ΕΛΛΗΝΙΚΟ ΑΝΟΙΚΤΟ ΠΑΝΕΠΙΣΤΗΜΙΟ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ ΚΑΙ ΤΕΧΝΟΛΟΓΙΑΣ

Μεταπτυχιακό Πρόγραμμα Σπουδών

"Μεταπτυχιακή Εξειδίκευση
στα Πληροφοριακά Συστήματα"

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Τίτλος:

"*Εφαρμογή σύγχρονων τεχνικών υπολογιστικής νοημοσύνης για την αποδοτική επίλυση του προβλήματος University Post Enrollment Course Timetabling (εύρεση βέλτιστου ωρολογίου προγράμματος μετά την αρχική δήλωση των μαθημάτων για τριτοβάθμια εκπαιδευτικά ιδρύματα). Εφαρμογή αλγόριθμου Particle Swarm Optimization (PSO) και υβριδικής παραλλαγής αυτού.*"

Ονοματεπώνυμο Φοιτητή:
Ιωάννης Α. Μαντούδης
Α.Μ: *73134*
E-mail:*gmantoudis@hol.gr*

Τηλ. Επικοινωνίας:
Τηλ: (+30)-6948892129
Τηλ: (+30)-2109639216

Ονοματεπώνυμο Επιβλέποντα Καθηγητή:
Γρηγόριος Ν. Μπεληγιάννης
E-mail: gbeligia@upatras.gr

ΑΘΗΝΑ 2013

"Εφαρμογή σύγχρονων τεχνικών υπολογιστικής νοημοσύνης για την αποδοτική επίλυση του προβλήματος University Post Enrollment Course Timetabling (εύρεση βέλτιστου ωρολογίου προγράμματος μετά την αρχική δήλωση των μαθημάτων για τριτοβάθμια εκπαιδευτικά ιδρύματα). Εφαρμογή αλγόριθμου Particle Swarm Optimization (PSO) και υβριδικής παραλλαγής αυτού."

ΙΩΑΝΝΗΣ Α. ΜΑΝΤΟΥΔΗΣ

Ονοματεπώνυμο Επιβλέποντα: Γρηγόριος Μπεληγιάννης

Ονοματεπώνυμο 1ου μέλους:

Ονοματεπώνυμο 2ου μέλους:

Περίληψη: Η παρούσα διπλωματική εργασία έχει ως αντικείμενο τη σχεδίαση και υλοποίηση ενός μεταευριστικού αλγόριθμου Υπολογιστικής Νοημοσύνης ο οποίος θα παράγει αυτόματα και με τη βοήθεια ηλεκτρονικού υπολογιστή, ωρολόγια προγράμματα που επιλύουν το πρόβλημα University Post Enrollment Course Timetabling (δηλαδή την κατασκευή ωρολογίου προγράμματος μετά την αρχική δήλωση των μαθημάτων από τους φοιτητές για τριτοβάθμια εκπαιδευτικά ιδρύματα). Ο αλγόριθμος που σχεδιάστηκε επιλύει το παραπάνω πρόβλημα ακολουθώντας τους περιορισμούς και τις προδιαγραφές του διεθνούς διαγωνισμού ITC 2007 – International Timetabling Competitionκαι πιο συγκεκριμένα της 2ης φάσης αυτού που σχετίζεται με το πρόβλημα PE – CCT.Ως εκ τούτου οι ελαστικοί και ανελαστικοί περιορισμοί του προβλήματος έχουν οριοθετηθεί από τους διοργανωτές του διαγωνισμού όπως επίσης και ο τρόπος αξιολόγησης των υποψήφιων λύσεων. Ο μεταευριστικός αλγόριθμος που χρησιμοποιήθηκε ως βάση για τον αλγόριθμο αυτής της διπλωματικής εργασίας είναι ο αλγόριθμος Particle Swarm Optimization (PSO), ο οποίος είναι ένας γνωστός αλγόριθμος στοχαστικής βελτιστοποίησης που ανήκει στην κατηγορία αλγορίθμων Υπολογιστικής Νοημοσύνης και ειδικότερα στην υποκατηγορία των αλγορίθμων Νοημοσύνης Σμήνους (Swarm Intelligence). Ο αλγόριθμος μας είναι ένας υβριδικός αλγόριθμος που βασίζεται στον PSO, αλλά έχει επιπλέον τροποποιηθεί και εμπλουτιστεί και με ιδέες από άλλους αλγορίθμους στοχαστικούς ή μη. Σκοπός μας ήταν η εξάλειψη ορισμένων εγγενών μειονεκτημάτων του κλασσικού PSO, ώστε να καταστεί δυνατόν να εξετάζει μεγαλύτερο φάσμα υποψηφίων λύσεων χωρίς να παγιδεύεται σε τοπικά βέλτιστα, να είναι πιο αποδοτικός και να παράγει στο προβλεπόμενο χρονικό

διάστημα ποιοτικότερες λύσεις, δηλαδή τελικώς λειτουργικότερα ωρολόγια προγράμματα.

Λέξεις κλειδιά: ωρολόγιο πρόγραμμα τριτοβάθμιων εκπαιδευτικών ιδρυμάτων, Post Enrollment Course Timetabling, PECCT, διαγωνισμός ITC 2007, Αλγόριθμοι βελτιστοποίησης σμήνους, Μεταευριστικοί αλγόριθμοι, Particle Swarm Optimization, PSO, Hybrid PSO, Υπολογιστική Νοημοσύνη, Μεταευρετικές μέθοδοι, Swarm Intelligence, Γραμμική και Μη-γραμμική βελτιστοποίηση.

"Use of Computational Intelligence techniques for the efficient solution of the Post Enrolment Based Timetabling problem (finding the optimal timetable after the initial statement of lessons for higher education institutions). Application of a hybrid Particle Swarm Optimization (PSO) algorithm."

IOANNIS A. MANTOUDIS

Name of Supervisor: GrigoriosBeligiannis

Name of 1st member:

Name of 2nd member:

Abstract: The main subject of the present thesis is the design and implementation of a Metaheuristic Computational Intelligence algorithm that produces efficient timetables, which can solve the University Post Enrolment Based Timetabling problem (that is producing efficient timetables after the initial statement of the lessons by higher education students). The algorithm we designed tackles the above problem following the instructions and regulations of the International Timetabling Competition (ITC – 2007) and more specifically the 2nd phase of this competition which deals with the Post Enrolment Based Timetabling problem. Therefore the hard and soft constraints imposed by the problem and the evaluation method of the solvers have been prescribed by the authors of this competition. The metaheuristic algorithm we used as the basis of our solver is the Particle Swarm Optimization (PSO) algorithm, a well-known stochastic optimization algorithm which belongs to the category of Computational Intelligence algorithms and more specific to the subcategory of Swarm Intelligence algorithms. Our algorithm is a hybrid algorithm that is based on PSO but furthermore it has been modified and enriched with ideas from other stochastic and non-stochastic algorithms. Our purpose was the elimination of some of the most profound and inherent limitations of the classic PSO in a way that it will be possible for the algorithm to examine more broad-based solutions without being trapped in local optima. As a result we have a more efficient algorithm that produces in the appropriate time limits better solutions, which in essence are more functional timetables.

Keywords: University – higher education timetables, Post Enrollment Course Timetabling, PECCT, International Timetabling Competition 2007,ITC 2007, Swarm

Intelligence Algorithms, Metaheuristic algorithms, Particle Swarm Optimization, PSO, Hybrid PSO, Computational Intelligence, Stochastic methods, Local Search Procedures, Linear and non-linear optimization.

Κεφάλαιο 1: Εισαγωγικές Έννοιες

Στο κεφάλαιο αυτό αναφέρονται και εισάγονται οι βασικές θεωρητικές έννοιες που εμπίπτουν στο αντικείμενο της παρούσας Διπλωματικής εργασίας και των εφαρμογών της. Αρχικά στην πρώτη ενότητα παρουσιάζεται η αναγκαιότητα δημιουργίας προγραμμάτων (timetables) σε διάφορες κατηγορίες οργανισμών και επιχειρήσεων που απασχολούν προσωπικό (εκπαιδευτικούς και μη), και οι ιδιαίτερα αυξημένες δυσκολίες που παρουσιάζονται κατά την κατάρτιση ενός τέτοιου προγράμματος από τον άνθρωπο. Στην δεύτερη ενότητα του κεφαλαίου γίνεται προσπάθεια να ενταχθεί το πρόβλημα στη θεματολογία συγκεκριμένων επιστημονικών πεδίων όπως της Επιχειρησιακής Έρευνας, του Μαθηματικού Προγραμματισμού, της Θεωρίας Αλγορίθμων και της Τεχνητής Νοημοσύνης.

Στην τρίτη ενότητα γίνεται η παρουσίαση του ακριβούς αντικειμένου που πραγματεύεται η συγκεκριμένη Διπλωματική Εργασία, το οποίο είναι η σχεδίαση και υλοποίηση ενός υβριδικού αλγόριθμου Υπολογιστικής Νοημοσύνης, που βασίζεται στον αλγόριθμο **PSO (Particle Swarm Optimization)** και αυτοματοποιεί τη διαδικασία δημιουργίας προγραμμάτων διδασκαλίας σε τριτοβάθμια εκπαιδευτικά ιδρύματα ΑΕΙ και ΤΕΙ, με τη χρήση Ηλεκτρονικού Υπολογιστή.

Τέλος στην ενότητα 1.4 παρουσιάζεται η διάρθρωση της Διπλωματικής Εργασίας με σύντομη ανάλυση του περιεχομένου και της δομής των 5 κεφαλαίων που την αποτελούν, καθώς επίσης των παραρτημάτων και της σχετικής βιβλιογραφίας που χρησιμοποιήθηκε.

1.1 Εφαρμογή των Ωρολογίων Προγραμμάτων σε Εκπαιδευτικούς ή άλλου είδους οργανισμούς

Ένας οργανισμός ή επιχείρηση που αποτελείται από υπαλλήλους με διάφορες ειδικότητες και εκτελεί εργασίες με συγκεκριμένα χρονοδιαγράμματα (ωράρια λειτουργίας, βάρδιες), πρέπει να οργανώνει τις εργασίες και τις υπηρεσίες του κατά τέτοιο τρόπο ώστε να διασφαλίζεται ανά πάσα στιγμή η εύρυθμη και απρόσκοπτη λειτουργία του.

Για να συμβεί αυτό θα πρέπει να εκπονείται και να τηρείται χρονολογικός προγραμματισμός που θα αναθέτει σε συγκεκριμένους χρόνους και στο κατάλληλο προσωπικό τις εργασίες που πρέπει να γίνουν. Με απλούστερα λόγια θα πρέπει να υπάρχει **Χρόνοπρογραμματισμός (Timescheduling)** όλων των εργασιών σε άμεση

συνάρτηση με την εξειδίκευση, τον αριθμό και την διαθεσιμότητα του προσωπικού. Η κατασκευή ενός τέτοιου προγράμματος χειρωνακτικά, στις περισσότερες των περιπτώσεων αποτελεί μια ιδιαίτερα δύσκολη υπόθεση. Σε ελάχιστες και ιδιαίτερα απλοϊκές περιπτώσεις μπορεί να γίνει μόνο από έναν άνθρωπο χωρίς τη χρήση του υπολογιστή, ακόμη όμως και τότε συχνά εμφανίζονται λάθη. Η δυσκολία στην κατάρτιση τέτοιων προγραμμάτων οφείλεται στην ύπαρξη πολλών συνιστωσών και απαιτήσεων που εισέρχονται στο πρόβλημα και πρέπει να ικανοποιηθούν ταυτόχρονα. Για να γίνει κατανοητή η δυσκολία σύνταξης ενός τέτοιου ωρολογίου προγράμματος θα εξετάσουμε έναν δημόσιο οργανισμό που λειτουργεί με βάρδιες όπως για παράδειγμα ένα πολυσύχναστο ΚΕΠ (Κέντρο Εξυπηρέτησης Πολιτών).Ο οργανισμός λειτουργεί καθημερινά σε δύο εξάωρες βάρδιες και μια επίσης εξάωρη βάρδια κάθε Σάββατο. Απασχολεί διοικητικό προσωπικό και προσωπικό εξυπηρέτησης των πολιτών. Το προσωπικό εξυπηρέτησης μοιράζεται σε ειδικότητες (πόστα) που συνήθως είναι περισσότερα των δύο (χωρίς αυτό να είναι δεσμευτικό για όλους τους συναφείς οργανισμούς, απλά το αναφέρουμε για την κατανόηση του παραδείγματος). Σε κάθε βάρδια κάθε πόστο πρέπει να έχει ένα ελάχιστο συγκεκριμένο αριθμό υπαλλήλων για να λειτουργεί αποτελεσματικά. Μερικοί υπάλληλοι έχουν την εκπαίδευση και τις ικανότητες να εργάζονται σε όλα τα πόστα, αυτό όμως δεν συμβαίνει με όλο το προσωπικό. Επειδή κάποια πόστα έχουν βαρύτερο φόρτο εργασίας είναι θεμιτό οι υπάλληλοι που απασχολούνται σε αυτά να δουλεύουν κυκλικά. Επίσης πρέπει να υπάρχει μεταξύ των υπαλλήλων εναλλαγή στην πρωινή και την απογευματινή βάρδια. Πολλοί υπάλληλοι για διαφορετικούς λόγους προτιμούν την $1^{η}$ βάρδια (πρωινή) ενώ άλλοι προτιμούν την $2^{η}$ βάρδια (απογευματινή). Αυτός είναι ένας άλλος παράγοντας που πρέπει να ληφθεί υπόψη κατά τη σύνταξη του προγράμματος. Τα πράγματα περιπλέκονται ακόμη περισσότερο όταν προκύπτουν απουσίες υπαλλήλων λόγω αδειών ή ασθένειας. Σε αυτή την περίπτωση οι διαθεσιμότητες των υπαλλήλων λειτουργούν σαν ανταγωνιστικές συνιστώσες και ο χρονοπρογραμματισμός των βαρδιών δυσκολεύει ακόμη περισσότερο. Γενικότερα όταν ο αριθμός των εργαζομένων πέσει κάτω από ένα ελάχιστο κατώφλι η λειτουργία του οργανισμού υπονομεύεται σημαντικά.

Ένα άλλο κλασσικό παράδειγμα που απαιτείται χρονοπρογραμματισμός είναι ένας εκπαιδευτικός οργανισμός, όπως για παράδειγμα σχολείο, κολλέγιο ή πανεπιστήμιο. Ένα τέτοιου είδους πρόγραμμα θα πρέπει να περιγράφει αναλυτικά τις ώρες που γίνεται η διδασκαλία μιας τάξης ή ενός τμήματος, τα μαθήματα που διδάσκονται,

τους καθηγητές που τα έχουν αναλάβει και τις αντίστοιχες αίθουσες. Πολλοί περιορισμοί και ανταγωνιστικές συνιστώσες μπορούν να προκύψουν κατά την κατάρτιση τέτοιων εκπαιδευτικών ωρολογίων προγραμμάτων. Συνήθως έχουν να κάνουν με την διαθεσιμότητα των τάξεων αλλά και των καθηγητών σε συγκεκριμένες ημέρες και ώρες, τους μαθητές – σπουδαστές, το είδος αλλά και τον τρόπο που διδάσκονται τα μαθήματα σε ότι έχει να κάνει με τις εβδομαδιαίες ώρες διδασκαλίας, τα εργαστήρια κλπ.

Σε περιπτώσεις όπως οι παραπάνω αλλά και σε άλλα προβλήματα χρονοπρογραμματισμού μέχρι πριν από λίγα χρόνια η κατάρτιση των προγραμμάτων γίνονταν από άνθρωπο, συνήθως από κάποιο έμπειρο υπάλληλο με διοικητικές ικανότητες. Η διαδικασία ήταν χρονοβόρα και επιρρεπής σε λάθη ενώ πολύ σπάνια το τελικό αποτέλεσμα ικανοποιούσε το σύνολο των υπαλλήλων. Πολλοί περιορισμοί δεν τηρούνταν προκειμένου όπως λέμε να «βγει» το πρόγραμμα. Το χειρότερο όλων ήταν ότι συχνά ο υπεύθυνος κατάρτισης του προγράμματος κατηγορούνταν για μεροληψία παρόλη την προσπάθεια που είχε καταβάλλει για την επίτευξη του βέλτιστου αποτελέσματος, αφού ένα πρόγραμμα με λάθη ήταν λογικό να ευνοεί περισσότερο κάποιους σε σχέση με άλλους συναδέλφους τους. Ευτυχώς η δυσάρεστη αυτή κατάσταση άλλαξε και συνεχίζει να αλλάζει προς το καλύτερο, με την είσοδο του ηλεκτρονικού υπολογιστή στους χώρους εργασίας. Η συνεχώς αυξανόμενη απόδοση τους αλλά και η βελτίωση των αλγορίθμων και των εφαρμογών που ασχολούνται με τον χρονοπρογραμματισμό, επιτρέπουν πλέον την αυτοματοποιημένη κατάρτιση ωρολογίων προγραμμάτων με ταχύτητα και σχεδόν βέλτιστο αποτέλεσμα. Να σημειωθεί εδώ ότι τέτοιου είδους εφαρμογές άργησαν να μπουν στον εργασιακό χώρο στην Ελλάδα και σε άλλες χώρες, σε σχέση με ποιο παραδοσιακές εργασίες όπως η Επεξεργασία Κειμένου, τα Λογιστικά Φύλλα και οι Βάσεις Δεδομένων, πιθανότητα επειδή υπήρχε άγνοια για τέτοιου είδους εφαρμογές αλλά και γιατί η αντίστοιχη ανάπτυξη αλγορίθμων δεν ήταν στο επιθυμητό επίπεδο. Είναι γεγονός πάντως πως τα τελευταία χρόνια υπάρχει σημαντική έρευνα από πολλούς επιστήμονες στον τομέα αυτό, πράγμα που έχει οδηγήσει τους ερευνητές και τους προγραμματιστές στην κατασκευή αλγορίθμων με εξαιρετική ποιότητα αποτελεσμάτων και μικρούς χρόνους εκτέλεσης. Υπάρχουν fora στο διαδίκτυο αλλά και ειδικά επιστημονικά περιοδικά που ασχολούνται με το αντικείμενο ενώ παράλληλα οργανώνονται διεθνή συνέδρια, εκθέσεις και διαγωνισμοί με το θέμα της μελέτης και κατασκευής τέτοιου είδους αλγορίθμων.

1. 2 Η Επιστημονική Κατηγοριοποίηση του προβλήματος

Γενικά το πρόβλημα της κατασκευής Ωρολογίων προγραμμάτων που πρέπει να πληρούν πολλές συνθήκες και περιορισμούς είναι ένα πρόβλημα βελτιστοποίησης. Ένα πρόγραμμα βελτιστοποίησης δεν δέχεται μονοσήμαντα μια λύση αλλά μπορεί να έχει πολλές αποδεκτές λύσεις, σε καθεμία από τις οποίες αντιστοιχεί μια τιμή. Έτσι το ζητούμενο σε αυτές τις περιπτώσεις είναι η εύρεση της βέλτιστης λύσης (μέγιστης ή ελάχιστης), από το σύνολο των λύσεων που ικανοποιεί τις απαιτήσεις μας. Ο ευρύτερος επιστημονικός κλάδος που μελετά τέτοιου είδους προβλήματα, είναι ο κλάδος των Εφαρμοσμένων Μαθηματικών που ονομάζεται Μαθηματικός Προγραμματισμός (Mathematical Programming) ή εξίσου συχνά Επιχειρησιακή Έρευνα (Operational Research). Ο κλάδος αυτός αναπτύχθηκε σημαντικά μετά το 1950 ακριβώς γιατί εκείνη την εποχή άρχισαν να εξελίσσονται ραγδαία οι πρώτοι ηλεκτρονικοί υπολογιστές. Επίσης άλλα επιστημονικά πεδία και υποπεδία που συμβάλλουν σημαντικά από τον ευρύτερο κλάδο της Πληροφορικής επιστήμης είναι και τα εξής:

- Θεωρία Αλγορίθμων και Πολυπλοκότητα (Algorithm Theory & Complexity)
- Τεχνητή Νοημοσύνη (Artificial Intelligence)
- Υπολογιστική Νοημοσύνη (Computational Intelligence)

Η Θεωρία Αλγορίθμων (Algorithm Theory) είναι μια κλασσική περιοχή στην Επιστήμη των Ηλεκτρονικών Υπολογιστών και ερευνά τον σχεδιασμό και την υλοποίηση αποδοτικών αλγορίθμων για την επίλυση υπολογιστικών προβλημάτων. Συγκεκριμένα θα εστιάσουμε από πλευράς Αλγοριθμικής στη μέθοδο του Δυναμικού Προγραμματισμού η οποία χρησιμοποιείται συχνά σε προβλήματα βελτιστοποίησης όπως αυτά που πραγματευόμαστε στην παρούσα εργασία.

