργές διαστάσεις ομοφιλίας, κατανέμει διαδοχικά τους πέντε κοινωνικούς ρόλους και παράγει αναλυτικές αναφορές σύνθεσης, ισορροπίας και ποιότητας για κάθε μέγεθος ομάδας.

Είσοδοι: 08_04_2_6_for_grouping.csv · 09_homophily_outputs/homophily_categorical_report.csv · 09_homophily_outputs/homophily_numeric_report.csv

Έξοδοι: 10_group_recommendation_outputs/10_group_recommendation_size_<n>_assignments.csv · ..._group_summary.csv · ..._balance_report.csv · ..._strata_deviation.csv · ..._group_quality.csv · ..._overall_quality.csv · ..._active_homophily_constraints.csv · ..._validation_report.csv · ..._run_report.txt

Πηγαίος κώδικας

```
# -*- coding: utf-8 -*-
"""
10_05_3_grouping_ALGORITHM_v2.py

Προσαρμοστικός role-aware και homophily-aware αλγόριθμος σύστασης ομάδων.

Έκδοση v2: διατηρεί τον κανόνα μέγιστης κάλυψης ομάδων και προσθέτει
ρητό validation/logging για μη αναγνωρισμένους ρόλους, μη αναγνωρισμένα
strata και ελλυπείς ή μη αριθμητικές τιμές.

Η έκδοση αυτή ευθυγραμμίζεται με το Κεφάλαιο 4 και την ανάλυση ομοφιλίας:
- Χρησιμοποιεί τους 5 τελικούς κοινωνικούς ρόλους:
  Ηγέτης, Γέφυρα, Δραστήριος Συμμετέχων,
  Περιφερειακός Συμμετέχων, Ανεπαρκής Αλληλεπίδραση.
- Χρησιμοποιεί το τελικό αρχείο προφίλ για ομαδοποίηση.
- Διαβάζει τα αποτελέσματα ομοφιλίας, όταν υπάρχουν.
- Ενεργοποιεί ισχυρότερες ποινές ομοιογένειας μόνο για χαρακτηριστικά
  στα οποία έχει εντοπιστεί στατιστικά σημαντική ομοφιλία.
- Εξάγει μόνο CSV και TXT.
- Καταγράφει ρητά όλα τα fallbacks/imputations στο run report.

Ενδεικτική εκτέλεση:
    py 10_05_3_grouping_ALGORITHM_v2.py --group-size 5
    py 10_05_3_grouping_ALGORITHM_v2.py --all-sizes
"""

from pathlib import Path
import argparse
import math
import time

import numpy as np
import pandas as pd

# =====
# CONFIG
# =====

BASE_DIR = Path(__file__).resolve().parent

# Το τελικό αρχείο του pipeline για τη σύσταση ομάδων.
# Αν στο δικό σου pipeline έχει ακριβώς άλλο όνομα, άλλαξε μόνο αυτή τη γραμμή.
INPUT_FILE = BASE_DIR / "08_04_2_6_for_grouping.csv"

# Αποτελέσματα του προηγούμενου σταδίου ομοφιλίας.
HOMOPHILY_DIR = BASE_DIR / "09_homophily_outputs"
HOMOPHILY_CATEGORICAL_FILE = HOMOPHILY_DIR / "homophily_categorical_report.csv"
HOMOPHILY_NUMERIC_FILE = HOMOPHILY_DIR / "homophily_numeric_report.csv"

OUTPUT_DIR = BASE_DIR / "10_group_recommendation_outputs"

DEFAULT_GROUP_SIZE = 5
DEFAULT_GROUP_SIZES_FOR_EXPERIMENTS = [5, 7]

RANDOM_SEED = 42

# Οι μεγάλες ποινές στις συναρτήσεις κόστους είναι σκόπιμα ιεραρχικές.
```

```

# Προϋπόθεση γενικευσιμότητας: τα upstream scores παραμένουν τυποποιημένα
# ή σε συγκρίσιμη κλίμακα, όπως ορίζεται στο pipeline του Κεφαλαίου 4.

ROLE_ORDER = [
    "Ηγέτης",
    "Γέφυρα",
    "Δραστήριος Συμμετέχων",
    "Περιφερειακός Συμμετέχων",
    "Ανεπαρκής Αλληλεπίδραση",
]

ACTIVE_CORE_ROLES = [
    "Ηγέτης",
    "Γέφυρα",
    "Δραστήριος Συμμετέχων",
]

NON_INSUFFICIENT_ROLES = [
    "Ηγέτης",
    "Γέφυρα",
    "Δραστήριος Συμμετέχων",
    "Περιφερειακός Συμμετέχων",
]

INSUFFICIENT_ROLE = "Ανεπαρκής Αλληλεπίδραση"

STRATA_ORDER = [
    "Πολύ Υψηλό",
    "Υψηλό",
    "Μεσαίο",
    "Χαμηλό",
    "Πολύ Χαμηλό",
]

NUMERIC_COLUMNS = [
    "profile_score_z",
    "social_role_score",
    "performance_score",
    "engagement_score",
    "frustration_score",
]

OPTIONAL_NUMERIC_COLUMNS = [
    "profile_score",
    "profile_score_z",
    "social_role_score",
    "performance_score",
    "engagement_score",
    "frustration_score",
]

REQUIRED_BASE_COLUMNS = [
    "user_id",
    "profile_stratum",
]

ROLE_COLUMN_CANDIDATES = [
    "network_role_el",
    "network_role",
    "role",
]

# Τιμές fallback που χρησιμοποιούνται μόνο όταν το dataset περιέχει κενή
# ή μη αναγνωρισμένη τιμή. Κάθε fallback καταγράφεται στο validation report.
ROLE_FALLBACK_VALUE = INSUFFICIENT_ROLE
STRATUM_FALLBACK_VALUE = "Μεσαίο"
NUMERIC_FALLBACK_VALUE = 0.0

# =====
# BASIC UTILITIES
# =====

def safe_float(x):
    try:
        if pd.isna(x):
            return 0.0
        return float(x)
    except Exception:
        return 0.0

def normalize_user_id(value):
    if pd.isna(value):
        return np.nan
    value = str(value).strip()
    value = value.replace("User", "").replace("USER", "").replace("user", "")

```

```

try:
    number = float(value)
    if number.is_integer():
        return str(int(number))
except ValueError:
    pass
return value

def read_csv_safely(path):
    path = Path(path)
    if not path.exists():
        raise FileNotFoundError(f"Δεν βρέθηκε το αρχείο: {path}")
    return pd.read_csv(path, encoding="utf-8-sig")

def find_first_existing_column(df, candidates):
    for col in candidates:
        if col in df.columns:
            return col
    return None

def normalize_role_value(value):
    """
    Κανονικοποιεί τιμή κοινωνικού ρόλου.

    Επιστρέφει:
    - canonical_role
    - used_fallback: True/False
    - original_value

    Δεν επιτρέπεται σιωπηλό fallback: κάθε μη αναγνωρισμένη τιμή
    καταγράφεται στο validation report.
    """
    original = value

    if pd.isna(value) or str(value).strip() == "":
        return ROLE_FALLBACK_VALUE, True, original

    value = str(value).strip()

    mapping = {
        "leader": "Ηγέτης",
        "Ηγετης": "Ηγέτης",
        "Ηγέτης": "Ηγέτης",
        "bridge": "Γέφυρα",
        "Γεφυρα": "Γέφυρα",
        "Γέφυρα": "Γέφυρα",
        "active": "Δραστήριος Συμμετέχων",
        "Δραστήριος Συμμετέχων": "Δραστήριος Συμμετέχων",
        "δραστήριος συμμετέχων": "Δραστήριος Συμμετέχων",
        "peripheral": "Περιφερειακός Συμμετέχων",
        "Περιφερειακός Παρατηρητής": "Περιφερειακός Συμμετέχων",
        "Περιφερειακός Συμμετέχων": "Περιφερειακός Συμμετέχων",
        "περιφερειακός συμμετέχων": "Περιφερειακός Συμμετέχων",
        "no_interaction": ROLE_FALLBACK_VALUE,
        "No interaction": ROLE_FALLBACK_VALUE,
        "Ανεπαρκής Αλληλεπίδραση": ROLE_FALLBACK_VALUE,
        "ανεπαρκής αλληλεπίδραση": ROLE_FALLBACK_VALUE,
    }

    if value in mapping:
        return mapping[value], False, original

    if value in ROLE_ORDER:
        return value, False, original

    return ROLE_FALLBACK_VALUE, True, original

def normalize_stratum_value(value):
    """
    Κανονικοποιεί τιμή στρώματος προφίλ και καταγράφει fallback.
    """
    original = value

    if pd.isna(value) or str(value).strip() == "":
        return STRATUM_FALLBACK_VALUE, True, original

    value = str(value).strip()

    mapping = {
        "Very High": "Πολύ Υψηλό",
        "very_high": "Πολύ Υψηλό",
        "Πολυ Υψηλο": "Πολύ Υψηλό",
        "Πολύ Υψηλό": "Πολύ Υψηλό",
    }

```

```

        "High": "Υψηλό",
        "high": "Υψηλό",
        "Υψηλο": "Υψηλό",
        "Υψηλό": "Υψηλό",
        "Medium": "Μεσαίο",
        "medium": "Μεσαίο",
        "Μεσαιο": "Μεσαίο",
        "Μεσαίο": "Μεσαίο",
        "Low": "Χαμηλό",
        "low": "Χαμηλό",
        "Χαμηλο": "Χαμηλό",
        "Χαμηλό": "Χαμηλό",
        "Very Low": "Πολύ Χαμηλό",
        "very_low": "Πολύ Χαμηλό",
        "Πολυ Χαμηλο": "Πολύ Χαμηλό",
        "Πολύ Χαμηλό": "Πολύ Χαμηλό",
    }

    if value in mapping:
        return mapping[value], False, original

    if value in STRATA_ORDER:
        return value, False, original

    return STRATUM_FALLBACK_VALUE, True, original

def build_validation_rows(validation_stats):
    """Μετατρέπει τα validation diagnostics σε dataframe."""
    rows = []

    rows.append({
        "check": "role_fallback_count",
        "value": validation_stats["role_fallback_count"],
        "details": validation_stats["role_fallback_details"],
    })

    rows.append({
        "check": "profile_stratum_fallback_count",
        "value": validation_stats["stratum_fallback_count"],
        "details": validation_stats["stratum_fallback_details"],
    })

    for col, info in validation_stats["numeric_imputations"].items():
        rows.append({
            "check": f"numeric_imputation_count::{col}",
            "value": info["count"],
            "details": info["details"],
        })

    return pd.DataFrame(rows)

def validate_and_prepare_dataframe(df):
    """
    Ελέγχει και προετοιμάζει το dataset.

    Σημαντικό για γενικευσιμότητα:
    - Δεν γίνονται πλέον σωληλά fallbacks.
    - Κάθε μη αναγνωρισμένος ρόλος/stratum καταγράφεται.
    - Κάθε ελλιπής ή μη αριθμητική τιμή που γίνεται 0.0 καταγράφεται.
    """
    df = df.copy()

    missing = [c for c in REQUIRED_BASE_COLUMNS if c not in df.columns]
    if missing:
        raise ValueError("Λείπουν απαιτούμενες στήλες: " + ", ".join(missing))

    role_col = find_first_existing_column(df, ROLE_COLUMN_CANDIDATES)
    if role_col is None:
        raise ValueError(
            "Δεν βρέθηκε στήλη κοινωνικού ρόλου. "
            "Αναμενόμενες στήλες: " + ", ".join(ROLE_COLUMN_CANDIDATES)
        )

    validation_stats = {
        "role_column_used": role_col,
        "role_fallback_count": 0,
        "role_fallback_details": "",
        "stratum_fallback_count": 0,
        "stratum_fallback_details": "",
        "numeric_imputations": {},
    }

    df["user_id"] = df["user_id"].apply(normalize_user_id)

    # Ρόλοι με logging fallback

```

```

role_results = df[role_col].apply(normalize_role_value)
df["network_role_el"] = role_results.apply(lambda x: x[0])
role_fallback_mask = role_results.apply(lambda x: x[1])
validation_stats["role_fallback_count"] = int(role_fallback_mask.sum())
if validation_stats["role_fallback_count"] > 0:
    details = (
        df.loc[role_fallback_mask, role_col]
        .fillna("<MISSING>")
        .astype(str)
        .value_counts()
        .to_dict()
    )
    validation_stats["role_fallback_details"] = str(details)

# Strata με logging fallback
stratum_results = df["profile_stratum"].apply(normalize_stratum_value)
df["profile_stratum"] = stratum_results.apply(lambda x: x[0])
stratum_fallback_mask = stratum_results.apply(lambda x: x[1])
validation_stats["stratum_fallback_count"] = int(stratum_fallback_mask.sum())
if validation_stats["stratum_fallback_count"] > 0:
    details = (
        df.loc[stratum_fallback_mask, "profile_stratum"]
        .fillna("<MISSING>")
        .astype(str)
        .value_counts()
        .to_dict()
    )
    # Επειδή η στήλη έχει ήδη αντικατασταθεί, καταγράφουμε τις αρχικές τιμές από το tuple.
    original_values = [str(x[2]) if not pd.isna(x[2]) else "<MISSING>" for x in stratum_results[stratum_fallback_mask]]
    validation_stats["stratum_fallback_details"] = str(pd.Series(original_values).value_counts().to_dict())

# Αριθμητικές στήλες με logging imputation
for col in OPTIONAL_NUMERIC_COLUMNS:
    if col not in df.columns:
        df[col] = NUMERIC_FALLBACK_VALUE
        validation_stats["numeric_imputations"][col] = {
            "count": len(df),
            "details": "missing_column -> filled_with_0.0",
        }
        continue

    original = df[col].copy()
    converted = pd.to_numeric(original, errors="coerce")
    imputed_mask = converted.isna()
    imputed_count = int(imputed_mask.sum())

    if imputed_count > 0:
        original_values = original[imputed_mask].fillna("<MISSING>").astype(str).value_counts().to_dict()
        validation_stats["numeric_imputations"][col] = {
            "count": imputed_count,
            "details": str(original_values),
        }

    df[col] = converted.fillna(NUMERIC_FALLBACK_VALUE).astype(float)

# Διατηρούμε και network_role για συμβατότητα με παλιότερα reports.
df["network_role"] = df["network_role_el"]

validation_report = build_validation_rows(validation_stats)

return df, validation_stats, validation_report

# =====
# HOMOPHILY DIAGNOSTIC INPUT
# =====

def load_active_homophily_constraints():
    """
    Διαβάζει τα reports ομοφιλίας και επιστρέφει ποιες διαστάσεις
    εμφανίζουν στατιστικά σημαντική ομοφιλία στο τρέχον dataset.

    Η μεθοδολογία είναι γενική: σε άλλο dataset μπορεί να ενεργοποιηθούν
    διαφορετικά χαρακτηριστικά.
    """
    active_categorical = []
    active_numeric = []

    if HOMOPHILY_CATEGORICAL_FILE.exists():
        cat = read_csv_safely(HOMOPHILY_CATEGORICAL_FILE)
        for _, row in cat.iterrows():
            attr = row.get("attribute")
            p_value = safe_float(row.get("p_value"))
            observed = safe_float(row.get("homophily_ratio"))
            null_mean = safe_float(row.get("null_mean"))

            if attr and p_value < 0.05 and observed > null_mean:
                active_categorical.append(attr)

