Ανοικτό Πανεπιστήμιο Κύπρου

Σχολή Θετικών και Εφαρμοσμένων Επιστημών

Μεταπτυχιακή Διατριβή
στα Πληροφοριακά Συστήματα

Ανάπτυξη Εξεταστικού Ημερολογίου

Γεώργιος Παπαϊωάννου

Επιβλέπων Καθηγητής
Δημήτριος Καλλές

Μάιος 2012

Ανοικτό Πανεπιστήμιο Κύπρου

Σχολή Θετικών και Εφαρμοσμένων Επιστημών

Ανάπτυξη Εξεταστικού Ημερολογίου

Γεώργιος Παπαϊωάννου

Επιβλέπων Καθηγητής
Δημήτριος Καλλές

Η παρούσα μεταπτυχιακή διατριβή υποβλήθηκε
προς μερική εκπλήρωση των απαιτήσεων για απόκτηση

μεταπτυχιακού τίτλου σπουδών
στα Πληροφοριακά Συστήματα

από τη Σχολή Θετικών και Εφαρμοσμένων Επιστημών
του Ανοικτού Πανεπιστημίου Κύπρου

Μάιος 2012

Περίληψη

Η κατάρτιση ενός εξεταστικού ημερολογίου σε ένα τριτοβάθμιο εκπαιδευτικό ίδρυμα είναι μία ιδιαίτερα επίπονη διαδικασία για το διοικητικό προσωπικό του πανεπιστημίου ιδιαίτερα όταν η διαδικασία αυτή γίνεται χωρίς κάποια αυτοματοποίηση.

Το συγκεκριμένο πρόβλημα ανήκει στην οικογένεια των προβλημάτων χρονοπρογραμματισμού (Scheduling) και ανήκει στην οικογένεια NP – Complete. Η κατάρτιση του εξεταστικού ημερολογίου έχει επιλυθεί με διάφορες τεχνικές προγραμματισμού.

Στην παρούσα διπλωματική θα επιλυθεί το πρόβλημα κατάρτισης του εξεταστικού ημερολογίου, με την βοήθεια των αλγορίθμων εξέλιξης (Evolutionary Algorithms). Με την βοήθεια των γενετικών αλγορίθμων έχουν επιλυθεί αρκετά προβλήματα NP - Complete. Οι αλγόριθμοι αυτοί προσομοιώνουν την βιολογική εξέλιξη των ειδών, σε ένα πρόβλημα προγραμματισμού. Η λύση του προβλήματος που προκύπτει από την εκτέλεση τους, ικανοποιεί τους περιορισμούς που μπορεί να περιέχει το πρόβλημα αλλά μπορεί να μην είναι η βέλτιστη δυνατή.

Ο αλγόριθμος που προτείνεται για την επίλυση του εξεταστικού ημερολογίου δίνει ιδιαίτερη έμφαση στην ικανοποίηση των περιορισμών που αφορούν τους φοιτητές και στην επιλογή των αιθουσών και των χρονοθυρίδων με τέτοιο τρόπο ώστε να μην υπάρχει περίπτωση να μην υπάρχουν διαθέσιμες θέσεις για τους φοιτητές που πρόκειται να εξεταστούν.

Στο τελευταίο κεφάλαιο ο αλγόριθμος εκτελείται με πραγματικά δεδομένα από το Ελληνικό Ανοικτό Πανεπιστήμιο σε μικρά προβλήματα (όσον αφορά τον αριθμό των μαθημάτων) και παράγει αποδεκτά εξεταστικά ημερολόγια. Σε μεγάλα προβλήματα παρατηρείτε ότι ο γενετικός αλγόριθμος έχει πάρα πολύ μεγάλες απαιτήσεις για υπολογιστική ισχύ, οπότε προτείνετε η εκτέλεση του αλγορίθμου σε Grid υπολογιστές.

Λέξεις Κλειδιά: Χρονοπρογραμματισμός γεγονότων, Γενετικοί Αλγόριθμοι, Κατασκευή Εξεταστικού Ημερολογίου, Αλγόριθμοι Βελτιστοποίησης

Περιεχόμενο: Κείμενο, Βιογραφικές Αναφορές, Παραρτήματα, Εικόνες, Κώδικας Προγράμματος

Summary

The examination timetable's construction, in an university, is a extremely time consumed procedure for the university personnel especially when it is not automated.

It belongs into the Scheduling Algorithm family and is been classified as an NP – Complete Problem. There are many programming methods that were used in order to produce feasible timetables with great success.

In this Master thesis, the examination timetable construction will be solved using Evolutionary Algorithms, which is a common method for solving NP – Complete Algorithmic problems. They simulate the biological evolution of species into a specific programming problem. The problem's solution might not be the best solution that can be afforded but it will satisfy all the constrains that the problem contains.

The proposed algorithm for the construction of the examination timetable emphasizes into the satisfaction of the constraints which are related to the students and the selection of the examination rooms and time slots in such a way that eliminates the luck of examination seats for the students that are going to be examined.

The evolution algorithm is executed with this year's students and lessons data from the Hellenic Open University and produces feasible examination timetables, when the population of lessons is smaller than sixty. In larger populations, the evolutionary algorithmic demands much processing power and time and we're proposed the usage of a Grid Computer for the algorithm's execution.

Keywords: Job Scheduling, Genetic Algorithms, Construction of Examination Timetable, Evolution Algorithms, NP – Complete Algorithms

Content: Text, References, Appendices, Pictures, Algorithm software

Ευχαριστίες

Ευχαριστώ τον Καθηγητή Δημήτριο Καλλέ για την αμέριστη αρωγή και συμπαράσταση του όποτε χρειάστηκε και τον Δρ. Κων/νο Καλπούζο για τις αμέτρητες συζητήσεις επί του θέματος. Επίσης, θα ήθελα να ευχαριστήσω το Μητρώο του Ελληνικού Ανοικτού Πανεπιστημίου για την παροχή των πραγματικών δεδομένων (ανωνυμοποιημένων) της περιόδου 2011 – 2012, τα οποία χρησιμοποιήθηκαν για την εκτέλεση των δοκιμών του Γενετικού Αλγόριθμου.

Αφιερωμένο στην Χρυσάνθη, στον Χάρη και στην Εμμανουέλα για την υπομονή τους και την συμπαράσταση τους σε όλη την διάρκεια αυτού του μεταπτυχιακού

Περιεχόμενα

Εισαγωγή 1

1 Χρονοπρογραμματισμός Γεγονότων 4
1.1 Ορισμός 4
1.2 Μορφές Προβλήματος Χρονοπρογραμματισμού Γεγονότων 5
1.2.1 Χρονοπρογραμματισμός χρήσης επιστημονικών οργάνων και μηχανημάτων 6
1.2.2 Εβδομαδιαίο Πρόγραμμα Βαρδιών Προσωπικού μιας εταιρείας ή ενός δημόσιου φορέα 6
1.2.3 Προγραμματισμός των Αγώνων ενός πρωταθλήματος 7
1.2.4 Προγραμματισμός Μαθημάτων και Εξετάσεων σε ένα Εκπαιδευτικό ίδρυμα 8

2 Προγραμματισμός Εξεταστικού Ημερολογίου Πανεπιστημιακού Ιδρύματος 9
2.1 Μέθοδοι Που Χρησιμοποιήθηκαν Για Την Επίλυση Του Προβλήματος 10
2.1.1 Γραφοκεντρικές Σειριακές Τεχνικές (Graph Based Sequential Techniques) 10
2.1.2 Τεχνικές που βασίζονται σε περιορισμούς (Constraint Based Techniques) 12
2.1.3 Τεχνικές Τοπικής Αναζήτησης (Local Search Based Techniques) 14
2.1.4 Αλγόριθμοι βασισμένοι σε πληθυσμούς (Population Based Algorithms) 15
2.1.5 Πολυκριτηριακές Τεχνικές (Multi – Criteria Techniques) 17
2.1.6 Υπέρ – Ευρετικές Τεχνικές (Hyper – heuristics Techniques) 17

3 Γενετικοί Αλγόριθμοι 18
3. Βελτιστοποίηση 19
3.2 Ιστορική Αναδρομή 20
3.3 Βασική Ορολογία Των Γενετικών Αλγορίθμων 20
3.4 Πλεονεκτήματα Των Γενετικών Αλγορίθμων Σε Σχέση Με Τους Συμβατικούς Αλγόριθμους Βελτιστοποίησης 21
3.5 Δομή Ενός Γενετικού Αλγορίθμου 22
3.6 Παράδειγμα Γενετικού Αλγόριθμου: Πρόβλημα Των Οκτώ Βασιλισσών 24
3.7 Προγραμματισμός Γενετικών Αλγορίθμων 26
3.8 Η Βιβλιοθήκη ECJ 28

4 Γενετικοί Αλγόριθμοι Υλοποίηση Εξεταστικού Ημερολογίου Με Την Χρήση Γενετικών Αλγορίθμων 29
4.1 Περιορισμοί Εξεταστικού Ημερολογίου 31
4.2 Μοντελοποίηση Διαγράμματος Οντοτήτων - Συσχετίσεων 32
4.2.1 Οντότητες Εξεταστικού Ημερολογίου 33
4.2.2. Συσχετίσεις Εξεταστικού Ημερολογίου 34
4.3 Σχεσιακό Μοντέλο Εξεταστικού Ημερολογίου 35
4.3.1 Μοντελοποίηση Περιορισμών Του Εξεταστικού Ημερολογίου 37
4.4 Αναπαράσταση Χρωμοσώματος Γενετικού Αλγορίθμου 38
4.5 Διάγραμμα UML Των Κλάσεων Του Προβλήματος Κατάρτισης Εξεταστικού Ημερολογίου .. 47
4.5.1 Κλάση Cities 48
4.5.2 Κλάση Groups 48
4.5.3 ΚλάσηLessonGroups 48
4.5.4 Κλάση Lessons 49
4.5.5 Κλάση Rooms 49
4.5.6 Κλάση StudentLessons 50
4.5.7 Κλάση Timeslots 50
4.5.8 Κλάση Timetable2 50
4.5.9 Κλάση Students 51
4.5.10 Κλάση Schools 51
4.6 Παράμετροι Γενετικού Αλγορίθμου 51
4.7 Αρχικοποίηση Αλγόριθμου 53
4.8 Συνάρτηση Καταλληλότητας 53

5 Αποτελέσματα 57
5.1 Προσδιορισμός Καλύτερου Πληθυσμού 58
5.2 Δοκιμές εκτέλεσης Γενετικού Αλγόριθμου με διάφορες παραμέτρους για το κύριο πρόβλημα 60
5.3 Παράδειγμα Εκτέλεσης του γενετικού αλγορίθμου για το κύριο πρόβλημα 62
5.4 Αποτελέσματα Εκτέλεσης για τα εναλλακτικά πρόβλημα 66

Επίλογος 68
Βιβλιογραφία 70

A **Τμήματα Κώδικα** A-1
A.1 Αρχείο Παραμέτρων ECJ A-1
A.2 Κώδικας Γενετικού Αλγορίθμου..... A-2
A.3 Ρουτίνα Evaluation A-3

Εισαγωγή

Η μεταπτυχιακή αυτή διατριβή πραγματεύεται το πρόβλημα της Ανάπτυξης του Εξεταστικού Ημερολογίου μίας εξεταστικής περιόδου σε ένα τριτοβάθμιο εκπαιδευτικό ίδρυμα. Το πρόβλημα αυτό έχει απασχολήσει τους επιστήμονες της πληροφορικής και έχει επιλυθεί με την χρήση διαφόρων αλγοριθμικών τεχνικών. Λόγω των πολλών παραμέτρων που υπάρχουν στο πρόβλημα αλλά και της πολυπλοκότητας του, το πρόβλημα ανήκει στην κατηγορία των NP-Complete προβλημάτων.

Η διάρθρωση της διπλωματικής διατριβής γίνεται σε πέντε κεφάλαια.

Στο ***Πρώτο κεφάλαιο*** ορίζεται το πρόβλημα του χρονοπρογραμματισμού γεγονότων και αναφέρονται οι κυριότερες μορφές που συναντιέται το πρόβλημα αυτό στην επιστήμη της πληροφορικής. Για κάθε μία από τις μορφές του προβλήματος παρουσιάζονται οι κυριότερες από τις λύσεις που έχουν προταθεί.

Στο ***Δεύτερο κεφάλαιο*** ορίζεται το πρόβλημα της ανάπτυξης του εξεταστικού ημερολογίου, καθώς και οι μέθοδοι που έχουν χρησιμοποιηθεί για την επίλυση του, οι οποίες περιλαμβάνουν την μέθοδο του χρωματισμού του γράφου, μαθηματικές μεθόδους, μεθόδους με αλγόριθμους πληθυσμών αλλά και ευρετικές και μετα – ευρετικές μεθόδους. Για κάθε μία από τις μεθόδους αυτές αναφέρονται οι κυριότερες δημοσιεύσεις που έχουν γίνει καθώς και ο τρόπος που χρησιμοποιήθηκε η μέθοδος για την επίλυση του προβλήματος.

Το ***Τρίτο κεφάλαιο*** είναι αφιερωμένο στους γενετικούς αλγόριθμους και στην εξέλιξη των ειδών γενικότερα. Προσπαθεί να εισάγει τον αναγνώστη στην ορολογία των αλγορίθμων αυτών, σε

σύγκριση πάντα με ένα βιολογικό πληθυσμό, και παράλληλα να εξηγήσει πως χρησιμοποιείται ο αλγόριθμος αυτός και ποια είναι τα πλεονεκτήματα και τα μειονεκτήματα του σε σχέση με άλλες προγραμματιστικές τεχνικές.

Στο ***Τέταρτο κεφάλαιο*** παρουσιάζεται η μέθοδος επίλυσης του προβλήματος του εξεταστικού ημερολογίου με την χρήση γενετικών αλγορίθμων που χρησιμοποιήθηκε στην παρούσα διπλωματική διατριβή. Περιγράφεται το διάγραμμα συσχετίσεων – οντοτήτων για το συγκεκριμένο πρόβλημα καθώς και το μαθηματικό του μοντέλο. Εξαιτίας της χρήσης των γενετικών αλγορίθμων εξηγείται η επιλογής της συγκεκριμένης μορφής του χρωμοσώματος και των παραμέτρων για την συνάρτηση καταλληλότητας. Στην συνέχεια παρουσιάζεται το UML Διάγραμμα που αναφέρεται στο πρόβλημα και παρουσιάζονται οι ρουτίνες για τις κλάσεις και τα αρχεία που χρησιμοποιήθηκαν για τον προγραμματισμό του αλγορίθμου.

Στο ***Πέμπτο κεφάλαιο*** παρουσιάζεται τα αποτελέσματα της εκτέλεσης του αλγόριθμου σε διάφορες περιπτώσεις, καθώς και τα συμπεράσματα που προκύπτουν από την εκτέλεση του. Σαν παράδειγμα παρουσιάζεται η περίπτωση της ανάπτυξης του εξεταστικού ημερολογίου στο οποίο πέρνουν μέρος 360 φοιτητές, οι οποίοι εξετάζονται σε 5 μαθήματα που αφορούν ένα συγκεκριμένο μεταπτυχιακό πρόγραμμα σπουδών, τα οποία εξετάζονται σε τρεις διαφορετικές πόλεις.

Κεφάλαιο 1
Χρονοπρογραμματισμός Γεγονότων

Στο κεφάλαιο αυτό δίνεται ένας αρχικός ορισμός για την έννοια του χρονοπρογραμματισμού γενονότων (scheduling), καθώς και μία παράθεση των μορφών των προβλημάτων του χρονοπρογραμματισμού που συναντιούνται στην καθημερινή ζωή. Για κάθε μία από τις μορφές του προβλήματος γίνεται παράθεση προγραμμάτων αλλά και δημοσιεύσεων που έχουν χρησιμοποιηθεί για την επίλυση τους.

1.1 Ορισμός

Ως Χρονοπρογραμματισμό Γεγονότων ή εργασιών σε ένα συγκεκριμένο χρονικό διάστημα, ορίζουμε την κατάταξη των γεγονότων ή των εργασιών σε συγκεκριμένες υποδιαιρέσεις του χρονικού διαστήματος, οι οποίες έχουν οριστεί εκ των προτέρων, με σκοπό την ικανοποίηση διαφόρων συνθηκών που αφορούν τα γεγονότα ή τις εργασίες που πρόκειται να προγραμματιστούν.

Ένας ορισμός για την γενικότερη έννοια του ημερολογίου έχει δοθεί από τους Burke, Kingston και de Werra (2004) [08] είναι: "Ένα πρόβλημα ημερολογίου είναι ένα πρόβλημα με τέσσερις παραμέτρους: *T* ένα πεπερασμένο σύνολο από χρονικές περιόδους, *R* ένα πεπερασμένο σύνολο από πηγές, *M* ένα

πεπερασμένο σύνολο από γεγονότα, *C* ένα πεπερασμένο σύνολο από περιορισμούς. Το πρόβλημα εστιάζεται στην εκχώρηση πηγών *R* και χρονικών περιόδων *T* στα γεγονότα *M* ώστε να ικανοποιούνται όσον το δυνατόν περισσότεροι από τους περιορισμούς *C* του προβλήματος".

Στην επιστήμη Η/Υ συναντιέται σαν ένα *NP - Complete* [02] πρόβλημα και έχει επιλυθεί στις διάφορες μορφές του με διαφορετικούς τρόπους όπως: χρωματισμός γράφου, γενετικούς αλγόριθμους, μεθόδους τοπικής αναζήτησης.

1.2 Μορφές Προβλήματος Χρονοπρογραμματισμού Γεγονότων

Το πρόβλημα του χρονοπρογραμματισμού γεγονότων συναντιέται στην καθημερινότητα σε διάφορες μορφές. Κοινή συνισταμένη όλων των μορφών του συγκεκριμένου προβλήματος είναι ότι γίνεται προσπάθεια για να βρεθεί η βέλτιστη λύση, λαμβάνοντας υπ' όψιν διάφορους περιορισμούς οι οποίοι είναι διαφορετικοί ανάλογα με την μορφή του προβλήματος. Οι περιορισμοί αυτοί που παρουσιάζονται σε κάθε μία από τις μορφές του προβλήματος που αναλύονται παρακάτω είναι ενδεικτικοί και δεν ισχύουν για όλες τις μορφές του προβλήματος παρά μόνο για την συγκεκριμένη στην οποία αναφέρονται.

Ανάλογα με την σημασία τους στην επίλυση του προβλήματος οι περιορισμοί – παράμετροι χωρίζονται σε δύο μεγάλες κατηγορίες:

1. Οι "ανελαστικοί" περιορισμοί (*hard constrains*), που πρέπει να ικανοποιούνται προκειμένου να παραχθεί ένα έγκυρο πρόγραμμα και

2. Οι "ελαστικοί" περιορισμοί (*soft constrains*), που αν ικανοποιηθούν δίνουν ένα πρόγραμμα το οποίο είναι καλύτερο από ένα άλλο.

Παρακάτω παρατίθενται χαρακτηριστικές μορφές του προβλήματος χρονοπρογραμματισμού γεγονότων:

1.2.1 Χρονοπρογραμματισμός χρήσης επιστημονικών οργάνων και μηχανημάτων

Είναι πάρα πολύ σημαντικό σε διάφορα επιστημονικά όργανα αλλά και σε διάφορα μηχανήματα που η χρήση τους είναι πολύ πυκνή, να ορίζεται το πρόγραμμα χρήσης τους λαμβάνοντας υπ' όψιν διάφορες παραμέτρους όπως:

1. Διαθεσιμότητα χρήσης σε συγκεκριμένες ημερομηνίες και ώρες

2. Βέλτιστη χρήση σε σχέση με το κόστος λειτουργίας. Είναι πολύ πιθανόν η τρέχουσα κατάσταση ενός οργάνου να ευνοεί την εργασία του χρήστη *A* σε σχέση με την εργασία του χρήστη *B*, εξαιτίας διαφόρων παραγόντων.

3. Προτεραιότητα χρήσης από ένα χρήστη σε σχέση με κάποιον άλλο. Ένας ερευνητής έχει μεγαλύτερη προτεραιότητα χρήσης του οργάνου σε σχέση με κάποιον φοιτητή

Το συγκεκριμένο πρόβλημα του προγραμματισμού εργασιών κάποιου επιστημονικού οργάνου (Job Scheduling) έχει αντιμετωπιστεί με διάφορους τρόπους όπως δυναμικό προγραμματισμό [27], αλγόριθμους μυρμηγκιών [36] και αλγόριθμους προσομοιωμένη ανόπτωσης [41] και ανήκει στην οικογένεια των NP-Complete προβλημάτων.

1.2.2 Εβδομαδιαίο Πρόγραμμα Βαρδιών Προσωπικού μιας εταιρείας ή ενός δημόσιου φορέα

Ο προγραμματισμός αυτός έχει σαν σκοπό την βέλτιστη ανάθεση του προσωπικού με σκοπό την μεγιστοποίηση της απόδοσης του κάθε ένα από τους εργαζόμενους, την ελαχιστοποίηση των περιορισμών που έχει θέσει αυτός αλλά και την δίκαιη κατανομή του ωραρίου σε όλους του εργαζόμενους.

Το πιο διαδεδομένο πρόβλημα κατάρτισης βάρδιας προσωπικού είναι η διαδικασία σε ένα νοσοκομείο. Οι σημαντικότεροι παράμετροι που λαμβάνονται υπ' όψιν για την συγκεκριμένη περίπτωση είναι:

- Οι νυκτερινές βάρδιες πρέπει να μοιράζονται ίσα για κάθε μία από τις νοσοκόμες.

- Ο λόγος του αριθμού των έμπειρων νοσοκόμων προς τον αριθμό των νέων νοσοκόμων σε όλες τις βάρδιες πρέπει να είναι σταθερός.

- Οι προσωπικές επιθυμίες των νοσοκόμων λαμβάνονται υπ' όψιν όσο το δυνατόν περισσότερο.

Ο προγραμματισμός της βάρδιας του προσωπικού ενός νοσοκομείου, έχει λυθεί με διάφορους τρόπους όπως: γραμμικό προγραμματισμό [28], έμπειρα συστήματα [13], ευρετικό προγραμματισμό [31] και γενετικοί αλγόριθμοι [04]. Το πρόβλημα του προγραμματισμού του προσωπικού ενός νοσοκομείου ανήκει στην οικογένεια των NP-Complete προβλημάτων.

Κυριότερες εφαρμογές για τον χρονοπρογραμματισμό βάρδιων προσωπικού είναι το *ShiftPlanner*[1], το *Time Force Scheduling*[2] και το *WorkSchedule.NET*[3].