Η Τεχνητή Νοημοσύνη (Artificial Intelligence) είναι η περιοχή της Πληροφορικής Επιστήμης που μελετά και σχεδιάζει υπολογιστικά συστήματα τα οποία επιλύουν δύσκολα προβλήματα που δεν μπορούν να επιλυθούν με την εξαντλητική εξέταση όλων των πιθανών λύσεων μια και αυτές μπορεί να είναι πάρα πολλές. Ο όρος εισήχθηκε πρώτα από τον John McCarthy το 1956 και ο ίδιος όρισε την Τεχνητή Νοημοσύνη ως τον κλάδο της Πληροφορικής που ασχολείται με την κατασκευή ευφυών μηχανών.

Η Υπολογιστική Νοημοσύνη (Computational Intelligence) αποτελεί υποκατηγορία της Τεχνητής Νοημοσύνης. Στηρίζεται σε Ευρετικούς Αλγόριθμους (Heuristic Algorithms), δηλαδή αλγόριθμους που επιλύουν σε λογικό χρόνο και με λογική κατανάλωση υπολογιστικών πόρων πρακτικά προβλήματα, χρησιμοποιώντας όμως συγκεκριμένα στιγμιότυπα δεδομένων εισόδου. Μερικοί τέτοιοι αλγόριθμοι είναι τα Ασαφή Συστήματα (Fuzzy Systems), τα Τεχνητά Νευρωνικά Δίκτυα (Artificial Neural Networks) και ο Εξελικτικός Προγραμματισμός (Evolutionary Computation).Οι αλγόριθμοι Υπολογιστικής Νοημοσύνης είναι σχεδιασμένοι ώστε να επιδεικνύουν κάποιο είδος ευφυΐας και καταφέρνουν να το πετύχουν αυτό συνδυάζοντας τεχνικές μάθησης, προσαρμογής και εξέλιξης. Μεγάλος αριθμός τέτοιων αλγορίθμων έχει επινοηθεί για την επίλυση προβλημάτων βελτιστοποίησης όπως αυτό που πραγματευόμαστε σε αυτή την εργασία. Παρακάτω αναφερόμαστε σε μερικούς από αυτούς παραθέτοντας παράλληλα και τα φυσικά φαινόμενα στα οποία στηρίζεται η λογική της κατασκευής τους.

- **Particle Swarm Optimization.** Πρόκειται για αλγόριθμο βελτιστοποίησης ο οποίος αναζητά τις βέλτιστες λύσεις του προβλήματος προσομοιώνοντας τη χορογραφία κινήσεων ενός σμήνους πτηνών που αναζητά την τροφή του. Προτάθηκε αρχικά από τους Kennedy, Eberhart και Shi.
- **Artificial Bee Colony.** Είναι αλγόριθμος βελτιστοποίησης που προσομοιώνει τον τρόπο που ένα σμήνος από μέλισσες αναζητά τροφή. Προτάθηκε το 2005 από τον Karaboga.
- **Ant Colony Optimization.**Όμοια με τα παραπάνω και εδώ έχουμε έναν μεταευριστικό αλγόριθμο βελτιστοποίησης που αναζητά βέλτιστες διαδρομές σε ένα γράφο, προσομοιώνοντας τη συμπεριφορά μιας αποικίας μυρμηγκιών καθώς αναζητούν την τροφή τους. Προτάθηκε το 1992 από τον Marco Rodrigo.
- **Simulated Annealing.** Είναι ένας μεταευριστικός αλγόριθμος στοχαστικής βελτιστοποίησης που έχει επηρεαστεί από την διαδικασία πυρώματος - ξεπυρώματος στην μεταλλουργία. Προτάθηκε στην αρχική του μορφή από τους Kirkpatrick, Gelatt και Vecchi το 1983.
- **Bacterial Foraging Optimization.** Ανήκει και αυτός στην κατηγορία τον στοχαστικών μεταευριστικών αλγόριθμων με κοινά στοιχεία με άλλους αλγόριθμους σμήνους όπως οι PSO, ABC που αναφέρθηκαν παραπάνω. Προσο-

μοιώνει τον τρόπο με τον οποίο μια καλλιέργεια βακτηρίων αναζητά τροφή μέσω χημικών ερεθισμάτων. Παρουσιάστηκε το 2002 από τον Passino.

1.3 Το Αντικείμενο της Διπλωματικής Εργασίας

Ο καταρτισμός ενός ωρολογίου προγράμματος για τις διδασκαλίες, τα εργαστήρια, τα σεμινάρια και τα επιβοηθητικά φροντιστήρια σε ένα τριτοβάθμιο εκπαιδευτικό ίδρυμα, κατά τέτοιο τρόπο ώστε να ικανοποιεί όλες τις απαιτήσεις καθηγητών και σπουδαστών είναι ένα δύσκολο πρόβλημα βελτιστοποίησης. Απαιτείται η ικανοποίηση πολλών περιορισμών προκειμένου το παραγόμενο πρόγραμμα να ικανοποιεί τους χρήστες του, έτσι ώστε να μην φορτώνονται με καθήκοντα σε τέτοιο βαθμό που να τους εμποδίζει να ρυθμίζουν ικανοποιητικά και τις μη – εκπαιδευτικές δραστηριότητες της ζωής τους.

Το παραπάνω πρόβλημα χρονοπρογραμματισμού διαφέρει βέβαια από ίδρυμα σε ίδρυμα ακόμη και εάν μιλάμε για εκπαιδευτικά ιδρύματα στο ίδιο κράτος, όμως σε κάθε περίπτωση οι βασικές παράμετροι που εμπλέκονται είναι κοινές. Τέτοιες παράμετροι είναι οι αίθουσες διδασκαλίας, τα εργαστήρια, τα αμφιθέατρα, ο αριθμός των φοιτητών ανά έτος και ανά μάθημα, ο αριθμός και η ειδικότητα των καθηγητών και τα διδασκόμενα μαθήματα. Στην γενική του περίπτωση το πρόβλημα που εξετάζουμε αναφέρεται σε ένα αριθμό γεγονότων (διδασκαλίες, σεμινάρια) που θα πρέπει να λάβουν χώρα σε συγκεκριμένες χρονικές περιόδους και σε συγκεκριμένους χώρους (αίθουσες, εργαστήρια) πληρώντας έναν αριθμό περιορισμών.

Οι περιορισμοί αυτοί κατηγοριοποιούνται σε δύο ομάδες:

- **Ανελαστικοί Περιορισμοί (Hard Constraints).** Πρόκειται για περιορισμούς που πρέπει να τηρούνται υποχρεωτικά προκειμένου το ωρολόγιο πρόγραμμα που προκύπτει να είναι εφικτό (feasible).
- **Ελαστικοί Περιορισμοί (Soft Constraints).** Πρόκειται για περιορισμούς που πρέπει να τηρούνται «εάν αυτό είναι δυνατόν» προκειμένου το παραγόμενο ωρολόγιο πρόγραμμα να είναι όσο περισσότερο λειτουργικό γίνεται για τους χρήστες του.

Μία αναλυτική παρουσίαση για τα παραπάνω δύο είδη περιορισμών θα γίνει στο 3^{ο} Κεφάλαιο όταν θα περιγράψουμε αναλυτικά το πρόβλημα του Post – Enrollment Course Timetable που παρουσιάζεται σε αυτή την εργασία.

Από τα παραπάνω γίνεται σαφές ότι η δημιουργία ενός ποιοτικού ωρολογίου προγράμματος είναι μια δύσκολη και χρονοβόρα εργασία που δύσκολα θα μπορούσε να διεκπεραιωθεί από κάποιο μεμονωμένο άτομο χωρίς επιπλέον βοήθεια. Η χρήση του υπολογιστή είναι απαραίτητη προϋπόθεση για την αποδοτική επίλυση αυτής της διαδικασίας, αρκεί βέβαια να διαθέτουμε τον κατάλληλο αλγόριθμο επίλυσης του προβλήματος και το αντίστοιχο λογισμικό. Πράγματι όπως αναφέραμε στην προηγούμενη ενότητα έχουν επινοηθεί αρκετοί τέτοιου είδους ευριστικοί και μεταευριστικοί αλγόριθμοι και μέθοδοι, που μπορούν να επιλύσουν το πρόβλημα με σχεδόν άριστο τρόπο. Η ποιότητα των παραγόμενων ωρολογίων προγραμμάτων μπορεί να είναι μετρήσιμη ώστε να υπάρχει σύγκριση μεταξύ των διαφορετικών αλγορίθμων.

Στην παρούσα διπλωματική εργασία ο σκοπός μας είναι η σχεδίαση ενός μεταευριστικού αλγόριθμου που θα επιλύει το πρόβλημα του Post Enrollment Course Timetabling σε τριτοβάθμια εκπαιδευτικά ιδρύματα, δηλαδή της κατασκευής ωρολογίου προγράμματος μετά την δήλωση των μαθημάτων κάθε εξαμήνου από τους ενδιαφερόμενους φοιτητές. Αναφορικά με το πρόβλημα αυτό και τους περιορισμούς του ο αλγόριθμος ικανοποιεί τους περιορισμούς έτσι όπως αυτή τέθηκαν από το διαγωνισμό της δεύτερης φάσης του 2nd International Timetabling Competition 2007 – 2008 αναλυτική παρουσίαση του οποίου θα γίνει στο 3ο κεφάλαιο. Περισσότερες πληροφορίες για το διαγωνισμό αυτό όπως επίσης και τα αρχεία εισόδου (input instances) που χρησιμοποιούνται από τον αλγόριθμο μας μπορούν να βρεθούν στην ιστοσελίδα του διαγωνισμού (www.cs.qub.ac.uk/itc2007/).

1. 4 Η Δομή της Διπλωματικής Εργασίας

Η παρούσα διπλωματική εργασία αποτελείται από 6 κεφάλαια, καθώς και από τα παραρτήματα και την αντίστοιχη βιβλιογραφία.

Στο 1ο κεφάλαιο παρουσιάζονται γενικά στοιχεία για το αντικείμενο της εργασίας και το πρακτικό πρόβλημα της κατασκευής ωρολογίων προγραμμάτων (Timetabling). Τίθεται επίσης και κατηγοριοποιείται το παρόν πρόβλημα στον επιστημονικό κλάδο στον οποίο ανήκει που είναι κατά βάση η Επιχειρησιακή Έρευνα και ο Γραμμικός Προγραμματισμός. Φυσικά στην επίλυση του προβλήματος πολύ σημαντική είναι η συμβολή κλάδων της Πληροφορικής όπως η Αλγοριθμική, η Υπολογιστική πολυπλοκότητα και η Τεχνητή Νοημοσύνη. Αναφέρονται επίσης επιγραμματικά διάφοροι αλγόριθμοι που ανήκουν στον κλάδο της Υπολογιστικής Νοημοσύνης που ανήκουν

στην κατηγορία των Μεταευριστικών αλγόριθμων και έχουν χρησιμοποιηθεί για την επίλυση του ίδιου προβλήματος αλλά και άλλων παρόμοιων προβλημάτων.

Στο 2^{ο} κεφάλαιο αναλύεται περισσότερο το πρόβλημα του Χρονοπρογραμματισμού και της κατασκευής ωρολογίων προγραμμάτων μαζί με μερικές από τις πιο κλασσικές εφαρμογές του. Μερικές από αυτές είναι το πρόβλημα του UniversityCourse Timetabling, το Nurse Rostering, το Bus Scheduling καθώς και το School Timetabling για την κατασκευή ωρολογίων προγραμμάτων για σχολεία δευτεροβάθμιας εκπαίδευσης. Στο κεφάλαιο αυτό γίνεται επίσης μια εισαγωγή στην NP – Πληρότητα και στην κλάση των NP – Complete προβλημάτων όπου ανήκει και ο αλγόριθμος της παρούσης εργασίας.

Στο 3^{ο} κεφάλαιο αναπτύσσεται αναλυτικά το πρόβλημα του Post Enrolment Course Timetabling (PE – CTT) και εξειδικεύεται στους κανόνες και τους περιορισμούς που είχαν τεθεί στον 2^{ο} διαγωνισμό International Timetabling Competition (ITC 2007). Παρουσιάζονται οι περιορισμοί, οι προδιαγραφές και ο τρόπος αξιολόγησης των υποψηφίων αλγορίθμων έτσι όπως αυτοί είχαν οριοθετηθεί από τους διοργανωτές του διαγωνισμού καθώς και μερικές από τις πιο αξιόλογες προτάσεις που συμμετείχαν.

Στο 4^{ο}κεφάλαιο παρουσιάζεται περιληπτικά η έννοια του Μεταευριστικού αλγόριθμου και αναλυτικότερα ο αλγόριθμος Particle Swarm Optimization που ανήκει σε αυτή την κατηγορία. Πρώτα αναλύεται ο αλγόριθμος στην κλασσική του μορφή με ταυτόχρονη επισκόπηση των βασικών του παραμέτρων και των ρυθμίσεων που αυτές επιδέχονται και στη συνέχεια γίνεται αναφορά στις υβριδικές παραλλαγές του αλγόριθμου που μπορούν να κατασκευάσουμε προκειμένου να βελτιώσουμε τις επιδόσεις του.

Στο 5^{ο} κεφάλαιο παρουσιάζεται αναλυτικά ο αλγόριθμος που επιλύει το πρόβλημα της παρούσης διπλωματικής εργασίας. Πρώτα δίνεται η κωδικοποίηση των particlesμε τρόπο που να συμβαδίζει με το υπό εξέταση πρόβλημα του PE – CCT. Στη συνέχεια δίνεται σε μορφή ψευδοκώδικα ο αλγόριθμος αρχικοποίησης και παρουσιάζεται αναλυτικά η αντικειμενική συνάρτηση που θα πρέπει να ελαχιστοποιηθεί για να έχουμε βέλτιστη λύση. Τέλος δίνεται ο ψευδοκώδικας για τον αλγόριθμο Local Search που χρησιμοποιείται για την εκλέπτυνση της λύσης μετατρέποντας τον αλγόριθμο μας σε υβριδικό PSO.

Κεφάλαιο 2:Χρονοπρογραμματισμός (TimeScheduling)

Στο 2^{ο} κεφάλαιο της παρούσας εργασίας παρουσιάζεται το γενικότερο πρόβλημα του Χρονοπρογραμματισμού (Timetabling – Scheduling) καθώς και μερικές από τι πιο σημαντικές εφαρμογές του. Το πρόβλημα αυτό κατατάσσεται στην ευρεία κλάση των «δύσκολων» υπολογιστικών προβλημάτων που ονομάζονται NP–Πλήρη προβλήματα (NP – Complete). Έτσι αρχικά στην ενότητα 2.1 δίνεται μια απλή περιγραφή της NP–Πληρότητας, χωρίς όμως να εισερχόμαστε σε τεχνικές λεπτομέρειες που ξεφεύγουν από τον σκοπό της παρούσης εργασίας.

Στις επόμενες ενότητες παρουσιάζονται κάποιες κλασσικές εργασίες που απαιτούν χρονοπρογραμματισμό και συγκεκριμένα στην ενότητα 2.2 εξετάζεται ο προγραμματισμός των δρομολογίων ενός οργανισμού συγκοινωνιών (Bus Scheduling), στην ενότητα 2.3 ο προγραμματισμός των βαρδιών ενός νοσοκομείου (Nurse Rostering), στην ενότητα 2.4 ο προγραμματισμός των διδασκαλιών αναφορικά με τις τάξεις και τους καθηγητές σε ένα σχολείο δευτεροβάθμιας εκπαίδευση (School Timetabling) και τέλος στην ενότητα 2.5 ο προγραμματισμός και η κατασκευή ενός προγράμματος εξετάσεων για τριτοβάθμια εκπαιδευτικά ιδρύματα.

Στο τρίτο κεφάλαιο της εργασίας θα παρουσιάσουμε αναλυτικά το πρόβλημα του Post Enrollment University Course Timetabling, το βέλτιστο προγραμματισμό του οποίου θα αντιμετωπίσει ο αλγόριθμος που πρόκειται να αναπτύξουμε.

2.1 Ντετερμινιστικοί Αλγόριθμοι

Ντετερμινιστικός Αλγόριθμος (Deterministic Algorithm) ονομάζεται ο αλγόριθμος που δεν χρησιμοποιεί τυχαιότητα – στοχαστικότητα στις επιλογές που κάνει προς εξεύρεση μιας λύσης. Σύμφωνα με τη βιβλιογραφία δεν υπάρχει ντετερμινιστικός αλγόριθμος που να επιλύει ικανοποιητικά το πρόβλημα του University Course Timetabling και κατά συνέπεια και τα υποπροβλήματα αυτού όπως το PE–CCT, αφού έχει αποδειχθεί πως πρόκειται για ένα NP–πλήρες πρόβλημα βελτιστοποίησης ειδικά όταν έχουμε μεγάλα στιγμιότυπα δεδομένων [21], [22]. Έτσι όπως σε όλα τα NP – πλήρη προβλήματα ισχύει και για το PE–CCT ότι ο χώρος λύσεων είναι τεράστιος για να μπορέσει να διερευνηθεί ικανοποιητικά σε πολυωνυμικό χρόνο (polynomial-time) από έναν ντετερμινιστικό αλγόριθμο που εκτελεί εξαντλητικές αναζητήσεις.

2.2 NP – Πληρότητα και η κλάση των NP – Completeπροβλημάτων

Στο επιστημονικό πεδίο της Αλγοριθμικής επίλυσης προβλημάτων η κλάση των NP–Complete προβλημάτων είναι πιθανόν η πιο γνωστή κλάση υπολογιστικών προβλημάτων με ανοιχτά και αναπάντητα ερωτήματα που απασχολούν για χρόνια την επιστημονική κοινότητα της Πληροφορικής και πιο συγκεκριμένα τους κλάδους της Αλγοριθμικής και της Υπολογιστικής Πολυπλοκότητας. Πρόκειται για μια κλάση ιδιαίτερα δύσκολων υπολογιστικών προβλημάτων, που παρόλο που δεν έχει αποδειχθεί ότι δεν μπορούν να λυθούν σε πολυωνυμικό χρόνο, δεν έχει βρεθεί ντετερμινιστικός αλγόριθμος που να τα επιλύει σε λογικό χρόνο και με λογικό υπολογιστικό κόστος. Έτσι ο μηχανικός λογισμικού έχοντας αυτή τη δυσκολία κατά νου, προσπαθεί να αναπτύξει ευριστικούς και μεταευριστικούς αλγόριθμους όπως ο PSO και άλλοι που αναφέρθηκαν στο προηγούμενο κεφάλαιο, για την επίλυση τέτοιων προβλημάτων σε λογικό χρόνο, ή να επιλύσει συγκεκριμένα στιγμιότυπα αυτών των προβλημάτων που πληρούν κάποια κριτήρια που έχει θέσει. Επίσης θα πρέπει να πούμε ότι από αλγοριθμική σκοπιά τα προβλήματα χρονοπρογραμματισμού και κατασκευής ωρολογίων προγραμμάτων, ανάγονται στο πρόβλημα του χρωματισμού ενός γράφου (graph coloring), ένα κλασσικό αλγοριθμικό πρόβλημα που έχει μελετηθεί διεξοδικά και ανήκει στην κλάση των NP – Complete προβλημάτων.

Για μια ευρύτερη ανάλυση αυτής της σημαντικής κλάσης υπολογιστικών προβλημάτων και των επιμέρους εφαρμογών της, ο αναγνώστης μπορεί να αναφερθεί στην προτεινόμενη βιβλιογραφία [4], [22].