```

```

if HOMOPHILY_NUMERIC_FILE.exists():
    num = read_csv_safely(HOMOPHILY_NUMERIC_FILE)
    for _, row in num.iterrows():
        attr = row.get("attribute")
        p_value = safe_float(row.get("p_value"))
        observed = safe_float(row.get("mean_abs_difference"))
        null_mean = safe_float(row.get("null_mean_abs_difference"))

        if attr and p_value < 0.05 and observed < null_mean:
            active_numeric.append(attr)

return {
    "active_categorical": active_categorical,
    "active_numeric": active_numeric,
}

# =====
# TARGETS / CAPACITIES
# =====

def build_group_capacities(n_students, group_size):
    num_groups = math.ceil(n_students / group_size)
    capacities = {gid: group_size for gid in range(1, num_groups + 1)}

    overflow = num_groups * group_size - n_students
    for i in range(overflow):
        capacities[num_groups - i] -= 1

    return num_groups, capacities

def build_strata_targets(df, num_groups):
    global_counts = (
        df["profile_stratum"]
        .value_counts()
        .reindex(STRATA_ORDER, fill_value=0)
        .to_dict()
    )

    base = {s: global_counts[s] // num_groups for s in STRATA_ORDER}
    remainder = {s: global_counts[s] % num_groups for s in STRATA_ORDER}

    targets = {
        gid: {s: base[s] for s in STRATA_ORDER}
        for gid in range(1, num_groups + 1)
    }

    for s in STRATA_ORDER:
        for gid in range(1, remainder[s] + 1):
            targets[gid][s] += 1

    return targets

def compute_global_numeric_means(df):
    return {col: df[col].astype(float).mean() for col in NUMERIC_COLUMNS if col in df.columns}

# =====
# SORTING PRIORITIES
# =====

def get_student_strength_score(student):
    """
    Βοηθητικός δείκτης για ταξινόμηση μέσα σε κάθε ρόλο.
    Δεν είναι τελική βαθμολογία φοιτητή και δεν εμφανίζεται στα αποτελέσματα.
    """
    return (
        2.0 * safe_float(student.get("profile_score_z", 0.0))
        + 1.5 * safe_float(student.get("social_role_score", 0.0))
        + 2.0 * safe_float(student.get("performance_score", 0.0))
        + 2.0 * safe_float(student.get("engagement_score", 0.0))
        - 1.5 * safe_float(student.get("frustration_score", 0.0))
    )

def sort_bucket(df, role_name):
    tmp = df.copy()
    tmp["_strength"] = tmp.apply(get_student_strength_score, axis=1)

    if role_name in ["Ηγέτης", "Γέφυρα"]:
        by = ["_strength", "social_role_score", "engagement_score", "profile_score_z", "frustration_score", "user_id"]
        ascending = [False, False, False, False, True, True]

    elif role_name == "Δραστήριος Συμμετέχων":

```

```

        by = ["_strength", "engagement_score", "performance_score", "profile_score_z", "frustration_score", "user_id"]
        ascending = [False, False, False, False, True, True]

    elif role_name == "Περιφερειακός Συμμετέχων":
        by = ["_strength", "profile_score_z", "performance_score", "engagement_score", "frustration_score", "user_id"]
        ascending = [False, False, False, False, True, True]

    else: # Ανεπαρκής Αλληλεπίδραση
        by = ["frustration_score", "_strength", "user_id"]
        ascending = [False, False, True]

    return tmp.sort_values(by=by, ascending=ascending).drop(columns=["_strength"]).reset_index(drop=True)

# =====
# CORE ALGORITHM
# =====

def assign_groups(df, group_size, homophily_constraints):
    n_students = len(df)
    num_groups, capacities = build_group_capacities(n_students, group_size)
    strata_targets = build_strata_targets(df, num_groups)
    global_means = compute_global_numeric_means(df)

    active_categorical = homophily_constraints.get("active_categorical", [])
    active_numeric = homophily_constraints.get("active_numeric", [])

    profile_stratum_homophily_active = "profile_stratum" in active_categorical
    role_homophily_active = "network_role_el" in active_categorical or "network_role" in active_categorical

    groups = {gid: [] for gid in range(1, num_groups + 1)}
    group_sizes = {gid: 0 for gid in range(1, num_groups + 1)}

    group_role_counts = {gid: {r: 0 for r in ROLE_ORDER} for gid in range(1, num_groups + 1)}
    group_strata_counts = {gid: {s: 0 for s in STRATA_ORDER} for gid in range(1, num_groups + 1)}
    group_numeric_sums = {gid: {col: 0.0 for col in NUMERIC_COLUMNS} for gid in range(1, num_groups + 1)}

    def add_student_to_group(gid, student):
        row = student.to_dict()
        groups[gid].append(row)
        group_sizes[gid] += 1

        role = student["network_role_el"]
        stratum = student["profile_stratum"]
        group_role_counts[gid][role] += 1
        group_strata_counts[gid][stratum] += 1

        for col in NUMERIC_COLUMNS:
            group_numeric_sums[gid][col] += safe_float(student[col])

    def active_core_count(gid):
        return sum(group_role_counts[gid][r] for r in ACTIVE_CORE_ROLES)

    def non_insufficient_count(gid):
        return sum(group_role_counts[gid][r] for r in NON_INSUFFICIENT_ROLES)

    def size_penalty(gid):
        if group_sizes[gid] >= capacities[gid]:
            return 1e12
        return group_sizes[gid] / max(capacities[gid], 1)

    def stratum_penalty(gid, stratum):
        """
        Ποινή υπερευγένητρησης ίδιου στρώματος.
        Αν η ομοφιλία στο profile_stratum είναι σημαντική, η ποινή ενισχύεται.
        """
        current = group_strata_counts[gid][stratum]
        target = strata_targets[gid][stratum]

        if current < target:
            return 0.0

        base_penalty = 2.5 + (current - target + 1) * 3.5
        multiplier = 2.0 if profile_stratum_homophily_active else 1.0
        return multiplier * base_penalty

    def projected_group_balance_penalty(gid, student):
        """
        Μετρά πόσο θα απέκλιναν οι μέσοι όροι της ομάδας από τους καθολικούς
        μέσους όρους μετά την τοποθέτηση του φοιτητή.

        Αυτό αντικαθιστά την παλιά mean_fit_penalty, η οποία μπορούσε να ευνοεί
        την εσωτερική ομοιογένεια. Εδώ στόχος είναι η ισορροπία μεταξύ ομάδων.
        """
        new_size = group_sizes[gid] + 1
        if new_size <= 0:
            return 0.0

```

```

score = 0.0
weights = {
    "profile_score_z": 1.0,
    "social_role_score": 0.6,
    "performance_score": 1.0,
    "engagement_score": 1.0,
    "frustration_score": 1.0,
}

for col in NUMERIC_COLUMNS:
    projected_mean = (group_numeric_sums[gid][col] + safe_float(student[col])) / new_size
    score += weights.get(col, 1.0) * abs(projected_mean - global_means[col])

return score

def numeric_homophily_penalty(gid, student):
    """
    Προσαρμοστική ποινή για αριθμητικά χαρακτηριστικά όπου έχει βρεθεί
    στατιστικά σημαντική ομοφιλία. Στο τρέχον dataset δεν ενεργοποιείται,
    αλλά παραμένει γενικεύσιμη για μελλοντικά Moodle datasets.
    """
    if group_sizes[gid] == 0 or not active_numeric:
        return 0.0

    penalty = 0.0
    for col in active_numeric:
        if col not in NUMERIC_COLUMNS:
            continue
        group_mean = group_numeric_sums[gid][col] / group_sizes[gid]
        distance = abs(safe_float(student[col]) - group_mean)
        # Όσο πιο κοντά είναι στον ήδη υπάρχοντα μέσο όρο, τόσο μεγαλύτερη η ποινή.
        penalty += 1.0 / (1.0 + distance)
    return penalty

def role_duplicate_penalty(gid, role):
    count = group_role_counts[gid][role]
    if count == 0:
        return 0.0

    if role == "Ηγέτης":
        return 120000.0 * count
    if role == "Γέφυρα":
        return 100000.0 * count
    if role == "Δραστήριος Συμμετέχων":
        return 80000.0 * count
    if role == "Περιφερειακός Συμμετέχων":
        return 800.0 * count
    return 2500.0 * count

def has_available_group_without_active_core():
    return any(
        group_sizes[gid] < capacities[gid]
        and active_core_count(gid) == 0
        for gid in range(1, num_groups + 1)
    )

def has_available_group_without_non_insufficient():
    return any(
        group_sizes[gid] < capacities[gid]
        and non_insufficient_count(gid) == 0
        for gid in range(1, num_groups + 1)
    )

def coverage_priority_penalty(gid, role):
    """
    Ρητός κανόνας κάλυψης πριν από τη βελτιστοποίηση.

    Αν υπάρχουν ακόμη διαθέσιμες ομάδες χωρίς ενεργό πυρήνα, οι ρόλοι
    ενεργού πυρήνα δεν επιτρέπεται να στοιβάζονται σε ομάδες που έχουν ήδη
    ενεργό πυρήνα, εκτός αν αυτό είναι αναπόφευκτο.

    Αν υπάρχουν ακόμη διαθέσιμες ομάδες χωρίς κανένα μη-ανεπαρκές μέλος,
    κάθε μη-ανεπαρκής φοιτητής πρέπει πρώτα να καλύπτει τέτοιες ομάδες.

    Με αυτόν τον τρόπο ο αλγόριθμος μεγιστοποιεί λεξικογραφικά την
    παιδαγωγική κάλυψη ομάδων πριν εφαρμόσει τις δευτερεύουσες ποινές
    ισορροπίας strata, ομοφιλίας και αριθμητικών χαρακτηριστικών.
    """
    penalty = 0.0

    if role in ACTIVE_CORE_ROLES:
        if has_available_group_without_active_core() and active_core_count(gid) > 0:
            penalty += 1e10

    if role in NON_INSUFFICIENT_ROLES:
        if has_available_group_without_non_insufficient() and non_insufficient_count(gid) > 0:

```

```

        penalty += 5e9

    return penalty

def base_score(gid, student):
    if group_sizes[gid] >= capacities[gid]:
        return 1e12

    role = student["network_role_el"]
    stratum = student["profile_stratum"]

    score = 0.0
    score += coverage_priority_penalty(gid, role)
    score += 10.0 * size_penalty(gid)
    score += 25.0 * stratum_penalty(gid, stratum)
    score += 8.0 * projected_group_balance_penalty(gid, student)
    score += 10.0 * numeric_homophily_penalty(gid, student)
    score += role_duplicate_penalty(gid, role)

    if role_homophily_active:
        score += 1.5 * role_duplicate_penalty(gid, role)

    return score

def score_for_leader(gid, student):
    score = base_score(gid, student)
    score += active_core_count(gid) * 250.0
    score += non_insufficient_count(gid) * 30.0
    return score

def score_for_bridge(gid, student):
    score = base_score(gid, student)
    score += active_core_count(gid) * 220.0
    score += non_insufficient_count(gid) * 25.0
    return score

def score_for_active(gid, student):
    score = base_score(gid, student)
    score += active_core_count(gid) * 150.0
    score += non_insufficient_count(gid) * 20.0
    return score

def score_for_peripheral(gid, student):
    score = base_score(gid, student)
    # Προτιμά ομάδες με μικρότερο αριθμό μη παθητικών μελών, ώστε να απλωθεί ο ρόλος.
    score += non_insufficient_count(gid) * 120.0
    return score

def score_for_insufficient(gid, student):
    score = base_score(gid, student)

    non_ins = non_insufficient_count(gid)
    active_core = active_core_count(gid)
    insufficient_count = group_role_counts[gid][INSUFFICIENT_ROLE]

    # Αποφεύγουμε ομάδα χωρίς κανένα ενεργό/μη παθητικό μέλος.
    if non_ins == 0:
        score += 80000.0
    elif active_core == 0:
        score += 5000.0

    # Αποφεύγουμε συσσώρευση φοιτητών ανεπαρκούς αλληλεπίδρασης.
    score += insufficient_count * 3000.0

    # Μικρή προτίμηση σε ομάδες με ενεργό πυρήνα.
    if active_core > 0:
        score -= 100.0

    return score

def place_bucket(bucket_df, scoring_function):
    for _, student in bucket_df.iterrows():
        best_gid = None
        best_score = None

        for gid in range(1, num_groups + 1):
            score = scoring_function(gid, student)
            if best_score is None or score < best_score:
                best_score = score
                best_gid = gid

        add_student_to_group(best_gid, student)

phases = [
    ("Ηγέτης", score_for_leader),
    ("Γέφυρα", score_for_bridge),
    ("Δραστήριος Συμμετέχων", score_for_active),