1.2.3 Προγραμματισμός των Αγώνων ενός πρωταθλήματος

Μία ακόμη προσέγγιση του προβλήματος χρονοπρογραμματισμού αποτελεί ο προγραμματισμός των αγώνων ενός πρωταθλήματος. Οι κυριότεροι περιορισμοί που τίθενται στην συγκεκριμένη περίπτωση είναι οι παρακάτω:

1. Καμία ομάδα δεν μπορεί να αγωνιστεί με τον εαυτό της.

2. Μία ομάδα μπορεί να αγωνιστεί εναντίον μίας μόνο άλλης ομάδας σε μία συγκεκριμένη χρονική στιγμή.

3. Μία ομάδα πρέπει να αγωνιστεί με όλες τις υπόλοιπες ομάδες που αγωνίζονται στο πρωτάθλημα προτού αγωνιστεί ξανά με μία ομάδα.

4. Αν η ομάδα Α αγωνίζεται εντός έδρας με την ομάδα Β την χρονική στιγμή t τότε η ομάδα Β αγωνίζεται εκτός έδρας με την ομάδα Α.

Το πρόβλημα αυτό έχει επιλυθεί με τους με διάφορους τρόπους όπως: Προγραμματισμός Ακεραίων [05], προγραμματισμός περιορισμών [37], ευρετικές μεθόδους [40] και γενετικούς αλγόριθμους.

Από τις διάφορες εφαρμογές που έχουν αναπτυχθεί ξεχωρίζουν οι εφαρμογές για το γερμανικό και αυστριακό πρωτάθλημα ποδοσφαίρου [03] καθώς και η αντίστοιχη που έχει αναπτυχθεί για το ιταλικό πρωτάθλημα [19].

1 ShiftPlanning: http://employee-scheduling-software-review.toptenreviews.com/shiftplanning-review.html
2 Task Force Scheduling: http://employee-scheduling-software-review.toptenreviews.com/timeforge-scheduling-review.html
3 WorkScheduler.NET http://employee-scheduling-software-review.toptenreviews.com/workschedule.net-review.html

1.2.4 Προγραμματισμός Μαθημάτων και Εξετάσεων σε ένα Εκπαιδευτικό ίδρυμα

Σε ένα εκπαιδευτικό ίδρυμα ο προγραμματισμός μαθημάτων ορίζεται ως την χρήση από ένα τμήμα φοιτητών ή μαθητών μίας αίθουσας σε μία συγκεκριμένη χρονική περίοδο. Η επιλογή μίας αίθουσας σε σχέση με κάποιαν άλλη εξαρτάται από :

- τον αριθμό των φοιτητών που έχουν δηλώσει να παρακολουθήσουν το μάθημα
- την χρήση ενός συγκεκριμένου οργάνου ή εξοπλισμού που υπάρχει στην αίθουσα π.χ. Βιντεοπροβολέας

Άλλες παράμετροι που λαμβάνονται υπ' όψιν για να επιλυθεί το συγκεκριμένο πρόβλημα είναι: ο μέγιστος αριθμός μαθημάτων που παρακολουθεί κάποιος μαθητής κατά την διάρκεια της ημέρας, αλλά και η διαθεσιμότητα του καθηγητή τις συγκεκριμένες ώρες.

Όπως και οι υπόλοιπες περιπτώσεις του χρονοπρογραμματισμού,ο προγραμματισμός των μαθημάτων σε ένα εκπαιδευτικό ίδρυμα είναι ένα πρόβλημα που ανήκει στην κατηγορία των NP-Complete προβλημάτων.

Το πρόβλημα αυτό έχει επιλυθεί με διάφορες μεθόδους όπως: γενετικούς αλγόριθμους [24], χρωματισμός γράφων, *Particle Swarm Optimization*.

Κεφάλαιο 2
Προγραμματισμός Εξεταστικού Ημερολογίου Πανεπιστημιακού Ιδρύματος

Σε ένα εκπαιδευτικό ίδρυμα ανάμεσα στις διάφορες εκπαιδευτικές διαδικασίες σημαντικό ρόλο παίζουν και οι εξετάσεις των διαφόρων μαθημάτων στο τέλος της ακαδημαϊκής περιόδου. Σε έρευνα που είχε γίνει στην Μεγάλη Βρετανία από τους Bruke et al το 1996 [07], βρέθηκε ότι ένα ποσοστό της τάξης του 21% μόνο χρησιμοποιούσε κάποιο λογισμικό για την παραγωγή ενός εξεταστικού ημερολογίου και σε όλες τις υπόλοιπες περιπτώσεις γινόταν χειροκίνητα.

Η υλοποίηση ενός εξεταστικού ημερολογίου σε ένα εκπαιδευτικό ίδρυμα χωρίς την βοήθεια κάποιου λογισμικού, αποδεικνύεται ότι είναι μία ιδιαίτερα δύσκολη και επίπονη εργασία για το διοικητικό προσωπικό του ιδρύματος, η οποία χρειάζεται πάρα πολλές εβδομάδες για να ολοκληρωθεί και πολλές φορές το τελικό αποτέλεσμα δεν ικανοποιεί τους εμπλεκόμενους.

Συνηθισμένα προβλήματα που μπορεί να προκύπτουν σε ένα εξεταστικό ημερολόγιο είναι:

1. Οι φοιτητές να μην έχουν την δυνατότητα να εξεταστούν σε όλα τα μαθήματα τα οποία παρακολούθησαν κατά την διάρκεια του ακαδημαϊκού έτους, αφού οι ώρες εξέτασης των

μαθημάτων συμπίπτουν μεταξύ τους, παρόλο που αυτό θεωρείται ένας από τους βασικούς περιορισμούς. Σε άλλη περίπτωση μπορεί να υπάρχουν μαθήματα τα οποία να εξετάζονται με μικρή χρονική απόσταση το ένα από το άλλο, οπότε ο φοιτητής δεν έχει την ευκαιρία να ξεκουραστεί αλλά και να έχει τον απαραίτητο χρόνο για την μελέτη του δεύτερου μαθήματος. Αυτό οδηγεί σε αύξηση της πιθανότητας αποτυχίας σε κάποιο από τα μαθήματα, με αποτέλεσμα την οικονομική επιβάρυνση των ίδιων στην περίπτωση που το τριτοβάθμιο ίδρυμα είναι ιδιωτικό αλλά και την λανθασμένη αξιολόγηση του ίδιου του ιδρύματος.

2. Οι εκπαιδευτικοί θα πρέπει να έχουν αρκετό διάστημα ανάμεσα σε δύο εξετάσεις ώστε να διορθώνουν τα γραπτά και να παραδίδουν έγκαιρα την βαθμολογία στις γραμματείες των τμημάτων. Με αυτό τον τρόπο η ανακοίνωση των βαθμολογιών γίνεται έγκαιρα και υπάρχει αρκετός χρόνος για να οργανώσουν οι φοιτητές καλύτερα τον χρόνο προετοιμασίας τους.

3. Ορισμένα εκπαιδευτικά ιδρύματα τα οποία δεν διαθέτουν ιδιόκτητους χώρους σε όλες τις περιοχές, στις οποίες έχουν τμήματα προς εξέταση Έτσι είναι πάρα πολύ σημαντικός ο χρόνος χρήσης των αιθουσών στην συγκεκριμένη περιοχή, προκειμένου το κόστος ενοικίασης των αιθουσών να είναι μικρότερο.

Το πρόβλημα Ανάπτυξης Εξεταστικού Ημερολογίου ανήκει στην οικογένεια των NP-Complete προβλημάτων και έχουν προταθεί διάφοροι τρόποι για την επίλυση του.

2.1 Μέθοδοι Που Χρησιμοποιήθηκαν Για Την Επίλυση Του Προβλήματος

Για την επίλυση του προβλήματος του εξεταστικού ημερολογίου έχουν χρησιμοποιηθεί διάφορες τεχνικές. Οι κυριότερες από αυτές απαριθμούνται από τον Qu R. et al (2006) [3] αλλά και τους Carter, M.W., Laporte, G (1996) [11] και είναι:

2.1.1 Γραφοκεντρικές Σειριακές Τεχνικές (Graph Based Sequential Techniques):

Είναι η πιο παλιά προσπάθεια επίλυσης του προβλήματος του εξεταστικού ημερολογίου. Η τεχνική επιλύει το πρόβλημα θεωρώντας ότι τα διαγωνίσματα είναι κορυφές του γράφου ενώ οι ανελαστικοί περιορισμοί που υπάρχουν στο πρόβλημα είναι οι ακμές που ενώνουν τα διαγωνίσματα μεταξύ τους.

Το πρόβλημα επιλύεται χρωματίζοντας τις κορυφές με τέτοιο χρώμα έτσι ώστε γειτονικές κορυφές που ενώνονται μεταξύ τους να μην έχουν το ίδιο χρώμα.

Οι ελαστικοί περιορισμοί δεν παίζουν ρόλο στην κατασκευή του γράφου αλλά λαμβάνονται υπόψιν σε μία συνάρτηση που μπορεί να δώσει ένα αποτέλεσμα της μέτρησης της ποιότητας του ημερολογίου. Κατά την υλοποίηση τα διαγωνίσματα εισάγονται ένα – ένα στο ημερολόγιο ανάλογα με τον αριθμό των ανελαστικών περιορισμών που τα συνδέουν με άλλα διαγωνίσματα. Έτσι πρώτο θα τοποθετηθεί το μάθημα με τους περισσότερους περιορισμούς και τελευταίο το μάθημα με τους λιγότερους.

Το πλεονέκτημα της συγκεκριμένης τεχνικής είναι ότι δίνει καλά αποτελέσματα σε μικρό υπολογιστικό χρόνο και είναι πολύ απλή στην υλοποίηση της. Ένα μειονέκτημα της είναι ότι μπορεί κάποιο από τα διαγωνίσματα που θα επιλεγεί νωρίς κατά την διαδικασία να οδηγήσει σε ένα ημερολόγιο που δεν θα είναι χρησιμοποιήσιμο αφού οι ανελαστικοί περιορισμοί που το συνδέουν με τα υπόλοιπα μαθήματα μπορεί να είναι τόσοι πολλοί ώστε κάποια άλλα μαθήματα να μην μπορούν να μπουν στον γράφο και κατά συνέπεια στο εξεταστικό ημερολόγιο.

Για την εύρεση της σειράς με την οποία θα μπουν τα μαθήματα στο ημερολόγιο έχουν χρησιμοποιηθεί διάφορες ευρετικές τεχνικές όπως:

1. **Βαθμός Κορεσμού** (*Saturation Degree*): Όσο πιο λίγες ελεύθερες θέσεις υπάρχει στις οποίες μπορεί να τοποθετηθεί το μάθημα τόσο πιο μεγάλη προτεραιότητα έχει.

2. **Μεγαλύτερος Βαθμός Συγκρούσεων** (*Largest Degree*): Η προτεραιότητα τοποθέτησης ενός μαθήματος στον γράφο είναι ανάλογος με τον αριθμό των συγκρούσεων που έχει με άλλα μαθήματα.

3. **Μεγαλύτερος Βαθμός Συγκρούσεων με βάρος** (*Largest Weighted Degree*): Το ίδιο με το προηγούμενο αλλά εδώ μπαίνει και σαν πολλαπλασιαστικός παράγοντας ο αριθμός των φοιτητών που παρακολουθούν το μάθημα.

4. **Μεγαλύτερος Αριθμός Εγγραφών Φοιτητών** (*Largest Enrollment*): Η προτεραιότητα τοποθέτησης ενός μαθήματος είναι ανάλογη με τις εγγραφές των φοιτητών στο μάθημα.

5. **Τυχαία Σειρά Κατάταξης** (*Random Ordering*): Τα μαθήματα τοποθετούνται στον γράφο με τυχαία σειρά κατάταξης

6. **Βαθμός Χρωματισμού του Γράφου** (*Colour Degree*): Η προτεραιότητα στην τοποθέτηση ενός μαθήματος στο γράφο είναι ανάλογη του αριθμού των συγκρούσεων του μαθήματος με τα υπόλοιπα μαθήματα που έχουν προγραμματιστεί για την ίδια εξεταστική ώρα.

Οι Carter, Laporte και Lee [12] το 1996, εξέτασαν τις πρώτες πέντε περιπτώσεις σε διάφορα εξεταστικά προβλήματα (πραγματικά αλλά και τυχαία). Μεγάλες ομάδες, αποτελούν μεγάλους υπογράφους όπου είχαν τις πλευρές τους γειτονικές με όλες τις άλλες, χρησιμοποιούνται σαν αρχικές λύσεις, βασισμένες σε ποία τεχνική θέλουν να ακολουθήσουν. Η ιδέα είναι ότι το πλήθος των στοιχείων της μεγαλύτερης ομάδας ορίζει τον ελάχιστο αριθμό των χρονικών περιόδων της εξέτασης που χρειάζονται από το πρόβλημα.

Οι Burke, Newall και Weare [10] το 1998 μελέτησαν την επίδραση της εισαγωγής ενός τυχαίου διαγωνίσματος στην διαδικασία των ευρετικών τεχνικών, δημιουργώντας δύο στρατηγικές επιλογής:

A. Η Tournament Selection όπου επιλέγεται ένα διαγώνισμα τυχαία από ένα υποσύνολο των πρώτων διαγωνισμάτων της ταξινομημένης λίστας και

B. Η Bias Selection όπου επιλέγει το πρώτο από τα διαγωνίσματα της ταξινομημένης λίστας που προέρχεται από ένα υποσύνολο όλων των διαγωνισμάτων.

Σήμερα οι γραφοκεντρικές τεχνικές χρησιμοποιούνται στα προβλήματα εξεταστικού ημερολογίου:

a. για την δημιουργία μιας ομάδας λύσεων του προβλήματος βάση των αρχικών παραμέτρων του, οι οποίες σε δεύτερο στάδιο τροφοδοτούν την είσοδο κάποιου άλλου αλγόριθμου σε αρχικό στάδιο.

b. για την επίλυση συγκεκριμένων τμημάτων του προβλήματος και εξαγωγή των λύσεων των τμημάτων αυτών με σκοπό την τροφοδότηση στην είσοδο κάποιου άλλου αλγορίθμου.

2.1.2 Τεχνικές που βασίζονται σε περιορισμούς (*Constraint Based Techniques*)

Οι συγκεκριμένες λύσεις περιγράφονται σαν Τεχνικές Ικανοποίησης Περιορισμών (*Constraint Satisfaction Techniques*) ή Λογικός Προγραμματισμός Περιορισμών (*Constraint Logic Programming*) και έχουν τις ρίζες τους στην Τεχνητή Νοημοσύνη. Αυτές μοντελοποιούν το πρόβλημα σαν θεωρώντας τα διαγωνίσματα σαν ένα σύνολο μεταβλητών με πεπερασμένο πεδίο ορισμού, στο οποίο θα πρέπει να

ανήκουν οι τιμές ώστε να ικανοποιούνται οι περιορισμοί. Οι τιμές των μεταβλητών αναπαριστούν τις χρονικές περιόδους αλλά και τις αίθουσες που θα γίνουν οι εξετάσεις.

Ο David [17] το 1998 χρησιμοποίησε τέτοιες τεχνικές για την μοντελοποίηση του εξεταστικού προβλήματος σε ένα σχολείο. Το σημαντικό σε αυτή την εργασία ήταν η χρονική πολυπλοκότητα, έτσι προτιμούνται μη ολοκληρωμένες λύσεις στις οποίες εφαρμόζονται τεχνικές τοπικής αναζήτησης ώστε να μετατραπούν σε ολοκληρωμένες. Προκειμένου να μην χαθούν πιθανές λύσεις, ο αλγόριθμος εκτελούνταν αρκετές φορές.

Οι Merlot et al [29] το 2003 χρησιμοποίησαν μία προγραμματιστική γλώσσα βελτιστοποίησης για την παραγωγή αρχικών λύσεων. Οι λύσεις αυτές βελτιστοποιούνταν με την βοήθεια τεχνικών όπως Προσομοιωμένη Ανόπτωση (*Simulating Annealing*) και Αναρρίχηση Λόφων (*Hill Climbing*). Οι μεταβλητές (διαγωνίσματα) ταξινομούνταν με βάση τα πεδία ορισμού τους (ελεύθερες περίοδοι εξεταστικού ημερολογίου) και έμπαιναν στο ημερολόγιο η μία μετά την άλλη.

Οι τεχνικές αυτές έχουν μεγάλο υπολογιστικό κόστος γιατί ο αριθμός των πιθανών εκχωρήσεων αυξάνει εκθετικά σε σχέση με τον αριθμό των μεταβλητών. Επίσης δεν μπορούν να δώσουν λύσεις πολύ καλής ποιότητας σε προβλήματα βελτιστοποίησης.

2. 1. 2. 1. Μέθοδος Ικανοποίησης Προβλήματος (SAT)

Με την έννοια SAT (Ικανοποίηση) ορίζεται σε μία συνάρτηση η απόδοση δυαδικών τιμών στις μεταβλητές της συνάρτησης με τέτοιο τρόπο ώστε το αποτέλεσμα να είναι αλήθεια. Αν το αποτέλεσμα της συνάρτησης είναι αλήθεια (TRUE) τότε το πρόβλημα έχει ικανοποιηθεί σε άλλη περίπτωση δεν έχει ικανοποιηθεί.

Η συνάρτηση θα πρέπει να είναι της μορφής:

$$(X_{11} \text{ OR } X_{12} \text{ OR } X_{13}) \text{ AND } (X_{21} \text{ OR } X_{22} \text{ OR } X_{23}) \text{ AND } (X_{31} \text{ OR } X_{32} \text{ OR } X_{33}) \text{ AND } ...$$

όπου οι μεταβλητές μπορεί να είναι είτε στην κανονική είτε στην αρνητική τους μορφή. Στο παράδειγμα που δόθηκε παραπάνω επειδή έχουμε τρεις μεταβλητές σε κάθε όρο της παραπάνω παράστασης οι οποίες διαχωρίζονται μεταξύ τους με πύλες OR. Οι επιμέρους όροι συνδέονται μεταξύ τους με πύλες AND. Αυτή η μορφή του προβλήματος ορίζεται σαν 3-SAT πρόβλημα. Το πρόβλημα του k-SAT και του 3-SAT είναι πρόβλημα NP-complete.

Για την επίλυση του SAT προβλήματος έχουν υλοποιηθεί αρκετοί αλγόριθμοι όπως:

1. Ο DPLL (*Davis – Putnam – Logemann – Loveland algorithm*) [18] που χρησιμοποιεί έναν αλγόριθμο οπισθοδρόμησης (*backtracking*) για να βρει την λύση του συγκεκριμένου προβλήματος.

2. Οι αλγόριθμοι GSAT και WalkSAT[4] [38], που είναι αλγόριθμοι τοπικής αναζήτησης, στην πρώτη φάση δίνουν τυχαίες τιμές στις μεταβλητές που απαρτίζουν την συνάρτηση. Αν η συνάρτηση δώσει αποτέλεσμα αλήθεια τότε έχει ικανοποιηθεί και ο αλγόριθμος σταματάει. Σε άλλη περίπτωση αλλάζει η κατάσταση μίας από τις μεταβλητές του προβλήματος και υπολογίζεται ξανά το αποτέλεσμα της συνάρτησης.

2.1.3 Τεχνικές Τοπικής Αναζήτησης (*Local Search Based Techniques*)

Η συγκεκριμένη τεχνική προσπαθεί να βρει λύση στο πρόβλημα ελέγχοντας την γειτονική περιοχή της. Ανάλογα με την δομή των περιοχών στις οποίες ψάχνει ο αλγόριθμος έχουν δημιουργηθεί και οι κατάλληλες τεχνικές. Οι αποτελεσματικότητα των τεχνικών εξαρτάται από τις παραμέτρους και τις ιδιότητες του διαστήματος μέσα στο οποίο θα γίνει η έρευνα. Το μεγάλο μειονέκτημα των τεχνικών αυτών είναι η μεγάλη προσπάθεια που χρειάζεται για την βελτιστοποίηση των παραμέτρων σε συγκεκριμένα προβλήματα ώστε να δοθούν λύσεις υψηλής ποιότητας. Στην συγκεκριμένη οικογένεια τεχνικών ανήκουν τεχνικές όπως τοπικής αναζήτησης με απαγόρευση κινήσεων, προσομοιωμένης ανόπτωσης.

Ο Di Gaspero και ο Schearf [21] το 2001, εργάστηκαν πάνω σε διάφορες τεχνικές Tabu Search όπου οι γειτονιές που δημιουργούνταν εξαρτώταν από τις παραβιάσεις των περιορισμών (ανελαστικών αλλά και ελαστικών). Το μήκος της λίστας ήταν δυναμικό και η συνάρτηση κόστους υπολογιζόταν συνεχώς κατά την διάρκεια του ελέγχου.

Παράδειγμα υλοποίησης βλέπουμε στη δημοσίευση [25].

4 http://www.cs.rochester.edu/u/kautz/walksat/

2.1.4 Αλγόριθμοι βασισμένοι σε πληθυσμούς (*Population Based Algorithms*):

Οι τεχνικές που έχουν δημιουργηθεί βασίζονται σε γενετικούς, μιμητικούς και αλγόριθμους μυρμηγκιών. Οι γενετικοί αλγόριθμοι είναι αυτοί που έχουν χρησιμοποιηθεί κατά κόρον στην δημιουργία του εξεταστικού ημερολογίου.

2.1.4.1 Γενετικοί Αλγόριθμοι

Οι γενετικοί αλγόριθμοι πήραν το όνομα τους από την θεωρία της εξέλιξης του Δαρβίνου. Κάθε σημείο του χώρου αναζήτησης ονομάζεται χρωμόσωμα (chromosome) και είναι συνήθως μία συμβολοσειρά (string) από μηδενικά και άσσους (allele). Κάθε ένα από τα ψηφία της συμβολοσειράς ονομάζεται γονίδιο (gene) . Το σύνολο των χρωμοσωμάτων που αποτελούν μία πιθανή λύση ονομάζεται γονότυπος (genotype).

Βασικό στοιχείο ενός γενετικού αλγορίθμου είναι η συνάρτηση καταλληλότητας (fitness function) του, η οποία αξιολογεί την ποιότητα του χρωμοσώματος του γενετικού αλγορίθμου και βάση αυτής λαμβάνεται η απόφαση αν το χρωμόσωμα θα πάρει μέρος σε κάποια από τις πράξεις του γενετικού αλγορίθμου. Η συνάρτηση καταλληλότητας συνήθως λαμβάνει υπόψιν τα γονίδια που απαρτίζουν το χρωμόσωμα.

Οι κυριότερες ενέργειες ενός γενετικού αλγορίθμου είναι:

1. Η Αναπαραγωγή (Selection) αντιγράφει κάποια χρωμοσώματα αυτούσια στην επόμενη γενιά.