2.3Προγραμματισμός δρομολογίων σε οργανισμό συγκοινωνιών (BusScheduling&Rostering)

Το πρόβλημα του Bus Schedulingείναι ένα NP – Completeπρόβλημα που μπορεί να επιλυθεί με μεταευριστικούς αλγόριθμους βελτιστοποίησης όπως για παράδειγμα είναι και ο PSO. Σκοπός εδώ είναι να κατασκευαστούν δίκτυα αστικών δρομολογίων που πληρούν διάφορους περιορισμούς, που μπορεί να αφορούν τον αριθμό και τον τύπο των διαθέσιμων λεωφορείων, την ελαχιστοποίηση του κόστους των δρομολογίων σε συνδυασμό με τη μείωση του χρόνου αναμονής και γενικότερα με την αύξηση της ικανοποίησης των επιβατών που χρησιμοποιούν τα συγκεκριμένα λεωφορεία.

2.4 Προγραμματισμός λειτουργίας και βαρδιών προσωπικού σε Νοσηλευτικό Ίδρυμα (Nurse Rostering)

Άλλο ένα πρόβλημα που ανήκει στην κλάση των NP – Completeπροβλημάτων είναι και το Nurse Rostering. Πρόκειται για ένα σύνθετο πρόβλημα χρονοπρογραμματισμού όπου πρέπει να κατασκευαστεί ένα πρόγραμμα βαρδιών προσωπικού για τη λειτουργία ενός νοσοκομείου που θα πληροί πολλούς ελαστικούς και ανελαστικούς περιορισμούς. Συγκεκριμένα το πρόγραμμα που θα κατασκευαστεί θα πρέπει να αναθέτει όλες τις διαφορετικές εργασίες ενός νοσηλευτικού ιδρύματος στα κατάλληλα άτομα που είναι διαθέσιμα σε μια δεδομένη χρονική στιγμή. Στόχος είναι η ικανοποίηση όλων των ανελαστικών λειτουργικών αναγκών του νοσοκομείου και των νοσηλευομένων, με ταυτόχρονη ικανοποίηση όσο των δυνατών περισσοτέρων ελαστικών περιορισμών και προτιμήσεων που τίθενται από το ιατρικό και το νοσηλευτικό προσωπικό. Για μια αναλυτικότερη και σε βάθος παρουσίαση του προβλήματος ο αναγνώστης προτρέπεται στην βιβλιογραφία [5].

2.5 Προγραμματισμός διδασκαλιών σε σχολεία (School Timetabling)

Είναι και αυτό πρόβλημα βελτιστοποίησης που ανήκει στην κλάση των NP – Completeπροβλημάτων με πολλές παραμέτρους και λύσεις που θα πρέπει να ικανοποιούν πολλούς περιορισμούς είτε στην ολότητα τους είτε μερικώς. Πιο συγκεκριμένα οι οντότητες που συμπλέκονται στο πρόβλημα αυτό είναι οι καθηγητές, τα τμήματα, οι διαθέσιμες τάξεις και το καθημερινό ωράριο μαθημάτων. Οι καθηγητές διδάσκουν μαθήματα σε συγκεκριμένες τάξεις και τμήματα και σε συγκεκριμένα ωράρια και οι ανελαστικοί αλλά και ελαστικοί περιορισμοί που προκύπτουν είναι πάρα πολλοί, προκειμένου να δημιουργηθεί ένα εφικτό αλλά και ποιοτικό πρόγραμμα που θα ικανοποιεί όσο το δυνατόν περισσότερο τους εμπλεκόμενους στο πρόβλημα. Αναλυτική μελέτη του προβλήματος αυτού μπορείτε να διαβάσετε στη βιβλιογραφία [2].

2.6Προγραμματισμός εξετάσεων σε εκπαιδευτικά ιδρύματα (Exams Timetabling)

Ο χρονοπρογραμματισμός εξετάσεων (Exams Timetabling) είναι και αυτό ένα πρόβλημα που ανήκει στην κλάση των NP – Complete προβλημάτων. Το πρόβλημα αφορά την αντιστοίχηση των εξεταζομένων μαθημάτων ενός τριτοβάθμιου εκπαιδευτικού ιδρύματος σε συγκεκριμένες χρονικές περιόδους και αίθουσες. Το παραγόμενο

πρόγραμμα εξετάσεων θα πρέπει να πληροί μια σειρά από ελαστικούς και ανελαστικούς περιορισμούς που έχουν να κάνουν με τα εξεταζόμενα μαθήματα, τις κατάλληλες αίθουσες και τα εργαστήρια από πλευράς εξοπλισμού και χωρητικότητας που θα λάβει χώρα η εξέταση, όπως επίσης τη διαθεσιμότητα των καθηγητών και τις προτιμήσεις που υπάρχουν σχετικά με την κατανομή των εξεταζομένων μαθημάτων στην εξεταστική περίοδο. [15] Το λογισμικό UniTime 3.2 διατίθεται δωρεάν στο διαδίκτυο και μπορεί να επιλύσει προβλήματα όπως το Exams Timetabling αλλά και το Post Enrollment Course Timetabling που πραγματεύεται αυτή η εργασία.

Κεφάλαιο 3: Επισκόπηση και Ανάλυση του προβλήματος Post Enrolment Based Course Timetabling

Στο παρόν κεφάλαιο θα αναπτυχθεί και θα καταγραφεί το πρόβλημα της κατασκευής ενός ωρολογίου προγράμματος που θα αφορά τα μαθήματα, τις διδασκαλίες και τις εξετάσεις σε ένα τριτοβάθμιο εκπαιδευτικό ίδρυμα. Συγκεκριμένα στην ενότητα 3.1 καταγράφεται το πρόβλημα στη γενικότερη του μορφή με τις υποπεριπτώσεις του και θέματα που αφορούν τα διάφορα πανεπιστημιακά ιδρύματα, ενώ στην ενότητα 3.2 το πρόβλημα εξειδικεύεται στο κομμάτι του Post Enrolment Based Course Timetabling (PE-CTT) με τους κανόνες και τους περιορισμούς που οριοθετήθηκαν από τον $2^{ο}$ διαγωνισμό International Timetabling Competition (ITC) που έλαβε χώρα το 2007 (www.cs.qub.ac.uk/itc2007). Στην ενότητα 3.3 περιγράφεται ο τρόπος αξιολόγησης των αλγόριθμων που συμμετείχαν στο διαγωνισμό σύμφωνα με τους κανόνες που έθεσαν οι διοργανωτές. Στην ενότητα 3.4 αναφέρουμε κάποιους από τους περιορισμούς του διαγωνισμού ITC 2007 για το μοντέλο PE – CCT, μαζί με επιπλέον περιορισμούς που είναι δυνατόν να εμφανιστούν στην πράξη. Τέλος στην ενότητα 3.5 αναφέρονται μερικές από τις πιο αξιόλογες προσπάθειες επίλυσης του προβλήματος που συμμετείχαν στον διαγωνισμό, τα αποτελέσματα των οποίων θα συγκριθούν με τα αποτελέσματα του δικού μας αλγόριθμου.

3.1 Εισαγωγή στον Χρονοπρογραμματισμό των γεγονότων ενός Τριτοβάθμιου Εκπαιδευτικού Ιδρύματος

Ο χρονοπρογραμματισμός και η δημιουργία ωρολογίων προγραμμάτων για τις διδασκαλίες, τις διαλέξεις, τα εργαστήρια και τα φροντιστηριακά μαθήματα ενός πανεπιστημιακού τμήματος, είναι ένα δύσκολο NP – Complete πρόβλημα που παρουσιάζει μεγάλη συνάφεια με τον χρονοπρογραμματισμό (Timetabling Schedule) του ωρολογίου προγράμματος ενός σχολείου δευτεροβάθμιας εκπαίδευσης. Σε όλα αυτά τα προβλήματα ο σκοπός δεν είναι μόνο η παραγωγή ενός προγράμματος που θα πληροί τις ελάχιστες απαιτήσεις λειτουργικότητας που τίθενται από το ίδρυμα, αλλά ενός «ποιοτικού» ωρολογίου προγράμματος που θα είναι λειτουργικό για όλους τους εμπλεκόμενους (καθηγητές, σπουδαστές, διοικητικό προσωπικό). Με άλλα λόγια μας ενδιαφέρει η παραγωγή ενός προγράμματος που θα μπορεί να χρησιμοποιηθεί από όλους τους ενδιαφερομένους, χωρίς να φορτώνει υπερβολικά με δραστηριότητες και εκπαιδευτικά γεγονότα τους χρήστες του στο σύνολο η μεμονωμένα.

Το πρόβλημα βέβαια θα ήταν σχεδόν απίθανο να επιλυθεί καθολικά σε ευρύ φάσμα εκπαιδευτικών ιδρυμάτων, αφού από τη φύση του εξαρτάται από τη χώρα, τις συνήθειες των χρηστών, το πανεπιστημιακό ίδρυμα και τις ιδιαιτερότητες του τμήματος στο οποίο εφαρμόζεται. Σε κάθε περίπτωση πάντως, για να επιτευχθεί η κατάρτιση ενός τέτοιου προγράμματος θα πρέπει να πληρούνται κάποιοι **Ανελαστικοί** περιορισμοί (**Hard Constraints**) και κάποιοι **Ελαστικοί** περιορισμοί (**Soft Constraints**) που έχουν τεθεί και είναι γνωστοί εξαρχής.

Οι Ανελαστικοί περιορισμοί πρέπει να πληρούνται υποχρεωτικά προκειμένου το παραγόμενο πρόγραμμα να είναι **Εφικτό (Feasible)** και να προσφέρει τι minimum λειτουργικότητα, ενώ οι Ελαστικοί περιορισμοί θα πρέπει να πληρούνται σε όσο το μεγαλύτερο βαθμό είναι αυτό δυνατό και εφικτό, προκειμένου το πρόγραμμα να είναι ποιοτικό, χρηστικό και λειτουργικό για τους χρήστες του.

Το πρόβλημα μπορεί επίσης να χωρισθεί σε υποπροβλήματα που χρήζουν ξεχωριστής αντιμετώπισης, αφού το κάθε ένα από αυτά παρουσιάζει ξεχωριστές δυσκολίες. Έτσι το βασικό πρόβλημα του University Course Timetabling μπορεί να χωριστεί στα παρακάτω υποπροβλήματα:

- **Curriculum Based Course Timetabling (CB-CTT).**Αναφέρεται στην κατάρτιση προγράμματος που αφορά καθηγητές, μαθητές και αίθουσες σε σχέση με τα εκπαιδευτικά προγράμματα που προσφέρει το πανεπιστήμιο στους φοιτητές του και είναι γνωστά εξαρχής.
- **Post Enrolment Based Course Timetabling (PE-CTT).** Σε αντίθεση με την προηγούμενη περίπτωση το παραγόμενο πρόγραμμα βασίζεται στις επιλογές μαθημάτων που κάνουν οι σπουδαστές σε κάθε εξάμηνο και η κατάρτιση του προγράμματος θα γίνει αφού δηλωθούν αυτές οι επιλογές.
- **Exams Timetabling.** Αφορά την κατάρτιση του προγράμματος των εξετάσεων του τμήματος με περιορισμούς σχετικούς με τα μαθήματα, τις ημερομηνίες εξετάσεων και τις διαθέσιμες αίθουσες.

Το γενικό πρόβλημα του University Course Timetabling είναι ένα πρόβλημα βελτιστοποίησης που έχει μελετηθεί εκτενώς σε σχέση με άλλα προβλήματα κατάρτισης εκπαιδευτικών προγραμμάτων [7] γεγονός που οφείλεται σε μεγάλο βαθμό και στους δύο διαγωνισμούς ITC 2002 και ITC 2007που είχαν ως θέμα τους το πρόβλημα αυτό. Έτσι στην επόμενη ενότητα αναλύεται το πρόβλημα της 2^{ης} φάσης του διαγωνισμού International Timetabling Competition του 2007 που αφορά το πρόβλημα PE-CTT

μαζί με όλους τους περιορισμούς που είχαν τεθεί στη συγκεκριμένη φάση του διαγωνισμού.

3.2 Ανάλυση του προβλήματος Post Enrolment Course Timetabling για το διαγωνισμό ITC 2007

Με το πέρασμα του χρόνου εμφανίστηκαν και μελετήθηκαν αρκετές διαφορετικές εκδόσεις του προβλήματος PE-CTT. Στην παρούσα εργασία εξετάζουμε τη βασική έκδοση αυτού του προβλήματος έτσι όπως διατυπώθηκε και χρησιμοποιήθηκε στο διαγωνισμό ITC 2007 και περιγράφεται παρακάτω. [8] Θα πρέπει να αναφέρουμε ότι η συγκεκριμένη μορφή του προβλήματος αποτελούσε τη $2^{η}$ φάση του διαγωνισμού, αφού η $1^{η}$ φάση εξέταζε το πρόβλημα του Curriculum Based Timetabling, ενώ η $3^{η}$ φάση το πρόβλημα του Exams Timetabling. ToPE-CTT που εξετάστηκε στη $2^{η}$ φάση βασίζονταν στο αντίστοιχο πρόβλημα που διατυπώθηκε στο $1^{ο}$ διαγωνισμό ITCπου έλαβε χώρα το 2002, εμπλουτισμένο όμως με περισσότερους περιορισμούς που δυσκολεύουν τη λύση του. Πιο συγκεκριμένα οι διοργανωτές πρόσθεσαν δύο επιπλέον ανελαστικούς περιορισμούς με σκοπό το πρόβλημα PE-CTT να αντικατοπτρίζει καλύτερα τις πραγματικές συνθήκες που αντιμετωπίζει ένα ίδρυμα κατά την κατάρτιση ενός παρόμοιου ωρολογίου προγράμματος. Αυτό είχε ως συνέπεια να γίνει πιο δύσκολη η ικανοποίηση των ανελαστικών περιορισμών από τους υποψήφιους αλγόριθμους - σε σχέση με τον $1^{ο}$ διαγωνισμό - όπου κάτι τέτοιο ήταν σχετικά εύκολα εφικτό αλλά και απολύτως απαραίτητο για να μπορεί να κριθεί ο αλγόριθμος. Έτσι, ως συνέπεια αυτού του γεγονότος, οι διοργανωτές άλλαξαν και τον τρόπο που γινόταν η αξιολόγηση και η βαθμολόγηση του κάθε αλγόριθμου, δεχόμενοι πλέον και αλγόριθμους που δεν πληρούσαν όλους τους ανελαστικούς περιορισμούς. Εξαιτίας του γεγονότος αυτού οι διοργανωτές εισήγαγαν την έννοια του **Έγκυρου (Valid)** ωρολογίου προγράμματος που θα αναλύσουμε παρακάτω κατά την παρουσίαση και μοντελοποίηση του προβλήματος. Για περισσότερες πληροφορίες αναφορικά με τον $1^{ο}$ διαγωνισμό ITC 2002 ο αναγνώστης παραπέμπεται στην ηλεκτρονική διεύθυνση (http://www.idsia.ch/Files/itcomp2002/).

3.2.1 Ορισμός του Προβλήματος

Το πρόβλημα Post Enrolment Based Course Timetabling (PE - CCT)που εξετάστηκε στη $2^{η}$ φάση του διαγωνισμού ITC 2007 αναλύεται ως εξής:

- Ένα σύνολο από n εκπαιδευτικά γεγονότα (events), E = $\{e_1,\ldots\ldots,e_n\}$που πρέπει να προγραμματιστούν σε 45 χρονικές περιόδους (timeslots), T = $\{t_{1,}\ldots\ldots,t_{45}\}$ μέσα στην εβδομάδα (δηλαδή 5 ημέρες επί 9 ώρες ημερησίως).
- Ένα σύνολο από m αίθουσες (rooms), R = $\{r_1,\ldots\ldots,r_m\}$ κάθε μία από τις οποίες έχει συγκεκριμένη χωρητικότητα και στις οποίες λαμβάνουν χώρα τα γεγονότα δηλαδή οι διαλέξεις, τα σεμινάρια, τα εργαστήρια κλπ.
- Ένα σύνολο από k σπουδαστές S = $\{s_{1,}\ldots\ldots, s_k\}$ που παρακολουθούν διάφορους συνδυασμούς από γεγονότα, δηλαδή μια σχέση $M \subseteq n \times k$ τέτοια ώστε $(n,k) \in M$ εάν ο μαθητής S παρακολουθεί το γεγονόςN.
- Ένα σύνολο από l προδιαγραφές (features), F = $\{f_1,\ldots\ldots..,f_l\}$ που πληρούνται από τις αίθουσες και απαιτούνται για να πραγματοποιηθούν τα διάφορα γεγονότα.
- Ένα σύνολο T = $\{t_{1,}\ldots\ldots,t_{45}\}$ από διαθέσιμες χρονικές περιόδους όπου μπορούν να πραγματοποιηθούν τα γεγονότα. Προφανώς δεν μπορούν όλα τα γεγονότα να είναι διαθέσιμα σε όλες τις χρονικές περιόδους.
- Μια καρτεσιανή σχέση $\Pi \subseteq N \times N$ τέτοια ώστε αν $(n_1, n_2) \in \Pi$ αυτό σημαίνει ότι τα δύο γεγονότα πρέπει να προγραμματιστούν σε χρονικές περιόδους t_1, t_2 με $t_1 < t_2$ δηλαδή διαδοχικά.

Σκοπός του προβλήματος είναι να γίνει η ανάθεση των γεγονότων στις διαθέσιμες χρονικές περιόδους και τις κατάλληλες αίθουσες έτσι ώστε να ικανοποιούνται οι παρακάτω **Ανελαστικοί** περιορισμοί:

1. **Hard Constraint 1 – Conflicts.** Κάθε σπουδαστής δεν θα πρέπει να απαιτείται να παρακολουθεί περισσότερα από ένα γεγονότα σε μια συγκεκριμένη χρονική περίοδο.
2. **Hard Constraint 2 – Compatibility.** Ένα γεγονός δεν θα πρέπει να προγραμματιστεί σε αίθουσα που δεν πληροί κάποια από τις προδιαγραφές που τίθενται από αυτό και επίσης δεν μπορεί να προγραμματιστεί σε αίθουσα με χωρητικότητα μικρότερη από τον αριθμό των σπουδαστών που έχουν δηλωθεί να το παρακολουθήσουν.
3. **Hard Constraint 3 – Occupancy.** Επιτρέπεται να προγραμματιστεί μόνο ένα γεγονός ανά αίθουσα για κάθε συγκεκριμένη χρονική περίοδο διδασκαλίας.

4. **Hard Constraint 4 – Availability.** Τα γεγονότα μπορούν να ανατεθούν σε χρονικές περιόδους που έχουν από πριν καθορισθεί ως διαθέσιμες για αυτά τα γεγονότα.
5. **Hard Constraint 5 – Precedences.** Οι χρονικές περίοδοι που θα ανατεθούν στα διάφορα γεγονότα θα πρέπει να πληρούν τη σχέση Π όπως περιγράφεται παραπάνω.

Οι Ανελαστικοί περιορισμοί 1), 2), 3) είναι ακριβώς ίδιοι με αυτούς του 1ου διαγωνισμού ITC 2002, ενώ οι περιορισμοί 4), 5) αποτελούν νέες προσθήκες.