```

```

        ("Περιφερειακός Συμμετέχων", score_for_peripheral),
        (INSUFFICIENT_ROLE, score_for_insufficient),
    ]

    for role_name, scoring_function in phases:
        bucket = sort_bucket(df[df["network_role_el"] == role_name], role_name)
        place_bucket(bucket, scoring_function)

    records = []
    for gid in range(1, num_groups + 1):
        for student in groups[gid]:
            row = dict(student)
            row["group_id"] = gid
            row["group_label"] = f"Ομάδα {gid:02d}"
            records.append(row)

    result = pd.DataFrame(records)
    result = result.sort_values(["group_id", "user_id"]).reset_index(drop=True)
    result["within_group_order"] = result.groupby("group_id").cumcount() + 1

    metadata = {
        "num_groups": num_groups,
        "capacities": capacities,
        "strata_targets": strata_targets,
        "active_categorical_homophily": active_categorical,
        "active_numeric_homophily": active_numeric,
        "profile_stratum_homophily_active": profile_stratum_homophily_active,
        "role_homophily_active": role_homophily_active,
    }

    return result, metadata

# =====
# REPORTS
# =====

def build_group_summary(result):
    summary = (
        result.groupby(["group_id", "group_label"])
        .agg(
            members=("user_id", "size"),
            mean_profile_score_z=("profile_score_z", "mean"),
            mean_social_role_score=("social_role_score", "mean"),
            mean_performance_score=("performance_score", "mean"),
            mean_engagement_score=("engagement_score", "mean"),
            mean_frustration_score=("frustration_score", "mean"),
            leaders=("network_role_el", lambda s: (s == "Ηγέτης").sum()),
            bridges=("network_role_el", lambda s: (s == "Γέφυρα").sum()),
            active=("network_role_el", lambda s: (s == "Δραστήριος Συμμετέχων").sum()),
            peripheral=("network_role_el", lambda s: (s == "Περιφερειακός Συμμετέχων").sum()),
            insufficient_interaction=("network_role_el", lambda s: (s == INSUFFICIENT_ROLE).sum()),
            very_high=("profile_stratum", lambda s: (s == "Πολύ Υψηλό").sum()),
            high=("profile_stratum", lambda s: (s == "Υψηλό").sum()),
            medium=("profile_stratum", lambda s: (s == "Μεσαίο").sum()),
            low=("profile_stratum", lambda s: (s == "Χαμηλό").sum()),
            very_low=("profile_stratum", lambda s: (s == "Πολύ Χαμηλό").sum()),
        )
        .reset_index()
    )

    summary["active_core_count"] = summary["leaders"] + summary["bridges"] + summary["active"]
    summary["non_insufficient_count"] = (
        summary["leaders"] + summary["bridges"] + summary["active"] + summary["peripheral"]
    )
    summary["role_diversity"] = summary[["leaders", "bridges", "active", "peripheral", "insufficient_interac-
tion"]].gt(0).sum(axis=1)

    return summary

def build_balance_report(result):
    group_means = (
        result.groupby("group_id")
        .agg(
            members=("user_id", "size"),
            profile_mean=("profile_score_z", "mean"),
            social_role_mean=("social_role_score", "mean"),
            performance_mean=("performance_score", "mean"),
            engagement_mean=("engagement_score", "mean"),
            frustration_mean=("frustration_score", "mean"),
        )
        .reset_index()
    )

    rows = []
    for metric in ["members", "profile_mean", "social_role_mean", "performance_mean", "engagement_mean", "frustration_mean"]:

```

```

        values = group_means[metric].astype(float)
        rows.append({
            "metric": metric,
            "global_mean": round(values.mean(), 4),
            "group_std": round(values.std(ddof=0), 4),
            "min_group_value": round(values.min(), 4),
            "max_group_value": round(values.max(), 4),
            "range": round(values.max() - values.min(), 4),
        })

    return pd.DataFrame(rows)

def build_strata_deviation(result):
    global_dist = result["profile_stratum"].value_counts(normalize=True).reindex(STRATA_ORDER, fill_value=0.0)

    group_props = (
        result.pivot_table(
            index="group_id",
            columns="profile_stratum",
            values="user_id",
            aggfunc="count",
            fill_value=0,
        )
        .reindex(columns=STRATA_ORDER, fill_value=0)
    )
    group_props = group_props.div(group_props.sum(axis=1), axis=0)
    deviation = (group_props - global_dist).abs().sum(axis=1)

    return pd.DataFrame({
        "group_id": deviation.index,
        "strata_deviation": deviation.values,
    }).sort_values("strata_deviation", ascending=False).reset_index(drop=True)

def build_quality_report(result):
    rows = []
    for gid, g in result.groupby("group_id"):
        roles = g["network_role_el"]
        rows.append({
            "group_id": gid,
            "members": len(g),
            "leaders": int((roles == "Ηγέτης").sum()),
            "bridges": int((roles == "Γέφυρα").sum()),
            "active": int((roles == "Δραστήριος Συμμετέχων").sum()),
            "peripheral": int((roles == "Περιφερειακός Συμμετέχων").sum()),
            "insufficient_interaction": int((roles == INSUFFICIENT_ROLE).sum()),
            "has_leader": int((roles == "Ηγέτης").any()),
            "has_bridge": int((roles == "Γέφυρα").any()),
            "has_active": int((roles == "Δραστήριος Συμμετέχων").any()),
            "has_peripheral": int((roles == "Περιφερειακός Συμμετέχων").any()),
            "has_active_core": int(roles.isin(ACTIVE_CORE_ROLES).any()),
            "has_non_insufficient": int(roles.isin(NON_INSUFFICIENT_ROLES).any()),
            "role_diversity": int(roles.nunique()),
        })
    return pd.DataFrame(rows)

def build_overall_quality(result, quality_report, strata_deviation, balance_report):
    rows = [
        {"metric": "students", "value": len(result)},
        {"metric": "num_groups", "value": result["group_id"].nunique()},
        {"metric": "groups_with_leader", "value": int(quality_report["has_leader"].sum())},
        {"metric": "groups_with_bridge", "value": int(quality_report["has_bridge"].sum())},
        {"metric": "groups_with_active", "value": int(quality_report["has_active"].sum())},
        {"metric": "groups_with_peripheral", "value": int(quality_report["has_peripheral"].sum())},
        {"metric": "groups_with_active_core", "value": int(quality_report["has_active_core"].sum())},
        {"metric": "groups_with_non_insufficient", "value": int(quality_report["has_non_insufficient"].sum())},
        {"metric": "mean_role_diversity", "value": round(quality_report["role_diversity"].mean(), 4)},
        {"metric": "max_insufficient_interaction_in_group", "value": int(quality_report["insufficient_interaction"].max())},
        {"metric": "mean_strata_deviation", "value": round(strata_deviation["strata_deviation"].mean(), 4)},
        {"metric": "max_strata_deviation", "value": round(strata_deviation["strata_deviation"].max(), 4)},
    ]

    for _, row in balance_report.iterrows():
        rows.append({"metric": f"{row['metric']}_std", "value": row["group_std"]})

    return pd.DataFrame(rows)

def write_outputs(result, metadata, group_size, validation_report, validation_stats):
    OUTPUT_DIR.mkdir(parents=True, exist_ok=True)

    prefix = f"10_group_recommendation_size_{group_size}"

    assignments_path = OUTPUT_DIR / f"{prefix}_assignments.csv"
    summary_path = OUTPUT_DIR / f"{prefix}_group_summary.csv"

```

```

balance_path = OUTPUT_DIR / f"{prefix}_balance_report.csv"
strata_dev_path = OUTPUT_DIR / f"{prefix}_strata_deviation.csv"
quality_path = OUTPUT_DIR / f"{prefix}_group_quality.csv"
overall_path = OUTPUT_DIR / f"{prefix}_overall_quality.csv"
homophily_constraints_path = OUTPUT_DIR / f"{prefix}_active_homophily_constraints.csv"
validation_path = OUTPUT_DIR / f"{prefix}_validation_report.csv"
run_report_path = OUTPUT_DIR / f"{prefix}_run_report.txt"

summary = build_group_summary(result)
balance_report = build_balance_report(result)
strata_deviation = build_strata_deviation(result)
quality_report = build_quality_report(result)
overall_quality = build_overall_quality(result, quality_report, strata_deviation, balance_report)

active_homophily_df = pd.DataFrame([
    {"constraint_type": "categorical", "attribute": attr}
    for attr in metadata["active_categorical_homophily"]
] + [
    {"constraint_type": "numeric", "attribute": attr}
    for attr in metadata["active_numeric_homophily"]
])

result.to_csv(assignments_path, index=False, encoding="utf-8-sig")
summary.to_csv(summary_path, index=False, encoding="utf-8-sig")
balance_report.to_csv(balance_path, index=False, encoding="utf-8-sig")
strata_deviation.to_csv(strata_dev_path, index=False, encoding="utf-8-sig")
quality_report.to_csv(quality_path, index=False, encoding="utf-8-sig")
overall_quality.to_csv(overall_path, index=False, encoding="utf-8-sig")
active_homophily_df.to_csv(homophily_constraints_path, index=False, encoding="utf-8-sig")
validation_report.to_csv(validation_path, index=False, encoding="utf-8-sig")

lines = []
lines.append("ΑΝΑΦΟΡΑ ΕΚΤΕΛΕΣΗΣ ΑΛΓΟΡΙΘΜΟΥ ΣΥΣΤΑΣΗΣ ΟΜΑΔΩΝ")
lines.append("=" * 70)
lines.append("")
lines.append(f"Μέγεθος ομάδας: {group_size}")
lines.append(f"Πλήθος φοιτητών: {len(result)}")
lines.append(f"Πλήθος ομάδων: {metadata['num_groups']}")
lines.append("")
lines.append("Ενεργές διαστάσεις ομοφιλίας:")
lines.append(f"- Κατηγορικές: {metadata['active_categorical_homophily']}")
lines.append(f"- Αριθμητικές: {metadata['active_numeric_homophily']}")
lines.append("")
lines.append("Έλεγχος ποιότητας εισόδου / fallbacks:")
lines.append(f"- Στήλη ρόλου που χρησιμοποιήθηκε: {validation_stats['role_column_used']}")
lines.append(f"- Μη αναγνωρισμένες/κενές τιμές ρόλου που τέθηκαν σε '{ROLE_FALLBACK_VALUE}': {validation_stats['role_fallback_count']}")
if validation_stats['role_fallback_details']:
    lines.append(f"  Λεπτομέρειες ρόλων: {validation_stats['role_fallback_details']}")
lines.append(f"- Μη αναγνωρισμένες/κενές τιμές profile_stratum που τέθηκαν σε '{STRATUM_FALLBACK_VALUE}': {validation_stats['stratum_fallback_count']}")
if validation_stats['stratum_fallback_details']:
    lines.append(f"  Λεπτομέρειες strata: {validation_stats['stratum_fallback_details']}")
for col, info in validation_stats['numeric_imputations'].items():
    lines.append(f"- Αριθμητικές τιμές που συμπληρώθηκαν με 0.0 στη στήλη {col}: {info['count']} ({info['details']})")
lines.append("")
lines.append("Κύριοι δείκτες ποιότητας:")
for _, row in overall_quality.iterrows():
    lines.append(f"- {row['metric']}: {row['value']}")
lines.append("")
lines.append("Αρχεία εξόδου:")
for p in [assignments_path, summary_path, balance_path, strata_dev_path, quality_path, overall_path, homophily_constraints_path, validation_path]:
    lines.append(f"- {p.name}")

run_report_path.write_text("\n".join(lines), encoding="utf-8")

return {
    "assignments": assignments_path,
    "summary": summary_path,
    "balance": balance_path,
    "strata_deviation": strata_dev_path,
    "quality": quality_path,
    "overall": overall_path,
    "homophily_constraints": homophily_constraints_path,
    "validation_report": validation_path,
    "run_report": run_report_path,
    "overall_quality": overall_quality,
}

# =====
# MAIN
# =====

def run_for_group_size(group_size):
    df = read_csv_safely(INPUT_FILE)

```

```

df, validation_stats, validation_report = validate_and_prepare_dataframe(df)

homophily_constraints = load_active_homophily_constraints()

result, metadata = assign_groups(df, group_size, homophily_constraints)
outputs = write_outputs(result, metadata, group_size, validation_report, validation_stats)

print("\n=== ΟΛΟΚΛΗΡΩΣΗ ===")
print(f"Μέγεθος ομάδας: {group_size}")
print(f"Φοιτητές: {len(result)}")
print(f"Ομάδες: {metadata['num_groups']}")
print(f"Ενεργές κατηγορικές διαστάσεις ομοφιλίας: {metadata['active_categorical_homophily']}")
print(f"Ενεργές αριθμητικές διαστάσεις ομοφιλίας: {metadata['active_numeric_homophily']}")
print(f"Fallbacks ρόλων: {validation_stats['role_fallback_count']}")
print(f"Fallbacks strata: {validation_stats['stratum_fallback_count']}")
numeric_total = sum(info['count'] for info in validation_stats['numeric_imputations'].values())
print(f"Αριθμητικές τιμές που συμπληρώθηκαν/imputed: {numeric_total}")
print("\nΑρχεία εξόδου:")
for key, path in outputs.items():
    if key != "overall_quality":
        print(f"- {path}")

print("\n=== OVERALL QUALITY ===")
print(outputs["overall_quality"].to_string(index=False))

def main():
    parser = argparse.ArgumentParser(description="Role-aware και homophily-aware σύσταση ομάδων.")
    parser.add_argument("--group-size", type=int, default=DEFAULT_GROUP_SIZE, help="Επιθυμητό μέγεθος ομάδας.")
    parser.add_argument("--all-sizes", action="store_true", help="Τρέχει για μεγέθη ομάδας 5 και 7.")
    args = parser.parse_args()

    start = time.time()

    if args.all_sizes:
        for size in DEFAULT_GROUP_SIZES_FOR_EXPERIMENTS:
            run_for_group_size(size)
    else:
        run_for_group_size(args.group_size)

    print(f"\nΣυνολικός χρόνος εκτέλεσης: {time.time() - start:.2f} sec")

if __name__ == "__main__":
    main()

```

A.21. 11_05_4_experimental_evaluation.py

Χαρακτήρας: Κύριο script - Συγκριτική αξιολόγηση

Περιγραφή: Εκτελεί και συγκρίνει τρεις στρατηγικές σχηματισμού ομάδων - Random baseline, Performance-only baseline και Proposed - με ακριβώς τις ίδιες μετρικές παιδαγωγικής λειτουργικότητας και δομικής ισορροπίας για μεγέθη ομάδας 5 και 7.