2. Η διασταύρωση (crossover) είναι η διαδικασία όπου τα χρωμοσώματα διασταυρώνονται μεταξύ τους.

3. Η μετάλλαξη (mutation) αλλάζει την τιμή ενός γόνου του χρωμοσώματος

Οι Terashima – Marin, Ross και Valenzuela – Rendon [39] το 1999 σχεδίασαν ένα σύστημα crossover με την χρήση ομάδων σε εξεταστικά προγράμματα τα οποία στην συνέχεια μετετράπησαν σε γραφοκεντρικά προβλήματα.

Οι γενετικοί αλγόριθμοι θα αναλυθούν περισσότερο σε ιδιαίτερο κεφάλαιο της διατριβής αφού αποτελούν την κύρια μέθοδο επίλυσης του προβλήματος σε αυτή την διπλωματική διατριβής.

2.1.4.2 Μιμητικοί Αλγόριθμοι:

Οι μιμητικοί Αλγόριθμοι [30] είναι μία επέκταση των γενετικών αλγορίθμων και η γενική ιδέα είναι ότι κατά την διάρκεια της ζωής τους τα άτομα που αποτελούν τον πληθυσμό βελτιώνουν διάφορες προσωπικές τους δεξιότητες αλλά και ιδέες. Έτσι υπάρχουν δύο στάδια βελτίωσης: Το πρώτο είναι ένας γενετικός αλγόριθμος ο οποίος υλοποιεί την βιολογική εξέλιξη και το δεύτερο στάδιο είναι η χρήση μίας τεχνικής τοπικής αναζήτησης, για παράδειγμα την αναρρίχηση λόφων η οποία υλοποιεί την εξέλιξη των ιδεών του ατόμου.

Το πλεονέκτημα της τεχνικής είναι ότι μπορεί να λυθούν προβλήματα μεγάλης δυσκολίας αλλά το κόστος σε υπολογιστικό χρόνο είναι πάρα πολύ μεγάλο, λόγω της ύπαρξης του αλγόριθμου τοπικής αναζήτησης.

Οι Newal, Burke και Weare [09] το 1996 δημιούργησαν ένα Μιμητικό αλγόριθμο που χρησιμοποιεί ισχυρές αλλά και ελαφριές μεταλλάξεις (mutations) για να αναδιατάξει μαθήματα είτε μόνα τους είτε σε ομάδες με σκοπό να αποφύγει τοπικά ελάχιστα. Δυστυχώς αυτές οι μεταλλάξεις δεν αύξησαν την ποιότητα στις λύσεις. Αντίθετα ο αλγόριθμος αναρρίχησης λόφων που χρησιμοποιήθηκε σε επόμενο στάδιο, βελτίωσε την ποιότητα στις λύσεις παρόλο που αύξησε τον χρόνο επεξεργασίας.

2.1.4.3 Αλγόριθμοι πληθυσμού Μυρμηγκιών:

Οι αλγόριθμοι πληθυσμού μυρμηγκιών [14] προσομοιώνουν την κίνηση των μυρμηγκιών από την φωλιά προς την τροφή τους. Κατά την διάρκεια της διαδρομής τους από και προς την τροφή, τα μυρμήγκια αφήνουν μία ποσότητα φερορμόνης στην διαδρομή. Η φερορμόνη αυτή έχει ένα ρυθμό ελάττωσης της ως προς τον χρόνο. Έτσι, αν τα περισσότερα μυρμήγκια επιλέξουν μία διαδρομή, η φερορμόνη θα είναι αυξημένη σε αυτή την διαδρομή οπότε και τα υπόλοιπα θα έχουν περισσότερες πιθανότητες να την ακολουθήσουν.

Ο Elay [23] έλυσε το πρόβλημα του εξεταστικού ημερολογίου με έναν αλγόριθμο μυρμηγκιών. Κάθε ένα από τα διαγωνίσματα αποτελεί ένα ψηφιακό μυρμήγκι και έναν κόμβο σε ένα γράφο. Το μυρμήγκι αυτό θα πρέπει να επισκεφτεί όλους τους κόμβους που χρειάζεται . Έπειτα ο αλγόριθμος βρίσκει την πιο κατάλληλη περίοδο εξέτασης για το διαγώνισμα και οι υπόλοιπες περίοδοι εξέτασης για το συγκεκριμένο διαγώνισμα λαμβάνουν μία ποινή.

2.1.5 Πολυκριτηριακές Τεχνικές (Multi – Criteria Techniques):

Στην πλειονότητα των αλγορίθμων που χρησιμοποιούνται σήμερα για την δημιουργία εξεταστικών ημερολογίων, αθροίζονται τιμές που έχουν συγκεκριμένο βάρος ανάλογα με τον περιορισμό που αναπαριστούν και το άθροισμα αυτό δείχνει την ποιότητα του ημερολογίου που έχει κατασκευαστεί. Οι πολυκριτηριακές τεχνικές αναπαριστούν τους περιορισμούς σαν ένα διάνυσμα αντί για μία τιμή. Κάθε στοιχείο του διανύσματος μπορεί να ελεγχθεί είτε αυτόνομα είτε σε σχέση με διάφορα άλλα στοιχεία. Τα κριτήρια μπορούν να αλλάζουν δυναμικά μέσα σε κάποια όριο και να παραχθεί εύκολα ένα εξεταστικό ημερολόγιο.

Οι Burke, Bykov και Petrovic [06] το 2001, δημιούργησαν μία τεχνική όπου έλεγχαν εννέα κριτήρια δύο επιπέδων. Στο 1ο επίπεδο χρησιμοποιήθηκε η Saturation Degree, μία γραφοκεντρική τεχνική για να επιστραφεί ένα αρχικό σύνολο λύσεων. Στο 2ο επίπεδο με ευρετικές τεχνικές οι λύσεις αυτές βελτιωνόταν. Η βαθμολογία των λύσεων γινόταν με βάση την απόσταση τους από ένα ιδεατό σημείο που οριζόταν από τα κριτήρια. Οι Petrovic και Bykov [33] το 2003, προσπάθησαν να βελτιώσουν αυτή την μέθοδο, βασίζοντας την τεχνική στον αλγόριθμο Great Deluge[5] [22].

2.1.6 Υπέρ – Ευρετικές Τεχνικές (Hyper – heuristics Techniques):

Οι τεχνικές αυτές μπορούν να οριστούν σαν ευρετικές τεχνικές οι οποίες επιλέγουν άλλες ευρετικές τεχνικές για να λύσουν το πρόβλημα. Ο στόχος είναι να υλοποιηθεί μία πιο γενική επιλογή παρά να υλοποιηθούν επιλογές εστιασμένες στο πρόβλημα που συχνά χρειάζονται περισσότερη προσπάθεια για την βελτιστοποίηση των παραμέτρων και μπορούν να χρησιμοποιηθούν μόνο για συγκεκριμένα προβλήματα.

Ο Qu και ο Burke [34] το 2006, εξέτασαν το αποτέλεσμα της χρησιμοποίησης διαφορετικών αλγορίθμων υψηλού επιπέδου [π.χ. Αλγόριθμος απότομης καθόδου (*Steepest Descent*)[6], τοπική αναζήτηση με απαγόρευση κινήσεων] για την ενοποίηση του γραφήματος των λύσεων . Βρέθηκε ότι όταν το διάστημα των λύσεων είναι μεγάλο, οι τεχνικές αυτές δίνουν πολύ καλά αποτελέσματα.

5 http://en.wikipedia.org/wiki/Great_Deluge_algorithm
6 http://en.wikipedia.org/wiki/Method_of_steepest_descent

Κεφάλαιο 3
Γενετικοί Αλγόριθμοι

Οι γενετικοί αλγόριθμοι είναι αλγόριθμοι βελτιστοποίησης που βασίζονται σε πληθυσμούς, οι οποίοι χρησιμοποιούν μηχανισμούς που έχουν εμπνευστεί από την βιολογία, όπως διασταύρωση, φυσική επιλογή, μετάλλαξη.

Το πλεονέκτημα των γενετικών αλγόριθμων συγκρινόμενες με άλλες μεθόδους βελτιστοποίησης είναι ο χαρακτήρας "μαύρου κουτιού" που έχουν. Έτσι γίνονται λίγες μόνο παραδοχές για τις συναρτήσεις καταλληλότητας τους. Επιπρόσθετα, οι ορισμοί των συναρτήσεων καταλληλότητας συνήθως απαιτούν λιγότερη διορατικότητα για την δομή του προβλήματος σε σχέση με την χειροκίνητη κατασκευή ενός αποδεκτού ευρετικού αλγόριθμου για το ίδιο πρόβλημα.

Οι γενετικοί αλγόριθμοι διατηρούν έναν πληθυσμό πιθανών λύσεων του προβλήματος που μας ενδιαφέρει, πάνω στον οποίο δουλεύουν, σε αντίθεση με άλλες μεθόδους αναζήτησης που επεξεργάζονται ένα μόνο σημείο του διαστήματος αναζήτησης. Έτσι ένας γενετικός αλγόριθμος πραγματοποιεί αναζήτηση σε πολλές κατευθύνσεις και υποστηρίζει καταγραφή και ανταλλαγή πληροφοριών μεταξύ αυτών των κατευθύνσεων [43].

3.1 Βελτιστοποίηση

Βελτιστοποίηση μίας κατάστασης ενός συστήματος ονομάζουμε την μεγιστοποίηση ή ελαχιστοποίηση μίας συνάρτησης μίας ή περισσότερων μεταβλητών των οποίων οι τιμές ορίζονται από την συγκεκριμένη κατάσταση στην οποία βρίσκεται το σύστημα.

Κλασικά παραδείγματα είναι:

1. Στον μικρόκοσμο όπου τα άτομα δημιουργούν μεταξύ τους δεσμούς για να μειώσουν τα ελεύθερα ηλεκτρόνια της εξωτερικής τους στοιβάδας και κατά συνέπεια την ενέργεια. Στο συγκεκριμένο παράδειγμα η συνάρτηση που πρέπει να ελαχιστοποιηθεί περιέχει τόσες μεταβλητές όσα και τα άτομα που παίρνουν μέρος στην χημική ένωση.

2. Στην βιολογία μέσω της θεωρίας της εξέλιξης, τα άτομα εξελίσσουν τις ιδιότητες τους με τέτοιο τρόπο ώστε να προσαρμόζονται όσο το δυνατόν καλύτερα στο περιβάλλον τους. Κλασσικό παράδειγμα είναι ο άνθρωπος που προσαρμόστηκε καλύτερα στο περιβάλλον του από όλα τα υπόλοιπα είδη και έχει κυριαρχήσει, δημιουργώντας ένα τοπικό μέγιστο.

3. Ο ίδιος ο άνθρωπος προσπαθεί να βελτιώσει διάφορες ιδιότητες του σε διάφορους τομείς, όπως για παράδειγμα τα οικονομικά του βρίσκοντας μία πιο προσοδοφόρα εργασία, τον προσδόκιμο χρόνο ζωής του εξελίσσοντας την φαρμακευτική αγωγή που λαμβάνει απέναντι στις διάφορες ασθένειες.

Έστω X το σύνολο των πιθανών καταστάσεων ενός προβλήματος και $f = \{f_1, f_2, f_3, \ldots, f_v\}$ το σύνολο των κριτηρίων βάση τον οποίων μπορεί να λεχθεί ότι μία κατάσταση x_1 είναι καλύτερη από την κατάσταση x_2.

Ονομάζουμε αντικειμενική συνάρτηση αξιολόγησης την μαθηματική συνάρτηση $g : X \to R$, η οποία είναι αποτέλεσμα μαθηματικών πράξεων μεταξύ των μελών του συνόλου των κριτηρίων και το αποτέλεσμα της δείχνει ότι μία κατάσταση του προβλήματος είναι καλύτερη ή χειρότερη από μία άλλη.

3.2 Ιστορική Αναδρομή

Το 1859, ο Δαρβίνος [16] δημοσίευσε την εργασία του "*On the origin of Species*" στην οποία όρισε τις αρχές της φυσικής επιλογής και της επιβίωσης του πιο ισχυρού ως τις κύριες δυνάμεις της βιολογικής εξέλιξης.

Το 1950 έγινε η πρώτη προσπάθεια για την χρήση των γενετικών αλγορίθμων από διάφορους βιολόγους προκειμένου να προσομοιώνουν πολύπλοκα βιολογικά συστήματα. Στις αρχές του 1970 ο John Holland [26] και οι συνεργάτες του, τους οδήγησαν στην σημερινή τους μορφή.

Οι εξελικτικοί αλγόριθμοι βασίστηκαν στην λογική της βιολογικής αυτής διεργασίας και εισήγαγαν ένα νεωτερισμό αφού όλη η εξέλιξη εξαρτιόνταν αποκλειστικά από το αποτέλεσμα.

Στους εξελικτικούς αλγόριθμους το πεδίου των λύσεων G παίζει το ρόλο του γενόματος και ένα μέλος του έστω g αποτελεί τον γονότυπο. Σε αντιστοιχία με τους ζωντανούς οργανισμούς, οι υποψήφιες λύσεις του προβλήματος (ή αλλιώς φαινότυπος) $x \in X$ στο πεδίο του προβλήματος X είναι ένα περιστατικό που παρουσιάζεται από την αντιστοιχία γενότυπου – φαινοτύπου. Η καταλληλότητα μετριέται από αντίστοιχες συναρτήσεις που έχουν σαν σκοπό την βελτιστοποίηση και να οδηγήσουν την εξέλιξη σε συγκεκριμένες καταστάσεις.

3.3 Βασική Ορολογία Των Γενετικών Αλγορίθμων

Η ορολογία που συναντιέται στους γενετικούς αλγόριθμους είναι αντίστοιχη με αυτήν που χρησιμοποιείται στην θεωρία της εξέλιξης στην επιστήμη της βιολογίας και ειδικότερα του τομέα της Φυσικής Γενετικής.

Μία πιθανή λύση του προβλήματος ονομάζεται γενότυπος (*genotype*) ή άτομο (*individual*). Το αποκωδικοποιημένο περιεχόμενο του γενότυπου ονομάζεται φαινότυπος. Το σύνολο των πιθανών λύσεων του προβλήματος ονομάζεται πληθυσμός (*population*). Κάθε μία από τις πιθανές λύσεις του προβλήματος αποτελείται από ένα χρωμόσωμα (*chromosome*) ή σε ελάχιστες περιπτώσεις περισσότερα. Αυτό είναι μία βασική διαφορά του γενετικού αλγόριθμου με την φύση, όπου ο κάθε γενότυπος αποτελείται από περισσότερα από ένα χρωμοσώματα.

Κάθε ένα χρωμόσωμα αποτελείται από γονίδια (*genes*). Το κάθε ένα από τα γονίδια επηρεάζει ένα συγκεκριμένο γνώρισμα του ατόμου που βρίσκεται σε συγκεκριμένες θέσεις του χρωμοσώματος και

ονομάζεται *loci*. Οι πιθανές τιμές - καταστάσεις που μπορεί να λάβει το γνώρισμα αυτό, ονομάζονται τιμές χαρακτηριστικού γνωρίσματος (*allele*).

3.4 Πλεονεκτήματα Των Γενετικών Αλγορίθμων Σε Σχέση Με Τους Συμβατικούς Αλγόριθμους Βελτιστοποίησης

Τα πλεονεκτήματα των γενετικών αλγορίθμων σε σχέση με τους συμβατικούς αλγόριθμους βελτιστοποίησης είναι ότι [42]:

1. Εργάζονται με μια κωδικοποίηση του συνόλου τιμών του προβλήματος και όχι με τις παραμέτρους του.

2. Κάνουν αναζήτηση σε πολλά σημεία ταυτόχρονα και όχι μόνο σε ένα μειώνοντας έτσι την πιθανότητα να βρεθεί ένα τοπικό μέγιστο και όχι ένα ολικό μέγιστο.

3. Χρησιμοποιούν μόνο την συνάρτηση καταλληλότητα και καμιά άλλη πληροφορία.

4. Δεν χρησιμοποιούν ντετερμινιστικούς κανόνες μετάβασης αλλά πιθανοθεωρητικούς.

3.5 Δομή Ενός Γενετικού Αλγορίθμου

Στην εικόνα 3.1 φαίνεται η γενική δομή ενός γενετικού αλγορίθμου [42].

$X^\star \longleftarrow \text{simpleEA}(\text{cmp}_F, ps)$

Input: cmp_F: the comparator function which allows us to compare the utility of two solution candidates
Input: ps: the population size
Data: t: the generation counter
Data: Pop: the population
Data: $Mate$: the mating pool
Data: v: the fitness function resulting from the fitness assigning process
Output: $X^\star$: the set of the best elements found

```
1  begin
2      t ⟵ 0
3      Pop ⟵ createPop(ps)
4      while ¬terminationCriterion() do
5          v ⟵ assignFitness(Pop, cmp_F)
6          Mate ⟵ select(Pop, v, ps)
7          t ⟵ t + 1
8          Pop ⟵ reproducePop(Mate)
9      return extractPhenotypes(extractOptimalSet(Pop))
10 end
```

Εικόνα 3.1: Βασική Δομή ενός γενετικού αλγορίθμου

Ο αλγόριθμος δέχεται σαν εισόδους:

- Την συνάρτηση σύγκρισης cmp_F με την οποία συγκρίνονται δύο υποψήφιες λύσεις του προβλήματος
- Το μέγεθος του πληθυσμού *ps*

Οι εσωτερικές μεταβλητές που χρησιμοποιούνται από τον αλγόριθμο είναι :

- Ο αριθμός της τρέχουσας γενιάς t.
- Ο πληθυσμός Pop.

Ο αλγόριθμος περιέχει τα παρακάτω βήματα:

1. Η συνάρτηση createPop παράγει τον αρχικό πληθυσμό ο οποίος αποτελείται από ps άτομα στην πρώτη επανάληψη του αλγορίθμου t=0. Ο αρχικός πληθυσμός παράγεται τυχαία και το μέγεθος του παραμένει σταθερό κατά την διάρκεια της εκτέλεσης του ΕΑ.

2. Το κριτήριο τερματισμού που αποφασίζει εάν ο αλγόριθμος θα συνεχίσει να εκτελείται παρόλο που δεν έχει φθάσει στην γενιά τερματισμού του. Αυτό συμβαίνει εάν η συνάρτηση καταλληλότητας έχει φτάσει στην επιθυμητή τιμή.

3. Οι περισσότεροι αλγόριθμοι εξέλιξης χρησιμοποιούν μία βαθμωτή συνάρτηση καταλληλότητας σε κάθε πληθυσμό συγκρίνοντας το διάνυσμα της συνάρτησης καταλληλότητας των επιμέρους ατόμων του πληθυσμού με άλλους πληθυσμούς. Το διάνυσμα αυτό δημιουργείται από την διαδικασία assignFitness.

4. Ένας αλγόριθμος επιλογής select ο οποίος επιλέγει *ps* "ενδιαφέροντα" άτομα και τα τοποθετεί στην λίστα των υποψηφίων για ζευγάρωμα. Για την επιλογή αυτή ο αλγόριθμος φροντίζει άτομα με καλύτερη απόδοση δηλαδή μεγαλύτερη τιμή στην συνάρτηση καταλληλότητας να έχουν μεγαλύτερη πιθανότητα να επιλεχθούν και να περάσουν στην επόμενη γενιά. Η επιλογή των χρωμοσωμάτων γίνεται με διάφορους τρόπους όπως:

   i. **Τυχαία Επιλογή** (*random selection*), όπου όπως λέει και το όνομα της τα χρωμοσώματα που θα αντιγραφούν επιλέγονται τυχαία

   ii. **Ανταγωνιστική Επιλογή** (*tournament selection*), όπου επιλέγονται δύο τυχαία χρωμοσώματα για το κάθε ένα από αυτά θα υπολογιστεί η συνάρτηση καταλληλότητας και θα καταταχθούν ανάλογα, από το μεγαλύτερο προς το μικρότερο. Έστω μία σταθερά λ επιλεγμένη από το διάστημα [0...1] και ένας τυχαίος αριθμός μ. Εάν λ>=μ τότε αναπαράγεται το χρωμόσωμα με την μεγαλύτερη συνάρτηση καταλληλότητας, ενώ σε άλλη περίπτωση το άλλο χρωμόσωμα. Τα χρωμοσώματα μπορούν να χρησιμοποιηθούν σε επόμενη επιλογή.

   iii. **Επιλογή με βάση την σειρά κατάταξης** (*ranking selection*) όπου τα χρωμοσώματα κατατάσσονται βάση του αποτελέσματος της συνάρτησης καταλληλότητας και επιλέγονται τα κ πρώτα.

   iv. **Επιλογή με βάση τον λόγο καταλληλότητας** (*fitness proportionate selection*) όπου το κ-οστό χρωμόσωμα θα αναπαραχθεί τόσες φορές όσες και το ακέραιο μέρος του πηλίκου του αποτελέσματος της συνάρτησης καταλληλότητας προς τον μέσο όρο των αποτελεσμάτων της συνάρτησης καταλληλότητας όλων των χρωμοσωμάτων.

v. **<u>Επιλογή με βάση τον αλγόριθμο ρουλέτας</u>** (*roulette ranking*), όπου υλοποιείται μία ρουλέτα με σχισμές και ο πληθυσμός ανάλογα με την τιμή που έχει υπολογιστεί από την συνάρτηση καταλληλότητας για τον ίδιο καταλαμβάνει μία σχισμή της ρουλέτας. Όσο πιο καλή είναι η τιμή της συνάρτησης καταλληλότητας τόσο πιο μεγάλη είναι η σχισμή.

5. Με την συνάρτηση reproducePop ένας νέο πληθυσμός δημιουργείται στην λίστα ζευγαριών με την χρήση μετάλλαξης (*mutation*) και διασταυρώσεων (*crossover*). Η διασταύρωση είναι η διαδικασία κατά την οποία δύο χρωμοσώματα χωρίζονται σε ένα ή περισσότερα σημεία και ενώνονται μεταξύ τους. Κλασικοί μέθοδοι διασταύρωσης είναι η διασταύρωση ενός σημείου (*1 – Point crossover*) και η διασταύρωση δύο σημείων (*2 – Points crossover*). Στην διασταύρωση ενός σημείου επιλέγεται ένα σημείο στο χρωμόσωμα και το τμήμα από την αρχή μέχρι εκείνο το σημείο αποτελεί το πρώτο μέρος του χρωμοσώματος ενώ το τμήμα από το σημείο μέχρι το τέλος του πρώτου χρωμοσώματος αποτελεί το δεύτερο μέρος του δεύτερου χρωμοσώματος. Τα νέα χρωμοσώματα που προκύπτουν παίρνουν την θέση των γονέων τους. Η μετάλλαξη επιλέγει τυχαία ένα γονίδιο του χρωμοσώματος και αλλάζει την τιμή του.