Ένα από τα βασικά χαρακτηριστικά των λύσεων του 1ου διαγωνισμού είναι ότι προκειμένου να αξιολογηθεί ένας αλγόριθμος θα έπρεπε να πληροί όλες τις ανελαστικές προϋποθέσεις. Αλγόριθμοι που δεν παρήγαγαν εφικτά ωρολόγια προγράμματα, δηλαδή δεν κάλυπταν τουλάχιστον τους ανελαστικούς περιορισμούς στο δοσμένο χρόνο δεν γίνονταν δεκτοί. Αυτός ο περιορισμός είχε τεθεί έτσι ώστε να υπάρχει ένα κοινό σημείο αναφοράς για την ποιότητα των παραγόμενων λύσεων του κάθε αλγόριθμου. Έτσι, αφού προκειμένου να αξιολογηθούν οι αλγόριθμοι των διαγωνιζομένων έπρεπε να πληρούν όλους του ανελαστικούς περιορισμούς, η αξιολόγηση τους και η βαθμολόγηση τους θα προέκυπτε αποκλειστικά από τον αριθμό των ελαστικών περιορισμών που πληρούσαν. Τα αντίστοιχα στιγμιότυπα του προβλήματος που παρείχαν οι διοργανωτές (υπό μορφή αρχείων απλού κειμένου), ήταν έτσι κατασκευασμένα ώστε να μπορεί να επιτευχθεί ο στόχος της εκπλήρωσης των ανελαστικών περιορισμών σχετικά εύκολα. Με την εισαγωγή των δύο νέων ανελαστικών περιορισμών που αναφέρθηκαν παραπάνω κάτι τέτοιο δεν ήταν πλέον τόσο εύκολο. Για το λόγο αυτό οι διοργανωτές επινόησαν την έννοια του **Έγκυρου (Valid)**προγράμματος. Για την καλύτερη κατανόηση αυτών των εννοιών παραθέτουμε τους παρακάτω ορισμούς:

- Ένα **Έγκυρο (Valid)** ωρολόγιο πρόγραμμα είναι ένα πρόγραμμα στο οποίο δεν παραβιάζεται κανένας ανελαστικός περιορισμός, αλλά δίνεται η δυνατότητα σε κάποια γεγονότα να μην τοποθετηθούν σε θέσεις του ωρολογίου προγράμματος.
- Ένα **Εφικτό (Feasible**) ωρολόγιο πρόγραμμα είναι ένα πρόγραμμα στο οποίο δεν παραβιάζεται κανένας ανελαστικός περιορισμός αλλά ταυτόχρονα όλα τα γεγονότα έχουν εισαχθεί σε θέσεις του ωρολογίου προγράμματος.

Δεδομένων των παραπάνω ορισμών η λύση που έδωσαν οι οργανωτές ήταν να γίνονται δεκτοί προς αξιολόγηση όλοι οι αλγόριθμοι που παρήγαγαν είτε έγκυρα είτε

εφικτά προγράμματα. Επίσης για την σωστότερη αξιολόγηση των αλγορίθμων, δημιούργησαν και ένα μέτρο της «Απόστασης από το εφικτό» (Distance to Feasibility) για την αξιολόγηση των υποψήφιων λύσεων το οποίο θα αναλυθεί παρακάτω στην ενότητα που παρουσιάζει την αξιολόγηση.

Επιπρόσθετα από τους πέντε ανελαστικούς περιορισμούς που αναφέρονται πιο πάνω, το πρόβλημα που εξετάζουμε πρέπει να πληροί και έναν αριθμό από **Ελαστικούς** περιορισμούς. Αυτοί είναι οι εξής:

1. **Soft Constraint 1 – Late Events.**Δεν θα πρέπει να έχει προγραμματιστεί γεγονός για να παρακολουθήσουν οι σπουδαστές στην τελευταία χρονική περίοδο κάθε ημέρας.
2. **Soft Constraint 2 – Consecutive Events.** Οι σπουδαστές δεν θα πρέπει να υποχρεούνται να παρακολουθήσουν τρία ή περισσότερα γεγονότα σε συνεχόμενες χρονικές περιόδους που συμβαίνουν την ίδια ημέρα.
3. **Soft Constraint 3 – Isolated Events.** Οι σπουδαστές δεν θα πρέπει να υποχρεούνται να παρακολουθήσουν μόνο ένα εκπαιδευτικό γεγονός σε κάποια ημέρα.

Σημειώνουμε εδώ ότι οι παραπάνω τρεις ελαστικοί περιορισμοί είναι ακριβώς ίδιοι με αυτούς που είχαν ορισθεί στον πρώτο διαγωνισμό. Οι λύσεις που δίνει ο κάθε υποψήφιος αλγόριθμος καταγράφονται με απλή κωδικοποίηση σε ένα αρχείο απλού κειμένου. Η κωδικοποίηση αυτή περιγράφεται στον δικτυακό τόπο του διαγωνισμού όπου μπορεί επιπλέον ο κάθε ενδιαφερόμενος να κατεβάσει και τα αρχεία με τα στιγμιότυπα του προβλήματος που πρέπει να επιλυθούν. Επίσης οι διοργανωτές κατασκεύασαν ένα πρόγραμμα (σε γλώσσα C++), που αξιολογεί τις λύσεις βάσει των περιορισμών που παραβιάζουν και είναι και αυτό διαθέσιμο, όπως και ο πηγαίος κώδικας του από την ίδια ιστοσελίδα.

3.2.2 Μοντελοποίηση του Προβλήματος

Στο πρόβλημα PE – CCTπου εξετάζουμε πρέπει να αναθέσουμε ένα γεγονός (μάθημα, διάλεξη, εργαστήριο), σε μια από τις 45 (5 x 9) χρονικές περιόδους που έχουμε διαθέσιμες μέσα στην εβδομάδα. Σύμφωνα με την ανάλυση του προβλήματος που έγινε παραπάνω αλλά και από τις συσχετίσεις των συνόλων που ορίζουν το πρόβλημα προκύπτουν οι παρακάτω πίνακες:

- Ο πρώτος πίνακας A_{kxn} ονομάζεται πίνακας *Student – Event* και δείχνει τα γεγονότα που παρακολουθούν συγκεκριμένοι σπουδαστές. Ισχύει ότι $a_{ij} = 1, a_{ij} \in A_{kxn}$ εάν ο σπουδαστής $S_i \in S$ παρακολουθεί το γεγονός $e_j \in E$, διαφορετικά η τιμή αυτή είναι 0.
- Ο δεύτερος πίνακας B_{nxn} ονομάζεται *Event – Conflict* και μας δείχνει εάν δύο γεγονότα μπορούν να προγραμματισθούν στην ίδια χρονική περίοδο ή όχι.
- Ο τρίτος πίνακας C_{mxl} ονομάζεται πίνακας *Room – Feature* και μας δείχνει τις προδιαγραφές που διαθέτει η κάθε αίθουσα. Έτσι ισχύει ότι $c_{ij} = 1$ εάν μια αίθουσα $r_i \in R$ διαθέτει την προδιαγραφή $f_j \in F$, αλλιώς σε διαφορετική περίπτωση το κελί έχει την τιμή 0.
- Ο τέταρτος πίνακας D_{lxn} ονομάζεται *Feature – Event*. Ισχύει ότι $d_{ij} = 1, d_{ij} \in D_{lxn}$ τότε το γεγονός $e_i \in E$ απαιτεί την προδιαγραφή αίθουσας $f_j \in F$, αλλιώς σε διαφορετική περίπτωση το κελί θα παίρνει την τιμή 0.
- Ο πέμπτος πίνακας G_{nxm} ονομάζεται *Event – Room* και καταγράφει όλες τις διαθέσιμες αίθουσες όπου μπορούν να τοποθετηθούν τα διάφορα γεγονότα. Μέσω αυτού του πίνακα, μπορούμε εύκολα να βρούμε όλες τις αίθουσες που η χωρητικότητα τους είναι επαρκής για το κάθε γεγονός. Χρησιμοποιούμε έτσι ένα πίνακα για να αναθέσουμε κάθε γεγονός σε μια αίθουσα $r_i \in R$ και σε μια συγκεκριμένη χρονική θέση t_i . Σε κάθε ζεύγος (r_i, t_i) ανατίθεται μια τιμή που αντιστοιχεί στο αντίστοιχο γεγονός. Εάν η αίθουσα r_i στη δεδομένη χρονική περίοδο t_i είναι κενή ή δεν έχει γίνει ακόμη καταχώρηση τότε το ζεύγος (r_i, t_i) έχει την τιμή -1 σε αυτή τη θέση. Με τον τρόπο αυτό εξασφαλίζουμε ότι δεν θα ανατεθεί στο ζεύγος (r_i, t_i) πάνω από ένα γεγονός κάθε φορά, καλύπτοντας έτσι τον τρίτο ανελαστικό περιορισμό.
- Ο έκτος πίνακας H_{nxt} ονομάζεται *Event – Availability* και κάθε του κελί λαμβάνει την τιμή 1 εάν το γεγονός $e_i \in E$ μπορεί να λάβει χώρα τη χρονική περίοδο $t_j \in T$. Σε αντίθετη περίπτωση η τιμή του κελιού είναι 0.

- Ο έβδομος και τελευταίος πίνακας που χρησιμοποιείται είναι ο πίνακας I_{nxn} που ονομάζεται *Event – Preference,* όπου κάθε τιμή του είναι $n_{ij} = 1$ αν το γεγονός $e_i \in E$ πρέπει να προγραμματισθεί πριν από το γεγονός $e_j \in E$ ή διαφορετικά έχουμε $n_{ij} = -1$αν το γεγονός $e_i \in E$ πρέπει να προγραμματισθεί μετά από το γεγονός $e_j \in E$. Σε περίπτωση που δεν έχουμε περιορισμό για τη διαδοχικότητα των δύο γεγονότων $e_{i,}e_j$ τότε ισχύει $n_{ij} = 0$.

Επιπλέον από τους πίνακες που περιγράφηκαν παραπάνω χρησιμοποιούμε και μια σειρά από λίστες ΕΕ. Κάθε στοιχείο $EE_i \in EE$ είναι μια λίστα γεγονότων που πρέπει να τοποθετηθούν στο ωρολόγιο πρόγραμμα πριν από το γεγονός $e_i \in E$. Η πληροφορία που παρέχεται από αυτές τις λίστες βοηθά στην κάλυψη του ανελαστικού περιορισμού 5 κατά την εκτέλεση του αλγόριθμου.

3.3 Η Αξιολόγηση των Λύσεων

Στην παρακάτω ενότητα θα περιγράψουμε τον τρόπο με τον οποίο οι διοργανωτές του διαγωνισμού ITC 2007 έκριναν και αξιολόγησαν την ποιότητα των αλγόριθμων που διαγωνίστηκαν. Καταρχήν, όπως ήδη έχει ειπωθεί, για να μπορέσει να διαγωνισθεί ένας αλγόριθμος θα πρέπει τουλάχιστον να παράγει έγκυρο πρόγραμμα. Υπενθυμίζουμε όμως ότι έγκυρη μπορεί να είναι και μια λύση που αφήνει κάποια γεγονότα εκτός ωρολογίου προγράμματος, προκειμένου να πληροί τους ανελαστικούς περιορισμούς. Έτσι επινοήθηκε η μετρική **«Απόσταση από το Εφικτό» (Distance to Feasibility)**, η οποία υπολογίζεται απλά ως το άθροισμα των σπουδαστών που έπρεπε να παρακολουθήσουν κάποια γεγονότα τα οποία δεν εισάχθηκαν τελικά στο πρόγραμμα. Έτσι για παράδειγμα εάν το παραγόμενο πρόγραμμα έχει αφήσει έξω τα γεγονότα ev1, ev2, ev6 τα οποία παρακολουθούν 6, 18, 20 σπουδαστές αντίστοιχα τότε η τιμή της παραπάνω μετρικής θα είναι:

Distance to Feasibility = 6 + 18 + 20 = 44

Εξ' ορισμού προκύπτει ότι ένα εφικτό (feasible) που ενσωματώνει όλα τα γεγονότα έχει τιμή 0 για την παραπάνω μετρική και μια έγκυρη λύση με μικρή τιμή είναι καλύτερη από μια αντίστοιχη με μεγαλύτερη τιμή. Επίσης εάν ένας σπουδαστής είναι δηλωμένος να παρακολουθεί δύο γεγονότα που δεν έχουν τοποθετηθεί στο πρόγραμμα τότε είναι λογικό να προσμετράτε δύο φορές στη μετρική.

Έχοντας μετρήσει τη μετρική Distance to Feasibilityεπόμενο βήμα στην αξιολόγηση των αλγορίθμων είναι η μέτρηση της παραβίασης των ελαστικών περιορισμών. Αυτό γίνεται με τη δημιουργία μιας αντικειμενικής συνάρτησης (η οποία θα δέχεται ακέραιες τιμές) και θα αποδίδει το κόστος παραβίασης των ελαστικών περιορισμών σύμφωνα με τους παρακάτω κανόνες:

- (*SoftCost1*) – Ένας φοιτητής δεν θα πρέπει να παρακολουθεί κάποιο γεγονός στην τελευταία χρονική περίοδο της ημέρας. Σε περίπτωση τέτοιου γεγονότος θα προστίθεται μια μονάδα για κάθε σπουδαστή που είναι υποχρεωμένος να το παρακολουθήσει.
- (*SoftCost*$_2$) –Ένας φοιτητής δεν θα πρέπει να παρακολουθεί πάνω από δύο συνεχόμενα γεγονότα σε μία ημέρα. Έτσι σε τέτοιες περιπτώσεις και για κάθε σπουδαστή 3 συνεχόμενα γεγονότα αντιστοιχούν σε 1 μονάδα, 4 συνεχόμενα γεγονότα σε 2 μονάδες κοκ. Για παράδειγμα εάν 4 σπουδαστές υποχρεούνται να παρακολουθήσουν 4 συνεχόμενα γεγονότα σε μια ημέρα η ποινή θα υπολογιστεί ως 4 * 2 = 8 μονάδες. Μαθήματα στην τελευταία περίοδο μιας ημέρας δεν θεωρούνται συνεχόμενα με μαθήματα που αρχίζουν την 1^η περίοδο της επόμενης ημέρας.
- (*SoftCost*$_3$) – Ένας σπουδαστής δεν θα πρέπει να παρακολουθεί μόνο ένα εκπαιδευτικό γεγονός ανά ημέρα. Εδώ προσμετρούνται με 1 μονάδα για κάθε σπουδαστή που πρέπει να κάνει κάτι τέτοιο σε μια ημέρα.

Η αντικειμενική συνάρτηση για τις παραβιάσεις των ελαστικών περιορισμών, την οποία θα ονομάζουμε από εδώ και στο εξής Soft Cost Violation προφανώς θα είναι το άθροισμα των SCost$_1$ +SCost$_2$ + SCost$_3$**.** Εν κατακλείδι η ποιότητα μιας υποψήφιας λύσης θα υπολογίζεται από μια συνάρτηση κόστους που είναι η σύνθεση των δύο μετρικών που αναλύσαμε, δηλαδή πρώτα υπολογίζεται η μετρική Distance to Feasibility και ο αλγόριθμος με την χαμηλότερη τιμή είναι ο νικητής. Εάν τώρα δύο αλγόριθμοι είναι ίση σε αυτή τη μετρική τότε υπολογίζεται για κάθε έναν από αυτούς η μετρική Soft Cost Violation και καλύτερος από αυτούς θεωρείται ο αλγόριθμος με τη χαμηλότερη τιμή.

3.4 Περιορισμοί του Μοντέλου του Διαγωνισμού

Στην προηγούμενη ενότητα εξηγήθηκε η μέθοδος που επέλεξαν οι διοργανωτές για μπορεί να γίνει μια δίκαιη αξιολόγηση των αλγορίθμων που μετείχαν στον διαγω-

νισμό. Η προσπάθεια τους επικεντρώθηκε στην εφαρμογή κανόνων αξιολόγησης που θα επιδοτούσαν περισσότερο τους αλγόριθμους που πράγματι δημιουργούν ποιοτικότερα ωρολόγια προγράμματα σε συνθήκες που είναι πολύ κοντά στις πραγματικές. Αυτό δεν σημαίνει βέβαια ότι δεν υπάρχουν και άλλες στρατηγικές αντιμετώπισης του προβλήματος με πλεονεκτήματα σε κάποιες περιπτώσεις αλλά και μειονεκτήματα. Στην βιβλιογραφία αλλά και στην επιχειρησιακή πρακτική υπάρχουν αλγόριθμοι που λειτουργούν διαφορετικά στον τρόπο που τοποθετούν τα εκπαιδευτικά γεγονότα στο υπό κατασκευή πρόγραμμα. Το σημαντικό πάντως στην περίπτωση αυτού του διαγωνισμού, ήταν να υπάρχει εκ των προτέρων μια καλώς ορισμένη και διατυπωμένη διαδικασία αξιολόγησης ώστε να μπορούν να συγκριθούν μεταξύ τους τα προγράμματα που παράγουν οι υποψήφιοι αλγόριθμοι.

Επίσης όπως αναφέρθηκε και παραπάνω οι πραγματικοί ελαστικοί και ανελαστικοί περιορισμοί που μπορεί να προκύψουν κατά την διαδικασία κατασκευής ενός ωρολογίου προγράμματος τύπου PE–CTT σε κάποιο εκπαιδευτικό ίδρυμα μπορεί να είναι πολύ περισσότεροι και με μεγαλύτερη ποικιλομορφία από αυτούς που προτείνει ο διαγωνισμός ITC. Αν και μια εξαντλητική καταγραφή αυτών των περιορισμών είναι μάλλον αδύνατο να γίνει στα πλαίσια αυτής της διπλωματικής, θα καταγράψουμε περιληπτικά μερικούς ακόμη περιορισμούς που μπορεί να συναντήσει ένας ερευνητής που έχει σκοπό να σχεδιάσει αλγόριθμους για την αντιμετώπιση αυτού του προβλήματος ή άλλων παρόμοιων.

- Γεγονότα που ενώ έχουν κοινούς σπουδαστές που τα παρακολουθούν δεν μπορούν να γίνουν σε διαδοχικές περιόδους, γιατί απαιτείται χρόνος για την μετακίνηση των σπουδαστών σε διαφορετικά σημεία του πανεπιστημιακού χώρου.
- Πολλά ιδρύματα έχουν κάποια ή κάποιες κενές ώρες συνήθως κοντά στο μεσημέρι για να εξασφαλίσουν ότι όλοι οι σπουδαστές θα έχουν την ευκαιρία για το μεσημεριανό τους γεύμα. Το γεγονός αυτό μπορεί να επιβάλλει κάποιους περιορισμούς στα γεγονότα που έχουν προγραμματισθεί σε εκείνες τις ώρες εάν δεν υπάρχει γενική διακοπή των μαθημάτων.
- Ένα ίδρυμα μπορεί να επιβάλλει κάποιους περιορισμούς σε κάποια μαθήματα που μπορεί να οφείλεται σε ιδιαίτερες σχέσεις και συνάφειες που υπάρχουν μεταξύ αυτών των γεγονότων. Έτσι για παράδειγμα μπορεί να επιβληθεί σε

δύο γεγονότα να πρέπει να διδάσκονται την ίδια ημέρα ή το αντίθετο εάν αυτό εξυπηρετεί καλύτερα τη διδασκαλία τους.

Στην καθημερινή πρακτική επίσης μπορεί να προκύψουν ελαστικοί και ανελαστικοί περιορισμοί που έχουν να κάνουν με την διαχείριση των αιθουσών διδασκαλίας, εργαστηρίων, αμφιθεάτρων κλπ.

Ενδεικτικά αναφέρουμε μερικές τέτοιες περιπτώσεις:

- Σε κάποιες συγκεκριμένες περιπτώσεις μαθημάτων η διδασκαλία τους δεν απαιτεί αίθουσα ή αμφιθέατρο αλλά μπορεί να γίνει στην ύπαιθρο ή περιλαμβάνει συχνά επισκέψεις εκτός του πανεπιστημιακού χώρου.
- Κάποιες αίθουσες ή αμφιθέατρα μπορεί να μην είναι διαθέσιμες σε κάποιες συγκεκριμένες ώρες για διοικητικού λόγους, συντήρηση ή επειδή έχουν προγραμματισθεί σε αυτές γεγονότα (διαλέξεις) εκτός του ιδρύματος.
- Κάποια ιδρύματα έχουν κανονισμούς που δεν επιτρέπουν σε γεγονότα με μικρό αριθμό σπουδαστών να εκμεταλλευθούν αίθουσες με μεγάλη χωρητικότητα ή εργαστήρια που χρησιμοποιούνται για πρακτική κλπ.