Είσοδοι: 08_04_2_6_for_grouping.csv · 10_05_3_grouping_ALGORITHM_v2.py

Έξοδοι: 11_experimental_evaluation_outputs/ με assignments και αναφορές ανά σενάριο και μέγεθος ομάδας, καθώς και συγκριτικά αρχεία overall_long, overall_wide, group_quality, balance και strata_deviation

Πηγαίος κώδικας

```
# -*- coding: utf-8 -*-
"""
11_05_4_experimental_evaluation.py

Πειραματική συγκριτική αξιολόγηση τριών στρατηγικών σχηματισμού ομάδων:

A. Random baseline
B. Performance-only baseline
C. Proposed role-aware και homophily-aware method

Το script αξιοποιεί απευθείας τις συναρτήσεις του τελικού αλγορίθμου
10_05_3_grouping_ALGORITHM_v2.py, ώστε το σενάριο Proposed να είναι
απολύτως ταυτόσημο με την υλοποίηση της Ενότητας 5.3.

Ενδεικτική εκτέλεση:
    py 11_05_4_experimental_evaluation.py --all-sizes
    py 11_05_4_experimental_evaluation.py --group-size 5
    py 11_05_4_experimental_evaluation.py --group-size 7
"""

from __future__ import annotations

from pathlib import Path
import argparse
import importlib.util
import math
import random
import time
from typing import Any

import pandas as pd

# =====
# CONFIG
# =====

BASE_DIR = Path(__file__).resolve().parent

INPUT_PROFILE_FILE = BASE_DIR / "08_04_2_6_for_grouping.csv"
PROPOSED_SCRIPT_FILE = BASE_DIR / "10_05_3_grouping_ALGORITHM_v2.py"

OUTPUT_DIR = BASE_DIR / "11_experimental_evaluation_outputs"

DEFAULT_GROUP_SIZE = 5
DEFAULT_GROUP_SIZES_FOR_EXPERIMENTS = [5, 7]
RANDOM_SEED = 42

ROLE_ORDER = [
    "Ηγέτης",
    "Γέφυρα",
    "Δραστήριος Συμμετέχων",
    "Περιφερειακός Συμμετέχων",
    "Ανεπαρκής Αλληλεπίδραση",
]

ACTIVE_CORE_ROLES = [
    "Ηγέτης",
    "Γέφυρα",
    "Δραστήριος Συμμετέχων",
]

NON_INSUFFICIENT_ROLES = [
```

```

        "Ηγέτης",
        "Γέφυρα",
        "Δραστήριος Συμμετέχων",
        "Περιφερειακός Συμμετέχων",
    ]

    INSUFFICIENT_ROLE = "Ανεπαρκής Αλληλεπίδραση"

    STRATA_ORDER = [
        "Πολύ Υψηλό",
        "Υψηλό",
        "Μεσαίο",
        "Χαμηλό",
        "Πολύ Χαμηλό",
    ]

    NUMERIC_COLUMNS = [
        "profile_score_z",
        "social_role_score",
        "performance_score",
        "engagement_score",
        "frustration_score",
    ]

    # =====
    # LOAD FINAL PROPOSED ALGORITHM
    # =====

    def load_proposed_module(script_path: Path):
        """Φορτώνει δυναμικά το τελικό script του προτεινόμενου αλγορίθμου."""
        if not script_path.exists():
            raise FileNotFoundError(
                f"Δεν βρέθηκε το τελικό script του προτεινόμενου αλγορίθμου: {script_path}"
            )

        spec = importlib.util.spec_from_file_location("proposed_grouping_algorithm", script_path)
        if spec is None or spec.loader is None:
            raise ImportError(f"Αδυναμία φόρτωσης του module: {script_path}")

        module = importlib.util.module_from_spec(spec)
        spec.loader.exec_module(module)
        return module

    def validate_proposed_module(module: Any) -> None:
        """Ελέγχει ότι το τελικό module διαθέτει τις συναρτήσεις που απαιτούνται."""
        required_functions = [
            "read_csv_safely",
            "validate_and_prepare_dataframe",
            "load_active_homophily_constraints",
            "assign_groups",
            "build_group_capacities",
            "build_strata_targets",
        ]

        missing = [name for name in required_functions if not hasattr(module, name)]
        if missing:
            raise AttributeError(
                "Το προτεινόμενο script δεν περιέχει τις απαιτούμενες συναρτήσεις: "
                + ", ".join(missing)
            )

    # =====
    # BASELINE ASSIGNMENTS
    # =====

    def assign_sequential_groups(
        df: pd.DataFrame,
        group_size: int,
        capacities: dict[int, int],
    ) -> pd.DataFrame:
        """Αναθέτει διαδοχικά τις ήδη ταξινομημένες γραμμές στις ομάδες."""
        frames: list[pd.DataFrame] = []
        start = 0

        for gid in sorted(capacities):
            capacity = capacities[gid]
            part = df.iloc[start:start + capacity].copy()
            part["group_id"] = gid
            part["group_label"] = f"Ομάδα {gid:02d}"
            frames.append(part)
            start += capacity

        if start != len(df):
            raise RuntimeError(

```

```

        f"Ασυνέπεια ανάθεσης baseline: ανατέθηκαν {start} από {len(df)} φοιτητές."
    )

    result = pd.concat(frames, ignore_index=True)
    result = result.sort_values(["group_id", "user_id"]).reset_index(drop=True)
    result["within_group_order"] = result.groupby("group_id").cumcount() + 1
    return result

def create_random_baseline(
    profiles_df: pd.DataFrame,
    group_size: int,
    capacities: dict[int, int],
    seed: int = RANDOM_SEED,
) -> pd.DataFrame:
    """
    Random baseline:
    τυχαία αναδιάταξη φοιτητών και διαδοχική κατανομή στις ομάδες.
    """
    rng = random.Random(seed)
    order = list(profiles_df.index)
    rng.shuffle(order)

    shuffled = profiles_df.loc[order].reset_index(drop=True)
    return assign_sequential_groups(shuffled, group_size, capacities)

def create_performance_only_baseline(
    profiles_df: pd.DataFrame,
    group_size: int,
    capacities: dict[int, int],
) -> pd.DataFrame:
    """
    Performance-only baseline:
    φθίνουσα ταξινόμηση κατά performance_score και διαδοχική κατανομή.

    Η διαδοχική κατανομή δημιουργεί ομάδες φοιτητών με παραπλήσια επίδοση,
    δηλαδή προσγγίζει ομοιογενές ability grouping.
    """
    sorted_df = profiles_df.sort_values(
        by=["performance_score", "user_id"],
        ascending=[False, True],
        kind="mergesort",
    ).reset_index(drop=True)

    return assign_sequential_groups(sorted_df, group_size, capacities)

# =====
# COMMON EVALUATION METRICS
# =====

def build_group_summary(result: pd.DataFrame) -> pd.DataFrame:
    """Σύνοψη σύνθεσης και αριθμητικών μέσων ανά ομάδα."""
    summary = (
        result.groupby(["group_id", "group_label"])
        .agg(
            members=("user_id", "size"),
            mean_profile_score_z=("profile_score_z", "mean"),
            mean_social_role_score=("social_role_score", "mean"),
            mean_performance_score=("performance_score", "mean"),
            mean_engagement_score=("engagement_score", "mean"),
            mean_frustration_score=("frustration_score", "mean"),
            leaders=("network_role_el", lambda s: int((s == "Ηγέτης").sum())),
            bridges=("network_role_el", lambda s: int((s == "Γέφυρα").sum())),
            active=("network_role_el", lambda s: int((s == "Δραστήριος Συμμετέχων").sum())),
            peripheral=("network_role_el", lambda s: int((s == "Περιφερειακός Συμμετέχων").sum())),
            insufficient_interaction=(
                "network_role_el",
                lambda s: int((s == INSUFFICIENT_ROLE).sum()),
            ),
            very_high=("profile_stratum", lambda s: int((s == "Πολύ Υψηλό").sum())),
            high=("profile_stratum", lambda s: int((s == "Υψηλό").sum())),
            medium=("profile_stratum", lambda s: int((s == "Μεσαίο").sum())),
            low=("profile_stratum", lambda s: int((s == "Χαμηλό").sum())),
            very_low=("profile_stratum", lambda s: int((s == "Πολύ Χαμηλό").sum())),
        )
        .reset_index()
    )

    summary["active_core_count"] = (
        summary["leaders"] + summary["bridges"] + summary["active"]
    )
    summary["non_insufficient_count"] = (
        summary["leaders"]
        + summary["bridges"]
        + summary["active"]
    )

```

```

    + summary["peripheral"]
)
summary["role_diversity"] = summary[
    ["leaders", "bridges", "active", "peripheral", "insufficient_interaction"]
].gt(0).sum(axis=1)

return summary

def build_group_quality(result: pd.DataFrame) -> pd.DataFrame:
    """Διαγνωστικοί δείκτες ποιότητας ανά ομάδα."""
    rows: list[dict[str, Any]] = []

    for gid, group in result.groupby("group_id", sort=True):
        roles = group["network_role_el"]

        leaders = int((roles == "Ηγέτης").sum())
        bridges = int((roles == "Γέφυρα").sum())
        active = int((roles == "Δραστήριος Συμμετέχων").sum())
        peripheral = int((roles == "Περιφερειακός Συμμετέχων").sum())
        insufficient = int((roles == INSUFFICIENT_ROLE).sum())

        rows.append({
            "group_id": int(gid),
            "group_label": f"Ομάδα {int(gid):02d}",
            "members": len(group),
            "leaders": leaders,
            "bridges": bridges,
            "active": active,
            "peripheral": peripheral,
            "insufficient_interaction": insufficient,
            "has_leader": int(leaders > 0),
            "has_bridge": int(bridges > 0),
            "has_active": int(active > 0),
            "has_peripheral": int(peripheral > 0),
            "has_active_core": int(roles.isin(ACTIVE_CORE_ROLES).any()),
            "has_non_insufficient": int(roles.isin(NON_INSUFFICIENT_ROLES).any()),
            "role_diversity": int(roles.nunique()),
        })

    return pd.DataFrame(rows)

def build_balance_report(result: pd.DataFrame) -> pd.DataFrame:
    """Διασπορά των μεγεθών και αριθμητικών μέσων μεταξύ ομάδων."""
    group_means = (
        result.groupby("group_id")
        .agg(
            members=("user_id", "size"),
            profile_mean=("profile_score_z", "mean"),
            social_role_mean=("social_role_score", "mean"),
            performance_mean=("performance_score", "mean"),
            engagement_mean=("engagement_score", "mean"),
            frustration_mean=("frustration_score", "mean"),
        )
        .reset_index()
    )

    rows: list[dict[str, Any]] = []
    for metric in [
        "members",
        "profile_mean",
        "social_role_mean",
        "performance_mean",
        "engagement_mean",
        "frustration_mean",
    ]:
        values = group_means[metric].astype(float)
        rows.append({
            "metric": metric,
            "global_mean": round(values.mean(), 6),
            "group_std": round(values.std(ddof=0), 6),
            "min_group_value": round(values.min(), 6),
            "max_group_value": round(values.max(), 6),
            "range": round(values.max() - values.min(), 6),
        })

    return pd.DataFrame(rows)

def build_strata_deviation(result: pd.DataFrame) -> pd.DataFrame:
    """
    L1 απόκλιση της κατανομής strata κάθε ομάδας από την κατανομή
    του συνολικού πληθυσμού.
    """
    global_dist = (
        result["profile_stratum"]

```

```

        .value_counts(normalize=True)
        .reindex(STRATA_ORDER, fill_value=0.0)
    )

    group_props = (
        result.pivot_table(
            index="group_id",
            columns="profile_stratum",
            values="user_id",
            aggfunc="count",
            fill_value=0,
        )
        .reindex(columns=STRATA_ORDER, fill_value=0)
    )

    group_props = group_props.div(group_props.sum(axis=1), axis=0)
    deviation = (group_props - global_dist).abs().sum(axis=1)

    return (
        pd.DataFrame({
            "group_id": deviation.index.astype(int),
            "strata_deviation": deviation.values,
        })
        .sort_values("strata_deviation", ascending=False)
        .reset_index(drop=True)
    )

def build_overall_metrics(
    result: pd.DataFrame,
    group_quality: pd.DataFrame,
    strata_deviation: pd.DataFrame,
    balance_report: pd.DataFrame,
) -> pd.DataFrame:
    """Συγκεντρωτικές μετρικές για τη σύγκριση των τριών σεναρίων."""
    rows: list[dict[str, Any]] = [
        {"metric": "students", "value": len(result)},
        {"metric": "num_groups", "value": result["group_id"].nunique()},
        {
            "metric": "groups_with_leader",
            "value": int(group_quality["has_leader"].sum()),
        },
        {
            "metric": "groups_with_bridge",
            "value": int(group_quality["has_bridge"].sum()),
        },
        {
            "metric": "groups_with_active",
            "value": int(group_quality["has_active"].sum()),
        },
        {
            "metric": "groups_with_peripheral",
            "value": int(group_quality["has_peripheral"].sum()),
        },
        {
            "metric": "groups_with_active_core",
            "value": int(group_quality["has_active_core"].sum()),
        },
        {
            "metric": "groups_with_non_insufficient",
            "value": int(group_quality["has_non_insufficient"].sum()),
        },
        {
            "metric": "mean_role_diversity",
            "value": round(group_quality["role_diversity"].mean(), 6),
        },
        {
            "metric": "max_insufficient_interaction_in_group",
            "value": int(group_quality["insufficient_interaction"].max()),
        },
        {
            "metric": "mean_strata_deviation",
            "value": round(strata_deviation["strata_deviation"].mean(), 6),
        },
        {
            "metric": "max_strata_deviation",
            "value": round(strata_deviation["strata_deviation"].max(), 6),
        },
    ]

    for _, row in balance_report.iterrows():
        rows.append({
            "metric": f"{row['metric']}_std",
            "value": row["group_std"],
        })

    return pd.DataFrame(rows)