6. Οι συναρτήσεις extractOptimalSet και extractPhenotype χρησιμοποιούνται για την εξαγωγή όλων των ατόμων του πληθυσμού που έχουν επικρατήσει μετά από όλες τις διαδικασίες και επιστρέφουν μόνο τον φαινότυπο τους.

3.6 Παράδειγμα Γενετικού Αλγόριθμου: Πρόβλημα Των Οκτώ Βασιλισσών

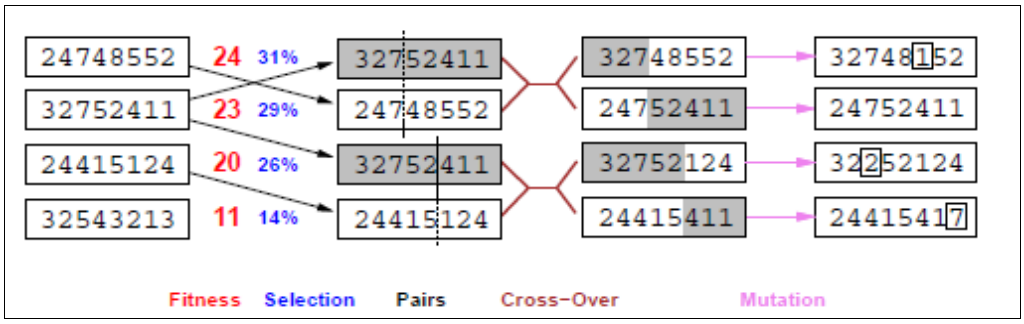
Ένα από τα πιο γνωστά προβλήματα στην επιστήμη της πληροφορικής είναι η τοποθέτηση πάνω σε μία σκακιέρα διάστασης κ x κ, έναν αριθμό από κ βασίλισσες με τέτοιο τρόπο ώστε να μην απειλείται καμία βασίλισσα. Το πρόβλημα είναι γνωστό και ως “n queens problem” και έχει επιλυθεί με διάφορους τρόπους. Σε αυτήν την παράγραφο, θα επιλυθεί με την βοήθεια των γενετικών αλγορίθμων [35].

Ο γενότυπος του προβλήματος θα είναι οι θέσεις των οκτώ βασιλισσών πάνω στην σκακιέρα. Το μαθηματικό μοντέλο το οποίο απεικονίζει την θέση της μάθε μίας από τις βασίλισσες στην σκακιέρα, ορίζεται ως εξής: Ορίζεται ένας πίνακας Π, μίας διάστασης, που ορίζει την στήλη της σκακιέρας. Οι τιμές (allele) που παίρνει το κάθε ένα στοιχείο του πίνακα που αποτελεί και το κύριο χαρακτηριστικό είναι

από 1 έως και 8 και ορίζει την γραμμή της βασίλισσας. Αυτό το μοντέλο ορίζει τον φαινότυπο του γενετικού αλγόριθμου.

Η συνάρτηση καταλληλότητας (fitness function) του αλγορίθμου θα είναι το άθροισμα των ζευγαριών των βασιλισσών που δεν δέχονται επίθεση η μία από την άλλη. Για το πρόβλημα των 8 βασιλισσών, η μέγιστη τιμή που μπορούν να πάρουν είναι 28. Έτσι λοιπόν το κριτήριο τερματισμού του αλγορίθμου (terminationCriterion), εάν δεν φτάσει ο επιθυμητός αριθμός των γενιών του γενετικού αλγορίθμου, είναι ότι η συνάρτηση καταλληλότητας να γίνει ίση με 28.

Στο πρώτο βήμα του αλγόριθμου δημιουργείτε ο πληθυσμός που αποτελείται από k τυχαίους πίνακες. Έστω ότι οι πληθυσμοί που έχουν δημιουργηθεί είναι οι παρακάτω: [2, 4, 7, 4, 8, 5, 5, 2], [3, 2, 7, 5, 2, 4, 1, 1], [2, 4, 4, 1, 5, 1, 2, 4], [3, 2, 5, 4, 3, 2, 1, 3].



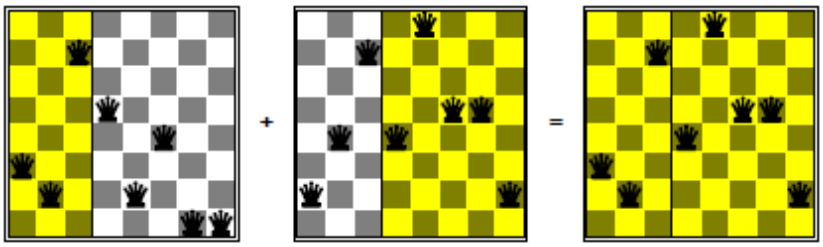
Σχήμα 3.1: Πράξεις των γενετικών Αλγορίθμων

Τα βήματα παραγωγής της επόμενης γενιάς φαίνονται στο σχήμα 3.1. Με κόκκινο χρώμα εμφανίζεται το αποτέλεσμα της συνάρτησης καταλληλότητας. Στον συγκεκριμένο αλγόριθμο η πιθανότητα επιλογής είναι ανάλογη του αποτελέσματος της συνάρτησης καταλληλότητας και σε ποσοστό εμφανίζεται στην $2^{η}$ στήλη του πίνακα με γαλάζιο χρώμα.

Στην τρίτη στήλη εμφανίζονται οι πληθυσμοί διατεταγμένα από αυτόν με την μεγαλύτερη τιμή προς αυτόν με την μικρότερη τιμή της συνάρτησης καταλληλότητας. Αυτό είναι ένας τρόπος επιλογής, που αποκόβει όλες τις λύσεις των οποίων η τιμή της συνάρτησης καταλληλότητας είναι κάτω από ένα επίπεδο. Παρατηρούμε ότι μία πιθανή λύση εμφανίζεται δύο φορές και μία άλλη καμία.

Στην επόμενη στήλη εμφανίζεται η διαδικασία της διασταύρωσης. Στην συγκεκριμένη περίπτωση χρησιμοποιήθηκε διασταύρωση ενός σημείου. Αυτό σημαίνει ότι ένα σημείο (αριθμός στήλης) επιλέγεται τυχαία για κάθε ένα από τα ζευγάρια που λαμβάνουν μέρος στη διασταύρωση.

Έστω λοιπόν ότι επιλέγεται το 3ο σημείο όπως φαίνεται και στην εικόνα 3.2. Η νέα πιθανή λύση του αλγορίθμου θα αποτελείται από τις στήλες 1 – 3 της πρώτης λύσης του από το ζευγάρι και από τις στήλες 4 – 8 της δεύτερη λύσης του ζευγαριού. Οι στήλες που δεν είναι χρωματισμένες, χάνονται κατά την διασταύρωση. Είναι χαρακτηριστικό ότι εάν οι γονείς είναι πολύ διαφορετικοί μεταξύ τους, οι απόγονοι δεν είναι σίγουρο ότι θα οδηγήσουν τον πληθυσμό σε μία κατάσταση που θα είναι καλύτερη σε σχέση με την προηγούμενη. Συνήθως όμως η διασταύρωση δίνει στα αρχικά στάδια αποτελέσματα που διαφέρουν μεταξύ τους πάρα πολύ, ενώ στην συνέχεια οι διαφορές είναι πολύ μικρότερες.



Εικόνα 3.2: Λειτουργία της διασταύρωσης

Το επόμενο βήμα είναι η εφαρμογή της διαδικασίας της μετάλλαξης (mutation). Η μετάλλαξη εφαρμόζεται σε μία πιθανή λύση αλλάζοντας τιμή σε ένα μόνο σημείο της. Στην περίπτωση που περιγράφηκε η μετάλλαξη εφαρμόστηκε στην πρώτη, τρίτη και τέταρτη πιθανή λύση και στο έκτο, τρίτο και όγδοο σημείο αντίστοιχα. Στο πρόβλημα που εξετάζεται η μετάλλαξη σημαίνει την μεταφορά μίας βασίλισσας από μία σειρά σε μία άλλη.

3.7 Προγραμματισμός Γενετικών Αλγορίθμων

Ο προγραμματισμός των γενετικών αλγορίθμων μπορεί να γίνει με την χρήση διάφορων εργαλείων και γλωσσών προγραμματισμού. Έτσι είναι εύκολο να γίνει ο προγραμματισμός στις πιο δημοφιλείς γλώσσες προγραμματισμού όπως *C*, *Java* και *Python,* είτε να χρησιμοποιηθούν έτοιμα πακέτα για τον προγραμματισμό σε διάφορα προγράμματα όπως *Matlab* και *Mathematica*.

Το πλεονέκτημα της χρήσης έτοιμων πακέτων είναι ότι έχουν υλοποιηθεί πάρα πολλές εκδόσεις των αλγόριθμων επιλογής, διασταύρωσης και μετάλλαξης και είναι υλοποιημένες με τέτοιο τρόπο ώστε η ταχύτητα εκτέλεσης τους να είναι αρκετά καλή και τα αποτελέσματα που παράγονται να είναι σωστά. Τα πακέτα αυτά δίνουν πάρα πολλές επιλογές στον χρήστη με αποτέλεσμα η επιλογή να γράψει κάποιος εξαρχής το δικό του πρόγραμμα για την υλοποίηση ενός γενετικού αλγόριθμου να μην κρίνεται αναγκαία εκτός από την περίπτωση που πρέπει να γίνει κάτι εντελώς ειδικό.

Τα κυριότερα παραδείγματα τέτοιων πακέτων είναι :

1. Το *PY Evolve* το οποίο είναι ένα πακέτο που αναφέρεται στην γλώσσα προγραμματισμού *Python*. Είναι ένα ολοκληρωμένο πακέτο που δίνει πάρα πολλές δυνατότητες στο χρήστη, όχι μόνο από άποψη πληθώρας αλγορίθμων για τα βήματα των γενετικών αλγορίθμων, αλλά και παρουσίασης των αποτελεσμάτων, όπως και της διαδικασίας εισόδου των δεδομένων στον Αλγόριθμο για παράδειγμα μέσω διαφόρων Βάσεων δεδομένων.[7]

2. Το *Global Optimization Toolbox* είναι το αντίστοιχο πακέτο για *MATLAB*. Επεκτείνει τις δυνατότητες της *MATLAB* για το κομμάτι των γενετικών αλγορίθμων. [8]

3. Το *GAlib* είναι μία συντομογραφία του *Genetic Algorithm Library* είναι το πιο γνωστό πακέτο σε *C++* για τους γενετικούς αλγορίθμους. Στην τελευταία του έκδοση υποστηρίζει και την *Visual C++* της *Microsoft* με αποτέλεσμα να είναι μία βιβλιοθήκη, που μπορεί να εκτελεστεί στα πιο γνωστά λειτουργικά συστήματα όπως *Linux*, *Windows*, *MAC OS* και συνεργάζεται με όλους σχεδόν τους compilers.[9]

4. Το *GAUL* μία συντομογραφία του *Genetic Algorithm Utility Library* και είναι μία βιβλιοθήκη γραμμένη σε *C* που μπορεί να εκτελεστεί σε *Linux* και *Windows*[10].

5. Το *GAGS* είναι μία βιβλιοθήκη που είχε αρχικά γραφεί για την γλώσσα προγραμματισμού *C*. Μετέπειτα η συγκεκριμένη βιβλιοθήκη μετατράπηκε ώστε να δουλεύει και σε άλλες γλώσσες προγραμματισμού όπως *Tcl/Tlk*, *C++*, *JAVA*.[11]

7 http://pyevolve.sourceforge.net/
8 http://www.mathworks.com/products/global-optimization/index.html;jsessionid=029fd7fce60d3f259e3e1a5c359e
9 http://lancet.mit.edu/ga/
10 http://gaul.sourceforge.net/
11 http://www.aridolan.com/ga/gaa/gaa.html#GAA

6. Το *ECJ* είναι μία συντομογραφία των λέξεων *Evolution Computational Java Research System* και είναι γραμμένο σε *JAVA*. Η συγκεκριμένη βιβλιοθήκη αναλύεται στην επόμενη παράγραφο.

3.8 Η Βιβλιοθήκη ECJ[12]

Η βιβλιοθήκη ECJ, δημιουργήθηκε στο τμήμα Η/Υ του πανεπιστήμιο *George Mason*, έχει φτάσει στην 20η έκδοση και είναι γραμμένο αποκλειστικά για *Java*. Είναι η πιο διαδεδομένη βιβλιοθήκη για χρήση με Java και έχει πάρα πολλές επιλογές για την αναπαράσταση του χρωμοσώματος. Άλλα πλεονεκτήματα είναι οι πιο ποιοτικές λύσεις αλλά και η πιο γρήγορη εκτέλεση του γενετικού αλγορίθμου. Το μόνο μειονέκτημα του είναι η φτωχή τεκμηρίωση του [32].

Κυριότερα χαρακτηριστικά είναι:

1. Γραφικό περιβάλλον με δυνατότητα αναπαράστασης με γράφημα

2. Παρουσίαση αρχείου κινήσεων ανεξάρτητα από το λειτουργικό σύστημα

3. Αρχείο παραμέτρων με συγκεκριμένη ιεραρχία

4. Πολυνηματικός προγραμματισμός

5. Γεννήτρια Τυχαίων Αριθμών με την μέθοδο *Mersenne Twister Random Number Generators*[13].

6. Δυνατότητα εφαρμογής στις κλάσεις του προγράμματος ώστε να μπορούν να υλοποιηθούν και άλλες μορφές των αλγορίθμων.

7. Πολλαπλές αναπαραστάσεις Γενετικών Αλγορίθμων: Σταθερού και Μεταβλητού μήκους.

8. Πολλές διαδικασίες για την λειτουργίες της επιλογής, της διασταύρωσης και της μετάλλαξης.

9. Δυνατότητα διαβάσματος και εγγραφής των δεδομένων από αρχείο με συγκεκριμένη μορφή.

10. Το πρόγραμμα μπορεί να χρησιμοποιηθεί και για εκτέλεση ενός γενετικού αλγόριθμου αλλά και για γενετικό προγραμματισμό.

12 http://cs.gmu.edu/~eclab/projects/ecj/
13 http://en.wikipedia.org/wiki/Mersenne_twister

. Κεφάλαιο 4
Υλοποίηση Εξεταστικού Ημερολογίου Με Την Χρήση Γενετικών Αλγορίθμων

Στα ιδρύματα της τριτοβάθμιας εκπαίδευσης όπως αναφέρθηκε και προηγούμενα, είναι αρκετά δύσκολο να υλοποιηθεί ένα εξεταστικό πρόγραμμα που καλύπτει τις ανάγκες όλων των εμπλεκόμενων πλευρών.

Ενδεικτικά για κάθε μία από τις ομάδες των ενδιαφερομένων για το πρόβλημα κατάρτισης του εξεταστικού ημερολογίου, οι ανάγκες είναι:

1. Οι φοιτητές επιθυμούν ένα εξεταστικό πρόγραμμα που θα τους δίνει το μεγαλύτερο δυνατόν περιθώριο ανάμεσα στα μαθήματα ώστε να υπάρχει αρκετός χρόνος για επανάληψη, χωρίς να υπάρχουν επικαλύψεις μαθημάτων και κατά προτίμηση τα πιο σημαντικά μαθήματα να έχουν μεγάλη χρονική απόσταση το ένα από το άλλο.

2. Οι εισηγητές επιθυμούν ένα εξεταστικό πρόγραμμα με όσο το δυνατόν μεγαλύτερο περιθώριο ανάμεσα στα μαθήματα που διδάσκουν ώστε να μπορούν να διορθώσουν τα εξεταστικά γραπτά με μεγαλύτερη άνεση, αλλά και να μπορέσουν να φτιάξουν πιο εύκολα τα θέματα στα οποία θα

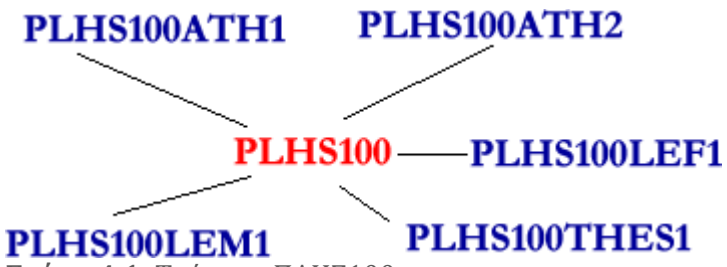
εξεταστούν οι φοιτητές. Επίσης και αυτοί θέλουν να μην υπάρχουν αλληλοεπικαλύψεις ανάμεσα στα μαθήματα τα οποία διδάσκουν για να είναι ανά πάσα στιγμή διαθέσιμοι στην αίθουσα εξέτασης για τυχόν απορίες και προβλήματα που μπορούν να προκύψουν.

3. Οι επιτηρητές επιθυμούν ένα εξεταστικό πρόγραμμα με όσον το δυνατόν περισσότερες επιτηρήσεις κατά την διάρκεια της ημέρας ώστε να μειώσουν τις ημέρες που θα απασχολούνται κατά την διάρκεια της εξέτασης.

4. Το διοικητικό προσωπικό του πανεπιστημίου επιθυμεί ένα εξεταστικό ημερολόγιο που θα τους δίνει την δυνατότητα να αναρτούνται οι βαθμολογίες όσο το δυνατόν πιο σύντομα.

Τα συμβατικά πανεπιστήμια (ΑΕΙ – ΤΕΙ) στεγάζονται σε συγκεκριμένα κτήρια, που μπορεί να μην βρίσκονται στον ίδιο φυσικό χώρο είτε ανήκουν στο εκπαιδευτικό ίδρυμα είτε ενοικιάζονται από αυτό για όλο τον χρόνο. Έτσι υπάρχει συγκεκριμένος αριθμός αιθουσών οι οποίες χρησιμοποιούνται στην εξεταστική περίοδο και ανάλογα με τις ανάγκες που θα προκύψουν μπορεί ο αριθμός αυτός να αλλάξει, χωρίς να μεταβληθεί το κόστος για το εκπαιδευτικό ίδρυμα. Η διάρκεια της εξεταστικής περιόδου είναι προκαθορισμένη από την έναρξη της χρονιάς.

Στα ανοικτά πανεπιστήμια λόγω της διαφορετικής φιλοσοφίας με τα συμβατικά πανεπιστήμια έχουμε την διάσπαση των μαθημάτων σε τμήματα ανάλογα με την πόλη την οποία έχει δηλώσει ο φοιτητής για να παρακολουθήσει το συγκεκριμένο μάθημα.



Σχήμα 4.1: Τμήματα ΠΛΗΣ100

Έτσι για παράδειγμα (Σχήμα 4.1) η θεματική ενότητα PLHS100 διδάσκεται σε τέσσερις πόλεις την Αθήνα, την Θεσσαλονίκη, την Λευκωσία και την Λεμεσό. Στην Αθήνα μάλιστα λόγω πολλών φοιτητών που έχουν δηλώσει αυτήν την θεματική ενότητα, έχει διασπαστεί σε δύο τμήματα τα PLHS100ATH1 και PLHS100ATH2. Αυτό σημαίνει ότι όλες αυτές οι θεματικές ενότητες θα πρέπει να εξεταστούν την ίδια εξεταστική ώρα.

4.1 Περιορισμοί Εξεταστικού Ημερολογίου

Οι ανάγκες που περιγράφηκαν στην προηγούμενη παράγραφο, μπορούν να κωδικοποιηθούν στην κατάρτιση του εξεταστικού ημερολογίου με την μορφή των παρακάτω περιορισμοί:

1. Ένας φοιτητές δεν μπορεί να εξετάζεται σε δύο μαθήματα ταυτόχρονα
2. Ένας εισηγητής δεν μπορεί να έχει ταυτόχρονα σε δύο μαθήματα εξέταση.
3. Ένας επιτηρητής δεν μπορεί να επιτηρεί δύο μαθήματα την ίδια χρονική στιγμή.
4. Οι φοιτητές που ανήκουν σε διαφορετικά τμήματα ενός μαθήματος πρέπει να εξετάζονται την ίδια χρονική στιγμή.
5. Ο αριθμός των φοιτητών ενός τμήματος που εξετάζεται σε μία χρονική στιγμή θα πρέπει να είναι ίσος ή μικρότερος από τον αριθμό των διαθέσιμων θέσεων προς εξέταση που προβλέπεται να υπάρχουν σε μία πόλη.
6. Το μήκος της εξεταστικής περιόδου θα πρέπει να είναι όσο το δυνατόν μεγαλύτερο ώστε να υπάρχει αρκετός χρόνος για τους φοιτητές να μελετήσουν.
7. Ο αριθμός των άδειων θέσεων σε μία αίθουσα στην οποία εξετάζεται ένα μάθημα θα πρέπει να είναι όσο το δυνατόν πιο κοντά στο μηδέν.
8. Η εξεταστική περίοδος θα πρέπει να είναι όσο το δυνατόν πιο μικρή

Όπως περιγράφηκε σε προηγούμενη παράγραφο, οι περιορισμοί ανάλογα με το πόσο σημαντικοί είναι χωρίζονται σε δύο κατηγορίες τους ανελαστικούς περιορισμούς και τους ελαστικούς περιορισμούς.

- **Ανελαστικοί "hard"** ονομάζονται οι περιορισμοί που πρέπει να ικανοποιηθούν προκειμένου το εξεταστικό ημερολόγιο που θα προκύψει να είναι *αποδεκτό (feasible)*, δηλαδή να μπορεί να χρησιμοποιηθεί από το Πανεπιστήμιο. Παραδείγματα ανελαστικών περιορισμών είναι από τους παραπάνω αναφερόμενους οι 1 – 5.

- **Ελαστικοί "soft"** ονομάζονται οι περιορισμοί που είναι καλό να ικανοποιηθούν, αλλά όχι απαραίτητο. Δηλαδή ένα εξεταστικό ημερολόγιο μπορεί να είναι αποδεκτό και να χρησιμοποιηθεί από ένα Πανεπιστήμιο, χωρίς να έχει ικανοποιηθεί κάποιος από τους ελαστικούς περιορισμούς του. Η ικανοποίηση των ελαστικών περιορισμών ορίζει εάν ένα εξεταστικό ημερολόγιο είναι πιο καλό ή όχι από ένα άλλο. Παραδείγματα ελαστικών περιορισμών είναι από τους παραπάνω αναφερόμενους οι 6 – 8.