Επιπλέον από όλους τους παραπάνω περιορισμούς το θέμα της φιλικότητας ενός προγράμματος έχει αρκετές ιδιαιτερότητες όπως προαναφέρθηκε για κάθε ξεχωριστό ίδρυμα. Έτσι για παράδειγμα:

- Πολλά ιδρύματα θεωρούν ότι πρέπει να υπάρχει μια κενή ημέρα στο πρόγραμμα κάθε εβδομάδος έτσι ώστε χρησιμοποιείται για έρευνα, μελέτη ή ξεκούραση.
- Επίσης μπορεί να υπάρχει μια κενή ημέρα κάθε εβδομάδα για τις αίθουσες υπολογιστών και τα άλλα είδη εργαστηρίων, ώστε αυτά να είναι ελεύθερα να χρησιμοποιούνται από τους σπουδαστές ή προγράμματα εκπαίδευσης εκτός των πανεπιστημιακών (Πχ Εκπαίδευση ενηλίκων, Σεμινάρια Ανέργων).
- Μεμονωμένοι περιορισμοί που επιβάλλουν οι διδάσκοντες και αφορούν συγκεκριμένα μαθήματα. Για παράδειγμα κάποιος καθηγητής μπορεί να θέλει όλες οι ώρες του να είναι προγραμματισμένες σε μια ημέρα λόγω διαβίωσης εκτός της πόλης που στεγάζεται το ίδρυμα κ.ά.

Επειδή θα ήταν μάλλον αρκετά απίθανο όλοι οι περιορισμοί που προαναφέραμε να εμφανιστούν ταυτόχρονα σε ένα εκπαιδευτικό ίδρυμα, οι διοργανωτές των διαγωνισμών ITC περιορίστηκαν σε αυτούς που θεώρησαν ότι είναι οι πιο γενικοί στη καθημερινή πρακτική και με τη μεγαλύτερη καθολικότητα. Για περισσότερες πληροφο-

ρίες σχετικά με τους κανόνες και τον τρόπο διεξαγωγής του διαγωνισμού ο αναγνώστης παραπέμπεται στον δικτυακό τόπο (www.cs.qub.ac.uk/itc2007) αλλά και στη βιβλιογραφία [8].

3.5 Αξιόλογες Προσπάθειες που συμμετείχαν στον διαγωνισμό

Στον διαγωνισμό ITC 2007 συμμετείχαν πολλοί και αξιόλογοι ερευνητές από όλο τον κόσμο. Πολλοί από αυτούς συμμετείχαν με προσαρμοσμένους αλγόριθμους και στα τρία υποπροβλήματα, ενώ κάποιοι άλλοι επικέντρωσαν τις προσπάθειές τους μόνο σε ένα από αυτά. Αναφορικά με το πρόβλημα του PE – CCTπου μας ενδιαφέρει οι τρεις πρώτες θέσεις είχαν ως εξής:

1. H. Cambazard και συνεργάτες. Χρησιμοποίησαν έναν υβριδικό αλγόριθμο που χρησιμοποιούσε Local Search τεχνικές και Constraint Programming.
2. M. Atsuta και συνεργάτες. Χρησιμοποίησαν έναν αλγόριθμο Tabu Search με ιδιαίτερες ρυθμίσεις στα χρησιμοποιούμενα βάρη των ανελαστικών και ελαστικών περιορισμών, που μπορούσαν να αλλάζουν ακόμη και κατά τη διάρκεια εκτέλεσης του αλγόριθμου.
3. M. Chiarardini και συνεργάτες. Χρησιμοποιήθηκε μεταευριστικός αλγόριθμος που συνδύαζε πολλές τεχνικές Stochastic Local Search.

Ιδιαιτέρως θα πρέπει να αναφέρουμε εδώ την προσπάθεια του T. Muller που δημιούργησε έναν αλγόριθμο χρησιμοποιώντας διάφορες τεχνικές από τη διαθέσιμη βιβλιοθήκη της Java που ονομάζεται Java Constraints Solver. Μερικές από αυτές τις τεχνικές είχαν ως βάση τους αλγόριθμους Hill Climbing, Simulated Annealing και Constraint Based Statistics. Ο παραπάνω συνθετικός αλγόριθμος τερμάτισε 5ος στο πρόβλημα του PE – CCT αλλά πρώτευσε στα άλλα δύο σκέλη του διαγωνισμού, δηλαδή στο Exams Timetabling και το CB – CCT και άτυπα ήταν ο αλγόριθμος με την καλύτερη συνολικά επίδοση στο γενικότερο πρόβλημα του University Course Timetabling.

Κεφάλαιο 4: Γενική Επισκόπηση του Αλγορίθμου PSO (Particle Swarm Optimization)

Αρχικά στην ενότητα 4.1 του παρόντος κεφαλαίου παρουσιάζεται η έννοια του Μεταευριστικού Αλγόριθμου (Metaheuristics Algorithm) και γίνεται η παρουσίαση του αλγόριθμου PSO που είναι χαρακτηριστικό δείγμα αυτής της κατηγορίας αλγορίθμων και της υποκατηγορίας των αλγορίθμων Νοημοσύνης Σμήνους (Swarm Intelligence). Στην ενότητα 4.2 αναλύεται ο αλγόριθμος στην κλασσική του μορφή, ενώ στην ενότητα 4.3 γίνεται επισκόπηση των παραμέτρων του αλγόριθμου και των ρυθμίσεων που αυτές επιδέχονται. Τέλος στην ενότητα 4.4 γίνεται μια πρώτη αναφορά στις υβριδικές παραλλαγές του PSO που μπορούμε να κατασκευάσουμε προκειμένου να βελτιώσουμε τις επιδόσεις.

4.1 Μεταευριστικοί Αλγόριθμοι και Αλγόριθμοι Νοημοσύνης Σμήνους

Οι Μεταευριστικοί Αλγόριθμοι ανήκουν στην κατηγορία αλγορίθμων Υπολογιστικής Νοημοσύνης που με τη σειρά τους ανήκουν στο ευρύτερο πεδίο της Τεχνητής Νοημοσύνης (Artificial Intelligence). Έχουν εφαρμογή σε μεγάλο αριθμό προβλημάτων βελτιστοποίησης όπου στη γενικότητα τους προσπαθούν με επαναληπτικό τρόπο, να υποδείξουν πρώτα και στη συνέχεια να βελτιώσουν μια υποψήφια λύση σ' ένα μεγάλο χώρο λύσεων. [16]

Σύμφωνα με τους Blum και Poli [17] οι μεταευριστικοί αλγόριθμοι χαρακτηρίζονται από τις παρακάτω βασικές ιδιότητες:

- Ερευνούν αποτελεσματικά το χώρο λύσεων προκειμένου να βρουν τις καλύτερες (σχεδόν βέλτιστες) λύσεις.
- Είναι μη – ντετερμινιστικοί προσεγγιστικοί αλγόριθμοι.
- Διαθέτουν μηχανισμούς που τους αποτρέπουν από το να παγιδεύονται σε τοπικές λύσεις του χώρου λύσεων.
- Εφαρμόζονται σε ευρεία γκάμα προβλημάτων βελτιστοποίησης.
- Είναι δυνατόν να διαθέτουν από πολύ απλές υπολογιστικές τεχνικές (τοπικές αναζητήσεις) έως και πολύ πολύπλοκες (πολύπλοκες διαδικασίες μάθησης).

Μια από τις γνωστές Μεταευρετικές αλγοριθμικές διαδικασίες είναι και ο αλγόριθμος PSO (Particle Swarm Optimization) που είναι ένας αλγόριθμος Υπολογιστικής Νοημοσύνης που ανήκει επιπλέον στην υποκατηγορία των αλγόριθμων Νοημοσύνης

Σμήνους (Swarm Intelligence). Ο αλγόριθμος PSO προτάθηκε πρώτα από τους Kennedy, Eberhart και Shi [20], [21], και είναι ένας αλγόριθμος που βασίζει τις διαδικασίες του στον τρόπο που ένα σμήνος πουλιών ή ψαριών κινείται στο χώρο συντονισμένα αναζητώντας την τροφή του. Όπως όλοι οι αλγόριθμοι Υπολογιστικής Νοημοσύνης δίνει έμφαση στη στρατηγική αναζήτησης λύσεων που με επαναληπτικές διαδικασίες θα δώσουν ένα σχεδόν βέλτιστο αποτέλεσμα. Η φιλοσοφία της παραπάνω υποκατηγορίας αλγορίθμων (Swarm Intelligence) στηρίζεται στην αρχή ότι η υψηλή συλλογική νοημοσύνη, μπορεί να προκύψει ως συμπεριφορά από τη συνεργασία ενός μεγάλου αριθμού λιγότερο ευφυών πρακτόρων (particles). Άλλοι παρόμοιοι αλγόριθμοι αυτής της κατηγορίας είναι ο Ant Colony Optimization, ο Swarm Fish Intelligence, ο Cat Swarm Optimization, ο Artificial Bee Colony και άλλοι.

Προκειμένου να αναπτύξουμε τη μεθοδολογία του αλγόριθμου PSO θα παρουσιάσουμε το παρακάτω παράδειγμα που περιγράφει [11] τον τρόπο λειτουργίας του.

Έστω λοιπόν ένα σμήνος πουλιών που αναζητά την τροφή του σε ένα συγκεκριμένο χώρο και ας υποθέσουμε πως υπάρχει μόνο ένα κομμάτι τροφής στη συγκεκριμένη περιοχή. Τα πουλιά δεν γνωρίζουν την ακριβή τοποθεσία όπου βρίσκεται η τροφή, έχουν όμως αίσθηση της απόστασης από αυτή και πόσο πλησιάζουν σε αυτή σε κάθε επανάληψη. Μια πολύ αποτελεσματική στρατηγική που μπορούν να χρησιμοποιήσουν είναι να ακολουθήσουν το πτηνό που βρίσκεται πιο κοντά στην τροφή, αφού κατά αυτόν τον τρόπο η απόσταση που θα έχουν από την τροφή θα μικραίνει συνεχώς και με ολοένα πιο γρήγορους ρυθμούς.

Κάνοντας τώρα την αντιστοιχία αυτού του φυσικού φαινομένου με τον αλγόριθμο PSO, παρατηρούμε ότι η έρευνα για την τροφή αντιστοιχεί στη διαδικασία βελτιστοποίησης μιας συνάρτησης, ενώ ο χώρος των λύσεων του προβλήματος βελτιστοποίησης αντιστοιχεί στην ευρύτερη περιοχή που βρίσκεται το κομμάτι της τροφής. Κάθε πουλί αντιστοιχεί σε μια εφικτή λύση στο χώρο των λύσεων και ονομάζεται **Particle**.Όλα τα particlesέχουν τιμές αποτίμησης (Fitness values) οι οποίες θα υπολογιστούν από επιμέρους συναρτήσεις αποτίμησης (Fitness functions). Αυτό που μας ενδιαφέρει λοιπόν είναι να ελαχιστοποιήσουμε ή να μεγιστοποιήσουμε τις παραπάνω συναρτήσεις αποτίμησης. Ως ιδιότητες τα particlesδιαθέτουν διανύσματα ταχυτήτων και σύμφωνα με τις τιμές αυτών καθορίζεται και η πορεία τους στο χώρο λύσεων. Τα particlesπετούν στο χώρο λύσεων ακολουθώντας τα particles με τις καλύτερες τιμές μέχρι εκείνη τη στιγμή.

Ο αλγόριθμος PSOαρχικοποιείται μέσω ανάθεσης τιμών σε μια σειρά από τυχαία particles (solutions) και στη συνέχεια αναζητά τις καλύτερες λύσεις μέσω διαδοχικών επαναλήψεων. Κάθε επανάληψη λέμε ότι αποτελεί μια **Γενεά Εξέλιξης (Generation).** Σε κάθε τέτοια γενεά εξέλιξης κάθε particleενημερώνεται βάσει των δύο παρακάτω τιμών:

1. **pbest$_i$** (Personal Best) – που είναι η βέλτιστη τιμή της λύσης που έχει επιτευχθεί μέχρι εκείνη τη στιγμή από το συγκεκριμένο particle.
2. **Gbest** (Global Best) – που είναι η βέλτιστη τιμή που έχει επιτευχθεί μέχρι εκείνη την επανάληψη από οποιοδήποτε particle(Ολικό Βέλτιστο).

Να αναφέρουμε επίσης ότι εάν θεωρήσουμε για κάθε particle ένα μέρος του πληθυσμού ως τοπολογικούς γείτονες, τότε το βέλτιστο καλείται **Lbest (Local Best).**

Αφού λοιπόν υπολογιστούν οι δύο αυτές τιμές τότε το κάθε particleενημερώνει την ταχύτητα και τη θέση του στο τέλος κάθε επανάληψης, με τη βοήθεια των παρακάτω σχέσεων:

(4.1 a) $$v_i^{k+1} = w \cdot v_i^k + c_1 \cdot rand() \cdot (pbest_i - s_i^k) + c_2 \cdot rand() \cdot (gbest_i - s_i^k)$$

(4.1 b) $$s_i^{k+1} = s_i^k + v_i^{k+1}$$

Όπου v_i^k είναι το διάνυσμα ταχύτητας του particle i στην k επανάληψη και s_i^k η θέση του particle I στην k επανάληψη. Τα gbest και $pbest_i$ έχουν ήδη οριστεί παραπάνω ενώ ο παράγοντας rand() αποδίδει έναν τυχαίο αριθμό στο διάστημα (0, 1). Ο τυχαίος αυτός αριθμός σχετίζεται με την επιτάχυνση του κάθε particle προς την κατεύθυνση των θέσεων $pbest_i$ και gbest του χώρου λύσεων. Οι συντελεστές c_1και c_2ονομάζονται παράγοντες μάθησης και συνήθως θέτονται ως $c_1 = c_2 = 2$.Η τιμή της ταχύτητας επίσης περιορίζεται σε κάποιο συγκεκριμένο διάστημα τις πιο πολλές φορές.

Ο συντελεστής (**w)** ονομάζεται **Συντελεστής Αδράνειας** και συνήθως δίνεται από την παρακάτω σχέση [18], [19]:

(4.1 c) $$w = w_{\max} - \frac{W_{\max} - W_{\min}}{iter_{\max}} \cdot iter$$

όπου

w_{max} – Αρχική τιμή αδράνειας

w_{min} – Τελική τιμή αδράνειας

$iter_{max}$ – Μέγιστος αριθμός επαναλήψεων του αλγόριθμου

iter – Αριθμός της τρέχουσας επανάληψης

Η παραπάνω μέθοδος υπολογισμού του συντελεστή wονομάζεται μέθοδος της Αδράνειας (IWA – Inertia Weights Approach) και ο συντελεστής w σε αυτή τη μέθοδο ελέγχει την επιρροή λόγω αδράνειας της προηγούμενης ταχύτητας στη νέα για κάθε particle [19]. Χρησιμοποιώντας την παραπάνω εξίσωση (4.1 c) της αδράνειας τα χαρακτηριστικά διαφοροποίησης των particles σιγά - σιγά φθίνουν και έτσι μπορούμε να υπολογίσουμε το διάνυσμα της ταχύτητας v_iγια κάθε ένα από αυτά, που θα οδηγήσει τη μέθοδο κοντά στη βέλτιστη λύση και στη θέση που θα υπολογιστεί από την εξίσωση (4.1 b) [19]. Από αυτή την εξίσωση το particleαποφασίζει πως θα κινηθεί στην επόμενη επανάληψη, δεδομένου ότι γνωρίζει την καλύτερη προσωπική του θέση $pbest_i$ αλλά και το gbest που είναι η καλύτερη θέση μέχρι εκείνη τη στιγμή για όλα τα particlestου σμήνους. Όλα τα particles του σμήνους τείνουν να κινηθούν σε κάθε επανάληψη προς τις καλύτερες θέσεις. Έτσι κατά αυτόν τον τρόπο η βέλτιστη λύση επιτυγχάνεται από τη συνδυασμένη προσπάθεια όλου του πληθυσμού του σμήνους.

4. 2 Παρουσίαση και Ανάλυση του κλασικού αλγόριθμου PSO

Παρακάτω παραθέτουμε σε μορφή ψευδοκώδικα τον αλγόριθμο PSO όπως αυτός περιγράφεται στην ιστοσελίδα www.swarmintelligence.org).

For each particle
Initialize particle
END
Set gbest to worst possible value
Do
For each particle
Calculate fitness value
If the fitness value is better than the best fitness value (pBest) in history
set current value as the new pBest
END

Choose the particle with the best fitness value of all the particles as the gBest
For each particle
Calculate particle velocity according equation (a)

Update particle position according equation (b)

END

While maximum iterations or minimum error criteria is not attained

Στην αρχή βλέπουμε πως αρχικοποιούνται τα particles με τυχαίο τρόπο, ενώ το gbest (ολικό βέλτιστο) λαμβάνει τη χειρότερη δυνατή τιμή. Στη συνέχεια όλα τα υπόλοιπα βήματα του αλγόριθμου θα επαναληφθούν όσες φορές είναι αυτό αναγκαίο για να πετύχουμε τα προσδοκώμενα αποτελέσματα κατά τον τερματισμό του. Έτσι για κάθε particleυπολογίζεται η τιμή της αντικειμενικής συνάρτησης (fitness value) και εάν αυτή είναι καλύτερη από την καλύτερη τιμή που έχει επιτύχει το συγκεκριμένο particleμέχρι εκείνη τη χρονική στιγμή, τότε αυτή ανατίθεται ως η νέα τιμή του pbest (personal best). Στη συνέχεια από όλες τις τιμές $pbest_i$ που έχουν ανατεθεί σταparticles, επιλέγεται αυτή με την καλύτερη τιμή αντικειμενικής συνάρτησης και συγκρίνεται με το ολικό βέλτιστο gbestπου έχει επιτύχει ο αλγόριθμος μέχρι εκείνη τη στιγμή. Εάν η $pbest_i$ που υπολογίστηκε είναι καλύτερη από το gbest τότε το αντικαθιστά ως η νέα τιμή του gbest. Τέλος επαναϋπολογίζεται η ταχύτητα και η θέση του κάθε particle και η επαναληπτική διαδικασία ξεκινά από την αρχή. Μόλις ολοκληρωθεί ο απαιτούμενος αριθμός επαναλήψεων (γενεών) της αλγοριθμικής διαδικασίας θα έχει γίνει ανάθεση στη μεταβλητή gbest της καλύτερης υπολογιζόμενης τιμής της αντικειμενικής συνάρτησης, η οποία θα αντιστοιχεί στο particleπου εκφράζει την καλύτερη λύση για το δεδομένο πρόβλημα. Υπενθυμίζεται βέβαια ότι ο αλγόριθμος PSOόπως και όλοι οι μεταευριστικοί αλγόριθμοι δεν παρέχουν βεβαιότητα για τον εάν μπορούν να επιτύχουν πάντα την καλύτερη λύση για το εξεταζόμενο πρόβλημα.

4. 3 Παραμετροποίηση του αλγόριθμου PSO

Τα τελευταία χρόνια ο αλγόριθμος PSO έχει χρησιμοποιηθεί σε αρκετές ερευνητικές περιοχές με εξαιρετικά αποτελέσματα. Η επιτυχία του οφείλεται και στο γεγονός ότι γενικά διαθέτει λίγες παραμέτρους, οι οποίες με τις κατάλληλες ρυθμίσεις επιτρέπουν στον αλγόριθμο να δίνει λύσεις σε μεγάλο φάσμα εφαρμοσμένων προβλημάτων βελτιστοποίησης.

Προσπαθώντας να προσαρμόσουμε τον αλγόριθμο PSO σε κάποιο συγκεκριμένο πρόβλημα βελτιστοποίησης παρατηρούμε ότι χρειάζεται να αναλύσουμε και να περιγράψουμε δύο βασικές έννοιες:

- Την κωδικοποίηση των Particles
- Τον καθορισμό της αντικειμενικής συνάρτησης

Ο αλγόριθμος PSO, σε αντίθεση με άλλους μεταευριστικούς αλγόριθμους δέχεται πραγματικούς αριθμούς ως particleskαι έτσι μπορούμε ως παράδειγμα να πούμε ότι κάθε particleμπορεί να είναι της μορφής $(x_1, x_2, x_3) \in \mathbb{R}^3$ και ως αντικειμενική συνάρτηση να θέσουμε την $f(x_1 x_2, x_3) = x_1^4 + x_2^4 + x_3^4$ (fitness function). Στη συνέχεια μπορούμε να εφαρμόσουμε τον αλγόριθμο για τον αριθμό επαναλήψεων που έχουμε ορίσει ή μέχρι την ικανοποίηση κάποιου άλλου κριτηρίου τερματισμού που έχουμε θέσει.