```

```

def evaluate_scenario(
    result: pd.DataFrame,
    scenario_name: str,
) -> dict[str, pd.DataFrame]:
    """Υπολογίζει ακριβώς τις ίδιες μετρικές για κάθε σενάριο."""
    summary = build_group_summary(result)
    quality = build_group_quality(result)
    balance = build_balance_report(result)
    strata = build_strata_deviation(result)
    overall = build_overall_metrics(result, quality, strata, balance)

    for frame in [summary, quality, balance, strata, overall]:
        frame.insert(0, "scenario", scenario_name)

    return {
        "group_summary": summary,
        "group_quality": quality,
        "balance_report": balance,
        "strata_deviation": strata,
        "overall_metrics": overall,
    }

# =====
# OUTPUT
# =====

def save_scenario_outputs(
    scenario_name: str,
    group_size: int,
    assignments: pd.DataFrame,
    evaluation: dict[str, pd.DataFrame],
) -> dict[str, Path]:
    """Αποθηκεύει τα αρχεία ενός σεναρίου."""
    OUTPUT_DIR.mkdir(parents=True, exist_ok=True)

    prefix = f"11_evaluation_{scenario_name}_size_{group_size}"

    paths = {
        "assignments": OUTPUT_DIR / f"{prefix}_assignments.csv",
        "group_summary": OUTPUT_DIR / f"{prefix}_group_summary.csv",
        "group_quality": OUTPUT_DIR / f"{prefix}_group_quality.csv",
        "balance_report": OUTPUT_DIR / f"{prefix}_balance_report.csv",
        "strata_deviation": OUTPUT_DIR / f"{prefix}_strata_deviation.csv",
        "overall_metrics": OUTPUT_DIR / f"{prefix}_overall_metrics.csv",
    }

    assignments.to_csv(paths["assignments"], index=False, encoding="utf-8-sig")
    for key in [
        "group_summary",
        "group_quality",
        "balance_report",
        "strata_deviation",
        "overall_metrics",
    ]:
        evaluation[key].to_csv(paths[key], index=False, encoding="utf-8-sig")

    return paths

def save_comparative_outputs(
    group_size: int,
    evaluations: dict[str, dict[str, pd.DataFrame]],
) -> dict[str, Path]:
    """Αποθηκεύει συγκριτικά αρχεία κοινής μορφής."""
    OUTPUT_DIR.mkdir(parents=True, exist_ok=True)

    overall_long = pd.concat(
        [pack["overall_metrics"] for pack in evaluations.values()],
        ignore_index=True,
    )

    overall_wide = (
        overall_long
        .pivot(index="metric", columns="scenario", values="value")
        .reset_index()
    )

    all_group_quality = pd.concat(
        [pack["group_quality"] for pack in evaluations.values()],
        ignore_index=True,
    )

    all_balance = pd.concat(
        [pack["balance_report"] for pack in evaluations.values()],

```

```

        ignore_index=True,
    )

    all_strata = pd.concat(
        [pack["strata_deviation"] for pack in evaluations.values()],
        ignore_index=True,
    )

    prefix = f"11_evaluation_comparison_size_{group_size}"
    paths = {
        "overall_long": OUTPUT_DIR / f"{prefix}_overall_long.csv",
        "overall_wide": OUTPUT_DIR / f"{prefix}_overall_wide.csv",
        "all_group_quality": OUTPUT_DIR / f"{prefix}_all_group_quality.csv",
        "all_balance": OUTPUT_DIR / f"{prefix}_all_balance.csv",
        "all_strata_deviation": OUTPUT_DIR / f"{prefix}_all_strata_deviation.csv",
    }

    overall_long.to_csv(paths["overall_long"], index=False, encoding="utf-8-sig")
    overall_wide.to_csv(paths["overall_wide"], index=False, encoding="utf-8-sig")
    all_group_quality.to_csv(
        paths["all_group_quality"], index=False, encoding="utf-8-sig"
    )
    all_balance.to_csv(paths["all_balance"], index=False, encoding="utf-8-sig")
    all_strata.to_csv(
        paths["all_strata_deviation"], index=False, encoding="utf-8-sig"
    )

    return paths

def print_comparison(group_size: int, overall_wide: pd.DataFrame) -> None:
    print("\n" + "=" * 90)
    print(f"ΤΕΛΙΚΗ ΣΥΓΚΡΙΣΗ – ΜΕΓΕΘΟΣ ΟΜΑΔΑΣ {group_size}")
    print("=" * 90)
    print(overall_wide.round(6).to_string(index=False))

# =====
# EXECUTION
# =====

def run_evaluation_for_size(
    module: Any,
    profiles_df: pd.DataFrame,
    validation_report: pd.DataFrame,
    validation_stats: dict[str, Any],
    homophily_constraints: dict[str, list[str]],
    group_size: int,
) -> None:
    """Εκτελεί και συγκρίνει τα τρία σενάρια για ένα μέγεθος ομάδας."""
    num_groups, capacities = module.build_group_capacities(
        len(profiles_df), group_size
    )

    # A. Random baseline
    random_assignments = create_random_baseline(
        profiles_df,
        group_size,
        capacities,
        seed=RANDOM_SEED,
    )

    # B. Performance-only baseline
    performance_assignments = create_performance_only_baseline(
        profiles_df,
        group_size,
        capacities,
    )

    # C. Proposed method – ακριβώς η τελική υλοποίηση της Ενότητας 5.3
    proposed_assignments, proposed_metadata = module.assign_groups(
        profiles_df,
        group_size,
        homophily_constraints,
    )

    scenarios = {
        "random": random_assignments,
        "performance_only": performance_assignments,
        "proposed": proposed_assignments,
    }

    evaluations = {
        name: evaluate_scenario(assignments, name)
        for name, assignments in scenarios.items()
    }

```

```

print("\n" + "=" * 90)
print(f"5.4 ΠΕΙΡΑΜΑΤΙΚΗ ΑΞΙΟΛΟΓΗΣΗ – ΜΕΓΕΘΟΣ ΟΜΑΔΑΣ {group_size}")
print("=" * 90)
print(f"Φοιτητές: {len(profiles_df)}")
print(f"Ομάδες: {num_groups}")
print(f"Random seed: {RANDOM_SEED}")
print(
    "Ενεργές κατηγορικές διαστάσεις ομοφιλίας: "
    f"{homophily_constraints.get('active_categorical', [])}"
)
print(
    "Ενεργές αριθμητικές διαστάσεις ομοφιλίας: "
    f"{homophily_constraints.get('active_numeric', [])}"
)
print(f"Fallbacks ρόλων: {validation_stats['role_fallback_count']}")
print(f"Fallbacks strata: {validation_stats['stratum_fallback_count']}")

for scenario_name, assignments in scenarios.items():
    save_scenario_outputs(
        scenario_name,
        group_size,
        assignments,
        evaluations[scenario_name],
    )

comparison_paths = save_comparative_outputs(group_size, evaluations)

overall_wide = pd.read_csv(
    comparison_paths["overall_wide"],
    encoding="utf-8-sig",
)
print_comparison(group_size, overall_wide)

# Αποθήκευση validation και homophily diagnostics της συγκεκριμένης εκτέλεσης.
validation_path = (
    OUTPUT_DIR / f"11_evaluation_validation_report_size_{group_size}.csv"
)
validation_report.to_csv(
    validation_path,
    index=False,
    encoding="utf-8-sig",
)

active_homophily = pd.DataFrame(
    [
        {"constraint_type": "categorical", "attribute": attr}
        for attr in homophily_constraints.get("active_categorical", [])
    ]
    + [
        {"constraint_type": "numeric", "attribute": attr}
        for attr in homophily_constraints.get("active_numeric", [])
    ]
)
active_homophily.to_csv(
    OUTPUT_DIR / f"11_evaluation_active_homophily_size_{group_size}.csv",
    index=False,
    encoding="utf-8-sig",
)

# Απλός έλεγχος ότι το proposed αποτέλεσμα συμφωνεί με το metadata.
if proposed_assignments["group_id"].nunique() != proposed_metadata["num_groups"]:
    raise RuntimeError(
        "Ασυνέπεια μεταξύ proposed assignments και proposed metadata."
    )

def main() -> None:
    parser = argparse.ArgumentParser(
        description="Συγκριτική αξιολόγηση Random, Performance-only και Proposed."
    )
    parser.add_argument(
        "--group-size",
        type=int,
        default=DEFAULT_GROUP_SIZE,
        help="Επιθυμητό μέγεθος ομάδας.",
    )
    parser.add_argument(
        "--all-sizes",
        action="store_true",
        help="Εκτελεί την αξιολόγηση για μεγέθη ομάδας 5 και 7.",
    )
    parser.add_argument(
        "--input-file",
        type=Path,
        default=INPUT_PROFILE_FILE,
        help="Αρχείο τελικού προφίλ για ομαδοποίηση.",
    )

```

```

parser.add_argument(
    "--proposed-script",
    type=Path,
    default=PROPOSED_SCRIPT_FILE,
    help="Τελικό script του προτεινόμενου αλγορίθμου.",
)
args = parser.parse_args()

if args.group_size < 2:
    raise ValueError("Το μέγεθος ομάδας πρέπει να είναι τουλάχιστον 2.")

start_time = time.time()

module = load_proposed_module(args.proposed_script)
validate_proposed_module(module)

raw_df = module.read_csv_safely(args.input_file)
profiles_df, validation_stats, validation_report = (
    module.validate_and_prepare_dataframe(raw_df)
)
homophily_constraints = module.load_active_homophily_constraints()

sizes = (
    DEFAULT_GROUP_SIZES_FOR_EXPERIMENTS
    if args.all_sizes
    else [args.group_size]
)

for group_size in sizes:
    run_evaluation_for_size(
        module=module,
        profiles_df=profiles_df,
        validation_report=validation_report,
        validation_stats=validation_stats,
        homophily_constraints=homophily_constraints,
        group_size=group_size,
    )

print("\n" + "=" * 90)
print(f"Ολοκληρώθηκε σε {time.time() - start_time:.2f} sec")
print(f"Αρχεία εξόδου: {OUTPUT_DIR}")

if __name__ == "__main__":
    main()

```

A.22. 12_05_4_repeated_random_evaluation.py

Χαρακτήρας: Κύριο script - Ανάλυση ευστάθειας αξιολόγησης

Περιγραφή: Επαναλαμβάνει το Random baseline 100 φορές με διαδοχικά seeds, συνοψίζει την εμπειρική κατανομή κάθε μετρικής και συγκρίνει την Proposed μέθοδο με τις τυχαίες εκτελέσεις μέσω διορθωμένης εμπειρικής p-value $(r+1)/(R+1)$.

Είσοδοι: 08_04_2_6_for_grouping.csv · 10_05_3_grouping_ALGORITHM_v2.py · 11_05_4_experimental_evaluation.py

Έξοδοι: 12_repeated_random_evaluation_outputs/ με raw metrics, περιγραφική σύνοψη, assignments και metrics των deterministic σεναρίων, proposed-vs-random, three-scenario comparison, validation, active homophily και run report

Πηγαίος κώδικας

```
# -*- coding: utf-8 -*-
"""
12_05_4_repeated_random_evaluation.py

Επαναλαμβανόμενη αξιολόγηση του Random baseline για τη διπλωματική εργασία.

Για κάθε εξεταζόμενο μέγεθος ομάδας:
1. Εκτελεί το Random baseline πολλές φορές με διαφορετικά seeds.
2. Υπολογίζει τις ίδιες μετρικές ποιότητας σε κάθε random run.
3. Συνοψίζει την εμπειρική κατανομή κάθε μετρικής:
   mean, std, median, min, max, 2.5ο και 97.5ο εκατοστημόριο.
4. Εκτελεί μία φορά το Performance-only baseline.
5. Εκτελεί μία φορά την Proposed μέθοδο της Ενότητας 5.3.
6. Υπολογίζει πόσα random runs πέτυχαν αποτέλεσμα ίσο ή καλύτερο
   από την Proposed μέθοδο και την αντίστοιχη διορθωμένη εμπειρική p-value.

Το script επαναχρησιμοποιεί τις συναρτήσεις του:
- 11_05_4_experimental_evaluation.py
- 10_05_3_grouping_ALGORITHM_v2.py

Ενδεικτική εκτέλεση:
py 12_05_4_repeated_random_evaluation.py --all-sizes
py 12_05_4_repeated_random_evaluation.py --all-sizes --random-runs 100
py 12_05_4_repeated_random_evaluation.py --group-size 5 --random-runs 100
"""

from __future__ import annotations

from pathlib import Path
import argparse
import importlib.util
import time
from typing import Any

import pandas as pd

# =====
# CONFIG
# =====

BASE_DIR = Path(__file__).resolve().parent

INPUT_PROFILE_FILE = BASE_DIR / "08_04_2_6_for_grouping.csv"
PROPOSED_SCRIPT_FILE = BASE_DIR / "10_05_3_grouping_ALGORITHM_v2.py"
EVALUATION_SCRIPT_FILE = BASE_DIR / "11_05_4_experimental_evaluation.py"

OUTPUT_DIR = BASE_DIR / "12_repeated_random_evaluation_outputs"

DEFAULT_GROUP_SIZE = 5
DEFAULT_GROUP_SIZES = [5, 7]
DEFAULT_RANDOM_RUNS = 100
DEFAULT_FIRST_SEED = 0

# Μετρικές στις οποίες μεγαλύτερη τιμή θεωρείται καλύτερη.
HIGHER_IS_BETTER = {
    "groups_with_leader",
    "groups_with_bridge",
    "groups_with_active",
    "groups_with_peripheral",
    "groups_with_active_core",
```

```

    "groups_with_non_insufficient",
    "mean_role_diversity",
}

# Μετρικές στις οποίες μικρότερη τιμή θεωρείται καλύτερη.
LOWER_IS_BETTER = {
    "max_insufficient_interaction_in_group",
    "mean_strata_deviation",
    "max_strata_deviation",
    "members_std",
    "profile_mean_std",
    "social_role_mean_std",
    "performance_mean_std",
    "engagement_mean_std",
    "frustration_mean_std",
}

# Περιγραφικές μετρικές που δεν χρησιμοποιούνται για σύγκριση ποιότητας.
DESCRIPTIVE_ONLY = {
    "students",
    "num_groups",
}

# =====
# DYNAMIC MODULE LOADING
# =====

def load_module(script_path: Path, module_name: str):
    """Φορτώνει δυναμικά ένα τοπικό Python script ως module."""
    if not script_path.exists():
        raise FileNotFoundError(f"Δεν βρέθηκε το script: {script_path}")

    spec = importlib.util.spec_from_file_location(module_name, script_path)
    if spec is None or spec.loader is None:
        raise ImportError(f"Αδυναμία φόρτωσης module από: {script_path}")

    module = importlib.util.module_from_spec(spec)
    spec.loader.exec_module(module)
    return module

def validate_modules(evaluation_module: Any, proposed_module: Any) -> None:
    """Ελέγχει ότι τα modules περιέχουν τις απαιτούμενες συναρτήσεις."""
    required_evaluation = [
        "create_random_baseline",
        "create_performance_only_baseline",
        "evaluate_scenario",
    ]
    required_proposed = [
        "read_csv_safely",
        "validate_and_prepare_dataframe",
        "load_active_homophily_constraints",
        "build_group_capacities",
        "assign_groups",
    ]

    missing_eval = [
        name for name in required_evaluation
        if not hasattr(evaluation_module, name)
    ]
    missing_proposed = [
        name for name in required_proposed
        if not hasattr(proposed_module, name)
    ]

    if missing_eval:
        raise AttributeError(
            "Λείπουν συναρτήσεις από το script 11: "
            + ", ".join(missing_eval)
        )

    if missing_proposed:
        raise AttributeError(
            "Λείπουν συναρτήσεις από το script 10: "
            + ", ".join(missing_proposed)
        )

# =====
# RANDOM RUNS
# =====

def run_repeated_random_baseline(
    evaluation_module: Any,
    profiles_df: pd.DataFrame,
    group_size: int,