Στον γενετικό αλγόριθμο οι ανελαστικοί περιορισμοί θα έχουν πολύ μεγαλύτερο βάρος στην συνάρτηση καταλληλότητας από τους ελαστικούς. Έτσι ακόμα και εάν δεν ικανοποιούνται οι

ελαστικοί περιορισμοί, η συνάρτηση καταλληλότητας θα πλησιάζει στην επιθυμητή τιμή και ο αλγόριθμος θα βρει μία λύση, παρόλο που αυτή μπορεί να μην είναι η καλύτερη δυνατή.

4.2 Μοντελοποίηση Διαγράμματος Οντοτήτων - Συσχετίσεων

Σε έναν γενετικό αλγόριθμο ένα από τα ποιο σημαντικά τμήματα είναι ο καθορισμός της αναπαράστασης. Τα πλεονεκτήματα μίας καλής αναπαράστασης είναι[14]:

1. Μικρός χρόνος εκτέλεσης του αλγορίθμου.

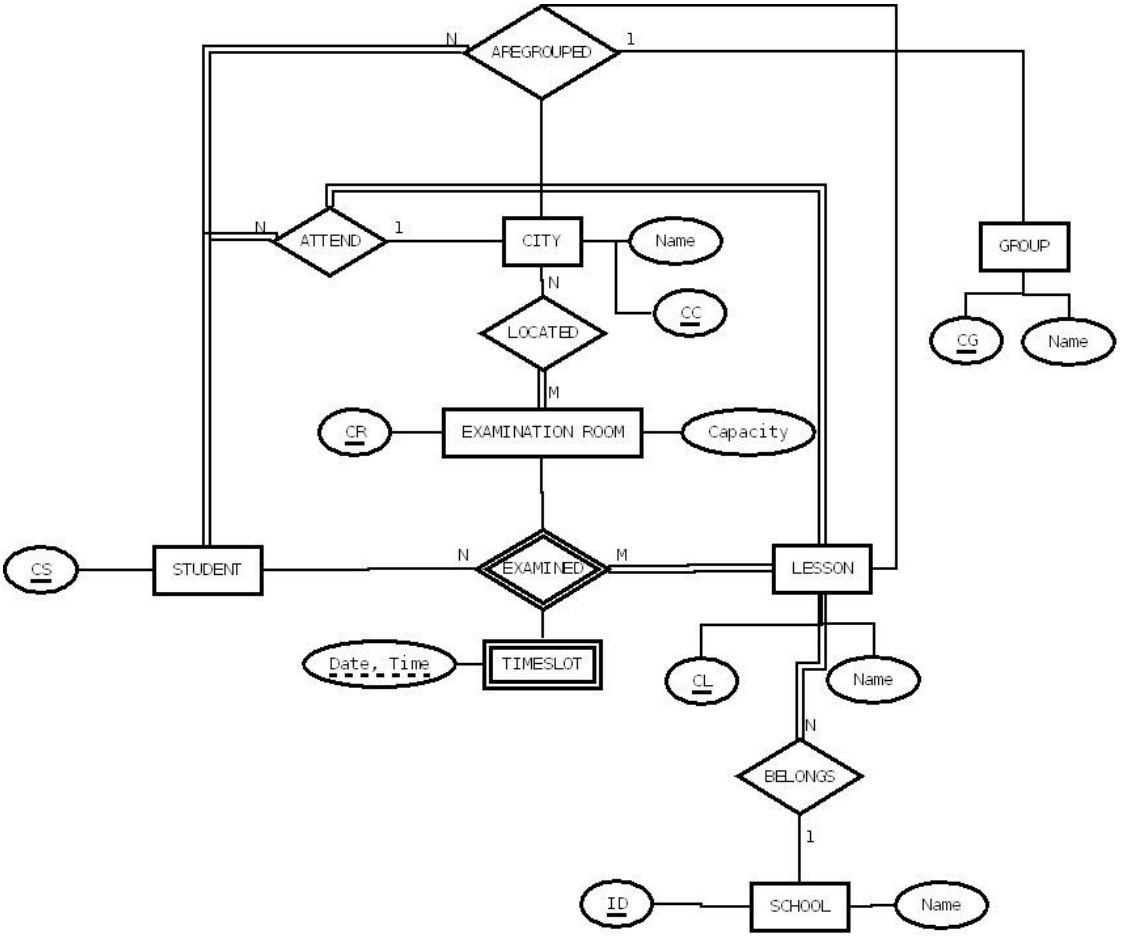
2. Κατά την διαδικασία της διασταύρωσης και της μετάλλαξης πρέπει οι διαδικασίες αυτές να λαμβάνουν υπόψιν τα όρια των δεδομένων προκειμένου να μην υπάρχει ανάμιξη αυτών.

3. Όσο πιο απλές είναι οι τιμές του χρωμοσώματος τόσο πιο εύκολα μπορούν να γίνουν οι διάφορες πράξεις του γενετικού αλγορίθμου.

Από τα παραπάνω φαίνεται πόσο σημαντική είναι η αναπαράσταση του χρωμοσώματος ενός γενετικού αλγόριθμου. Ένας από τους πιο αξιόπιστους τρόπους για να αναπαρασταθούν σωστά τα δεδομένα είναι το Διάγραμμα Οντοτήτων – Συσχετίσεων. Το διάγραμμα αυτό χρησιμοποιείται κυρίως στην κατασκευή βάσεων δεδομένων αλλά με μεγάλη ευκολία μπορεί να χρησιμοποιηθεί για την κατασκευή ενός UML Διαγράμματος και κατ' επέκταση στον ορισμό των ιδιοτήτων και των μεθόδων των αντικειμένων που χρησιμοποιούνται για την επίλυση ενός προβλήματος.

14 http://webcache.googleusercontent.com/search?q=cache:ybvzENwiYTwJ:en.wikipedia.org/wiki/Genetic_algorithm+genetic+algorithm+representation+and+quality&cd=1&hl=el&ct=clnk&gl=gr&client=firefox-a

Στην σχήμα 4.2 φαίνεται το διάγραμμα οντοτήτων – συσχετίσεων της βάσης δεδομένων που θα μπορούσε να υποστηρίξει το εξεταστικό ημερολόγιο σε ένα ανοικτό πανεπιστήμιο. Το διάγραμμα αυτό περιέχει τα απολύτως απαραίτητα στοιχεία για να μοντελοποιηθεί το τμήμα του προβλήματος που αφορά τον φοιτητή, το μάθημα, την σχολή, την αίθουσα εξέτασης και τον χρόνο εξέτασης.



Σχήμα 4.2: Διάγραμμα Οντοτήτων - Συσχετίσεων

4.2.1 Οντότητες Εξεταστικού Ημερολογίου

Οι οντότητες που διακρίνονται στο διάγραμμα οντοτήτων – συσχετίσεων είναι οι:

- Ο Φοιτητής (*STUDENT*) είναι ο εξεταζόμενος σε ένα συγκεκριμένο μάθημα. Κύρια ιδιότητα του είναι ο Αριθμός Μητρώου (*CS*).

- Το μάθημα (*LESSON*) αποτελεί την οντότητα των μαθημάτων τα οποία παρακολουθούνται από τους φοιτητές και θα εξεταστούν. Οι κυριότερες ιδιότητες της οντότητας είναι ο κωδικός

μαθήματος (*CL*) που αποτελεί και κύριο κλειδί της οντότητας, το όνομα του μαθήματος (*NAME*) .

- Η Σχολή (*SCHOOL*) αποτελεί την οντότητα της σχολής στην οποία ανήκει το μάθημα στο οποίο πρόκειται να εξεταστούν οι φοιτητές. Οι ιδιότητες της οντότητας είναι ο κωδικός της σχολής (*ID*) που αποτελεί και το κύριο κλειδί και το όνομα της σχολής *(NAME).*

- Η Αίθουσα (*EXAMINATION ROOM*) είναι η οντότητα των εξεταστικών αιθουσών σε μία πόλη, όπου γίνεται η εξέταση του μαθήματος. Οι ιδιότητες της οντότητας είναι ο κωδικός της (*CR*) που αποτελεί και το κύριο κλειδί και η χωρητικότητα της (*CAPACITY*).

- Η ώρα Εξέτασης (*TIMESLOT*) είναι η ασθενής οντότητα που περιγράφει την ημερομηνία και ώρα μιας εξέτασης. Οι ιδιότητες της είναι η ημερομηνία (*DATE*), η ώρα (*TIME*), ο συνδυασμός των οποίων αποτελεί κλειδί για την οντότητα.

- Η πόλη (*CITY*) είναι η πόλη στην οποία υπάρχουν οι αίθουσες εξέτασης. Οι ιδιότητες της είναι η ονομασία (*NAME*) και ο κωδικός της (*CC*) που είναι και το κύριο κλειδί της.

- Το τμήμα (*GROUP*) είναι η οντότητα που περιγράφει το τμήμα στο οποίο ανήκει ο φοιτητής. Οι ιδιότητες της είναι ο κωδικός τμήματος (*CG*) που αποτελεί και το κύριο κλειδί της οντότητας καθώς και η ονομασία (*NAME*) του τμήματος. Στα συμβατικά πανεπιστήμια δημιουργείται ένα τμήμα για κάθε μάθημα, ενώ στα ανοικτά δημιουργείται ένα συνήθως τμήμα για κάθε πόλη στην οποία υπάρχουν φοιτητές που παρακολουθούν το μάθημα.

4.2.2 Συσχετίσεις Εξεταστικού Ημερολογίου

Οι οντότητες συνδέονται μεταξύ τους με διάφορες συσχετίσεις. Οι κύριες συσχετίσεις που δημιουργούνται είναι:

1. Η συσχέτιση *LOCATED* ενώνει τις οντότητες *CITY* και *EXAMINATION ROOM* και έχει την έννοια ότι η αίθουσα στην οποία θα γίνει η εξέταση ανήκει σε μία συγκεκριμένη πόλη. Η συσχέτιση είναι 1:N αφού σε μία πόλη μπορεί να υπάρχουν πολλές αίθουσες εξέτασης, αλλά μία αίθουσα εξέτασης ανήκει μόνο σε μία πόλη.

2. Η συσχέτιση *EXAMINED* ενώνει τις οντότητες *EXAMINATION ROOM, STUDENT, TIMESLOT* και *LESSON*. Είναι η κεντρική συσχέτιση στην οποία ορίζεται σε ποιο εξεταστικό δωμάτιο θα εξεταστεί ο φοιτητής σε ένα μάθημα και η ακριβής χρονική στιγμή της εξέτασης.

3. Η συσχέτιση *ATTEND* συνδέει τις οντότητες *CITY, STUDENT* και *LESSON*. Ορίζει την πόλη στην οποία ένας φοιτητής παρακολουθεί τις διαλέξεις για ένα μάθημα. Η συσχέτιση είναι 1:N γιατί σε μία πόλη υπάρχουν πολλοί φοιτητές που παρακολουθούν ένα μάθημα ενώ ένα φοιτητής μπορεί να παρακολουθεί ένα μάθημα σε μία μόνο πόλη.

4. Η συσχέτιση *BELONG* συνδέει τις οντότητες *SCHOOL* και *LESSON*. Ορίζει σε ποια σχολή του πανεπιστημίου ανήκει το μάθημα που θα εξεταστεί. Η συσχέτιση είναι 1:N γιατί το μάθημα ανήκει σε μία μόνο σχολή, ενώ η σχολή έχει πολλά μαθήματα.

5. Η συσχέτιση *AREGROUPED* συνδέει τις οντότητες *GROUP*, *LESSON*, *CITY* και *STUDENT*. Ορίζει την ομαδοποίηση των μαθημάτων σε GROUP με βάση τους φοιτητές που παρακολουθούν ένα μάθημα σε μία πόλη.

4.3 Σχεσιακό Μοντέλο Εξεταστικού Ημερολογίου

Έχοντας υπόψιν το E-R Διάγραμμα που περιγράφηκε στην προηγούμενη παράγραφο, ορίζονται τα παρακάτω σύνολα:

1. Το S είναι το σύνολο των φοιτητών που πρόκειται να πάρουν μέρος στην συγκεκριμένη εξεταστική περίοδο, με S_i συμβολίζεται ο φοιτητής i.

2. Το L είναι το σύνολο των μαθημάτων που πρόκειται να εξεταστούν στην συγκεκριμένη εξεταστική περίοδο, με L_i συμβολίζεται το μάθημα i.

3. Το T είναι το σύνολο των εξεταστικών διαστημάτων. Σαν εξεταστικό διάστημα ορίζεται το διάστημα με ώρα έναρξης t_1 και ώρα λήξης t_2. Κάθε ένα από τα εξεταστικά διαστήματα μπορεί να μην έχουν την ίδια διάρκεια μεταξύ τους. Με T_i, συμβολίζεται το εξεταστικό διάστημα i.

4. Το C είναι το σύνολο των πόλεων στις οποίες έχουν δημιουργεί τμήματα των μαθημάτων που πρόκειται να διδαχθούν. Με C_i, συμβολίζεται η πόλη i.

5. Το *R* είναι το σύνολο των εξεταστικών δωματίων στα οποία θα γίνουν οι εξετάσεις. Με R_i, συμβολίζεται το εξεταστικό δωμάτιο *i*.

6. Με *SL* απεικονίζεται η συσχέτιση που συνδέει το σύνολο των μαθητών *S* με το σύνολο των μαθημάτων *L*. Η συγκεκριμένη συσχέτιση αποτελεί το καρτεσιανό γινόμενο του *S* και του *L*, δηλαδή $SL = S\,x\,L$. Τα στοιχεία που συμπεριλαμβάνονται σε αυτό είναι τα μαθήματα στα οποία εξετάζονται οι φοιτητές του πανεπιστημίου.

7. Με *LC* απεικονίζεται η συσχέτιση που συνδέει το σύνολο των μαθημάτων *L* με το σύνολο των πόλεων *C*. Η συγκεκριμένη συσχέτιση αποτελεί το καρτεσιανό γινόμενο του *C* και του *L*, δηλαδή $LC = C\,x\,L$. Τα στοιχεία που συμπεριλαμβάνονται σε αυτό είναι τα μαθήματα τα οποία διδάσκονται στις διάφορες πόλεις που έχει τμήματα το πανεπιστήμιο.

8. Με *EX* απεικονίζεται η συσχέτιση που συνδέει το σύνολο των φοιτητών *S*, το σύνολο των εξεταστικών διαστημάτων *T*, το σύνολο των μαθημάτων *L* και το σύνολο των δωμάτιο στα οποία γίνεται η εξέταση *R*. Η συγκεκριμένη συσχέτιση αποτελεί το καρτεσιανό γινόμενο του *S*, του *T*, του *L* και του *R*, δηλαδή $EX = S\,x\,L\,x\,T\,x\,R$. Τα στοιχεία που περιλαμβάνονται σε αυτήν είναι η αίθουσα στην οποία εξετάζεται ο φοιτητής σε ένα συγκεκριμένο μάθημα μία συγκεκριμένη εξεταστική ώρα.

9. Με *RC* απεικονίζεται η συσχέτιση που συνδέει των σύνολο των πόλεων *C* με το σύνολο των εξεταστικών δωματίων *R*. Η συγκεκριμένη συσχέτιση αποτελεί το καρτεσιανό γινόμενο του *C* και του *R*, δηλαδή $RC = C\,x\,R$.Περιλαμβάνονται όλοι οι συνδυασμοί των εξεταστικών αιθουσών με τις πόλεις στις οποίες ανήκουν.

Ο Γενετικός αλγόριθμος είναι ένας αλγόριθμος βελτιστοποίησης, οπότε πρέπει να βρεθεί μία συνάρτηση της οποίας το αποτέλεσμα στην περίπτωση του εξεταστικού ημερολογίου πρέπει να ελαχιστοποιηθεί.

Η συνάρτηση αυτή θα αποτελείται από το άθροισμα των γινομένων των μεταβλητών κόστους των περιορισμών επί το βάρος των περιορισμών αυτών στην συνάρτηση. Δηλαδή θα είναι της μορφής

$$fitness = \sum_{i=1}^{n} cost_i * w_i$$

όπου $cost_\iota$ είναι η μεταβλητή για τον ι περιορισμό και w_i είναι η μεταβλητή για το βάρος του i περιορισμού

4.3.1 Μοντελοποίηση Περιορισμών Του Εξεταστικού Ημερολογίου

Από τους περιορισμούς που αναφέρθηκαν στην παράγραφο 4.1 θα μοντελοποιηθούν οι περιορισμοί αυτοί οι οποίοι αναφέρονται στον φοιτητή, στα μαθήματα αλλά και στο μέγεθος της αίθουσας

1. ***Περιορισμός 1ος: Κανείς φοιτητής δεν μπορεί να εξεταστεί σε περισσότερα από ένα μαθήματα την ίδια χρονική στιγμή.*** Έστω L_{ij} το σύνολο των μαθημάτων στα οποία εξετάζεται ο φοιτητής s_i την χρονική στιγμή t_j, ορίζεται ως εξής: $L1_{ij} = L * \pi_l(\sigma_{t=t_i \wedge s=s_j}(EX))$. Το τμήμα $\sigma_{t=t_i \wedge s=s_j}(EX)$ κάνει την επιλογή των πλειάδων της σχέσης *EX* που περιέχουν τον φοιτητή s_i και την χρονική στιγμή t_j. Με την πράξη της προβολής επιλέγεται μόνο το σύνολο των κωδικών των μαθημάτων. Ακολουθεί η φυσική συνένωση με το σύνολο L ώστε να επιστραφούν τα ζητούμενα μαθήματα. Ο αριθμός των φοιτητών που εξετάζονται σε περισσότερα από ένα μαθήματα την ίδια χρονική στιγμή αποθηκεύεται στην μεταβλητή *many_Lessons* και ορίζεται σαν $many\ Lessons = \sum_{s \in S} \sum_{t \in T} |L1_{st}|$

2. ***Περιορισμός 2ος: Όλα τα μαθήματα που βρίσκονται μέσα στο πρόγραμμα εξετάσεων πρέπει να εξεταστούν.*** Έστω L_2 το σύνολο των μαθημάτων τα οποία δεν βρίσκονται μέσα στο πρόγραμμα των εξετάσεων, που ορίζεται ως εξής: $L_2 = \{l_k \in L : [L * (\pi_l(\sigma_{l=l_k}(EX))) = \emptyset]\}$. Η μεταβλητή *not_Timetable* περιέχει τον αριθμό των μαθημάτων που περιέχονται στο σύνολο L_2, δηλαδή $not\ Timetable = |L_2|$.

3. ***Περιορισμός 3ος: Δύο μαθήματα του ίδιου τμήματος είναι καλό να μην εξετάζονται την ίδια χρονική στιγμή.*** Έστω L_{3k} το σύνολο των μαθημάτων που εξετάζονται την ίδια εξεταστική περίοδο k, αυτό ορίζεται ως $L_{3k} = \{L * \pi_l(\sigma_{t=t_k}(EX))\}$. Η μεταβλητή *sameTime* περιέχει τον αριθμό των μαθημάτων που περιέχονται στο σύνολο L_{3k}, δηλαδή $sameTime = \sum_{t \in T} |L3_t|$

4. ***Περιορισμός 4ος: Οι φοιτητές που πρόκειται να εξεταστούν σε ένα μάθημα σε μία πόλη πρέπει να χωράνε στις διαθέσιμες θέσεις που υπάρχουν στην πόλη.*** Το σύνολο των φοιτητών S_k που εξετάζονται στην πόλη k την εξεταστική περίοδο t ορίζεται ως

$S_k = \left\{ S * \pi_s (\sigma_{t=t_i \wedge c=c_k}(EX)) \right\}$. Ο συνολικός αριθμός των εξεταζόμενων στην πόλη k την εξεταστική περίοδο t είναι . $totalExamined_k = |S_k|$ Η χωρητικότητα της αίθουσας περιέχεται στην μεταβλητή capacity. Το σύνολο των εξεταστικών δωματίων της πόλης k ορίζεται ως: $R_k = \left\{ R * \pi_r (\sigma_{c=c_k}(RC)) \right\}$. Η συνολική χωρητικότητα μίας πόλης, ορίζεται ως $totalCapacity_k = \sum_{r \in R_k} R_k.capacity$. Για να ισχύει ο περιορισμός πρέπει: $totalExamined_k \leq totalCapacity_k$.

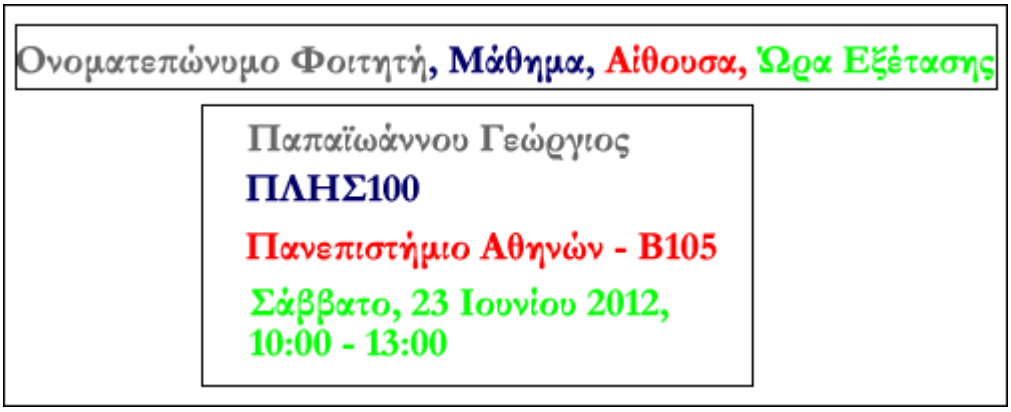
Η συνάρτηση καταλληλότητας δίνεται από την σχέση $fitness = hard_1 w_1 + hard_2 w_2 + hard_1 w_3 + hard_4 w_4$. Όσο μεγαλύτερο είναι ένα βάρος w_i του περιορισμού *i* τόσο πιο μεγάλη συμμετοχή έχει ο περιορισμός αυτός στην συνάρτηση καταλληλότητας.

4.4 Αναπαράσταση Χρωμοσώματος Γενετικού Αλγορίθμου

Ένα από τα πιο σημαντικά τμήματα για ένα γενετικό αλγόριθμο είναι η αναπαράσταση του χρωμοσώματος δηλαδή η επιλογή του φαινότυπου του αλγόριθμου. Όπως αναφέρθηκε προηγούμενα, ο φαινότυπος είναι συνήθως μία σειρά από μηδενικά (0) και άσσους (1), τα οποία αντιστοιχούν στον ζητούμενο γονότυπο.