Η διάδοση και η επιτυχία του PSOοφείλεται όπως ήδη έχουμε αναφέρει και στο γεγονός ότι δεν χρειάζεται να εμπλακούμε με τη ρύθμιση πολλών και περίπλοκων παραμέτρων. Παρακάτω ακολουθεί η παρουσίαση των βασικών παραμέτρων του PSO μαζί με συνηθισμένες τιμές που μπορεί να λάβουν:

- Ο αριθμός των particles συνήθως κυμαίνεται μεταξύ 20 – 40 σύμφωνα με τη βιβλιογραφία. Σε πολλές περιπτώσεις ακόμη και 10 – 20 particles είναι ικανά για την επίτευξη λύσης σε κάποια προβλήματα, ενώ από την άλλη πλευρά για την επίτευξη λύσης σε ιδιαίτερα δύσκολα προβλήματα κάποιος μπορεί να δοκιμάσει 100 ή και 200 particles.
- Η διάσταση των particles καθορίζεται από τη φύση και τις απαιτήσεις του κάθε προβλήματος.
- Για τους παράγοντες μάθησης όπως αναφέραμε ορίζεται μια τιμή συνήθως στο διάστημα [0, 4]. Στη δική μας περίπτωση θέσαμε $c_1 = c_2 = 2$.
- Αναφορικά με τον αριθμό των επαναλήψεων του αλγόριθμου αυτός εξαρτάται από τη φύση, τη δυσκολία και την πολυπλοκότητα του προς επίλυση προβλήματος καθώς επίσης και από τη διαθέσιμη υπολογιστική ισχύ.
- Ο συντελεστής αδράνειας w υπολογίζεται όπως είδαμε από τη σχέση (4. 1 c).
- Τα κριτήρια τερματισμού της αλγοριθμικής διαδικασίας καθορίζονται από το μέγιστο χρόνο εκτέλεσης που έχει τεθεί αλλά και το δεδομένο περιθώριο σφάλματος.

Παρατηρήσεις

Αναφορικά με τον αριθμό των particles για το πρόβλημα που εξετάζουμε θα πρέπει να πούμε ότι το Post Enrollment University Course Timetabling,όπως και άλλα παρόμοια προβλήματα αυτής της κατηγορίας θεωρούνται αρκετά δύσκολα και έτσι εάν επιλέξουμε μικρό αριθμό particles η λύση που θα επιτευχθεί είναι εύκολο να παγιδευτεί σε τοπικό βέλτιστο (Local Best) και να μην είναι ικανοποιητική. Από τη άλλη πλευρά εάν επιλέξουμε μεγάλο αριθμό από particles (> 100), οι απαιτήσεις υπολογιστικής ισχύος θα αυξηθούν δραματικά χωρίς ιδιαίτερα μεγάλη αύξηση στην ποιότητα της παραγόμενης λύσης. Ένας καλός συμβιβασμός είναι να θέσουμε τον αριθμό των particlesμεταξύ 40 και 50. Στον δικό μας αλγόριθμο επιλέχθηκε για τον αριθμό των particlesη τιμή 50.

4.4 Υβριδικές Παραλλαγές του αλγόριθμου PSO

Ο αλγόριθμος PSO είναι ένας μεταευριστικός αλγόριθμος που έχει χρησιμοποιηθεί ως βάση για τη δημιουργία πολλών υβριδικών αλγόριθμων βελτιστοποίησης που ανήκουν στην κατηγορία των αλγόριθμων νοημοσύνης σμήνους (Swarm Intelligence), αλλά και της ευρύτερης κατηγορίας των αλγόριθμων Υπολογιστικής Νοημοσύνης. Μερικές τέτοιες περιπτώσεις υβριδικών αλγόριθμωνPSO είναι οι παρακάτω [16], [19]:

- Συνδυασμός του PSO και του αλγόριθμου Bacterial Foreaging Optimization που ανήκει και αυτός στην κατηγορία Swarm Intelligence.
- Συνδυασμός του PSO και του Γενετικού Αλγόριθμου (GA – Genetic Algorithm).
- Συνδυασμός του PSO και του αλγόριθμου Tabu Search.

Οι συνδυασμοί που αναφέρονται παραπάνω είναι πολύ συχνοί σε προβλήματα βελτιστοποίησης όπως το PE-CTT που αναφέρεται σε αυτή την εργασία. Ο αλγόριθμος PSO στη βασική του μορφή συγκλίνει γρήγορα σε κάποια λύση, αλλά δυστυχώς όμως αυτό ταυτόχρονα αποτελεί και το βασικό του μειονέκτημα. Αυτό γιατί μπορεί να συγκλίνει σε λύση που είναι είτε τοπικό βέλτιστο ή αρκετά μακριά από το ολικό βέλτιστο που ψάχνουμε. Έτσι ο αλγόριθμος παγιδεύεται τοπικά και τερματίζεται σε μη - βέλτιστες λύσεις που δεν δίνουν ικανοποιητικό αποτέλεσμα.

Ένα επιπλέον πρόβλημα είναι ότι η απόδοση στοχαστικών αλγορίθμων όπως ο PSO εξαρτάται πολύ από την κατηγορία του προς επίλυση προβλήματος. Η εξάρτηση αυτή έχει να κάνει με την ρύθμιση των παραμέτρων του αλγόριθμου για το εκάστοτε πρόβλημα στο οποίο εφαρμόζεται. Πολύ συχνά η διακύμανση αυτή στις επιδόσεις του αλγόριθμου είναι πολύ μεγάλη. Για παράδειγμα αυξάνοντας την παράμετρο(w - inertia weight) αυξάνουμε την ταχύτητα των particles με αποτέλεσμα αυτά να μπορούν να καλύψουν μεγαλύτερο εύρος στο χώρο λύσεων (search space)με κόστος όμως την μικρότερη κάλυψη τοπικών λύσεων και κατ' επέκταση την ταχύτητα. Προφανώς το ίδιο συμβαίνει και αντίστροφα για μικρή τιμή της παραμέτρου w. Κατά συνέπεια η σωστή ρύθμιση των παραμέτρων για το κάθε επιμέρους πρόβλημα μπορεί να αποτελεί επίπονη εργασία για τον ερευνητή που αντιμετωπίζεται με πειραματισμό και εξαντλητικές δοκιμές

Τα παραπάνω μειονεκτήματα του απλού PSO μπορούν να αντιμετωπιστούν με επιτυχία μέσω της ενσωμάτωσης υβριδικών τεχνικών στον αλγόριθμο έτσι ώστε να συνδυάζονται τα πλεονεκτήματα από κάθε επιμέρους μέθοδο. Τελικά ο τρόπος που ο ερευνητής θα κατασκευάσει έναν υβριδικό αλγόριθμο νοημοσύνης σμήνους εξαρτάται από πολλούς παράγοντες όπως για παράδειγμα ο τύπος και η δυσκολία του προβλήματος βελτιστοποίησης, η διάκριση της συνάρτησης βελτιστοποίησης (συνεχής, διακριτή), οι απαιτήσεις για ταχύτητα, η μεγάλη κάλυψη χώρου λύσεων, οι χρονικοί περιορισμοί κ.ά. [19]

Στην παρούσα ερευνητική εργασία επιχειρήθηκε η δημιουργία ενός υβριδικού αλγόριθμου PSO που συνδυάζει μια τεχνική τοπικής διαδικασίας εκλέπτυνσης (Local Search Procedure) προκειμένου να επιτυγχάνεται η ανατάραξη των υποψήφιων λύσεων ώστε αυτές να μην παγιδεύονται σε τοπικά βέλτιστα. Η διαδικασία αφορά τον $3^{ο}$ ελαστικό περιορισμό και θα παρουσιαστεί αναλυτικότερα, μαζί με τον αντίστοιχο ψευδοκώδικα, στο επόμενο κεφάλαιο όπου γίνεται η παρουσίαση του αλγόριθμουPSO.

Κεφάλαιο 5: Παρουσίαση του Αλγόριθμου

5.1 Η Κωδικοποίηση των Particles

Για την κωδικοποίηση του particleχρησιμοποιούμε ένα δυσδιάστατο πίνακα με γραμμές όσες και οι αίθουσες (rooms) του ιδρύματος και 45 στήλες, δηλαδή όσος είναι ο αριθμός τον εκπαιδευτικών γεγονότων που μπορούν να προγραμματιστούν σε μια αίθουσα μέσα σε μια εβδομάδα (5 x9 = 45). Σε κελί αυτού του πίνακα αποτελεί μπορεί να αντιστοιχηθεί ένα εκπαιδευτικό γεγονός (event) με τη μορφή ενός αριθμού στο διάστημα από 0 μέχρι event_number - 1. Εάν η αίθουσα σε κάποια χρονική περίοδο είναι κενή, χωρίς δηλαδή να έχει προγραμματιστεί κάποιο γεγονός τότε εισάγετε ο αριθμός -1. Εδώ θα πρέπει να αναφέρουμε ότι χρησιμοποιούμε αριθμούς αντί για ονόματα γεγονότων, γιατί με αυτόν τον τρόπο απλοποιείται η λειτουργία του αλγορίθμου. Έτσι για παράδειγμα εάν θέσουμε ως x[r] το r-particle, τότε ο συμβολισμός x[r][3][10] = 12 σημαίνει ότι στην αίθουσα 3 κατά τη $10^{η}$ ώρα του ωρολογίου προγράμματος (δηλαδή την $1^{η}$ ώρα της Τρίτης) στεγάζει το γεγονός 12. Με όμοιο τρόπο ο συμβολισμός x[r][10][30] = -1 μας δείχνει ότι στην αίθουσα με αύξων αριθμό 10 δεν έχει προγραμματιστεί γεγονός κατά την $30^{η}$ χρονική περίοδο της εβδομάδος, δηλαδή την $3^{η}$ ώρα της Πέμπτης.

Στον παρακάτω πίνακα μπορούμε να δούμε την κωδικοποίηση του κάθε particleόπως αυτή προκύπτει από τα παραπάνω:

	Timeslot 1	**Timeslot 2**	**…**	**Timeslot 45**
Room 1				
Room 2				
Room 3				
…				
Room m				

Πίνακας 1- Κωδικοποίηση του Particle

5.2 Η Αρχικοποίηση (Initialization) του κάθε Particle

Η διαδικασία της αρχικοποίησης των particles έχει ως στόχο την απόδοση αρχικών τιμών σε κάθε particleκατά τυχαίο τρόπο. Σκοπός μας είναι να εξασφαλίσουμε την όσο το δυνατόν μεγαλύτερη ποικιλομορφία των αρχικών λύσεων, έτσι ώστε να

δώσουμε τη δυνατότητα στον αλγόριθμο να εξερευνήσει όλο το χώρο του προβλήματος χωρίς να συγκλίνει πρόωρα σε τοπικά μέγιστα ή ελάχιστα.

Η αρχικοποίηση του αλγόριθμου σε μορφή ψευδοκώδικα δίνεται παρακάτω:

```
For each particle p do
{
    For each room r do
    {
        For each timeslot tdo
        {
            x[p][r][t] = -1
        } /* END For */
    /* initial value -1 indicates that there is noevent assignment in this room */
        For each event that takes places in room r do
        {
            While there are unassigned hours in room r do
            {
                choose a timeslot t from 1 to 45 at random
                if(x[p][r][t] == -1 )
                {
                    /*  if there is no event to this timeslot */
                    x[p][c][t] = event
                    reduce by one the teaching hours in room r
                }
            } /* END while */
        } /* END for */
    } /* END for */
} /* ENDfor */
```

Παρατηρούμε πως με την τυχαία ανάθεση τιμών (μέσω γεννήτριας τυχαίων αριθμών) σε κάθε Particleμπορούμε να αυξήσουμε την ποικιλομορφία των παραγόμενων λύσεων, αποτρέποντας την παγίδευση του αλγόριθμου σε τοπικά βέλτιστες λύσεις που θα επιδρούσαν αρνητικά στην ικανότητα του αλγόριθμου να ψάχνει σε όλο το χώρο λύσεων για το ολικό ακρότατο της αντικειμενικής συνάρτησης.

5.3 Η Αντικειμενική συνάρτηση (Fitness Function)

Η αντικειμενική συνάρτηση (fitness function) έχει κεφαλαιώδη σημασία για τη λειτουργία του αλγόριθμου. Μέσω αυτής αξιολογείται το κόστος παραβίασης των ελαστικών και ανελαστικών περιορισμών και κατά συνέπεια η ποιότητα του παραγόμενου προγράμματος. Για κάθε υποψήφιο particleλύσεων υπολογίζεται η τιμή της συνάρτησης αυτής και στη συνέχεια συγκρίνονται οι τιμές που αντιστοιχούν σε κάθε particleπροκειμένου να αναδειχθεί το πιο ποιοτικό από αυτά που αντιστοιχεί και στο καλύτερο ωρολόγιο πρόγραμμα. Η τιμή της αντικειμενικής συνάρτησης είναι αυτή που χρησιμοποιεί ο αλγόριθμος για να ενημερώνει την μεταβλητή personal best του κάθε particlεσε κάθε γενεά – επανάληψη και κατ' επέκταση και την τιμή της μεταβλητής global best που αντιστοιχεί στο particleμε την ποιοτικότερη τιμή, δηλαδή την ελάχιστη τιμή κόστους παραβίασης των περιορισμών. Υπενθυμίζουμε εδώ ότι το gbest είναι το καλύτερο particle, μέχρι εκείνη τη στιγμή εξέλιξης του αλγόριθμου και παίρνει ως τιμή την καλύτερη τιμή της αντικειμενικής συνάρτησης που έχει καταγραφεί μέχρι εκείνη τη στιγμή.

Η αντικειμενική συνάρτηση στον αλγόριθμό μας, που όπως είδαμε οδηγεί την αναζήτηση των λύσεων, είναι συνδυασμός των παρακάτω δύο πραγμάτων:

- Παραβίαση των ανελαστικών περιορισμών που στη ουσία σημαίνει κάποια τιμή για την παράμετρο Distance to Feasibility.
- Παραβίαση των ελαστικών περιορισμών (SCV – Soft Constraint Violation) όπως αυτές τέθηκαν στην παράγραφο3.3 του 3ου κεφαλαίου δηλαδή το άθροισμα SCost1 + SCost2 + SCost3.

Τελικά η αντικειμενική συνάρτηση f(t) για ένα ωρολόγιο πρόγραμμα t (particle t) δίνεται ως το άθροισμα HCV (Hard Constraint Violation) πολλαπλασιαζόμενο με το κατάλληλο βάρος C και των ελαστικών παραβιάσεων SCV (Soft Constraint Violation) και είναι:

$$\boxed{f(t) = HCV(t) \cdot C + SCV(t)}$$

όπου C είναι μια σταθερά που θα πρέπει να είναι μεγαλύτερη από τον μέγιστο αριθμό των πιθανών ελαστικών παραβιάσεων. Ο περιορισμός αυτός μας εξασφαλίζει ότι εφικτά (feasible) αλλά μη ποιοτικά προγράμματα, δηλαδή προγράμματα με παρα-

βιάσεις τον ελαστικών περιορισμών, πολύ σπάνια θα έχουν μεγαλύτερο κόστος από μη εφικτά ωρολόγια προγράμματα.

Η αντικειμενική συνάρτηση αυτή έχει χρησιμοποιηθεί και από άλλους ερευνητές όπως οι Rossi – Doria & Paechter (2004) και οι Naseem Jat – Yang το 2010 [23], [24].Στο σημείο αυτό μπορούμε να πούμε ότι ένα σπουδαίο εύρημα των παραπάνω ερευνητών είναι ότι η σταθερά C δηλαδή το Hard Constraint Weight αρκεί να έχει την τιμή 1 (C = 1), με την έννοια ότι αυτή η τιμή επαρκεί ώστε ο αλγόριθμος να παράγει εφικτά προγράμματα. Αυτό το υπό μία έννοια αναπάντεχο εύρημα εξηγείται με το ότι κάθε ανελαστικός περιορισμός έχει κόστος ανάλογο με τον αριθμό των εμπλεκομένων σπουδαστών, ενώ κάθε ελαστικός περιορισμός έχει κόστος που αναλογεί σε πολύ μικρότερο αριθμό σπουδαστών. Κατά αυτόν τον τρόπο δεν είναι αναγκαίο να δοθεί στην σταθερά C τιμή μεγαλύτερη του 1 και έτσι το HCW στον αλγόριθμο παραμένει σε αυτή τιμή για όλα τα πειράματα.

5.4 Ο Αλγόριθμος

Ο αλγόριθμος που παρουσιάζουμε στη συνέχεια βασίζεται στον αλγόριθμο Νοημοσύνης Σμήνους Particle Swarm Optimization (PSO). Επιπλέον εκτός των μετατροπών και τροποποιήσεων που έχει δεχτεί για να προσαρμοστεί στο πρόβλημα του PE – CCTτου διαγωνισμού ITC 2007, ενσωματώνει και άλλες αλγοριθμικές τεχνικές ώστε να επιτευχθεί η ανατάραξη (perturbation) των υποψήφιων λύσεων και να μην παγιδεύεται σε τοπικά βέλτιστα. Ο αλγόριθμος σε μορφή ψευδοκώδικα παρουσιάζεται παρακάτω:

Initialize p = 50 particles

Initialize personal best fitness of each particle to worst possible

Initialize global best fitness to worst possible

```
For 10000 generations repeat
{
   For each particle x[p] do
   {
         fitness_x = fitness of current particle
         If (fitness_x< personal best fitness of current particle)
         {
               Update the personal best matrix and the personal best fitness
               ofthe current particle
```

If (fitness_x< global best fitness)

Update the global best matrix and the global best fitness

}

If there are hard clashes at particle x[p]

Pick a timeslot1 with hard clashes, at random

Else

Pick a timeslot1 from 1 to 35 at random

Pick a timeslot2 from 1 to 35 at random, different from timeslot1

/* Up to this point two different timeslots have been chosen with one *of themhaving hard clash provided that there are hard clashes at * the particle x[p].Otherwise, just two different timeslots have been *chosen at random.

****/***

Perform swap with probability (timeslot1, timeslot2, current particle, produced S)

/* First step of perturbation on current particle completed producing * the matrix produced S.

****/***

Pick a timeslot1 from 1 to 35 at random

Change random(S, timeslot1, personal best of current particle, producedW)

/* Second step of perturbation on matrix S completed producing the * matrixproduced W.

****/***

Pick a timeslot2 from 1 to 35 at random

Change random (W, timeslot2, global best, produced transformed particle)

/* Third step of perturbation on matrix W completed producing a * transformed particle originated from initial current particle.

```
*Now, an optimization of the transformed particle's fitness and
* structure is attempted by the following while-loop.
*/

fitness_x = fitness of produced transformed particle
While (fitness_x> global best fitness)
{
        Escape = random number in the interval [0, 1]
        If the while loop has completed another 10 more times of
        running
        AND
        Escape < 1/ (1 + current generation) exit the while loop
        /* That means we give the while-loop the opportunity to
        * terminate, no matter what the fitness value is, every time 10
        * more loops are performed.
        */

        /* Assertion: at this point, the while-loop did not terminate
        * due to escape probability. Moreover, the fitness of the
        * transformed particle  is greater than global best fitness.
        * Hence, a new perturbation of matrix W is attempted next,
        * hopping that the resulted particle will have fitness smaller
        * than or equal to the fitness of global best.
        */

        Pick a timeslot1 from 1 to 35 at random
        Change random (W, timeslot1, global best, produced trans-
        formedparticle)
        fitness_x = fitness of produced transformed particle
    } /* end of while loop */
  } /* end of for loop */
} /* end of for - repeat loop */
```

5.5 Θεμελιώδεις Διαδικασίες

Κομβικής σημασίας στοχαστικές διαδικασίες για τον τρόπο με τον οποίο ο αλγόριθμος αναζητά τη λύση είναι οι παρακάτω:

- Perform Swap With Probability (columnslot1, columnslot2, P, produced S)
- Change Random (P, columnslot1, A, produced S)

Λόγω του κομβικού τους ρόλου οι διαδικασίες αυτές αναλύονται πιο διεξοδικά παρακάτω.