```

```

        capacities: dict[int, int],
        n_runs: int,
        first_seed: int,
    ) -> pd.DataFrame:
        """
        Εκτελεί το Random baseline n_runs φορές.

        Επιστρέφει long-format dataframe με μία γραμμή ανά:
        run_id x metric.
        """
        all_metrics: list[pd.DataFrame] = []

        for run_index in range(n_runs):
            seed = first_seed + run_index

            assignments = evaluation_module.create_random_baseline(
                profiles_df=profiles_df,
                group_size=group_size,
                capacities=capacities,
                seed=seed,
            )

            evaluation = evaluation_module.evaluate_scenario(
                assignments,
                scenario_name="random",
            )

            metrics = evaluation["overall_metrics"][["metric", "value"]].copy()
            metrics.insert(0, "seed", seed)
            metrics.insert(0, "run_id", run_index + 1)
            metrics.insert(0, "group_size", group_size)
            all_metrics.append(metrics)

        if not all_metrics:
            raise RuntimeError("Δεν πραγματοποιήθηκε καμία τυχαία εκτέλεση.")

        return pd.concat(all_metrics, ignore_index=True)

def summarize_random_runs(random_metrics: pd.DataFrame) -> pd.DataFrame:
    """Παράγει περιγραφικά στατιστικά ανά μετρική."""
    required = {"metric", "value"}
    missing = required.difference(random_metrics.columns)
    if missing:
        raise ValueError(
            "Λείπουν στήλες από τα random metrics: " + ", ".join(sorted(missing))
        )

    summary = (
        random_metrics
        .groupby("metric", as_index=False)["value"]
        .agg(
            random_mean="mean",
            random_std="std",
            random_median="median",
            random_min="min",
            random_max="max",
        )
    )

    quantiles = (
        random_metrics
        .groupby("metric")["value"]
        .quantile([0.025, 0.975])
        .unstack()
        .reset_index()
        .rename(columns={
            0.025: "random_p2_5",
            0.975: "random_p97_5",
        })
    )

    summary = summary.merge(quantiles, on="metric", how="left")

    numeric_cols = [
        "random_mean",
        "random_std",
        "random_median",
        "random_min",
        "random_max",
        "random_p2_5",
        "random_p97_5",
    ]
    summary[numeric_cols] = summary[numeric_cols].round(6)

    return summary

```

```

# =====
# DETERMINISTIC SCENARIOS
# =====

def evaluate_deterministic_scenarios(
    evaluation_module: Any,
    proposed_module: Any,
    profiles_df: pd.DataFrame,
    group_size: int,
    capacities: dict[int, int],
    homophily_constraints: dict[str, list[str]],
) -> tuple[pd.DataFrame, pd.DataFrame, pd.DataFrame, pd.DataFrame]:
    """
    Εκτελεί μία φορά:
    - Performance-only baseline
    - Proposed method

    Επιστρέφει assignments και overall metrics και για τα δύο.
    """
    performance_assignments = evaluation_module.create_performance_only_baseline(
        profiles_df=profiles_df,
        group_size=group_size,
        capacities=capacities,
    )
    performance_eval = evaluation_module.evaluate_scenario(
        performance_assignments,
        scenario_name="performance_only",
    )
    performance_metrics = performance_eval["overall_metrics"][
        ["metric", "value"]
    ].copy()

    proposed_assignments, proposed_metadata = proposed_module.assign_groups(
        profiles_df,
        group_size,
        homophily_constraints,
    )
    proposed_eval = evaluation_module.evaluate_scenario(
        proposed_assignments,
        scenario_name="proposed",
    )
    proposed_metrics = proposed_eval["overall_metrics"][
        ["metric", "value"]
    ].copy()

    if proposed_assignments["group_id"].nunique() != proposed_metadata["num_groups"]:
        raise RuntimeError(
            "Ασυνέπεια μεταξύ proposed assignments και metadata."
        )

    return (
        performance_assignments,
        performance_metrics,
        proposed_assignments,
        proposed_metrics,
    )

# =====
# EMPIRICAL COMPARISON
# =====

def metric_direction(metric: str) -> str:
    if metric in HIGHER_IS_BETTER:
        return "higher_is_better"
    if metric in LOWER_IS_BETTER:
        return "lower_is_better"
    if metric in DESCRIPTIVE_ONLY:
        return "descriptive_only"
    return "not_classified"

def compare_proposed_with_random(
    random_metrics: pd.DataFrame,
    random_summary: pd.DataFrame,
    proposed_metrics: pd.DataFrame,
    n_runs: int,
) -> pd.DataFrame:
    """
    Συγκρίνει την Proposed τιμή κάθε μετρικής με την εμπειρική
    κατανομή των random runs.

    empirical_p_value = (equal_or_better + 1) / (n_runs + 1)

    Η τιμή αυτή είναι εμπειρική πιθανότητα από randomization runs,
    όχι παραμετρικό p-value.
    """

```

```

"""
proposed_lookup = dict(
    zip(proposed_metrics["metric"], proposed_metrics["value"])
)
summary_lookup = random_summary.set_index("metric").to_dict("index")

rows: list[dict[str, Any]] = []

for metric, proposed_value in proposed_lookup.items():
    direction = metric_direction(metric)

    values = random_metrics.loc[
        random_metrics["metric"] == metric,
        "value",
    ].astype(float)

    if values.empty:
        continue

    if direction == "higher_is_better":
        equal_or_better = int((values >= float(proposed_value)).sum())
        proposed_better_than_random = int((values < float(proposed_value)).sum())
    elif direction == "lower_is_better":
        equal_or_better = int((values <= float(proposed_value)).sum())
        proposed_better_than_random = int((values > float(proposed_value)).sum())
    else:
        equal_or_better = None
        proposed_better_than_random = None

    empirical_p = (
        (equal_or_better + 1) / (n_runs + 1)
        if equal_or_better is not None
        else None
    )

    random_info = summary_lookup.get(metric, {})

    rows.append({
        "metric": metric,
        "direction": direction,
        "proposed_value": round(float(proposed_value), 6),
        "random_mean": random_info.get("random_mean"),
        "random_std": random_info.get("random_std"),
        "random_median": random_info.get("random_median"),
        "random_min": random_info.get("random_min"),
        "random_max": random_info.get("random_max"),
        "random_p2_5": random_info.get("random_p2_5"),
        "random_p97_5": random_info.get("random_p97_5"),
        "random_runs_equal_or_better": equal_or_better,
        "random_runs_proposed_better": proposed_better_than_random,
        "random_runs": n_runs,
        "proposed_better_percentage": (
            round(100.0 * proposed_better_than_random / n_runs, 2)
            if proposed_better_than_random is not None
            else None
        ),
        "empirical_p_value": (
            round(empirical_p, 6)
            if empirical_p is not None
            else None
        ),
    })

return pd.DataFrame(rows)

def build_three_scenario_comparison(
    random_summary: pd.DataFrame,
    performance_metrics: pd.DataFrame,
    proposed_metrics: pd.DataFrame,
) -> pd.DataFrame:
    """Συνδυάζει Random mean, Performance-only και Proposed."""
    performance = performance_metrics.rename(
        columns={"value": "performance_only"}
    )
    proposed = proposed_metrics.rename(columns={"value": "proposed"})

    comparison = (
        random_summary[
            [
                "metric",
                "random_mean",
                "random_std",
                "random_min",
                "random_max",
                "random_p2_5",
                "random_p97_5",
            ]

```

```

    ]
    ].merge(performance, on="metric", how="outer")
    ].merge(proposed, on="metric", how="outer")
)

comparison.insert(
    1,
    "direction",
    comparison["metric"].apply(metric_direction),
)

return comparison

# =====
# SAVE / REPORT
# =====

def write_run_report(
    path: Path,
    group_size: int,
    n_students: int,
    n_groups: int,
    n_runs: int,
    first_seed: int,
    homophily_constraints: dict[str, list[str]],
    comparison: pd.DataFrame,
) -> None:
    """Γράφει συνοπτική αναφορά εκτέλεσης σε TXT."""
    lines: list[str] = [
        "ΕΠΑΝΑΛΑΜΒΑΝΟΜΕΝΗ ΑΞΙΟΛΟΓΗΣΗ RANDOM BASELINE",
        "=" * 72,
        "",
        f"Μέγεθος ομάδας: {group_size}",
        f"Πλήθος φοιτητών: {n_students}",
        f"Πλήθος ομάδων: {n_groups}",
        f"Πλήθος random runs: {n_runs}",
        f"Seeds: {first_seed} έως {first_seed + n_runs - 1}",
        "",
        "Ενεργές διαστάσεις ομοφιλίας:",
        f"- Κατηγορικές: {homophily_constraints.get('active_categorical', [])}",
        f"- Αριθμητικές: {homophily_constraints.get('active_numeric', [])}",
        "",
        "Σημείωση:",
        "Η empirical_p_value υπολογίζεται ως (r + 1) / (R + 1),",
        "όπου r είναι τα random runs με αποτέλεσμα ίσο ή καλύτερο από",
        "την Proposed μέθοδο και R το συνολικό πλήθος random runs.",
        "",
        "ΣΥΓΚΡΙΣΗ PROPOSED ME RANDOM",
        "-" * 72,
        comparison.to_string(index=False),
    ]

    path.write_text("\n".join(lines), encoding="utf-8")

def save_outputs(
    group_size: int,
    random_metrics: pd.DataFrame,
    random_summary: pd.DataFrame,
    performance_assignments: pd.DataFrame,
    performance_metrics: pd.DataFrame,
    proposed_assignments: pd.DataFrame,
    proposed_metrics: pd.DataFrame,
    proposed_vs_random: pd.DataFrame,
    three_scenario_comparison: pd.DataFrame,
    validation_report: pd.DataFrame,
    homophily_constraints: dict[str, list[str]],
    n_students: int,
    n_groups: int,
    n_runs: int,
    first_seed: int,
) -> dict[str, Path]:
    OUTPUT_DIR.mkdir(parents=True, exist_ok=True)

    prefix = f"12_repeated_random_size_{group_size}"

    paths = {
        "random_runs_raw": OUTPUT_DIR / f"{prefix}_raw_metrics.csv",
        "random_summary": OUTPUT_DIR / f"{prefix}_summary.csv",
        "performance_assignments": OUTPUT_DIR / f"{prefix}_performance_assignments.csv",
        "performance_metrics": OUTPUT_DIR / f"{prefix}_performance_metrics.csv",
        "proposed_assignments": OUTPUT_DIR / f"{prefix}_proposed_assignments.csv",
        "proposed_metrics": OUTPUT_DIR / f"{prefix}_proposed_metrics.csv",
        "proposed_vs_random": OUTPUT_DIR / f"{prefix}_proposed_vs_random.csv",
        "three_scenario_comparison": OUTPUT_DIR / f"{prefix}_three_scenario_comparison.csv",
    }

```

```

        "validation_report": OUTPUT_DIR / f"{prefix}_validation_report.csv",
        "active_homophily": OUTPUT_DIR / f"{prefix}_active_homophily.csv",
        "run_report": OUTPUT_DIR / f"{prefix}_run_report.txt",
    }

    random_metrics.to_csv(
        paths["random_runs_raw"], index=False, encoding="utf-8-sig"
    )
    random_summary.to_csv(
        paths["random_summary"], index=False, encoding="utf-8-sig"
    )
    performance_assignments.to_csv(
        paths["performance_assignments"], index=False, encoding="utf-8-sig"
    )
    performance_metrics.to_csv(
        paths["performance_metrics"], index=False, encoding="utf-8-sig"
    )
    proposed_assignments.to_csv(
        paths["proposed_assignments"], index=False, encoding="utf-8-sig"
    )
    proposed_metrics.to_csv(
        paths["proposed_metrics"], index=False, encoding="utf-8-sig"
    )
    proposed_vs_random.to_csv(
        paths["proposed_vs_random"], index=False, encoding="utf-8-sig"
    )
    three_scenario_comparison.to_csv(
        paths["three_scenario_comparison"], index=False, encoding="utf-8-sig"
    )
    validation_report.to_csv(
        paths["validation_report"], index=False, encoding="utf-8-sig"
    )

    active_homophily = pd.DataFrame(
        [
            {"constraint_type": "categorical", "attribute": attr}
            for attr in homophily_constraints.get("active_categorical", [])
        ]
        + [
            {"constraint_type": "numeric", "attribute": attr}
            for attr in homophily_constraints.get("active_numeric", [])
        ]
    )
    active_homophily.to_csv(
        paths["active_homophily"], index=False, encoding="utf-8-sig"
    )

    write_run_report(
        path=paths["run_report"],
        group_size=group_size,
        n_students=n_students,
        n_groups=n_groups,
        n_runs=n_runs,
        first_seed=first_seed,
        homophily_constraints=homophily_constraints,
        comparison=proposed_vs_random,
    )

    return paths

# =====
# EXECUTION PER GROUP SIZE
# =====

def run_for_group_size(
    evaluation_module: Any,
    proposed_module: Any,
    profiles_df: pd.DataFrame,
    validation_report: pd.DataFrame,
    homophily_constraints: dict[str, list[str]],
    group_size: int,
    n_runs: int,
    first_seed: int,
) -> None:
    """Εκτελεί ολόκληρη την επαναλαμβανόμενη αξιολόγηση."""
    n_students = len(profiles_df)
    n_groups, capacities = proposed_module.build_group_capacities(
        n_students,
        group_size,
    )

    print("\n" + "=" * 92)
    print(f"12. ΕΠΑΝΑΛΑΜΒΑΝΟΜΕΝΗ RANDOM ΑΞΙΟΛΟΓΗΣΗ – ΜΕΓΕΘΟΣ {group_size}")
    print("=" * 92)
    print(f"Φοιτητές: {n_students}")
    print(f"Ομάδες: {n_groups}")