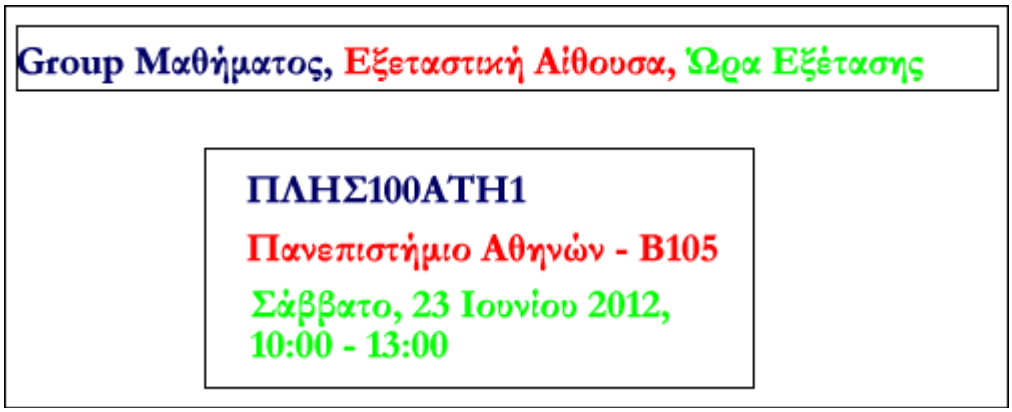
Ο τελικός στόχος του πανεπιστημίου είναι η παραγωγή τετράδων της μορφής *(Ονοματεπώνυμο Φοιτητή, Εξεταζόμενο Μάθημα, Αίθουσα Εξέτασης, Εξεταστική Ώρα)* και η αποστολή τους σε όλους τους ενδιαφερόμενους.

Ένα παράδειγμα της τετράδας αυτής φαίνεται, στην επόμενη σελίδα, στο σχήμα 4.3, όπου ο **φοιτητής** *Γεώργιος Παπαϊωάννου*, θα εξεταστεί στο **μάθημα** *ΠΛΗΣ100*, στην **αίθουσα** του *Πανεπιστημίου Αθηνών Β105*, την **εξεταστική περίοδο** *Σάββατο 23 Ιουνίου 2012, 10:00 – 13:00*.



Σχήμα 4.3: Μορφή διατεταγμένων τετράδων του Εξεταστικού Ημερολογίου

Ένας άλλος τρόπος που περιγράφει μία πιο συμπυκνωμένη μορφή των εγγραφών του εξεταστικού ημερολογίου είναι η αναπαράσταση σε διατεταγμένες τριάδες της μορφής: (*Group Μαθήματος, Εξεταστική Αίθουσα, Εξεταστική Περίοδος*). Ένα παράδειγμα της τριάδας αυτής φαίνεται στο Σχήμα 4.4, όπου η **ομάδα μαθήματος** *ΠΛΗΣ100ΑΤΗ1* θα εξεταστεί στην **αίθουσα** του *Πανεπιστημίου Αθηνών Β105*, την **εξεταστική περίοδο** *Σάββατο 23 Ιουνίου 2012, 10:00 – 13:00*.



Σχήμα 4.4: Μορφή Διατεταγμένων Τριάδων Εξεταστικού Ημερολογίου

Στην επίλυση του εξεταστικού ημερολογίου με γενετικούς αλγόριθμους έχουν χρησιμοποιηθεί διάφορες αναπαραστάσεις (φαινότυποι). Οι κυριότερες από αυτές φαίνονται παρακάτω:

1. Το κάθε χρωμόσωμα αποτελείται από γονίδια που το κάθε ένα από αυτά απεικονίζει μία χρονοθυρίδα. Η τιμή που παίρνει το κάθε ένα από τα γενόματα ισούται με έναν δυαδικό αριθμό που ορίζει τα μαθήματα που πρόκειται να εξεταστούν την συγκεκριμένη χρονική στιγμή. Έστω για παράδειγμα ότι υπάρχουν 4 μαθήματα τα Μ1, Μ2, Μ3, Μ4 και την χρονική στιγμή Τ1 θα εξεταστούν τα μαθήματα Μ1 και Μ3. Με 1 συμβολίζονται τα μαθήματα που εξετάζονται και

σχηματίζεται η συμβολοσειρά 1010. Αν μετατραπεί σε ακέραιος αριθμός το 1010 ισούται με 10 που είναι και η τιμή του γενόματος. Η αναπαράσταση αυτή έχει ένα βασικό μειονέκτημα το οποίο ονομάζεται label replacement problem, δηλαδή κατά την διάρκεια της διασταύρωσης μπορεί αρκετά διαγωνίσματα να μην εξετάζονται με αποτέλεσμα το εξεταστικό ημερολόγιο να μην είναι αποδεκτό [01].

2. Το κάθε χρωμόσωμα αποτελείται από γονίδια που το κάθε ένα από αυτά απεικονίζει ένα μάθημα. Η τιμή που παίρνει το κάθε ένα από τα γονίδια ισούται με έναν αριθμό που ορίζει την ώρα που πρόκειται να εξεταστεί [15].

3. Έστω ότι έχουμε *s* μάθήματα. Το χρωμόσωμα έχει μήκος 2s όπου οι τιμές από 0 – (s - 1) απεικονίζουν την εξεταστική περίοδο (χρονοθυρίδα) στην οποία θα εξεταστεί το μάθημα και οι τιμες απο s εώς και 2s – 1, απεικονίζουν την αίθουσα στην οποία θα εξεταστεί το συγκεκριμένο μάθημα [20].

Στην παρούσα διπλωματική διατριβή αποφασίστηκε να επιλεγεί η δεύτερη απεικόνιση. Το πρόβλημα που πρόκειται να λυθεί περιέχει εκτός από την ημερομηνία και την ώρα εξέτασης του μαθήματος και τον χώρο εξέτασης του. Στο πέμπτο κεφάλαιο της μεταπτυχιακής διατριβής παρατηρήθηκε ότι η αύξηση του χρόνου εκτέλεσης μεγαλώνει όσο μεγαλώνει το χρωμόσωμα και ότι για μεγάλα χρωμοσώματα είναι απαγορευτική η εκτέλεση σε συμβατικούς υπολογιστές.

Παράλληλα ο φαινότυπος θα πρέπει να είναι όσο πιο απλός γίνεται και ο αλγόριθμος θα πρέπει να μπορεί να επιλύει σε ικανοποιητικό χρόνο τους ταυτόχρονα τους παρακάτω έξι περιορισμούς:

1. Ένα μάθημα μπορεί να εξεταστεί μόνο μία φορά.

2. Όλα τα μαθήματα που περιέχονται στο εξεταστικό πρόγραμμα πρέπει να εξεταστούν.

3. Δύο μαθήματα που ανήκουν στην ίδια ομάδα μαθημάτων πρέπει να εξετάζονται στην ίδια εξεταστική περίοδο.

4. Σε μία εξεταστική περίοδο, σε μία πόλη, ο αριθμός των φοιτητών που εξετάζονται πρέπει να είναι μικρότερος ή ίσος με τον συνολικό αριθμό των διαθέσιμων θέσεων που υπάρχουν στην πόλη.

5. Ένα μάθημα δεν μπορεί να εξεταστεί σε μία αίθουσα με μικρότερη χωρητικότητα από τον αριθμό των φοιτητών που έχουν δηλώσει το μάθημα

6. Ένας φοιτητής είναι καλό να εξετάζεται μόνο σε ένα μάθημα σε μία συγκεκριμένη εξεταστική περίοδο.

Η προσέγγιση για την αναπαράσταση του χρωμοσώματος είναι ότι κάθε θέση του αποτελεί ένα τμήμα των φοιτητών μίας ομάδος, ενώ η τιμή κάθε θέσης του χρωμοσώματος αναπαριστά την αίθουσας εξετάσεων και την εξεταστικής περιόδου .

Το παρακάτω παράδειγμα θα διευκολύνει την κατανόηση της αναπαράστασης: Έστω η δομή των τμημάτων που περιγράφεται στην Εικόνα 4. Το χρωμόσωμα βάση της αρχικής υπόθεσης θα έπρεπε να έχει μήκος πέντε (5) και η κάθε μία από τις θέσεις του θα είναι ένα τμήμα της *PLHS100* π.χ [PLHS100ATH1, PLHS100ATH2, PLHS100LEM1, PLHS100THES1, PLHS100LEF1].

Έστω, για παράδειγμα, ότι στην Αθήνα υπάρχουν δύο αίθουσες η *B105* και η *B106* με χωρητικότητα δέκα και τριάντα πέντε άτομα αντίστοιχα. Για να είναι το παράδειγμα πιο κατανοητό, θεωρήθηκε ότι οι εξετάσεις των τμημάτων γίνονται την ίδια εξεταστική ώρα. Στην Αθήνα διδάσκονται το *PLHS100ATH1* με είκοσι άτομα και το *PLHS100ATH2* με είκοσι πέντε άτομα αντίστοιχα. Με την απεικόνιση της προηγούμενης παραγράφου είναι αδύνατον να εξεταστούν παράλληλα και τα δύο τμήματα αφού δεν υπάρχει δυνατότητα διάσπασης του τμήματος σε δύο.

Έτσι ο γενετικός αλγόριθμος θα δεχθεί σαν λύση αυτή που περιγράφεται στο Σχήμα 4.5, η οποία οδηγεί σε ένα εξεταστικό ημερολόγιο το οποίο δεν είναι αποδεκτό. Αυτό συμβαίνει γιατί ο αλγόριθμος έχει οδηγηθεί σε ένα τοπικό ελάχιστο και θα πρέπει να οδηγηθεί έξω από αυτό είτε με την βοήθεια μίας άλλης τεχνικής είτε με επανεκκίνηση του αλγόριθμου προκειμένου να βρεθεί μία άλλη λύση, εάν υπάρχει.

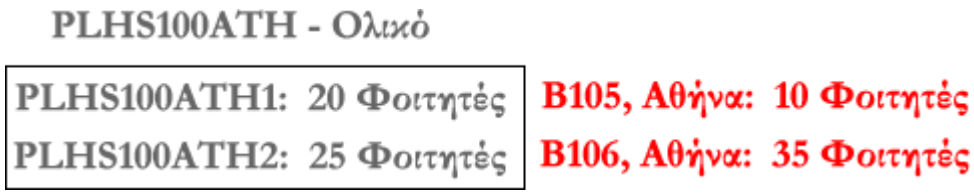
Σχήμα 4.5: Παράδειγμα Τμημάτων & Αιθουσών στην Αθήνα

Προκειμένου να ξεπεραστεί αυτό το σημείο, εφαρμόστηκε η λογική του προβλήματος του σάκου (*Knapsack Problem*), κάνοντας δύο υποθέσεις, οι οποίες δεν επηρεάζουν την γενικότητα του προβλήματος:

1. οι φοιτητές των τμημάτων μίας πόλης όταν εξετάζονται μπορούν να μπουν σε οποιαδήποτε αίθουσα εξετάζονται τα τμήματα αυτά

2. δεν υπάρχει προτεραιότητα στις θέσεις που κάθονται οι φοιτητές κατά την εξέταση, δηλαδή η λογική που ισχύει είναι First In – First Served.

Ακολουθείται η παρακάτω διαδικασία για τον διαχωρισμό σε τμήματα εξετάσεων.



Σχήμα 4.6: Πρώτο Βήμα Αλγόριθμου Συνένωσης

1. Συνενώνονται τα δύο τμήματα της πόλης σε ένα, τοποθετώντας το δεύτερο τμήμα στο τέλος του πρώτου. Το τελικό group ονομάζεται *PLS100ATH-Ολικό.*

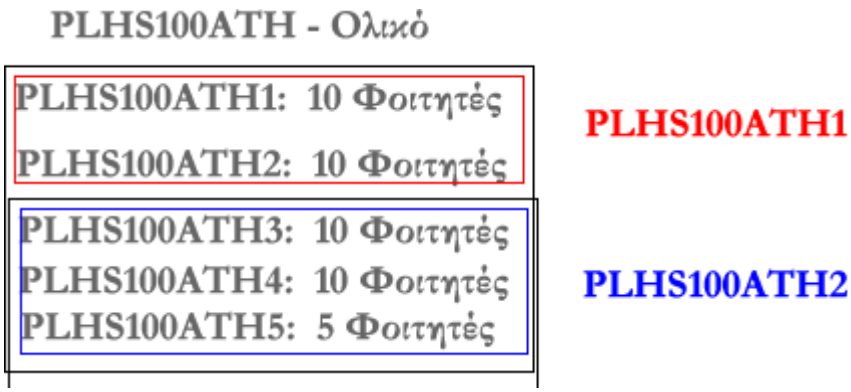
2. Το αποτελέσματα του πρώτου βήματος (Σχήμα 4.6) χωρίζεται σε ομάδες με μέγεθος ίσο με το μέγεθος της μικρότερης αίθουσας. Τα νέα τμήματα που θα δημιουργηθούν φαίνονται στον παρακάτω πίνακα:

Τμήμα	Άτομα προς Εξέταση
PLHS100ATH1	10
PLHS100ATH2	10
PLHS100ATH3	10
PLHS100ATH4	10
PLHS100ATH5	5

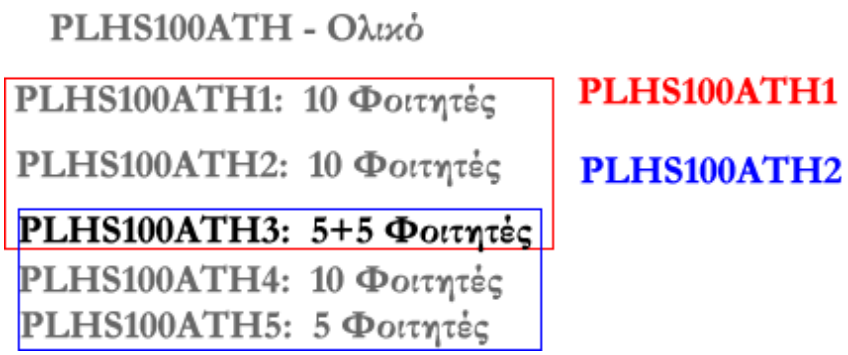
Πίνακας 4.1: Χωρητικότητα Τμημάτων Αθήνας μετά την διάσπαση

Η διαδικασία για το παράδειγμα φαίνεται στην Εικόνα 4.7. Με μαύρο χρώμα φαίνεται τα περιεχόμενα του *PLHS100ATH-Ολικό*. Αυτά διασπώνται σε δεκάδες, εκτός από το τελευταίο τμήμα που περιέχει μόνο πέντε φοιτητές. Στην εικόνα στο τετράγωνο με το κόκκινο χρώμα περιέχονται οι ομάδες που έχουν προκύψει από το τμήμα διδασκαλίας *PLHS100ATH1* και με μπλε αυτές που έχουν προκύψει από το τμήμα διδασκαλίας *PLHS100ATH2*.

Στην περίπτωση που τα τμήματα είχαν αντίστροφα τα μεγέθη τους, το τρίτο τμήμα θα περιείχε φοιτητές και από τα δύο τμήματα διδασκαλίας *PLHS100ATH1* και *PLHS100ATH2*, όπως φαίνεται στην Εικόνα 4.8. Το τμήμα εξετάσεων *PLHS100ATH3* που προκύπτει περιέχει πέντε φοιτητές από το τμήμα διδασκαλίας *PLHS100ATH1* και άλλους πέντε από το τμήμα διδασκαλίας *PLHS100ATH2*.



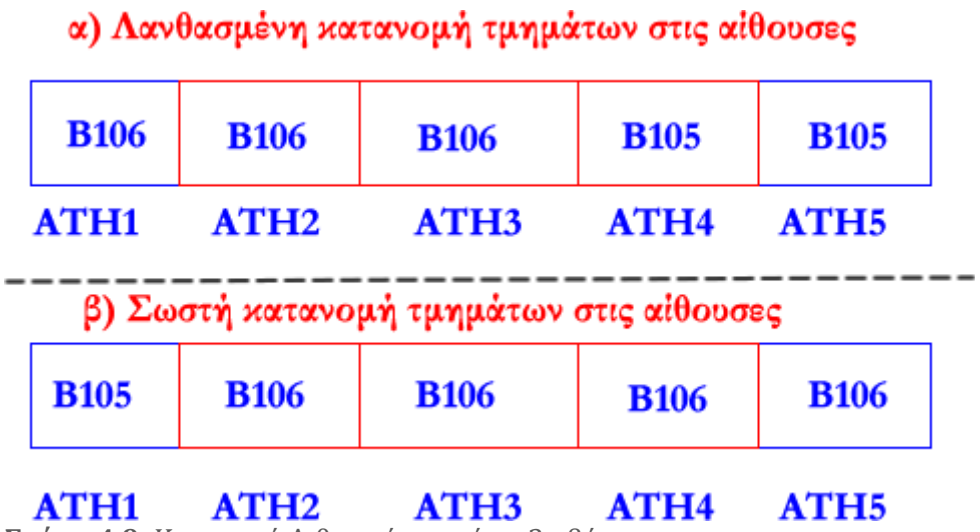
Σχήμα 4.7: Διάσπαση Τμημάτων και Δημιουργία νέων (Βήμα 2 Αλγόριθμου)



Σχήμα 4.8: Διάσπαση Τμημάτων και Δημιουργία Νέων (2ο Παράδειγμα)

Τα τμήματα αυτά τοποθετούνται στις αίθουσες και ο αλγόριθμος με την βελτίωση της συνάρτησης καταλληλότητας θα καταφέρει να τις τοποθετήσει στην σωστή θέση. Στο σχήμα 4.9, στην οποία φαίνεται και το τελευταίο τμήμα του παραδείγματος που χρησιμοποιήθηκε, στην πρώτη περίπτωση στην αίθουσα *B105* θα εξεταστούν είκοσι άτομα, το οποίο είναι αδύνατον γιατί η χωρητικότητα της είναι δέκα άτομα. Στην δεύτερη περίπτωση στην αίθουσα *B105* θα εξεταστούν δέκα άτομα (όση και η

χωρητικότητα της) ενώ στην αίθουσα *B106* θα εξεταστούν τριάντα πέντε άτομα, αριθμός που συμπίπτει επίσης με την χωρητικότητα της.



Σχήμα 4.9: Κατανομή Αιθουσών κατά το 3ο βήμα

Στην πραγματικότητα, όπως έχει αναφερθεί, για την αναπαράσταση του φαινότυπου θα πρέπει να ληφθεί υπόψιν και η εξεταστική περίοδος στην οποία εξετάζεται η ομάδα μαθημάτων.

Έστω ότι υπάρχουν μ εξεταστικές αίθουσες και κ εξεταστικές περίοδοι. Ορίζεται ο σύνθετος_αριθμός που περιλαμβάνει την ι – Εξεταστική Αίθουσα και την λ – Εξεταστική περίοδο σύμφωνα με την παρακάτω σχέση:

$$\Sigma\acute{\upsilon}\nu\theta\varepsilon\tau\sigma\varsigma A\rho\iota\theta\mu\acute{\sigma}\varsigma = \iota\,\kappa + \lambda$$

όπου:

1. *ι* είναι ο αύξων αριθμός της εξεταστικής αίθουσας η οποία παίρνει τιμές από 0 έως και μ-1

2. *λ* είναι ο αύξων αριθμός της εξεταστικής περιόδου η οποία παίρνει τιμές από 0 έως και κ -1

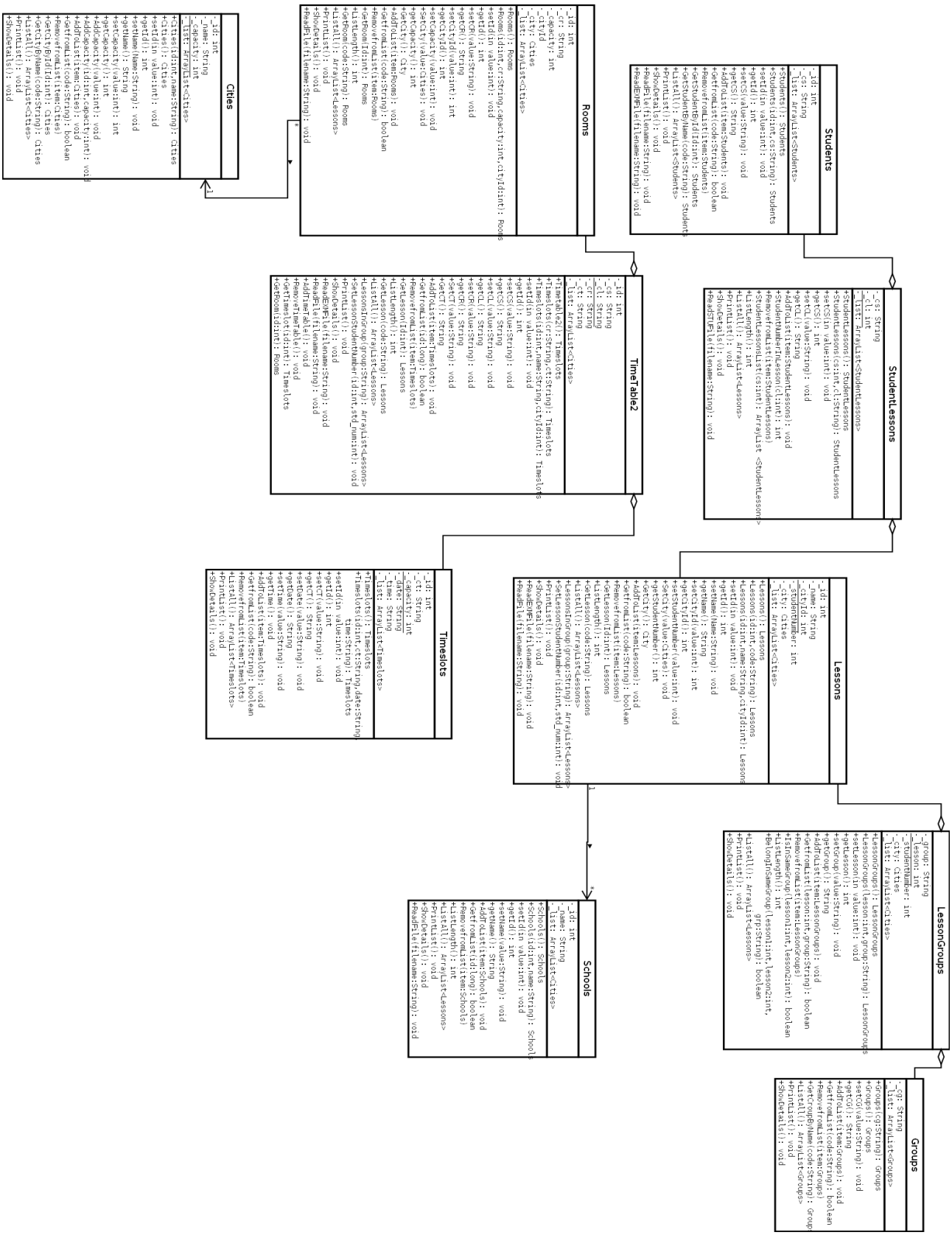
Είναι πολύ πιθανόν σε κάποια από τα μαθήματα οι τιμές του γενόματος να είναι ίσες μεταξύ τους. Εάν αυτό δεν πρέπει να ισχύει θα βαθμολογηθεί αντίστοιχα από την συνάρτηση καταλληλότητας και σε επόμενο στάδιο ο αλγόριθμος θα επιχειρήσει να βρει μία πιο καλή λύση.