5.5.1 Perform Swap With Probability (columnslot1, columnslot2, P, Produced S)

Μια διαδεδομένη τεχνική στη βιβλιογραφία [25] σχετικά με τη δημιουργία PSOαλγορίθμων για την επίλυση προβλημάτων Timetabling,είναι οι διαδοχικές ανταλλαγές (swaps) σε ζεύγη εκπαιδευτικών γεγονότων (events) προκειμένου να επιτευχθεί ποιοτικότερο particle.Η συγκεκριμένη συνάρτηση επιχειρεί την ανταλλαγή ολόκληρων στηλών του πίνακα P που κωδικοποιεί το particleστην περίπτωση που η ανταλλαγή αυτή είναι **έγκυρη** δηλαδή δεν επιφέρει παραβιάσεις ανελαστικών και ελαστικών περιορισμών. Το παραπάνω σκεπτικό μπορεί να φαίνεται προφανές προκειμένου να επιτευχθεί ο στόχος της βελτιστοποίησης του fitnessτου δοσμένου particle, όμως σημαντική διαφορά της συνάρτησης αυτής είναι ότι επιχειρεί στοχαστικά αλλαγές και μεταξύ μη έγκυρων swaps προκειμένου να βελτιώσει την ποικιλομορφία της παραγόμενης λύσης. Προτού όμως αναλύσουμε τη λειτουργία αυτής της συνάρτησης θα πρέπει πρώτα να παραθέσουμε ορισμούς για το τι είναι ένα μη έγκυρο swap και σε ποιες κατηγορίες μπορεί να χωριστεί.

Έτσι έχουμε:

- **Ελαστική fitness** είναι η τιμή που μπορεί να πάρει η αντικειμενική συνάρτηση σε ένα particle όταν για τον υπολογισμό της ελέγχονται και προσμετρούνται μόνο οι ελαστικοί περιορισμοί.
- **Έγκυρο** swap είναι αυτό το οποίο όταν εκτελεστεί στο particleδεν θα επιφέρει παραβιάσεις Ανελαστικών περιορισμών (Hard Constraint clashes), αλλά και επιπλέον εάν επιφέρει παραβιάσεις ελαστικών περιορισμών αυτές δεν θα έχουν επιβαρυντικό κόστος στην ελαστική fitnessπου είχε το particleπριν από το swap.

- **Μη Έγκυρο** swap είναι αυτό το οποίο είτε προκαλεί παραβίαση ανελαστικού περιορισμού είτε προκαλεί παραβίαση ελαστικού περιορισμού με αρνητικό κόστος για την ελαστική fitnesstου particle.

Ο σημαντικός νεωτερισμός αυτής της συνάρτησης είναι ότι όχι μόνο εκτελεί τα έγκυρα swaps, αλλά και με κάποια πιθανότητα εκτελεί και μη έγκυρα swapskαι των δύο παραπάνω κατηγοριών.

Το μη έγκυρο swapεκτελείται εάν ισχύει η παρακάτω συνθήκη:

$r_1 > e^{-\frac{currentCost}{currentGeneration}}$ όπου r_1 είναι τυχαίος πραγματικός αριθμός με $r_1 \in [0,1]$

Η παραπάνω τιμή currentCost είναι το ολικό βέλτιστο της αντικειμενικής συνάρτησης (gbest) που έχει επιτευχθεί μέχρι την τρέχουσα επανάληψη, ενώ η τιμή currentGeneration είναι η τιμή της αντικειμενικής συνάρτησης για την τρέχουσα επανάληψη.

Η παραπάνω διαδικασία perform Swap With Probability επενεργεί στον πίνακα P που είναι το τρέχων particle και τα swapsεκτελούνται μεταξύ των δύο στηλών που περιέχουν τα timeslot1 και timeslot2. Το αποτέλεσμα είναι να δημιουργηθεί ο πίνακας producedS που έχει την ίδια δομή με το particle.

5.5.2 Change Random (P, columnslot1, A, produced S)

Στον αλγόριθμο μας εφαρμόζεται μια στοχαστική αλγοριθμική ιδέα που έχει χρησιμοποιηθεί και από άλλους ερευνητές παρόμοιων προβλημάτων [25], κατά την οποία επιχειρείται η τυχαία επιλογή μιας στήλης από το particlekαι η ανταλλαγή της από τον πίνακα A δηλαδή το globalBest. Το αποτέλεσμα αυτής της διαδικασίας είναι ο παραγόμενος πίνακας producedS. Η διαφορά εδώ με τον αλγόριθμο της βιβλιογραφίας είναι ότι επιχειρείται η αντικατάσταση ολόκληρης στήλης γεγονότων στο particlekαι όχι μόνο μεμονωμένων timeslots. Επιλέγεται δηλαδή τυχαία μια στήλη (columnslot 1) από τον πίνακα P και αντικαθίσταται τα γεγονότα αυτής με τα αντίστοιχα της στήλης του πίνακα A.

5.6 Αλγόριθμος εκλέπτυνσης της λύσης (Local Search Procedure)

Στον αλγόριθμο μας γίνεται χρήση μιας επαναληπτικής διαδικασίας εκλέπτυνσης της λύσης που αφορά την ελαχιστοποίηση του κόστους παραβίασης του 3^{ου} ελαστικού περιορισμού. Πιο συγκεκριμένα η διαδικασία αυτή επιχειρεί να ελαχιστοποιήσει τα μεμονωμένα εκπαιδευτικά γεγονότα σε επίπεδο ημέρας έτσι ώστε ένας σπουδαστής να μην απαιτείται να παρακολουθεί ένα μόνο μεμονωμένο εκπαιδευτικό γεγονός

μέσα σε μια ημέρα (3ος ελαστικός περιορισμός). Η διαδικασία έχει την μορφή τοπικής διαδικασίας έρευνας (Local Search Procedure), γίνεται όπως είπαμε σε επίπεδο ημέρας και εκτελεί ένα μεγάλο αριθμό ανταλλαγών (swaps) στις αναθέσεις των εκπαιδευτικών γεγονότων προκειμένου να περιοριστούν οι μεμονωμένες διδασκαλίες. Ο αλγόριθμος της διαδικασίας σε μορφή ψευδοκώδικα φαίνεται παρακάτω:

```
/* Set a soft constraint weight for cost3 as double of the other soft cost values */
SCcost3  = 2
For each day of the week
{
    F1 = fitness of global best concerning the current day only, calculated
    using new weights.
    if (F1 == 0)
        /* That means that this day has the optimal structure */
        Proceed to the next day
    else
    {
            for 500000 times repeat
            {
            timeslot1 = one of the current day's timeslots, (random chosen)
            timeslot2 = one of the current day's timeslots, different from
                    timeslot1, (random chosen)

/* At this point two different timeslots in the range of the current day have been
* elected at random. The Local Search Procedure now tries  a perturbation of
* the corresponding day structure in hope of producing better global best for
*this day. After the 500000 iterations the LS procedure hopefully produces a
*  better global best for the particle
*/
            Perform swap with probability (timeslot1, timeslot2, global
            best's day, produced transformed day)
            F2 = fitness of global best concerning the transformed day only,
```

```
            calculated using new weights.

        if (F2 <= F1 and there are no gaps for the classes of the
    transformed day)
                {
                Copy the transformed day to the corresponding day of
                global best

    /* i.e. fix the structure of the transformed day by copying it to
    * the global best, until a new transformed day with smaller or
    * equal fitness is resulted during the rest of the 500000 looping
    * times. If this happens then the new transformed day's
    * structure will be fixed copied at global best.
    */
                F1 = F2
                } /* ENDIf */

        } /* END repeat */
        } /* ENDif */
    } /* ENDfor */
```

Αρχικά θα πρέπει να ορίσουμε ένα βάρος για τον 3ο ελαστικό περιορισμό κατά τέτοιο τρόπο ώστε αυτός να αποκτήσει προτεραιότητα ως παράγοντας καθορισμού της λύσης σε σχέση με τους άλλους δύο ελαστικούς περιορισμούς. Υπενθυμίζουμε ότι ο $1^{ος}$ ελαστικός περιορισμός αφορά τα μεμονωμένα γεγονότα της τελευταίας ώρας κάθε ημέρας, ενώ ό $4^{ος}$ ελαστικός περιορισμός αφορά την παρακολούθηση πολλαπλών εκπαιδευτικών γεγονότων σε συνεχόμενες ώρες.

Στη συνέχεια ο αλγόριθμος εκλέπτυνσης υπολογίζει την τιμή fitness της συνάρτησης κόστους από τον πίνακα globalbest αλλά σε επίπεδο ημέρας, δηλαδή ξεχωριστά για κάθε ημέρα. Ο υπολογισμός αυτής της τιμής fitness γίνεται με το νέο βάρος SCcost3 για τον 3ο ελαστικό περιορισμό. Με άλλα λόγια ελέγχονται οι παραβιάσεις των ελαστικών περιορισμών, δίνοντας ιδιαίτερο βάρος στον 3ο περιορισμό, οι οποίες μπορεί να συμβαίνουν σε κάποια συγκεκριμένη ημέρα. Εάν τώρα προκύψει μηδενική

τιμή για τη νέα συνάρτηση κόστους αυτό θα σημαίνει ότι η μέρα που εξετάζεται έχει τη βέλτιστη δομή και δεν επιδέχεται βελτίωση. Αν όμως η συνάρτηση κόστους έχει τιμή μεγαλύτερη του μηδενός, τότε το πρόγραμμα αυτής της ημέρας μπορεί να βελτιωθεί.

Έτσι λοιπόν ξεκινάει ένα loop, το οποίο εκτελείται 500000 φορές, που προσπαθεί να βελτιώσει τη δομή της υπό εξέταση ημέρας. Σε κάθε επανάληψη του αλγόριθμου εκλέπτυνσης επιλέγονται δύο διαφορετικές ώρες από τη συγκεκριμένη ημέρα από τον πίνακα globalbest και εκτελείται η συνάρτηση *performSwapWithProbability* όπως αυτή περιεγράφηκε αναλυτικά νωρίτερα στο κεφάλαιο. Ως παραμέτρους εισόδου στη συνάρτηση έχουμε τις δύο ώρες που έχουν επιλεγεί και το μέρος εκείνο του πίνακα globalbest που αναφέρεται στην υπό εξέταση ημέρα. Ως έξοδο της συνάρτησης λαμβάνουμε έναν πίνακα που περιέχει μόνο τη συγκεκριμένη ημέρα μετά τον μετασχηματισμό που έχει υποστεί. Στη συνέχεια η δομή της παραγόμενης ημέρας αξιολογείται συγκρίνοντας την fitnesstης με την αντίστοιχη τιμή που είχε πριν από το μετασχηματισμό της. Έτσι σε περίπτωση που η νέα τιμή είναι μικρότερη ή ίση από την παλαιά αλλά και επιπλέον δεν παρουσιάζονται κενά στο πρόγραμμα της ημέρας, τότε η δομή της παραχθείσας ημέρας εισέρχεται στον πίνακα globalbest και η τιμή fitnessενημερώνεται με το νέο καλύτερο αποτέλεσμα, διαφορετικά ο πίνακας globalbest παραμένει αμετάβλητος. Σημειώνεται ότι τα κενά στο πρόγραμμα επιβάλλεται να ελεγχθούν, γιατί οι ανταλλαγές (swaps) που συμβαίνουν κατά την επαναληπτική διαδικασία του αλγόριθμου θα μπορούσαν να επηρεάσουν το πρόγραμμα, στις περιπτώσεις που αυτό σε κάποιες ημέρες δεν είναι συμπληρωμένο και στις εννέα του ώρες. Τέλος είναι σημαντικό να αναφέρουμε, ότι αφού οι μετασχηματισμοί συμβαίνουν σε επίπεδο ημέρας, διατηρούνται τα αποτελέσματα που έχει επιτύχει ο κύριος αλγόριθμος μέχρι εκείνη τη στιγμή αναφορικά με τους άλλους δύο ελαστικούς περιορισμούς.

Βιβλιογραφικές Αναφορές

1. S. Fotakis, D. Spirakis, Algorithms and Complexity, Hellenic Open University 2002.
2. G. Beligiannis, I. Tassopoulos, A hybrid particle swarm optimization based algorithm for high school timetabling problems, Applied Soft Computing 12 (2012).
3. P. Kechagiopoulos, G. Beligiannis, Solving the Urban Transit Routing Problem using a particle swarm optimization based algorithm, Applied Soft Computing 21 (2014).
4. T. Cormen, C. Leiserson, R. Rivest, C. Stein, Introduction to Algorithms 2nd edition, MIT Press, McGraw – Hill Book Company.
5. G. Beligiannis, C. Moschopoulos and S. D. Likothanasis. A genetic algorithm approach to school timetabling. Journal of the Operational Reaserch Society, 2009.
6. Thomas Weiss. Global Optimization Algorithms – Theory and Application, 2009.
7. S. Ceschia, L. Di Gaspero, A. Schaerf, Design, engineering and experimental analysis of a simulated annealing approach to the Post Enrolment course timetabling problem. Computers & Operations Research Journal 39 (2012) 1615 – 1624.
8. Rhydian Lewis, Ben Paechter, Barry McCollum, Post Enrolment based Course Timetabling: A description of the problem used for track two of the Second International Timetabling Competition. Cardiff Business School, Cardiff University, August 2007.
9. http://en.wikipedia.org/wiki/Artificial_intelligence
10. http://en.wikipedia.org/wiki/Computational_intelligence
11. http://www.swarmintelligence.org/tutorials.php
12. http://en.wikipedia.org/wiki/Metaheuristic
13. http://en.wikipedia.org/wiki/Particle_swarm_optimization
14. http://en.wikipedia.org/wiki/Ant_Colony_Optimization_Algorithms
15. http://www.unitime.org/exam_description.php

16. Brownlee Jason PhD. Clever Algorithms: Nature – Inspired Programming Recipes
17. Blum C., Roli A. Metaheuristics in Combinatorial Optimization: Overview and Conceptual comparison, ACM Computing Surveys (CSUR) 2003.
18. Kwang .Y. Lee, M. El - Sharkawi. Modern Heuristic Optimization Techniques, Theory and Applications to Power Systems, 2008.
19. Premalatha K., Natarajan A. M. Hybrid PSO & GA algorithm for Global Maximization, Konga Engineering College, India, 2009.
20. Van den Bergh F. and Engelbrecht A. P. A Cooperative Approach to Particle Swarm Optimization, IEEE Transactions on Evolutionary Computation, pp 225 – 239, 2005.
21. Even S., Hui A., Shamir A. On the complexity of timetable and multi commodity flow problems, SIAM Journal on Computing, 691 – 703, 1976.
22. M. R Garey, D. S Johnson. Computers and Intractability – A guide to NP – Completeness, 1st edition, San Francisco, WH Freeman and Co, 1979.
23. Rossi – Doria O. and Paechter B. A memetic algorithm for university course timetabling, In procedure of combinatorial optimization, 2004.
24. NaseemJat S. and Yang S. A hybrid algorithm and tabu search approach for post enrolment course timetabling, published online, 3 november 2010.
25. Shu Chuan Chu, Yi – Tin Chen, JiunHuei Ho. Timetabling Scheduling Using Particle Swarm Optimization. Proceedings of the first conference of Innovative Computing Information and Control (ICICIC 2006).

Παράρτημα

Περιγραφή της μορφής και της κωδικοποίησης των αρχείων εισόδου

Το κάθε αρχείο εισόδου πρέπει να είναι ή μπορεί να μετατραπεί σε αρχείο απλού κειμένου (txtfile), ώστε να μπορεί να διαβαστεί από το πρόγραμμα Στην παρουσίαση της δομής των αρχείων που ακολουθεί χρησιμοποιήθηκε σε μεγάλο βαθμό η περιγραφή που είχε δοθεί από τους διοργανωτές στην αντίστοιχη σελίδα του διαγωνισμού που αναφέρετε στα στιγμιότυπα εισόδου και δίνεται παρακάτω:

(http://www.cs.qub.ac.uk/itc2007/postenrolcourse/course_post_index_files/Inputformat.htm).

Τα αρχεία του διαγωνισμού έχουν την ίδια κωδικοποίηση με αυτή του προγενέστερου διαγωνισμού ITC 2003, εκτός από δύο νέους πίνακες που έχουν προστεθεί στο τέλος. Όλα τα αρχεία περιέχουν ακέραιους αριθμούς και τα διαχωριστικά σε μια γραμμή είναι κενά διαστήματα. Παρακάτω αρχικά παρουσιάζεται η κωδικοποίηση των αρχείων εισόδου με τρόπο παρόμοιο με αυτόν της επίσημης ιστοσελίδας και στη συνέχεια θα δοθεί πραγματικό στιγμιότυπο (dataset1.txt) με αναλυτική παρουσίαση της κωδικοποίησης:

1η Γραμμή

Αρ. Γεγονότων – Αρ. Αιθουσών – Αρ. Προδιαγραφών – Αρ. Σπουδαστών

Μία γραμμή ανά αίθουσα

Χωρητικότητα της αίθουσας

Μία γραμμή για κάθε σπουδαστή/γεγονός

Η τιμή της γραμμής είναι είτε (1) είτε (0) ανάλογα με το εάν ο σπουδαστής παρακολουθεί το συγκεκριμένο γεγονός είτε όχι. Έτσι για παράδειγμα εάν έχουμε 3 σπουδαστές και 4 γεγονότα τότε το αρχείο περιέχει τις παρακάτω 12 γραμμές:

0
1
0
0
1
1

0
0
0
0
1
0

Οι γραμμές αυτές μετασχηματίζονται στον παρακάτω 3 x 4 πίνακα (students/events):

0	1	0	0
1	1	0	0
0	0	1	0

Από τον παραπάνω πίνακα προκύπτει ότι:

- ο $1^{ος}$ σπουδαστής παρακολουθεί το $2^{ο}$ γεγονός
- ο $2^{ος}$ σπουδαστής παρακολουθεί τα γεγονότα 1 και 2
- ο $3^{ος}$ σπουδαστής παρακολουθεί το $3^{ο}$ γεγονός

Μία γραμμή για κάθε αίθουσα/προδιαγραφή

Η τιμή της γραμμής είναι είτε (1) είτε (0) ανάλογα με το εάν η αίθουσα πληροί τη συγκεκριμένη προδιαγραφή ή όχι. Έτσι για παράδειγμα εάν έχουμε 3 αίθουσες και 4 προδιαγραφές τότε το αρχείο περιέχει τις παρακάτω 12 γραμμές:

0
1
0
0
1
1
0
0

0
0
1
0

Οι γραμμές αυτές μετασχηματίζονται στον παρακάτω 3 x 4 πίνακα (room/features):

0	1	0	0
1	1	0	0
0	0	1	0

Από τον παραπάνω πίνακα προκύπτει ότι:

- η $1^{η}$ αίθουσα πληροί τη $2^{η}$ προδιαγραφή
- η $2^{η}$ αίθουσα πληροί τις προδιαγραφές 1 και 2
- η $3^{η}$ αίθουσα πληροί την $3^{η}$ προδιαγραφή

Μία γραμμή για κάθε γεγονός/προδιαγραφή

Η τιμή της γραμμής είναι είτε (1) είτε (0) ανάλογα με το εάν το εκπαιδευτικό γεγονός απαιτεί μια συγκεκριμένη προδιαγραφή ή εάν κάτι τέτοιο δεν απαιτείται. Έτσι για παράδειγμα εάν έχουμε 3 γεγονότα και 4 προδιαγραφές τότε το αρχείο περιέχει τις παρακάτω 12 γραμμές:

0
1
0
0
1
1
0
0
0

0
1
0

Οι γραμμές αυτές μετασχηματίζονται στον παρακάτω 3 x 4 πίνακα (events/features):

0	1	0	0
1	1	0	0
0	0	1	0

Από τον παραπάνω πίνακα προκύπτει ότι:

- το $1^{ο}$ γεγονός απαιτεί τη $2^{η}$ προδιαγραφή
- το $2^{ο}$ γεγονός απαιτεί τις προδιαγραφές 1 και 2
- το $3^{ο}$ γεγονός απαιτεί την $3^{η}$ προδιαγραφή