```

```

print(f"Random runs: {n_runs}")
print(f"Seeds: {first_seed}-{first_seed + n_runs - 1}")

random_metrics = run_repeated_random_baseline(
    evaluation_module=evaluation_module,
    profiles_df=profiles_df,
    group_size=group_size,
    capacities=capacities,
    n_runs=n_runs,
    first_seed=first_seed,
)
random_summary = summarize_random_runs(random_metrics)

(
    performance_assignments,
    performance_metrics,
    proposed_assignments,
    proposed_metrics,
) = evaluate_deterministic_scenarios(
    evaluation_module=evaluation_module,
    proposed_module=proposed_module,
    profiles_df=profiles_df,
    group_size=group_size,
    capacities=capacities,
    homophily_constraints=homophily_constraints,
)

proposed_vs_random = compare_proposed_with_random(
    random_metrics=random_metrics,
    random_summary=random_summary,
    proposed_metrics=proposed_metrics,
    n_runs=n_runs,
)

three_scenario_comparison = build_three_scenario_comparison(
    random_summary=random_summary,
    performance_metrics=performance_metrics,
    proposed_metrics=proposed_metrics,
)

paths = save_outputs(
    group_size=group_size,
    random_metrics=random_metrics,
    random_summary=random_summary,
    performance_assignments=performance_assignments,
    performance_metrics=performance_metrics,
    proposed_assignments=proposed_assignments,
    proposed_metrics=proposed_metrics,
    proposed_vs_random=proposed_vs_random,
    three_scenario_comparison=three_scenario_comparison,
    validation_report=validation_report,
    homophily_constraints=homophily_constraints,
    n_students=n_students,
    n_groups=n_groups,
    n_runs=n_runs,
    first_seed=first_seed,
)

print("\n--- PROPOSED VS RANDOM ---")
display_cols = [
    "metric",
    "proposed_value",
    "random_mean",
    "random_std",
    "random_min",
    "random_max",
    "random_runs_equal_or_better",
    "empirical_p_value",
]
print(
    proposed_vs_random[
        proposed_vs_random["direction"].isin(
            ["higher_is_better", "lower_is_better"]
        )
    ][display_cols]
    .round(6)
    .to_string(index=False)
)

print("\nΑρχεία εξόδου:")
for label, path in paths.items():
    print(f"- {label}: {path.name}")

```

```

# =====
# MAIN
# =====

```

```

def main() -> None:
    parser = argparse.ArgumentParser(
        description=(
            "Επαναλαμβανόμενη αξιολόγηση Random baseline και σύγκριση "
            "με Performance-only και Proposed."
        )
    )
    parser.add_argument(
        "--group-size",
        type=int,
        default=DEFAULT_GROUP_SIZE,
        help="Επιθυμητό μέγεθος ομάδας.",
    )
    parser.add_argument(
        "--all-sizes",
        action="store_true",
        help="Εκτελεί την αξιολόγηση για μεγέθη ομάδας 5 και 7.",
    )
    parser.add_argument(
        "--random-runs",
        type=int,
        default=DEFAULT_RANDOM_RUNS,
        help="Πλήθος ανεξάρτητων random runs. Προεπιλογή: 100.",
    )
    parser.add_argument(
        "--first-seed",
        type=int,
        default=DEFAULT_FIRST_SEED,
        help="Πρώτο seed. Τα επόμενα αυξάνονται διαδοχικά κατά 1.",
    )
    parser.add_argument(
        "--input-file",
        type=Path,
        default=INPUT_PROFILE_FILE,
        help="Τελικό αρχείο προφίλ για ομαδοποίηση.",
    )
    parser.add_argument(
        "--proposed-script",
        type=Path,
        default=PROPOSED_SCRIPT_FILE,
        help="Τελικό script του προτεινόμενου αλγορίθμου.",
    )
    parser.add_argument(
        "--evaluation-script",
        type=Path,
        default=EVALUATION_SCRIPT_FILE,
        help="Script 11 με τις κοινές συναρτήσεις αξιολόγησης.",
    )
    args = parser.parse_args()

    if args.group_size < 2:
        raise ValueError("Το μέγεθος ομάδας πρέπει να είναι τουλάχιστον 2.")
    if args.random_runs < 2:
        raise ValueError(
            "Το πλήθος random runs πρέπει να είναι τουλάχιστον 2."
        )

    start_time = time.time()

    evaluation_module = load_module(
        args.evaluation_script,
        "experimental_evaluation_module",
    )
    proposed_module = load_module(
        args.proposed_script,
        "proposed_grouping_module",
    )
    validate_modules(evaluation_module, proposed_module)

    raw_df = proposed_module.read_csv_safely(args.input_file)
    profiles_df, validation_stats, validation_report = (
        proposed_module.validate_and_prepare_dataframe(raw_df)
    )
    homophily_constraints = (
        proposed_module.load_active_homophily_constraints()
    )

    numeric_imputation_total = sum(
        info["count"]
        for info in validation_stats["numeric_imputations"].values()
    )

    print("=== ΕΛΕΓΧΟΣ ΕΙΣΟΔΟΥ ===")
    print(f"Φοιτητές: {len(profiles_df)}")
    print(f"Fallbacks πόλων: {validation_stats['role_fallback_count']}")
    print(f"Fallbacks strata: {validation_stats['stratum_fallback_count']}")

```

```

print(f"Αριθμητικές imputations: {numeric_imputation_total}")
print(
    "Ενεργές κατηγορικές διαστάσεις ομοφιλίας: "
    f"{homophily_constraints.get('active_categorical', [])}"
)
print(
    "Ενεργές αριθμητικές διαστάσεις ομοφιλίας: "
    f"{homophily_constraints.get('active_numeric', [])}"
)

group_sizes = DEFAULT_GROUP_SIZES if args.all_sizes else [args.group_size]

for group_size in group_sizes:
    run_for_group_size(
        evaluation_module=evaluation_module,
        proposed_module=proposed_module,
        profiles_df=profiles_df,
        validation_report=validation_report,
        homophily_constraints=homophily_constraints,
        group_size=group_size,
        n_runs=args.random_runs,
        first_seed=args.first_seed,
    )

print("\n" + "=" * 92)
print(f"Ολοκληρώθηκε σε {time.time() - start_time:.2f} sec")
print(f"Φάκελος αποτελεσμάτων: {OUTPUT_DIR}")

if __name__ == "__main__":
    main()

```


Παράρτημα Β

Αναλυτικά αποτελέσματα στατιστικής και πειραματικής αξιολόγησης

Το παρόν παράρτημα παραθέτει τα πλήρη αποτελέσματα του post-hoc ελέγχου Dunn για τη στατιστική επικύρωση των κοινωνικών ρόλων και τα αναλυτικά αποτελέσματα της επαναλαμβανόμενης αξιολόγησης του Random baseline για μεγέθη ομάδας 5 και 7 ατόμων.

B.1 Πλήρη αποτελέσματα post-hoc ελέγχου Dunn

Οι ακόλουθοι πίνακες παρουσιάζουν τις διορθωμένες τιμές p του post-hoc ελέγχου Dunn με διόρθωση Bonferroni. Με έντονη γραφή επισημαίνονται οι στατιστικά σημαντικές διαφορές ($p < 0,05$). Οι συντομευμένες επικεφαλίδες «Δραστήριος», «Περιφερειακός» και «Ανεπαρκής» αντιστοιχούν στους ρόλους «Δραστήριος Συμμετέχων», «Περιφερειακός Συμμετέχων» και «Ανεπαρκής Αλληλεπίδραση».

Degree

Degree	Ηγέτης	Γέφυρα	Δραστήριος	Περιφερειακός	Ανεπαρκής
Ηγέτης	—	0,526	0,311	$3,48 \times 10^{-10}$	$1,38 \times 10^{-20}$
Γέφυρα	0,526	—	1,000	0,193	$1,76 \times 10^{-5}$
Δραστήριος	0,311	1,000	—	$7,59 \times 10^{-6}$	$6,78 \times 10^{-16}$
Περιφερειακός	$3,48 \times 10^{-10}$	0,193	$7,59 \times 10^{-6}$	—	$3,52 \times 10^{-5}$
Ανεπαρκής	$1,38 \times 10^{-20}$	$1,76 \times 10^{-5}$	$6,78 \times 10^{-16}$	$3,52 \times 10^{-5}$	—

Πίνακας B.1. Post-hoc έλεγχος Dunn (Bonferroni) — Degree. Με έντονη γραφή: $p < 0,05$.

Weighted Degree

Weighted Degree	Ηγέτης	Γέφυρα	Δραστήριος	Περιφερειακός	Ανεπαρκής
Ηγέτης	—	0,456	0,331	$4,09 \times 10^{-10}$	$2,04 \times 10^{-20}$
Γέφυρα	0,456	—	1,000	0,241	$2,78 \times 10^{-5}$
Δραστήριος	0,331	1,000	—	$7,44 \times 10^{-6}$	$7,87 \times 10^{-16}$
Περιφερειακός	$4,09 \times 10^{-10}$	0,241	$7,44 \times 10^{-6}$	—	$3,95 \times 10^{-5}$
Ανεπαρκής	$2,04 \times 10^{-20}$	$2,78 \times 10^{-5}$	$7,87 \times 10^{-16}$	$3,95 \times 10^{-5}$	—

Πίνακας B.2. Post-hoc έλεγχος Dunn (Bonferroni) — Weighted Degree. Με έντονη γραφή: $p < 0,05$.

Betweenness

Betweenness	Ηγέτης	Γέφυρα	Δραστήριος	Περιφερειακός	Ανεπαρκής
Ηγέτης	—	1,000	0,611	$3,66 \times 10^{-6}$	$1,20 \times 10^{-7}$
Γέφυρα	1,000	—	0,041	$4,29 \times 10^{-6}$	$2,08 \times 10^{-7}$
Δραστήριος	0,611	0,041	—	0,004	$1,52 \times 10^{-4}$

Περιφερειακός	$3,66 \times 10^{-6}$	$4,29 \times 10^{-6}$	0,004	—	1,000
Ανεπαρκής	$1,20 \times 10^{-7}$	$2,08 \times 10^{-7}$	$1,52 \times 10^{-4}$	1,000	—

Πίνακας Β.3. Post-hoc έλεγχος Dunn (Bonferroni) — Betweenness. Με έντονη γραφή: $p < 0,05$.

Closeness

Closeness	Ηγέτης	Γέφυρα	Δραστήριος	Περιφερειακός	Ανεπαρκής
Ηγέτης	—	1,000	1,000	$3,69 \times 10^{-7}$	$2,58 \times 10^{-17}$
Γέφυρα	1,000	—	1,000	0,434	$3,93 \times 10^{-5}$
Δραστήριος	1,000	1,000	—	$1,96 \times 10^{-5}$	$4,25 \times 10^{-16}$
Περιφερειακός	$3,69 \times 10^{-7}$	0,434	$1,96 \times 10^{-5}$	—	$1,01 \times 10^{-5}$
Ανεπαρκής	$2,58 \times 10^{-17}$	$3,93 \times 10^{-5}$	$4,25 \times 10^{-16}$	$1,01 \times 10^{-5}$	—

Πίνακας Β.4. Post-hoc έλεγχος Dunn (Bonferroni) — Closeness. Με έντονη γραφή: $p < 0,05$.

PageRank

PageRank	Ηγέτης	Γέφυρα	Δραστήριος	Περιφερειακός	Ανεπαρκής
Ηγέτης	—	0,706	0,043	$3,11 \times 10^{-9}$	$1,70 \times 10^{-20}$
Γέφυρα	0,706	—	1,000	0,239	$9,51 \times 10^{-6}$
Δραστήριος	0,043	1,000	—	0,003	$4,93 \times 10^{-13}$
Περιφερειακός	$3,11 \times 10^{-9}$	0,239	0,003	—	$5,80 \times 10^{-6}$
Ανεπαρκής	$1,70 \times 10^{-20}$	$9,51 \times 10^{-6}$	$4,93 \times 10^{-13}$	$5,80 \times 10^{-6}$	—

Πίνακας Β.5. Post-hoc έλεγχος Dunn (Bonferroni) — PageRank. Με έντονη γραφή: $p < 0,05$

B.2 Πλήρη αποτελέσματα ανάλυσης ευστάθειας

Η ανάλυση βασίζεται σε 100 ανεξάρτητες εκτελέσεις του Random baseline με διαφορετικές τιμές seed. Για κάθε μετρική παρουσιάζονται η μέση τιμή, η τυπική απόκλιση, η διάμεσος, η ελάχιστη και η μέγιστη τιμή, καθώς και το εμπειρικό διάστημα από το 2,5ό έως το 97,5ό εκατοστημόριο. Σε ξεχωριστό πίνακα παρουσιάζονται οι τιμές των στρατηγικών Performance-only και Proposed, το πλήθος r των τυχαίων εκτελέσεων που πέτυχαν αποτέλεσμα ίσο ή καλύτερο από την Proposed μέθοδο, το πλήθος και το ποσοστό των εκτελέσεων έναντι των οποίων υπερείχε η Proposed μέθοδος και η διορθωμένη εμπειρική τιμή $p = (r + 1)/(R + 1)$, με $R = 100$.

Το σύμβολο $\uparrow$ δηλώνει ότι μεγαλύτερη τιμή είναι προτιμότερη, ενώ το $\downarrow$ ότι μικρότερη τιμή είναι προτιμότερη. Η στήλη «Raw variable» διατηρεί την ακριβή ονομασία της μεταβλητής στα αρχεία εξόδου των scripts.

B.2.1 Ομάδες 5 ατόμων

Πλαίσιο εκτέλεσης: 383 φοιτητές, 77 ομάδες, 100 τυχαίες εκτελέσεις.