Συνοψίζοντας λοιπόν για την αναπαράσταση του φαινότυπου ακολουθείται η παρακάτω διαδικασία:

1. Για κάθε πόλη όπου υπάρχουν μαθήματα προς εξέταση βρίσκεται το μέγεθος της μικρότερης αίθουσας, έστω *minlength.*
2. Κάθε μάθημα της πόλης χωρίζεται σε επιμέρους μαθήματα με μέγεθος μεγαλύτερο ή ίσο με το *minlenght*.
3. Το σύνολο των μαθημάτων όλων των πόλεων αποτελούν τον φαινότυπο του γενετικού αλγορίθμου
4. Η τιμή του κάθε ενός από τα μαθήματα αποτελείται από ένα σύνθετο αριθμό, ο οποίος αναπαριστά την αίθουσα εξέτασης και την ώρα εξέτασης του μαθήματος.

Συνοψίζοντας η προτεινόμενη αναπαράσταση πλεονεκτεί σε σχέση με άλλες αναπαραστάσεις γιατί:

1. Κάθε μάθημα εμφανίζεται μόνο μία φορά κατά την διάρκεια της εξεταστικής περιόδου, λόγω της δομής της αναπαράστασης.
2. Το χρωμόσωμα είναι αρκετά απλό, αφού περιέχει ένα αριθμό, ο οποίος συγχωνεύει την πληροφορία του χρόνου και του χώρου της εξέτασης για το συγκεκριμένο μάθημα.



Σχήμα 4.13: Διάγραμμα UML

4.5 Διάγραμμα UML Των Κλάσεων Του Προβλήματος Κατάρτισης Εξεταστικού Ημερολογίου

Στο σχήμα 4.13 της προηγούμενης σελίδας φαίνεται το διάγραμμα UML όπου περιέχονται όλες οι βοηθητικές κλάσεις και οι σχέσεις αυτών που χρησιμοποιήθηκαν για την δημιουργία του εξεταστικού ημερολογίου. Στην πραγματικότητα υπάρχουν δύο ακόμα κλάσεις οι οποίες είναι οι κεντρικές κλάσεις των αρχείων που πρόκειται να εκτελεστούν και είναι οι *ExaminationTimetable* και η *CreateLessonGroups*. Η πρώτη είναι υπεύθυνη για την προετοιμασία και εκτέλεση του γενετικού αλγόριθμου και η δεύτερη χωρίζει τα τμήματα σε μικρότερα ανάλογα με την δυνατότητα της πόλης στην οποία πρόκειται να εξεταστούν.

Οι κλάσεις που υλοποιήθηκαν έχουν τα παρακάτω χαρακτηριστικά:

1. Όλες οι ιδιότητες τους (*attributes*) είναι ιδιωτικές (*private*) μεταβλητές. Προκειμένου να διαβαστούν ή να αλλάξουν οι τιμές τους έχουν δημιουργηθεί οι κατάλληλες δημόσιες (*public*) μέθοδοι.
2. Για κάθε μία κλάση δημιουργείτε μία ιδιωτική λίστα αντικειμένων αυτής (*_list*) με τρεις ρουτίνες την *AddToList* με την οποία προστίθενται αντικείμενα στην λίστα, την *RemoveFromList* με την οποία αφαιρούνται αντικείμενα από την λίστα και την *GetFromList* η οποία πιστοποιεί αν ένα αντικείμενο υπάρχει ή όχι στην λίστα. Η *PrintList* εκτυπώνει όλη την λίστα με τα αντικείμενα στην προκαθορισμένη έξοδο, η *ListAll* επιστρέφει ένα πίνακα με τα αντικείμενα αυτά και η *ListLength* επιστρέφει τον αριθμό των αντικειμένων που βρίσκονται μέσα στην λίστα.
3. Με την *ReadFile* διαβάζονται τα δεδομένα ενός αρχείου και δημιουργείται μία λίστα με αντικείμενα, η οποία φορτώνεται στην *_list* και με την *ShowDetails* εμφανίζονται στην οθόνη οι λεπτομέρειες του τρέχοντος αντικειμένου.

Όλες οι κλάσεις που περιγράφονται παρακάτω και αποτελούν την ραχοκοκαλιά του αλγόριθμου προέρχονται από το διάγραμμα συσχετίσεων – οντοτήτων που περιγράφηκε σε προηγούμενη παράγραφο.

4.5.1 Κλάση Cities

Το αντικείμενο *Cities* αποτελεί την αναπαράσταση της οντότητας *CITY* και απεικονίζει στον πραγματικό κόσμο μία πόλη στην οποία γίνεται η εξέταση ενός μαθήματος και περιλαμβάνει και εξεταστικές αίθουσες.

Περιλαμβάνει σαν ιδιωτικές ιδιότητες τον κωδικό της πόλης (*ID*), το όνομα αυτής (*Name*) και την χωρητικότητα της σε διαθέσιμες θέσεις εξέτασης (*Capacity*).

1. Η μέθοδος *GetCityById* επιστρέφει ένα αντικείμενο *Cities* με βάση τον κωδικό της πόλης *id.*

2. Η μέθοδος *GetCityByName* επιστρέφει ένα αντικείμενο *Cities* με βάση το όνομα της πόλης *Name.*

3. Η αύξηση της χωρητικότητας του τρέχοντος αντικειμένου της πόλης γίνεται με την μέθοδο *AddCapacity.*

4.5.2 Κλάση Groups

Το αντικείμενο *Groups* αποτελεί την αναπαράσταση της οντότητας *GROUP* και απεικονίζει στον πραγματικό κόσμο μία ομάδα μαθημάτων τα οποία πρόκειται να εξεταστούν.

Η ιδιωτική ιδιότητα είναι ο κωδικός της ομάδος (*CG*). Υπάρχει η δυνατότητα εύρεσης ενός συγκεκριμένου αντικείμενου *Groups* με τον κωδικό με την αντίστοιχη μέθοδο *GetGroupByName*.

4.5.3 Κλάση LessonGroups

Το αντικείμενο *LessonGroups* αποτελεί την αναπαράσταση ενός τμήματος της οντότητας AREGROUPED και απεικονίζει στον πραγματικό κόσμο την ομαδοποίηση μαθημάτων σε ομάδες.

Η κλάση περιλαμβάνει σαν ιδιωτικές ιδιότητες τον κωδικό της ομάδας μαθημάτων (*group*) και τον χαρακτηριστικό αριθμό του μαθήματος.

1. Η μέθοδος *IsInSameGroup* εξετάζει αν δύο μαθήματα ανήκουν στην ίδια ομάδα μαθημάτων και

2. η μέθοδος *BelongInSameGroup* εξετάζει εάν δύο μαθήματα ανήκουν σε μία συγκεκριμένη ομάδα μαθημάτων με όνομα *grp*.

4.5.4 Κλάση Lessons

Το αντικείμενο *Lessons* αποτελεί την αναπαράσταση της οντότητας *LESSON* και απεικονίζει στον πραγματικό κόσμο ένα μάθημα το οποίο πρόκειται να εξεταστεί.

Περιλαμβάνει σαν ιδιωτικές ιδιότητες τον κωδικό του μαθήματος (*ID*), το όνομα του (Name), τον κωδικό της πόλης στην οποία εξετάζεται (*CityId*) και τον αριθμό των φοιτητών οι οποίοι πρόκειται να εξεταστούν στο συγκεκριμένο μάθημα (*StudentNumber*).

1. Με την μέθοδο *SetLessonStudentNumber* ορίζεται ο αριθμός των φοιτητών που θα εξεταστούν στο μάθημα με κωδικό *id*.

2. Η μέθοδος *GetLesson* επιστρέφει ένα αντικείμενο μαθήματος ανάλογα με την κλήση της. Η πρώτη έκδοση έχει όρισμα το όνομα του μαθήματος και η δεύτερη έκδοση της τον κωδικό του μαθήματος.

3. Η μέθοδος *LessonsInGroup* δέχεται σαν όρισμα επιστρέφει την λίστα μαθημάτων που ανήκουν σε μία συγκεκριμένη ομάδα συμβολαίων.

4. Η μέθοδος *ReadEXMFile* δέχεται σαν όρισμα το όνομα του αρχείου με τα μαθήματα και επιστρέφει μία λίστα με τα μαθήματα προς εξέταση.

4.5.5 Κλάση Rooms

Το αντικείμενο *Rooms* αποτελεί την αναπαράσταση της οντότητας *ROOM* και απεικονίζει στον πραγματικό κόσμο μία εξεταστική αίθουσα στην οποία θα μπορούν να εξεταστούν διάφορα μαθήματα.

Περιλαμβάνει σαν ιδιωτικές ιδιότητες τον κωδικό του εξεταστικού δωματίου (*ID*), την ονομασία του (*CR*) τον κωδικό της πόλης στην οποία βρίσκεται (*CC*) και την χωρητικότητα του σε διαθέσιμες θέσεις εξέτασης (*Capacity*).

Η μέθοδος *GetRoom* επιστρέφει ένα αντικείμενο εξεταστικού δωματίου ανάλογα με τα ορίσματα που της έχουν δοθεί. Η πρώτη μορφή δέχεται σαν όρισμα την ονομασία του εξεταστικού δωματίου και η δεύτερη έκδοση τον κωδικό του.

4.5.6 Κλάση StudentLessons

Το αντικείμενο *StudentLessons* αποτελεί την αναπαράσταση της συσχέτισης ATTEND και απεικονίζει στον πραγματικό κόσμο το γεγονός ότι ένας φοιτητής πρόκειται να εξεταστεί σε ένα μάθημα.

Περιλαμβάνει σαν ιδιωτικές ιδιότητες τον κωδικό του φοιτητή που εξετάζεται (*CS*) και τον κωδικό του μαθήματος που πρόκειται να εξεταστεί (*CL*).

1. Η μέθοδος *StudentNumberInLesson* επιστρέφει τον αριθμό των φοιτητών οι οποίοι εξετάζονται στο συγκεκριμένο μάθημα.

2. Η μέθοδος *ReadSTUFile* προσθέτει στην λίστα των φοιτητών δεδομένα για να χρησιμοποιηθούν μαζί με τα δεδομένα από το αρχείο *EXM*.

4.5.7 Κλάση Timeslots

Το αντικείμενο *Timeslots* αποτελεί την αναπαράσταση της οντότητας *TIMESLOT* και απεικονίζει στον πραγματικό κόσμο μία χρονοθυρίδα στην οποία θα εξεταστούν διαφορετικά μαθήματα σε διαφορετικές εξεταστικές αίθουσες.

Περιλαμβάνει σαν ιδιωτικές ιδιότητες τον κωδικό της χρονοθυρίδας (*ID*), την ημερομηνία της (*Date*) και την ώρα (*Time*).

4.5.8 Κλάση Timetable2

Το αντικείμενο *Timetable2* αποτελεί την αναπαράσταση της συσχέτισης EXAMINE και απεικονίζει στον πραγματικό κόσμο μια εγγραφή του εξεταστικού ημερολογίου, δηλαδή σε ποια εξεταστική αίθουσα και πότε ακριβώς θα .εξεταστεί ένα συγκεκριμένο μάθημα.

Περιλαμβάνει σαν ιδιωτικές ιδιότητες τον κωδικό της εγγραφής (*ID*), τον κωδικό του μαθήματος (*CL*), τον κωδικό εξεταστικού δωματίου (*CR*) και τον κωδικό της χρονοθυριδας στην οποία εξετάζεται το μάθημα (*CT*).

1. Η μέθοδος *GetTimeslot* επιστρέφει τον κωδικό της χρονοθυρίδας η οποία περιλαμβάνεται στην συγκεκριμένη εγγραφή με κωδικό *id*.

2. Η μέθοδος *GetRoom* επιστρέφει τον κωδικό του εξεταστικού δωματίου το οποίο περιλαμβάνεται στην συγκεκριμένη εγγραφή με κωδικό *id*.

4.5.9 Κλάση Students

Το αντικείμενο *Students* αποτελεί την αναπαράσταση της οντότητας STUDENT και απεικονίζει στον πραγματικό κόσμο τους φοιτητές που πρόκειται να εξεταστούν στην συγκεκριμένη εξεταστική περίοδο.

Περιλαμβάνει σαν ιδιωτική ιδιότητα τον κωδικό του φοιτητή (*CS*).

1. Η μέθοδος *GetStudentByName* επιστρέφει ένα αντικείμενο του φοιτητή ο οποίος έχει σαν όρισμα τον κωδικό του

2. Η μέθοδος *ReadEXMFile* φορτώνει σε μία λίστα του φοιτητές που περιέχονται σε ένα αρχείο με κωδικοποίηση *EXM* .

4.5.10 Κλάση Schools.

Το αντικείμενο *Schools* αποτελεί την αναπαράσταση της οντότητας SCHOOL και απεικονίζει στον πραγματικό κόσμο μία σχολή στην οποία θα ανήκουν οι φοιτητές και τα μαθήματα τα οποία πρόκειται να εξεταστούν.

Περιλαμβάνει σαν ιδιωτικές ιδιότητες τον κωδικό της σχολής (*ID*) και το όνομα αυτής (*Name*).

4.6 Παράμετροι Γενετικού Αλγορίθμου

Προκειμένου να εκτελεστεί ο γενετικός αλγόριθμος θα πρέπει να οριστούν διάφορες παράμετροι εκτέλεσης του. Αυτό επιτρέπεται από το ECJ Toolkit που χρησιμοποιήθηκε στην διπλωματική διατριβή, είτε στην αρχή της εκτέλεσης του προγράμματος με την μορφή ενός αρχείου παραμέτρων είτε με την κλήση των κατάλληλων μεθόδων πριν την εκτέλεση της βασικής ρουτίνας του αλγόριθμου.

Οι κύριες παράμετροι που πρέπει να οριστούν προκειμένου να εκτελεστεί σωστά ο γενετικός αλγόριθμος είναι:

1. **Το μέγεθος του πληθυσμού του γενετικού αλγορίθμου:** Η τιμή που επιλέχθηκε είναι 300 άτομα.

2. **Ο αριθμός των γενεών του γενετικού αλγόριθμου.** Η τιμή που επιλέχθηκε είναι ίση με 100000

3. **Η απεικόνιση του γενόματος**. Η τιμή που επιλέχθηκε είναι διάνυσμα ακεραίων αριθμών .

4. **Η μικρότερη τιμή που μπορεί να πάρει το γένομα.** Η τιμή που επιλέχθηκε ειναι 0.

5. **Η μεγαλύτερη τιμή που μπορεί να πάρει το γένομα.** Η τιμή εξαρτάται από τον αριθμό των αιθουσών και των χρονοθυρίδων που χρησιμοποιούνται στα δεδομένα του προβλήματος. Ο λόγος αναλύεται στην *4.4*

6. **Το μέγεθος του γενόματος.** Η τιμή εξαρτάται από τον αριθμό των τμημάτων των μαθημάτων τα οποία εξετάζονται κατά την διάρκεια της εξεταστικής περιόδου. Ο αριθμός των τμημάτων προκύπτει μετά την εφαρμογή του αλγόριθμου που περιγράφηκε στην παράγραφο *4.4.*

7. **Ο τύπος της διασταύρωσης:** Η τιμή που επιλέχθηκε είναι one, η οποία υποδηλώνει το διασταύρωση ενός σημείου[15].

8. **Την πιθανότητα μετάλλαξης.** Η τιμή που επιλέχθηκε είναι 0.03

9. **Η μέθοδος επιλογής.** Η τιμή που επιλέχθηκε ήταν η ανταγωνιστική επιλογή[16] με μέγεθος 2.

10. Για την πιο γρήγορη προσέγγιση της βέλτιστης λύσης επιλέγεται να κρατάτε στην επόμενη γενιά του γενετικού αλγόριθμου η καλύτερη λύση της προηγούμενης γενιάς. Αυτή η διαδικασία ονομάζεται **ελιτισμός (elitism)**.

11. **Επιλογή της κύριας κλάσης του προβλήματος που πρόκειται να εκτελεστεί**. Η κλάση είναι η Exams.examsGA

15 Παράγραφος 4.6. Αναλύεται στο παράδειγμα της τοποθέτησης των οκτώ βασιλισσών.
16 Παράγραφος 3.6

4.7 Αρχικοποίηση Αλγόριθμου

Η αρχικοποίηση του γενετικού αλγορίθμου γίνεται με την απόδοση τυχαίων αριθμών στο χρωμόσωμα. Οι αριθμοί αυτοί επιλέγονται από τον συνδυασμό αιθουσών και χρονοθυρίδων που αποτελούν τον σύνθετο αριθμό που περιγράφηκε στην παράγραφο 4.4.

Έστω ότι υπάρχουν δέκα αίθουσες στις οποίες θα εξεταστούν τα μαθήματα και υπάρχουν τρεις εξεταστικές χρονοθυρίδες. Το σύνολο των πιθανών τιμών του γενόματος θα είναι από 0 έως και 29.

Οι τιμές αυτές αλλάζουν σε κάθε εκτέλεση του αλγόριθμου και εξαρτώνται αποκλειστικά από τον αριθμό των αιθουσών και των χρονοθυρίδων που χρησιμοποιούνται κατά την εξέταση του μαθήματος. Στην συγκεκριμένη διπλωματική διατριβή, γίνεται η υπόθεση ότι σε κάθε χρονοθυρίδα είναι δυνατόν να χρησιμοποιηθούν όλες οι αίθουσες των πόλεων για να εξεταστεί ένα μάθημα.

4.8 Συνάρτηση Καταλληλότητας

Η συνάρτηση καταλληλότητας μίας υποψήφιας λύσης αποτελεί για έναν γενετικό αλγόριθμο το κριτήριο το οποίο δείχνει πόσο κοντά είναι η συγκεκριμένη λύση στην ικανοποίηση συγκεκριμένων στόχων που έχουν τεθεί για τον γενετικό αλγόριθμο.

Στο πρόβλημα του εξεταστικού ημερολογίου, η συνάρτηση καταλληλότητας όπως αναφέρθηκε σε προηγούμενη παράγραφο θα πρέπει να συμπεριλαμβάνει όλους τους περιορισμούς που εξετάζονται για το συγκεκριμένο πρόβλημα.

Το ECJ Toolkit που χρησιμοποιήθηκε για την εκτέλεση του γενετικού αλγορίθμου στην παρούσα διπλωματική διατριβή, προσπαθεί να ελαχιστοποιήσει την τιμή της συνάρτησης καταλληλότητας. Αυτό σημαίνει ότι η τιμή της συνάρτησης καταλληλότητας είναι αντιστρόφως ανάλογη της απόστασης από την λύση ενός προβλήματος.

Έστω $f(x)$ η τιμή του αθροίσματος των τιμών των περιορισμών της x προτεινόμενης λύσης του προβλήματος πολλαπλασιασμένων με τα κατάλληλα βάρη. Η συνάρτηση καταλληλότητας $fit(x)$, θα δίνεται από τον τύπο $fit(x)=\frac{1}{f(x)}$. Όταν η λύση είναι βέλτιστη η τιμή του αθροίσματος θα είναι ίση με το μηδέν $f(x)=0$. Αυτό σημαίνει ότι όλοι οι περιορισμοί ικανοποιούνται για αυτήν την

λύση. Όμως $f(x)=0 \rightarrow \frac{1}{0} \rightarrow fit(x)=\infty$. Για να αποφευχθεί αυτό μετατρέπεται η συνάρτηση καταλληλότητας σε $fit(x)=\frac{1}{1+f(x)}$. Έτσι εάν $f(x)=0$ τότε $fit(x)=\frac{1}{1+0}=\frac{1}{1}=1$.

Στην παράγραφο 4.1 αναφέρθηκαν οι περιορισμοί οι οποίοι πρέπει να ικανοποιούνται ώστε ένα ημερολόγιο να είναι αποδεκτό. Αυτοί είναι:

1. Δύο μαθήματα που ανήκουν στην ίδια ομάδα μαθημάτων πρέπει να εξετάζονται στην ίδια εξεταστική περίοδο. Ορίζεται η μεταβλητή *lesGrpSameGroup* η οποία αυξάνεται κατά ένα σε κάθε περίπτωση που υπάρχει μάθημα το οποίο δεν εξετάζεται μαζί με τα υπόλοιπα μαθήματα της ομάδος του.

2. Ένα μάθημα πρέπει να εξετάζεται στην ίδια πόλη την οποία διδάσκεται. Ορίζεται η μεταβλητή *wrongLessonRoom* η οποία αυξάνεται κατά ένα σε κάθε περίπτωση που υπάρχει μάθημα το οποίο δεν εξετάζεται στην πόλη που διδάσκεται.

3. Σε μία εξεταστική περίοδο, σε μία πόλη, ο αριθμός των φοιτητών που εξετάζονται πρέπει να είναι μικρότερος ή ίσος με τον συνολικό αριθμό των διαθέσιμων θέσεων που υπάρχουν στην πόλη. Ορίζεται η μεταβλητή *wrongRoomCapacity* η οποία αυξάνεται κατά ένα σε κάθε περίπτωση που ο συνολικός αριθμός των φοιτητών που εξετάζεται στις αίθουσες μία συγκεκριμένης πόλης σε μία συγκεκριμένη χρονοθυρίδα είναι μεγαλύτερος από τον συνολικό αριθμό των διαθέσιμων θέσεων που υπάρχουν στην πόλη.

4. Ένας φοιτητής είναι καλό να εξετάζεται μόνο σε ένα μάθημα σε μία συγκεκριμένη εξεταστική χρονοθυρίδα. Ορίζεται η μεταβλητή *stdSameHour* η οποία αυξάνεται κατά ένα σε κάθε περίπτωση που εξετάζεται ο φοιτητής σε περισσότερα από ένα μαθήματα την ίδια εξεταστική περίοδο.

5. Όλα τα μαθήματα πρέπει να εξεταστούν αλλά μόνο μία φορά το κάθε ένα από αυτά. Ο περιορισμός αυτός καλύπτεται από την αναπαράσταση του φαινότυπου του γενετικού αλγόριθμου.

Για κάθε έναν από τους περιορισμούς ορίζεται ένα βάρος w_i όπου $i \in (1,2,3,4)$. Όλα τα βάρη λαμβάνουν τιμές από το σύνολο R^*_+ των πραγματικών θετικών αριθμών.