Μία γραμμή για κάθε γεγονός/χρονική περίοδο

Η τιμή της γραμμής είναι είτε (1) είτε (0) ανάλογα με το εάν το εκπαιδευτικό γεγονός μπορεί να προγραμματιστεί σε μια χρονική περίοδο (1) ή εάν δε μπορεί (0). Έτσι για παράδειγμα εάν έχουμε 3 γεγονότα και 4 διαθέσιμες χρονικές περιόδους τότε το αρχείο περιέχει τις παρακάτω 12 γραμμές:

1
0
1
1
0
0
1
0
1
1

1
1

Οι γραμμές αυτές μετασχηματίζονται στον παρακάτω 3 x 4 πίνακα (events/timeslots):

1	0	1	1
0	0	1	0
1	1	1	1

Από τον παραπάνω πίνακα προκύπτει ότι:

- το $1^{ο}$ γεγονός μπορεί να προγραμματισθεί στις περιόδους 1, 3, 4 αλλά όχι στην περίοδο 2
- το $2^{ο}$ γεγονός μπορεί να προγραμματισθεί μόνο στην περίοδο 3
- το $3^{ο}$ γεγονός μπορεί να προγραμματισθεί σε οποιαδήποτε από τις 4 διαθέσιμες χρονικές περιόδους

Μία γραμμή για κάθε γεγονός/γεγονός

Η τιμή της γραμμής είναι (1) εάν το $1^{ο}$ γεγονός πρέπει να προγραμματισθεί πριν από το $2^{ο}$, (-1) εάν το $1^{ο}$ γεγονός πρέπει να προγραμματισθεί μετά το $2^{ο}$ και (0) εάν το εκπαιδευτικό γεγονός δεν απαιτεί συγκεκριμένη διαδοχικότητα. Έτσι για παράδειγμα εάν έχουμε 4 γεγονότα τότε το αρχείο περιέχει τις παρακάτω 16 γραμμές:

0
-1
0
0
1
0
0
1
1

0
0
0
0
-1
0
0

Οι γραμμές αυτές μετασχηματίζονται στον παρακάτω 4x 4 πίνακα (events/events):

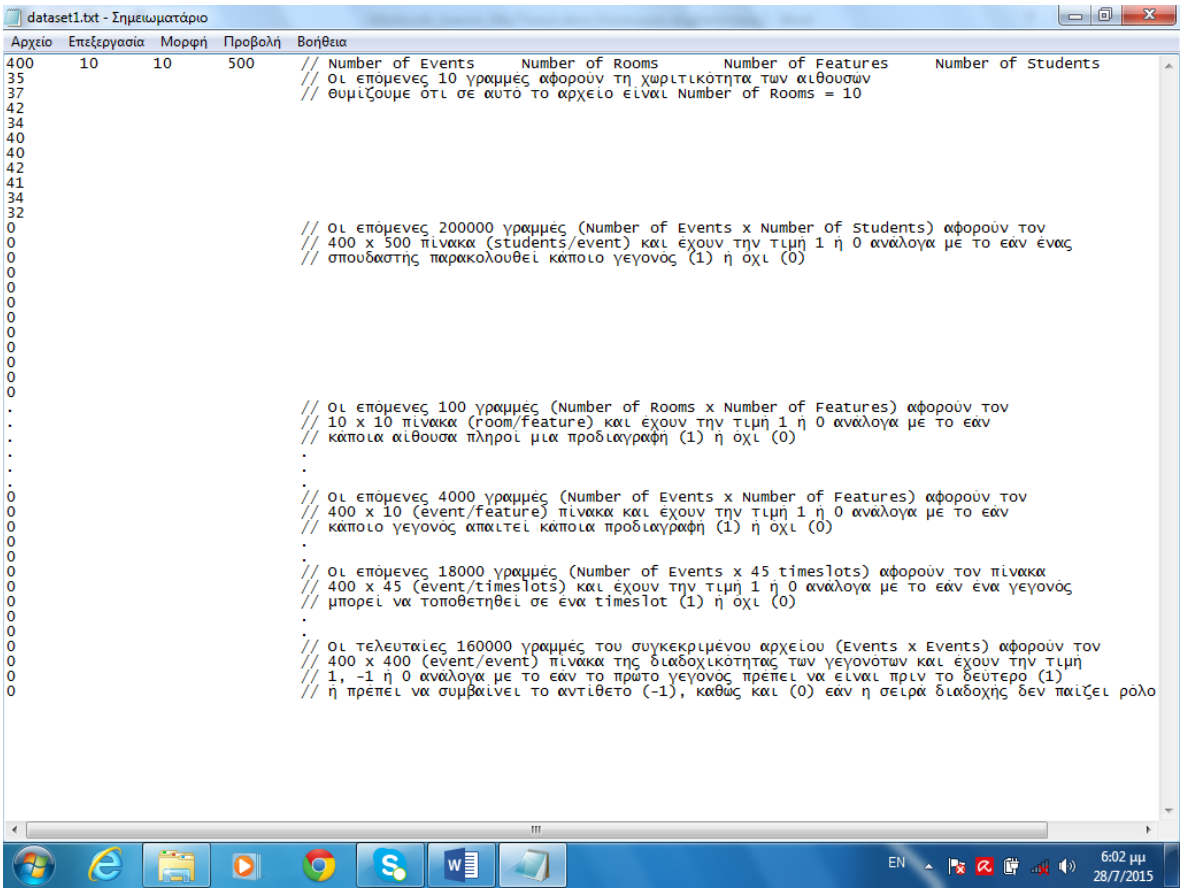
0	-1	0	0
1	0	0	1
1	0	0	0
0	-1	0	0

Από τον παραπάνω πίνακα προκύπτει ότι:

- το $2^{ο}$ γεγονός πρέπει να προγραμματισθεί πριν από το $1^{ο}$ και παρομοίως το $1^{ο}$ πρέπει να προγραμματισθεί μετά το 2ο
- το $2^{ο}$ γεγονός πρέπει να προγραμματισθεί πριν από το $4^{ο}$ γεγονός και παρομοίως το $4^{ο}$ γεγονός πρέπει να προγραμματισθεί μετά το $2^{ο}$

Παρακάτω δίνεται μια απεικόνιση οθόνης (screenshot) που δείχνει ένα μικρό μόνο μέρος από το dataset1.txt,το οποίο επιπλέον διαφέρει από το κανονικό αρχείο απλού κειμένου στο ότι περιέχει σχόλια για τα δεδομένα που αναγράφονται προκειμένου να διευκολυνθεί ο αναγνώστης.

Επειδή κάθε τέτοιο αρχείο περιέχει πολλές χιλιάδες γραμμές χρησιμοποιούμε τις τελείες (…) για να δείξουμε ότι στη θέση τους μπαίνουν γραμμές με ψηφία, ενώ προφανώς κάτι τέτοιο δεν συμβαίνει στα πραγματικά αρχεία. Επίσης στην περιγραφή της δομής των αρχείων, όπως αναφέρθηκε και παραπάνω έχουν προστεθεί σχόλια μετά από το σύμβολο «//» που και αυτά προφανώς δεν εμφανίζονται στο αρχείο κειμένου που θα διαβάσει ο αλγόριθμος.


dataset1.txt - Σημειωματάριο
Αρχείο Επεξεργασία Μορφή Προβολή Βοήθεια
400 10 10 500 // Number of Events Number of Rooms Number of Features Number of Students
35 // Οι επόμενες 10 γραμμές αφορούν τη χωριτικότητα των αιθουσων
37 // θυμίζουμε ότι σε αυτό το αρχείο είναι Number of Rooms = 10
42
34
40
40
42
41
34
32
0 // Οι επόμενες 200000 γραμμές (Number of Events x Number Of Students) αφορούν τον
0 // 400 x 500 πίνακα (students/event) και έχουν την τιμή 1 ή 0 ανάλογα με το εάν ένας
0 // σπουδαστης παρακολουθεί κάποιο γεγονός (1) ή όχι (0)
0
0
0
0
0
0
0
0
0
. // Οι επόμενες 100 γραμμές (Number of Rooms x Number of Features) αφορούν τον
. // 10 x 10 πίνακα (room/feature) και έχουν την τιμή 1 ή 0 ανάλογα με το εάν
. // κάποια αίθουσα πληροί μια προδιαγραφή (1) ή όχι (0)
.
.
.
0 // Οι επόμενες 4000 γραμμές (Number of Events x Number of Features) αφορούν τον
0 // 400 x 10 (event/feature) πίνακα και έχουν την τιμή 1 ή 0 ανάλογα με το εάν
0 // κάποιο γεγονός απαιτεί κάποια προδιαγραφή (1) ή όχι (0)
0
0
0 // Οι επόμενες 18000 γραμμές (Number of Events x 45 timeslots) αφορούν τον πίνακα
0 // 400 x 45 (event/timeslots) και έχουν την τιμή 1 ή 0 ανάλογα με το εάν ένα γεγονός
0 // μπορεί να τοποθετηθεί σε ένα timeslot (1) ή όχι (0)
0
0
0 // Οι τελευταίες 160000 γραμμές του συγκεκριμένου αρχείου (Events x Events) αφορούν τον
0 // 400 x 400 (event/event) πίνακα της διαδοχικότητας των γεγονοτων και έχουν την τιμη
0 // 1, -1 ή 0 ανάλογα με το εάν το πρώτο γεγονός πρέπει να είναι πριν το δεύτερο (1)
0 // ή πρέπει να συμβαίνει το αντίθετο (-1), καθώς και (0) εάν η σειρα διαδοχής δεν παίζει ρολο
EN
6:02 μμ
28/7/2015

Παρουσίαση των βασικότερων συναρτήσεων του προγράμματος

Παρακάτω περιγράφονται όλες οι συναρτήσεις που χρησιμοποιούνται από το πρόγραμμα μας και υλοποιούν τον αλγόριθμο PSO. Για ένα σημαντικό αριθμό από αυτές τις συναρτήσεις – εκτός ίσως από τις ιδιαίτερα απλές – έχουν δημιουργηθεί βοηθητικά προγράμματα (testers) που περιέχουν hard – codedδεδομένα για την σωστή επαλήθευση της λειτουργίας τους. Επίσης έχει τοποθετηθεί πιο περιορισμένος περιληπτικός σχολιασμός στην αρχή του προγράμματος κατά την παράθεση των πρωτοτύπων αυτών των συναρτήσεων.

1. *int* ***initializeRandomness(****int* ***seed).***Βοηθητική συνάρτηση που αρχικοποιεί με seed που δίνεται από το χρήστη τη ρουτίνα που περιγράφεται παρακάτω για την παραγωγή τυχαίων αριθμών σε ένα διάστημα. Κατά αυτόν τον τρόπο το seedδίνεται από τον χρήστη αντί να παράγεται τυχαία από το ρολόϊ του συστήματος δίνοντας ένα μικρό επιπλέον έλεγχο στο χρήστη του προγράμματος.
2. *int* ***random(****int* ***lower,*** *int* ***upper).***Βοηθητική συνάρτηση που παράγει έναν τυχαίο ακέραιο αριθμό μέσα στο διάστημα ακεραίων [lower, upper].Αν και πρόκειται για ιδιαίτερα απλή συνάρτηση είναι θεμελιώδης για έναν μεταευριστικό αλγόριθμο, αφού μέσω αυτής μπορούμε να προσδώσουμε τυχαιότητα και στοχαστικότητα (μέσω πιθανοτήτων) σε συναρτήσεις του αλγόριθμου που εκτελούν ανταλλαγές στηλών και γραμμών στον πίνακα που κωδικοποιεί το particle.
3. *int* ***copyMatrices(****int* ***begin,*** *int* ***end,*** *int* ***b[ ][45],*** *int* ***a[ ][45],*** *int* ***dimension).***Η συνάρτηση αυτή αντιγράφει τα περιεχόμενα του πίνακα A στον πίνακα B. Παρατηρούμε ότι οι πίνακες έχουν 45 στήλες όσες δηλαδή και το particletης λύσης σύμφωνα με την κωδικοποίηση που έχει γίνει.Μέσω των παραμέτρων της συνάρτησης έχουμε πλήρη έλεγχο για το ποιο μέρος του πίνακα aθα αντιγραφεί στον πίνακα b. Η παράμετρος *dimension*καθορίζει πόσες γραμμές του πίνακα θα αντιγραφτούν, ενώ μέσω των παραμέτρων *begin*και *end*καθορίζουμε πόσες από τις 45 στήλες του aπίνακα θα αντιγραφούν στον b. Έτσι για παράδειγμα δίνοντας την εντολή *copyMatrices(0, 45, b, a, 3)*εξασφαλίζουμε την αντιγραφή όλων των στοιχείων του aστον πίνακα b.
4. *int* ***swap(****int* ***a[][45],*** *int* ***room1,*** *int* ***timeslot1,*** *int* ***timeslot2).*** Εκτελεί εναλλαγή δύο κελιών στην ίδια γραμμή του πίνακα. Σύμφωνα με την κωδικοποίηση που έχουμε κάνει για τα particles στις γραμμές αναγράφονται οι διαθέσιμες αίθου-

σες (rooms), κατά συνέπεια η συνάρτηση αυτή εναλλάσσει δύο διαφορετικές χρονικές στιγμές (timeslots) για κάποια συγκεκριμένη αίθουσα. Για παράδειγμα η εντολή *swap(a, 0, 2, 4)*εναλλάσσει το timeslot3 με το timeslot5 για την $1^{η}$ αίθουσα (γραμμή 0) του πίνακα των particle.

5. *int* ***swapTwoColumns(****int* ***a[][45],*** *int* ***timeslot1,*** *int* ***timeslot2,*** *int* ***roomsNumber).*** Χρησιμοποιεί την προηγούμενη συνάρτηση swap για να εκτελέσει εναλλαγές timeslots που αφορούν περισσότερες γραμμές (rooms) ή και όλες του πίνακα των particles. Ο αριθμός αυτός καθορίζεται από την μεταβλητή *roomsNumber*. Για παράδειγμα δίνοντας την εντολή *swapTwoColumns(a, 0, 2, 2)*στον πίνακα a[3][45] εξασφαλίζουμε την εναλλαγή των στηλών 1, 2, 3 του συγκεκριμένου πίνακα.
6. *int* ***checkSingleClassEmptyPeriods****(int* ***begin****, int* ***end****, int* ***a[][45]****, int* ***roomNumber,****int* ***showResults).*** Η συνάρτηση επιστρέφει τις κενές χρονικές περιόδους που έχει ανά πάσα στιγμή κάποια αίθουσα του εκπαιδευτικού ιδρύματος. Στη συνάρτηση μπορούμε να περάσουμε ως παράμετρο την ημέρα της εβδομάδος που θα αρχίσει η μέτρηση των κενών και την αίθουσα για την οποία θα ισχύσει η μέτρηση (η οποία αντιστοιχεί σε κάποια γραμμή του πίνακα των particles. Θα πρέπει να πούμε ότι η τελευταία ώρα κάθε ημέρας εάν είναι κενή δεν προσμετράτε, αφού θα ήταν παραβίαση ανελαστικού περιορισμού να τοποθετήσουμε εκεί μεμονωμένο γεγονός.
7. *void* ***swapTableElements****(int* ***table[][2]****, int* ***point1****, int* ***point2).*** Εναλλάσσει δύο γραμμές σε έναν πίνακα 2 στηλών. Έτσι για παράδειγμα δίνοντας την εντολή *swapTableElements(m, 0, 3)*στον πίνακα m[2][4] εναλλάσονται τα στοιχεία της $1^{ης}$ με την $4^{η}$ γραμμή.
8. *int* ***swapIsValid*** *(int* ***begin****, int* ***end****, int* ***a[[45],*** *int* ***room1****, int* ***numberOfEvents****, int* ***numberOfRooms****, int* ***numberOfStudents,*** *int* ***timeslot1****, int* ***timeslot2,*** *int* ***sAvail[numberOfStudents][45]).*** Η συνάρτηση αυτή επιστρέφει (1) εάν είναι επιτρεπτή μια ανταλλαγή στοιχείων στο particle. Αυτό συμβαίνει εάν κατά την ανταλλαγή αυτή δεν παρουσιάζονται παραβιάσεις ανελαστικών περιορισμών. Στη συνέχεια η συνάρτηση ελέγχει για παραβιάσεις ελαστικών περιορισμών που τυχών μπορεί να συμβαίνουν μετά από κάποιο swap και υπολογίζει το αντίστοιχο κόστος παραβίασης των περιορισμών αυτών. Εάν το κόστος παραβίασης μετά την ανταλλαγή είναι μεγαλύτερο από ότι πριν την

ανταλλαγή η συνάρτηση επιστρέφει την τιμή (2) μαζί με την απόλυτη διαφορά του κόστους πριν και μετά την ανταλλαγή. Η συνάρτηση αυτή αποτελεί βασικό κομμάτι της πολύ σημαντικής συνάρτησης *performSwapWithProbability* που αποτελεί ένα από τα πιο σημαντικά τμήματα του αλγόριθμου.

9. *Int* ***acceptSwapWithProbability(****int* ***begin****, int* ***end****, int* ***a[] [45],*** *int* ***room1****, int* ***numberOfEvents****, int* ***numberOfRooms****, int* ***numberOfStudents,*** *int* ***timeslot1****, int* ***timeslot2,*** *int* ***sAvail[numberOfStudents][45]).*** Αποτελεί και αυτή μέρος της συνάρτησης *performSwapWithProbability* και επιστρέφει την τιμή (1) αν το swap μεταξύ δύο χρονικών στιγμών είναι αποδεκτό ή (-1) εάν δεν είναι. Στην δεύτερη περίπτωση το swap μπορεί να εκτελεστεί βάσει συγκεκριμένης πιθανότητας.
10. *int* ***performSwapWithProbability(*****int** ***begin,*** **int** ***end,*** **int** ***p,*** **int** ***a[][45],*** **int** ***timeslot1,*** **int** ***timeslot2,*** *int* ***numberOfRooms,*** *int* ***numberOfEvents****, int* ***numberOfStudents,*** *int* ***times,*** *int* ***sAvail[numberOfStudents][45]).*** Η διαδικασία αυτή έχει εξηγηθεί αναλυτικά στο $5^{ο}$ κεφάλαιο. Εδώ μόνο να πούμε πως αυτή η συνάρτηση συνδυάζει και συνοψίζει τα αποτελέσματα των δύο προηγούμενων συναρτήσεων προκειμένου να εκτελέσει τις αντίστοιχες εναλλαγές στηλών στο particle βάσει των πιθανοτήτων που έχουν ήδη αναφερθεί στο αντίστοιχο κεφάλαιο.
11. *int* ***calculateSoftConstraintFitness*** *(int* ***a[][45]****, int* ***numberOfRooms,*** *int* ***numberOfEvents****, int* ***numberOfStudents****, int* ***sAvail[numberOfStudents][45])****.* Η εν λόγω συνάρτηση επιστρέφει το συνολικό κόστος παραβίασης των ελαστικών περιορισμών ενός particle *a[] [45]*.Ως παραμέτρους εκτός από το particle, δέχεται τον αριθμό των αιθουσών, τον αριθμό των γεγονότων, τον αριθμό των σπουδαστών και τον παραγόμενο πίνακα *sAvail[][45]*. Εκτός από το particle – υποψήφιο ωρολόγιο πρόγραμμα, τα υπόλοιπα arguments έχουν διαβαστεί κατά την έναρξη του αλγόριθμου από το αρχείο δεδομένων απλού κειμένου.
12. *Int* ***distToFeasibility(****int****a[][45]****, int* ***numberOfRooms,*** *int* ***numberOfEvents****, int* ***numberOfStudents****, int* ***attendsMtrx[numberOfStudents][numberOfEvents]).*** Η συνάρτηση αυτή επιστρέφει το μέτρο της «απόστασης από το εφικτό», δηλαδή των αριθμό των σπουδαστών που μετέχουν σε εκπαιδευτικά γεγονότα που δεν έχουν τοποθετηθεί στο

ωρολόγιο πρόγραμμα λόγω των ανελαστικών περιορισμών που υπάρχουν. Οι παράμετροι της συνάρτησης είναι όμοιοι με την συνάρτηση *calculateSoftConstraintFitness* εκτός από τον πίνακα *attendsMtrx* που στην ουσία είναι ο πίνακας *students/events* που διαβάζετε από το αρχείο δεδομένων και μας δείχνει ποιοι σπουδαστές παρακολουθούν τα διαθέσιμα γεγονότα.