Μετρική	Raw variable	Κατ.	M.O.	T.A.	Διάμ.	Ελάχ.	Μέγ.	P2,5	P97,5
Ομάδες με τουλάχιστον έναν Ηγέτη	groups_with_leader	↑	16,530	1,039	16,000	14,000	18,000	14,475	18,000
Ομάδες με τουλάχιστον μία Γέφυρα	groups_with_bridge	↑	6,830	0,378	7,000	6,000	7,000	6,000	7,000
Ομάδες με τουλάχιστον έναν Δραστήριο Συμμετέχοντα	groups_with_active	↑	24,180	1,610	24,000	20,000	27,000	21,000	27,000
Ομάδες με τουλάχιστον έναν Περιφερειακό Συμμετέχοντα	groups_with_peripheral	↑	41,060	2,339	41,000	35,000	45,000	36,475	45,000
Ομάδες με ενεργό πυρήνα	groups_with_active_core	↑	39,920	2,073	40,000	35,000	46,000	35,000	43,525
Ομάδες με τουλάχιστον ένα μέλος χωρίς Ανεπαρκή Αλληλεπίδραση	groups_with_non_insufficient	↑	62,200	2,399	62,000	56,000	68,000	58,000	66,000
Μέση ποικιλία κοινωνικών ρόλων	mean_role_diversity	↑	2,148441	0,043770	2,155844	2,025974	2,233766	2,064935	2,214610
Μέγιστος αριθμός μελών Ανεπαρκούς Αλληλεπίδρασης ανά ομάδα	max_insufficient_interaction_in_group	↓	5,000	0,000	5,000	5,000	5,000	5,000	5,000
Μέση απόκλιση κατανομής στρωμάτων	mean_strata_deviation	↓	0,733809	0,022408	0,733450	0,660100	0,781849	0,692851	0,770770
Μέγιστη απόκλιση κατανομής στρωμάτων	max_strata_deviation	↓	1,310034	0,132559	1,338903	1,127937	1,738903	1,153525	1,467363
Τυπική απόκλιση μεγέθους ομάδων	members_std	↓	0,159058	0,000000	0,159058	0,159058	0,159058	0,159058	0,159058
Τυπική απόκλιση μέσου σύνθετου προφίλ μεταξύ ομάδων	profile_mean_std	↓	0,448324	0,031468	0,447287	0,376096	0,543232	0,387058	0,505842
Τυπική απόκλιση μέσου κοινωνικού ρόλου μεταξύ ομάδων	social_role_mean_std	↓	0,463052	0,031735	0,459340	0,383605	0,548636	0,401255	0,528984
Τυπική απόκλιση μέσης επίδοσης μεταξύ ομάδων	performance_mean_std	↓	0,365868	0,023812	0,365455	0,299164	0,416248	0,326822	0,408760
Τυπική απόκλιση μέσης συμμετοχής μεταξύ ομάδων	engagement_mean_std	↓	0,262140	0,017814	0,259902	0,224015	0,306274	0,227656	0,295776
Τυπική απόκλιση μέσης απογοήτευσης μεταξύ ομάδων	frustration_mean_std	↓	0,443488	0,033898	0,447310	0,352701	0,509044	0,368867	0,494533

Πίνακας Β.6α. Εμπειρική κατανομή των 100 εκτελέσεων του Random baseline — ομάδες 5 ατόμων.

Σύγκριση στρατηγικών και εμπειρική σημαντικότητα — ομάδες 5 ατόμων

Μετρική	Raw variable	Κατ.	Performance-only	Proposed	r ίσο/καλύτερο	Proposed καλύτερη (runs)	Proposed καλύτερη (%)	Εμπειρικό p	p < 0,05
Ομάδες με τουλάχιστον έναν Ηγέτη	groups_with_leader	↑	16	18	21	79	79,00%	0,217822	Όχι
Ομάδες με τουλάχιστον μία Γέφυρα	groups_with_bridge	↑	7	7	83	17	17,00%	0,831683	Όχι
Ομάδες με τουλάχιστον έναν Δραστήριο Συμμετέχοντα	groups_with_active	↑	25	28	0	100	100,00%	0,009901	Ναι
Ομάδες με τουλάχιστον έναν Περιφερειακό Συμμετέχοντα	groups_with_peripheral	↑	43	54	0	100	100,00%	0,009901	Ναι
Ομάδες με ενεργό πυρήνα	groups_with_active_core	↑	42	53	0	100	100,00%	0,009901	Ναι
Ομάδες με τουλάχιστον ένα μέλος χωρίς Ανεπαρκή Αλληλεπίδραση	groups_with_non_insufficient	↑	62	77	0	100	100,00%	0,009901	Ναι
Μέση ποικιλία κοινωνικών ρόλων	mean_role_diversity	↑	2,181818	2,389610	0	100	100,00%	0,009901	Ναι
Μέγιστος αριθμός μελών Ανεπαρκούς Αλληλεπίδρασης ανά ομάδα	max_insufficient_interaction_in_group	↓	5	4	0	100	100,00%	0,009901	Ναι
Μέση απόκλιση κατανομής στρωμάτων	mean_strata_deviation	↓	0,914506	0,493568	0	100	100,00%	0,009901	Ναι
Μέγιστη απόκλιση κατανομής στρωμάτων	max_strata_deviation	↓	1,738903	0,872063	0	100	100,00%	0,009901	Ναι
Τυπική απόκλιση μεγέθους ομάδων	members_std	↓	0,159058	0,159058	100	0	0,00%	1,000000	Όχι
Τυπική απόκλιση μέσου σύνθετου προφίλ μεταξύ ομάδων	profile_mean_std	↓	0,660257	0,198512	0	100	100,00%	0,009901	Ναι
Τυπική απόκλιση μέσου κοινωνικού ρόλου μεταξύ ομάδων	social_role_mean_std	↓	0,480007	0,245800	0	100	100,00%	0,009901	Ναι
Τυπική απόκλιση μέσης επίδοσης μεταξύ ομάδων	performance_mean_std	↓	0,838919	0,218340	0	100	100,00%	0,009901	Ναι
Τυπική απόκλιση μέσης συμμετοχής μεταξύ ομάδων	engagement_mean_std	↓	0,305492	0,132668	0	100	100,00%	0,009901	Ναι
Τυπική απόκλιση μέσης απογοήτευσης μεταξύ ομάδων	frustration_mean_std	↓	0,466628	0,415536	22	78	78,00%	0,227723	Όχι

Πίνακας Β.6β. Σύγκριση Performance-only και Proposed με την εμπειρική κατανομή του Random baseline — ομάδες 5 ατόμων.

B.2.2 Ομάδες 7 ατόμων

Πλαίσιο εκτέλεσης: 383 φοιτητές, 55 ομάδες, 100 τυχαίες εκτελέσεις.

Μετρική	Raw variable	Κατ.	M.O.	T.A.	Διάμ.	Ελάχ.	Μέγ.	P2,5	P97,5
Ομάδες με τουλάχιστον έναν Ηγέτη	groups_with_leader	↑	16,050	1,149	16,000	13,000	18,000	14,000	18,000
Ομάδες με τουλάχιστον μία Γέφυρα	groups_with_bridge	↑	6,650	0,500	7,000	5,000	7,000	6,000	7,000
Ομάδες με τουλάχιστον έναν Δραστήριο Συμμετέχοντα	groups_with_active	↑	22,570	1,559	22,500	18,000	26,000	20,000	25,000
Ομάδες με τουλάχιστον έναν Περιφερειακό Συμμετέχοντα	groups_with_peripheral	↑	35,880	2,266	36,000	29,000	42,000	32,000	39,525
Ομάδες με ενεργό πυρήνα	groups_with_active_core	↑	35,390	2,064	35,000	30,000	41,000	31,000	39,525
Ομάδες με τουλάχιστον ένα μέλος χωρίς Ανεπαρκή Αλληλεπίδραση	groups_with_non_insufficient	↑	49,230	1,814	49,000	43,000	53,000	46,000	52,525
Μέση ποικιλία κοινωνικών ρόλων	mean_role_diversity	↑	2,475455	0,051450	2,490909	2,290909	2,563636	2,372272	2,545455
Μέγιστος αριθμός μελών Ανεπαρκούς Αλληλεπίδρασης ανά ομάδα	max_insufficient_interaction_in_group	↓	7,000	0,000	7,000	7,000	7,000	7,000	7,000
Μέση απόκλιση κατανομής στρωμάτων	mean_strata_deviation	↓	0,563337	0,027476	0,562887	0,480052	0,617425	0,514962	0,613896
Μέγιστη απόκλιση κατανομής στρωμάτων	max_strata_deviation	↓	1,111113	0,135341	1,150317	0,878777	1,467363	0,883737	1,436031
Τυπική απόκλιση μεγέθους ομάδων	members_std	↓	0,187193	0,000000	0,187193	0,187193	0,187193	0,187193	0,187193
Τυπική απόκλιση μέσου σύνθετου προφίλ μεταξύ ομάδων	profile_mean_std	↓	0,380202	0,032796	0,381874	0,299884	0,476525	0,321598	0,449407
Τυπική απόκλιση μέσου κοινωνικού ρόλου μεταξύ ομάδων	social_role_mean_std	↓	0,387371	0,031103	0,387076	0,333323	0,489169	0,339946	0,456524
Τυπική απόκλιση μέσης επίδοσης μεταξύ ομάδων	performance_mean_std	↓	0,306510	0,025730	0,304990	0,246254	0,365512	0,258477	0,360239

Τυπική απόκλιση μέσης συμμετοχής μεταξύ ομάδων	engagement_mean_std	↓	0,224406	0,018070	0,226095	0,184862	0,266040	0,189119	0,257808
Τυπική απόκλιση μέσης απογοήτευσης μεταξύ ομάδων	frustration_mean_std	↓	0,375002	0,035067	0,374739	0,298055	0,457763	0,312269	0,439133

Πίνακας Β.7α. Εμπειρική κατανομή των 100 εκτελέσεων του Random baseline — ομάδες 7 ατόμων.

Σύγκριση στρατηγικών και εμπειρική σημαντικότητα — ομάδες 7 ατόμων

Μετρική	Raw variable	Κατ.	Performance-only	Proposed	r ίσο/καλύτερο	Proposed καλύτερη (runs)	Proposed καλύτερη (%)	Εμπειρικό p	p < 0,05
Ομάδες με τουλάχιστον έναν Ηγέτη	groups_with_leader	↑	16	18	10	90	90,00%	0,108911	Όχι
Ομάδες με τουλάχιστον μία Γέφυρα	groups_with_bridge	↑	7	7	66	34	34,00%	0,663366	Όχι
Ομάδες με τουλάχιστον έναν Δραστήριο Συμμετέχοντα	groups_with_active	↑	21	28	0	100	100,00%	0,009901	Ναι
Ομάδες με τουλάχιστον έναν Περιφερειακό Συμμετέχοντα	groups_with_peripheral	↑	37	54	0	100	100,00%	0,009901	Ναι
Ομάδες με ενεργό πυρήνα	groups_with_active_core	↑	35	53	0	100	100,00%	0,009901	Ναι
Ομάδες με τουλάχιστον ένα μέλος χωρίς Ανεπαρκή Αλληλεπίδραση	groups_with_non_insufficient	↑	48	55	0	100	100,00%	0,009901	Ναι
Μέση ποικιλία κοινωνικών ρόλων	mean_role_diversity	↑	2,472727	2,945455	0	100	100,00%	0,009901	Ναι
Μέγιστος αριθμός μελών Ανεπαρκούς Αλληλεπίδρασης ανά ομάδα	max_insufficient_interaction_in_group	↓	7	6	0	100	100,00%	0,009901	Ναι
Μέση απόκλιση κατανομής στρωμάτων	mean_strata_deviation	↓	0,791760	0,137088	0	100	100,00%	0,009901	Ναι
Μέγιστη απόκλιση κατανομής στρωμάτων	max_strata_deviation	↓	1,436031	0,570683	0	100	100,00%	0,009901	Ναι
Τυπική απόκλιση μεγέθους ομάδων	members_std	↓	0,187193	0,187193	100	0	0,00%	1,000000	Όχι
Τυπική απόκλιση μέσου σύνθετου προφίλ μεταξυ ομάδων	profile_mean_std	↓	0,602444	0,105683	0	100	100,00%	0,009901	Ναι
Τυπική απόκλιση μέσου κοινωνικού ρόλου μεταξυ ομάδων	social_role_mean_std	↓	0,387929	0,144156	0	100	100,00%	0,009901	Ναι
Τυπική απόκλιση μέσης επίδοσης μεταξύ ομάδων	performance_mean_std	↓	0,836295	0,143171	0	100	100,00%	0,009901	Ναι
Τυπική απόκλιση μέσης συμμετοχής μεταξύ ομάδων	engagement_mean_std	↓	0,246351	0,101990	0	100	100,00%	0,009901	Ναι

Τυπική απόκλιση μέσης απογοήτευσης μεταξύ ομάδων	frustration_mean_std	↓	0,400463	0,248768	0	100	100,00%	0,009901	Ναι
--	----------------------	---	----------	-----------------	---	-----	---------	-----------------	------------

Πίνακας Β.7β. Σύγκριση Performance-only και Proposed με την εμπειρική κατανομή του Random baseline — ομάδες 7 ατόμων.

Σημείωση: Το μέγεθος των ομάδων καθορίζεται εκ των προτέρων από τον αλγόριθμο. Για τον λόγο αυτό η μετρική members_std είναι σταθερή μεταξύ των στρατηγικών και δεν αποτελεί διαφοροποιητικό δείκτη ποιότητας. Διατηρείται στους πίνακες για λόγους πληρότητας και αναπαραγωγιμότητας.

Υπεύθυνη Δήλωση Συγγραφέα:

Δηλώνω ρητά ότι, σύμφωνα με το άρθρο 8 του Ν.1599/1986, η παρούσα εργασία αποτελεί αποκλειστικά προϊόν προσωπικής μου εργασίας, δεν προσβάλλει κάθε μορφής δικαιώματα διανοητικής ιδιοκτησίας, προσωπικότητας και προσωπικών δεδομένων τρίτων, δεν περιέχει έργα/εισφορές τρίτων για τα οποία απαιτείται άδεια των δημιουργών/δικαιούχων και δεν είναι προϊόν μερικής ή ολικής αντιγραφής, οι πηγές δε που χρησιμοποιήθηκαν περιορίζονται στις βιβλιογραφικές αναφορές και μόνον και πληρούν τους κανόνες της επιστημονικής παράθεσης.