Πρόταση: Το βάρος ενός περιορισμού *j* που είναι πιο ανελαστικός από έναν περιορισμό *k* , θα είναι μικρότερο από το βάρος του περιορισμού *k*.

Απόδειξη:Έστω ένα εξεταστικό ημερολόγιο με N περιορισμούς και *j, k* περιορισμοί με $0<j<k<N-1$.Ο *j* περιορισμός θα είναι πιο πολύ ανελαστικός σε σχέση με τον *k* περιορισμό. Έστω w_j και w_k τα βάρη των περιορισμών . Είναι δυνατόν να εξεταστούν δύο περιπτώσεις:

1. Ο περιορισμός j δεν ικανοποιείται μ φορές. Τότε η συνάρτηση καταλληλότητας θα είναι ίση με

$$fit(J)=\frac{1}{1+f(J)}=\frac{1}{1+w_j\mu} \quad .$$

2. Ο περιορισμός k δεν ικανοποιείται μ φορές. Τότε η συνάρτηση καταλληλότητας θα είναι ίση με

$$fit(K)=\frac{1}{1+f(K)}=\frac{1}{1+w_k\mu} \quad .$$

Σε κάθε περίπτωση για το μ το οποίο δείχνει τις φορές που παραβιάζεται ο περιορισμός πρέπει να ισχύει ότι $\mu\in\mathbb{N}$, δηλαδή ότι είναι ένας φυσικός αριθμός μεγαλύτερος από το μηδέν. Εάν είναι μηδέν τότε δεν υφίσταται περιορισμός.

Αφού ο περιορισμός *j* είναι πιο ανελαστικός από τον *k* πρέπει η ελάχιστη μεταβολή της συνάρτησης καταλληλότητας του *j* περιορισμού να είναι πιο μεγάλη από την ελάχιστη μεταβολή του *k* περιορισμού. Η ελάχιστη αυτή μεταβολή είναι ίση με 1 δηλαδή από *μ* γίνεται *μ+1.*

Έτσι λοιπόν ισχύει ότι:

$$\Delta fit(J)>\Delta fit(K)\Rightarrow \frac{1}{1+w_j(\mu+1)}-\frac{1}{1+w_j\mu}>\frac{1}{1+w_k(\mu+1)}-\frac{1}{1+w_k\mu}\Rightarrow \quad .$$

$$\frac{1+w_j\mu-(1+w_j(\mu+1))}{[1+w_j(\mu+1)][1+w_j\mu]}>\frac{1+w_k\mu-(1+w_k(\mu+1))}{[1+w_k(\mu+1)][1+w_k\mu]}\Rightarrow$$

$$\frac{1+w_j\mu-1-w_j\mu-w_j}{(1+w_j\mu+w_j)(1+w_j\mu)}>\frac{1+w_k\mu-1-w_k\mu-w_k}{(1+w_k\mu+w_k)(1+w_k\mu)}\Rightarrow$$

$$\frac{-w_j}{1+w_j\mu+w_j\mu+w_j^2\mu^2+w_j+w_j^2\mu}>\frac{-w_k}{1+w_k\mu+w_k\mu+w_k^2\mu^2+w_j+w_k^2\mu}\Rightarrow$$

$$\frac{-w_j}{1+w_j(2\mu+1)+w_j^2(\mu^2+\mu)}>\frac{-w_k}{1+w_k(2\mu+1)+w_k^2(\mu^2+\mu)}\Rightarrow$$

$$\frac{w_j}{1+w_j(2\mu+1)+w_j^2(\mu^2+\mu)}<\frac{w_k}{1+w_k(2\mu+1)+w_k^2(\mu^2+\mu)}[1]$$

Το τριώνυμο $1+w_j(2\mu+1)+w_j^2(\mu^2+\mu)$ έχει λύσεις τις $\frac{1}{\mu+1}$ και $\frac{-1}{\mu}$. Η ελάχιστη τιμή που μπορεί να πάρει το μ εάν υπάρχει περιορισμός είναι 1. Σε αυτή την περίπτωση οι ρίζες του τριωνύμου θα είναι -1 και 0.5. Όπως αναφέρθηκε προηγούμενα τα βάρη πρέπει να είναι αριθμοί θετικοί. Εάν επιλεχθούν τιμές του περιορισμού w_k που είναι μεγαλύτερες από το 0,5 το τριώνυμο θα έχει θετικές τιμές. Το ίδιο ακριβώς ισχύει και για το τριώνυμο $1+w_k(2\mu+1)+w_k^2(\mu^2+\mu)$.

Στο συγκεκριμένο εξεταστικό ημερολόγιο ενδιαφέρει οι φοιτητές να εξετάζονται στην πόλη την οποία παρακολουθούν το μάθημα και όλα τα μαθήματα που ανήκουν στην ίδια ομάδα να εξετάζονται την ίδια χρονική στιγμή. Τα βάρη που επιλέχθηκαν για τους περιορισμούς φαίνονται στον παρακάτω πίνακα 4.2:

Μεταβλητή Περιορισμού	***Τιμή Συντελεστή***
lesGrpSameGroup	*50*
wrongLessonRoom	*1*
wrongRoomCapacity	*1000*
stdSameHour	*5000*

Πίνακας 4.2: Συντελεστές Βαρύτητας Περιορισμών

Κεφάλαιο 5
Αποτελέσματα

Ο γενετικός αλγόριθμος που περιγράφηκε στην προηγούμενη ενότητα δοκιμάστηκε με διαφορετικά εξεταστικά προβλήματα. Το κύριο εξεταστικό πρόβλημα που χρησιμοποιήθηκε για τους σκοπούς της διπλωματικής αποτελείται από: 360 φοιτητές, 5 μαθήματα που αφορούν ένα συγκεκριμένο μεταπτυχιακό πρόγραμμα σπουδών, τα οποία εξετάζονται σε τρεις διαφορετικές πόλεις. Όλα τα δεδομένα έχουν αντληθεί ανωνυμοποιημένα από το Ελληνικό Ανοικτό Πανεπιστήμιο και αφορούν την περίοδο 2011 – 2012.

Το υπολογιστικό σύστημα που χρησιμοποιήθηκε αποτελείται από έναν Η/Υ με επεξεργαστή Intel Core 2 Duo (2.4 GHz) και μνήμη 8GB. Το λειτουργικό σύστημα που ήταν εγκαταστημένο στον Η/Υ ήταν Windows Vista 64 Bit.

Ο σκοπός της εκτέλεσης του αλγόριθμου όπως αναφέρθηκε στο προηγούμενο κεφάλαιο είναι να παραχθεί ένα “αποδεκτό” ημερολόγιο το οποίο θα ικανοποιεί όλους τους ανελαστικούς περιορισμούς που έχουν τεθεί από τον εκάστοτε υπεύθυνο.

Για την επίτευξη αυτού του στόχου και με βάση όσα έχουν αναφερθεί στο προηγούμενο κεφάλαιο, ορίζονται σε πρώτη φάση τα τμήματα των μαθημάτων ανά πόλη, ο αριθμός των οποίων είναι 18 για το συγκεκριμένο παράδειγμα. Για την εκτέλεση του αλγόριθμου θα χρησιμοποιηθούν 56 τμήματα τα

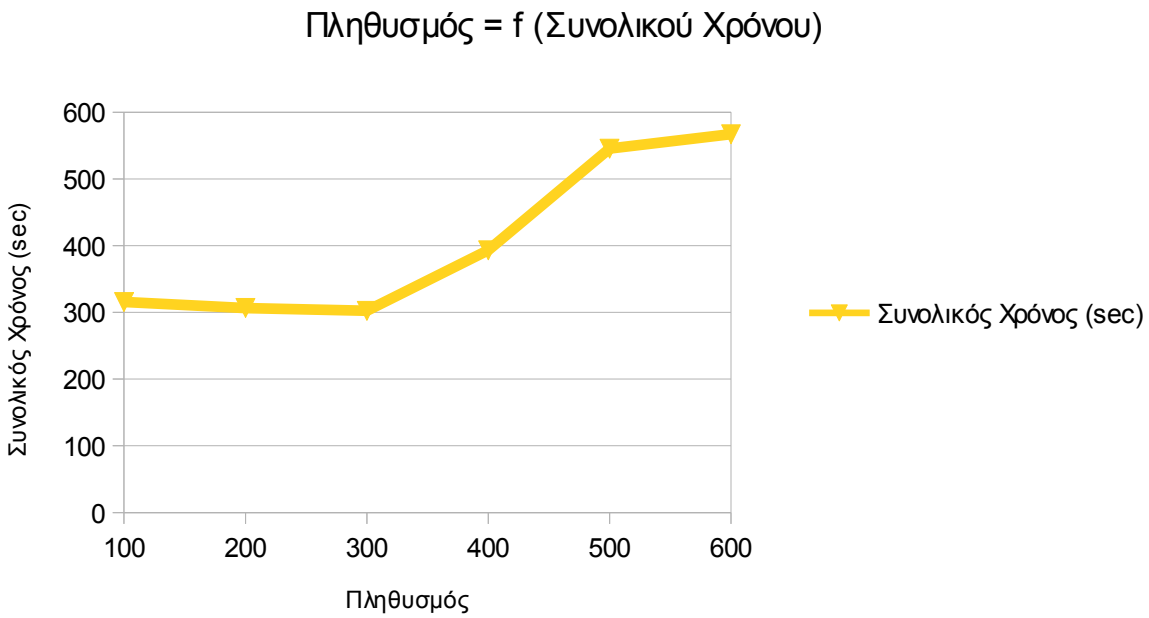
οποία προκύπτουν από την διαίρεση των ατόμων κάθε τμήματος με την ελάχιστη χωρητικότητα του δωματίου στην οποία ανήκουν.

5.1 Προσδιορισμός Καλύτερου Πληθυσμού

Οι δοκιμές έγιναν για το συγκεκριμένο εξεταστικό πρόβλημα το οποίο θα έπρεπε να υλοποιηθεί σε 24 χρονοθυρίδες με δυνατότητα χρήσης όλων των εξεταστικών δωματίων. Οι πληθυσμοί που χρησιμοποιήθηκαν είναι 100, 200, 300, 400, 500 και 600 άτομα αντίστοιχα. Στον παρακάτω πίνακα (5.1) φαίνεται ο μέσος όρος των αποτελεσμάτων των δοκιμών του αλγόριθμου σε 100 εκτελέσεις του. Σημαντικό είναι να αναφέρουμε ότι η διασπορά του συνόλου των αποτελεσμάτων από το μέσο όρο ήταν σε ένα διάστημα +/- 10%:

Πληθυσμός	***Μέγιστη Γενιά***	***Χρόνος Εκτέλεσης μίας Γενιάς (sec)***	***Συνολικός Χρόνος (sec)***
100	*944*	*0,33*	*315,52*
200	*613*	*0,5*	*306,5*
300	*458*	*0,65*	*302,25*
400	*397*	*0,99*	*393,03*
500	*376*	*1,45*	*545,2*
600	*315*	*1,6*	*567,0*

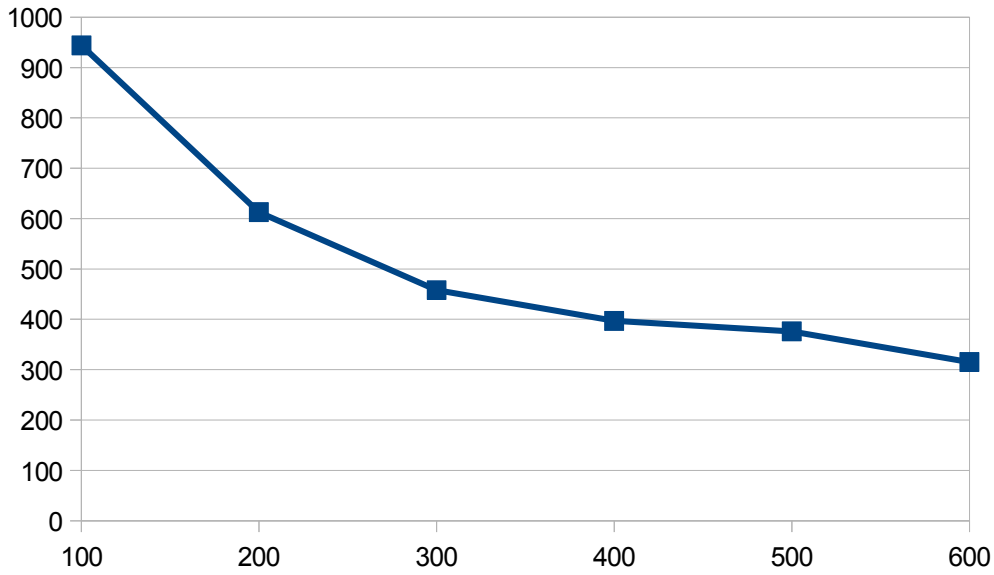
Πίνακας 5.1: Αποτελέσματα εκτέλεσης γενετικού αλγόριθμου με διάφορους πληθυσμούς. Τα αποτελέσματα δείχνουν το χρόνο εκτέλεσης μίας γενιάς, τον συνολικό χρόνο εκτέλεσης του αλγόριθμου και την γενιά στην οποία σταμάτησε.



Σχήμα 5.1 : *Εύρεση Καλύτερου Αριθμού Ατόμων*

Στο σχήμα 5.1, παρατηρείται ότι ο καλύτερος χρόνος εκτέλεσης του γενετικού αλγόριθμου για την παραγωγή ενός αποδεκτού εξεταστικού ημερολογίου επιτυγχάνεται με τον πληθυσμό των τριακοσίων (300) ατόμων. Το αποτέλεσμα αυτό αποτελεί ένα εμπειρικό προσδιορισμό του αριθμού των ατόμων του πληθυσμού βάση του οποίου θα γίνουν τα πειράματα και οι συγκρίσεις.

Στο σχήμα 5.2, φαίνεται η γραφική παράσταση της συνάρτησης των αριθμού των γενεών του γενετικού αλγόριθμου που χρειάζονται για να φτάσει ο αλγόριθμος σε ένα αποδεκτό ημερολόγιο σε σχέση με τον πληθυσμό των ατόμων που χρησιμοποιούνται στον γενετικό αυτόν αλγόριθμο.



Σχήμα 5.2: *Γενιά = f (Πληθυσμός)*

5.2 Δοκιμές εκτέλεσης Γενετικού Αλγόριθμου με διάφορες παραμέτρους για το κύριο πρόβλημα

Για τις δοκιμές που έγιναν επιλέχθηκε για τον πληθυσμό η τιμή των 300 ατόμων πάνω από την οποία όπως φαίνεται έχουμε μείωση των γενεών που χρειάζεται για να επιστρέψει ο αλγόριθμος ένα αποδεκτό ημερολόγιο αλλά δεν είναι πάρα πολύ μεγάλη. Από την άλλη ο χρόνος εκτέλεσης με την αύξηση του πληθυσμού αυξάνεται πάρα πολύ με αποτέλεσμα ο πληθυσμός των 300 ατόμων να είναι η βέλτιστη επιλογή για τον συνολικό χρόνο εκτέλεσης του αλγόριθμου.

Οι δοκιμές που έχουν γίνει αφορούν συνδυασμούς πέντε διαφορετικών αριθμών χρονοθυρίδων και δύο συνολικών αριθμών εξεταστικών δωματίων. Τα αποτελέσματα φαίνονται στον πίνακα 5.2.

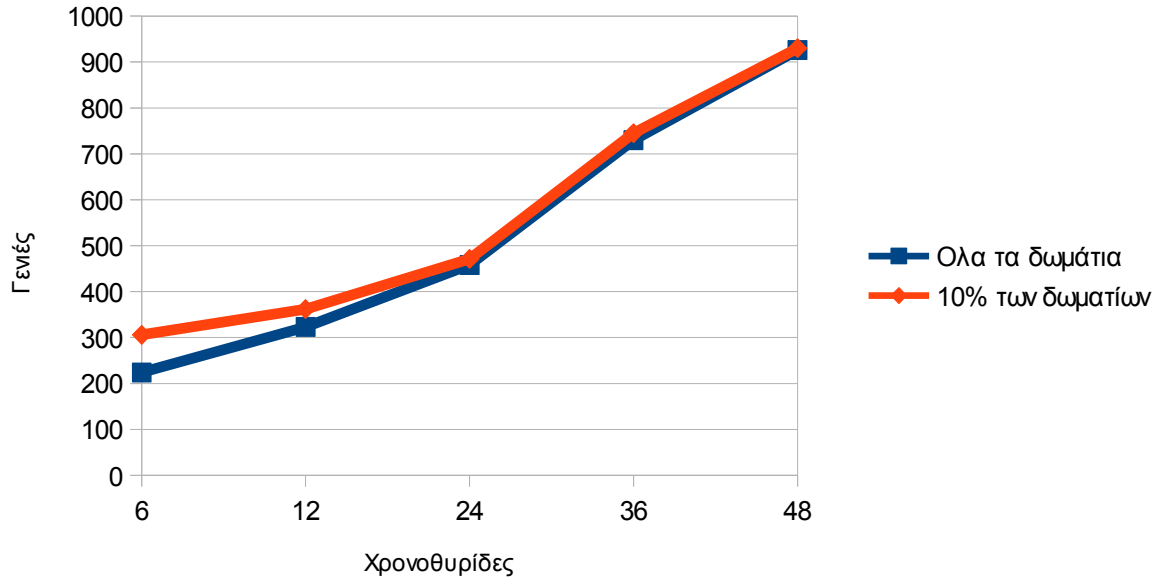
	Όλα τα δωμάτια			*10% των δωματίων*		
Χρονοθυρίδα	***Μέγιστη γενιά***	***Χρόνος Γενιάς (sec)***	***Συνολικός Χρόνος (sec)***	***Μέγιστη γενιά***	***Χρόνος Γενιάς (sec)***	***Συνολικός Χρόνος (sec)***
6	*224*	*0,71*	*159,04*	*306*	*0,64*	*195,84*
12	*323*	*0,71*	*229,33*	*362*	*0,64*	*231,68*
24	*458*	*0,71*	*325,18*	*471*	*0,64*	*301,44*
36	*730*	*0,71*	*518,30*	*745*	*0,64*	*476,80*
48	*926*	*0,71*	*657,46*	*930*	*0,64*	*595,20*

Πίνακας 5.2: Αποτελέσματα εκτέλεσης αλγορίθμου με διαφορετικούς αριθμούς χρονοθυρίδων

Η γραφική αναπαράσταση των αποτελεσμάτων φαίνεται στο σχήμα 5.3 όπου απεικονίζονται οι γενεές που χρειάζεται ο γενετικός αλγόριθμος για να παράγει ένα αποδεκτό ημερολόγιο σε συνάρτηση με τον αριθμό των χρονοθυρίδων που είναι διαθέσιμες.

Παρατηρείτε ότι όσο αυξάνονται οι χρονοθυρίδες του εξεταστικού ημερολογίου τόσο πιο πολύ συγκλίνουν οι γενιές που χρειάζονται για να παραχθεί ένα αποδεκτό εξεταστικό ημερολόγιο. Αυτό συμβαίνει γιατί με την αύξηση τους, υπάρχουν περισσότερες επιλογές για την εύρεση μιας χρονοθυρίδας όπου θα μπορέσουν να τοποθετηθούν τα μαθήματα, ανεξάρτητα από τον αριθμό των αιθουσών που είναι διαθέσιμες.

Ένα ακόμα συμπέρασμα είναι ότι ο χρόνος εκτέλεσης μίας γενιάς του αλγόριθμου είναι ανεξάρτητος από τον αριθμό των χρονοθυρίδων που έχουν οριστεί στα δεδομένα του προβλήματος. Αυτό συμβαίνει γιατί οι πράξεις που γίνονται κατά την εκτέλεση του γενετικού αλγόριθμου εξαρτώνται από τον αριθμό των φοιτητών, τον αριθμό των ομάδων των μαθημάτων.



Σχήμα 5.3: *Γενιές = f(Αρ. Χρονοθυρίδων)*

5.3 Παράδειγμα εκτέλεσης του γενετικού αλγόριθμου στο βασικό πρόβλημα

Στην παράγραφο αυτή θα παρουσιαστεί η ολοκληρωμένη λύση που προτείνεται από τον γενετικό αλγόριθμο για την λύση του βασικού προβλήματος.

Το βασικό πρόβλημα περιλαμβάνει 360 φοιτητές, 5 μαθήματα τα οποία εξετάζονται σε 3 πόλεις. Οι πόλεις αυτές έχουν διαθέσιμα 24 εξεταστικά δωμάτια και οι διαθέσιμες χρονοθυρίδες είναι 6. Στον πίνακα 5.4 φαίνονται οι αίθουσες και οι χωρητικότητες τους.

Οι χρονοθυρίδες φαίνονται στον πίνακα 5.3

	Δευτέρα	**Τρίτη**	**Τετάρτη**	**Πέμπτη**	**Παρασκευή**
10:00 – 13:00	0	1	2	3	4
17:00 – 20:00			5		

Πίνακας 5.3 : Χρονοθυρίδες Παραδείγματος

Κωδικός	Πόλη	Χωρητικότητα
108	Αθήνα	20
109	Αθήνα	20
110	Αθήνα	17
111	Αθήνα	16
112	Αθήνα	19
113	Αθήνα	19
114	Αθήνα	17
115	Αθήνα	19
116	Αθήνα	20
303	Αθήνα	20
304	Αθήνα	20
ΑΜΦ1	Αθήνα	110
ΑΜΦ2	Αθήνα	60
601	Θεσσαλονίκη	10
602	Θεσσαλονίκη	20
603	Θεσσαλονίκη	28
604	Θεσσαλονίκη	12
615	Θεσσαλονίκη	25
610	Θεσσαλονίκη	45
611	Θεσσαλονίκη	45
ΑΜΦ	Θεσσαλονίκη	100
11	Πάτρα	15
13	Πάτρα	100
21	Πάτρα	15

Πίνακας 5.4 : Δεδομένα Αιθουσών

Το τελικό πρόγραμμα που προέκυψε από τον αλγόριθμο φαίνεται στον πίνακα 5.4. Παρατηρείται, όπως ήταν αναμενόμενο, ότι τα τμήματα του ίδιου μαθήματος εξετάζονται στην ίδια χρονοθυρίδα αλλά και ότι δεν έχει γίνει κανένα λάθος στην πόλη, δηλαδή το τμήμα να διδάσκεται σε διαφορετική πόλη από αυτήν που εξετάζεται.

Η συνάρτηση καταλληλότητας είναι ίση με 1 και αυτό σημαίνει ότι όλοι οι περιορισμοί έχουν ικανοποιηθεί δηλαδή κανένας από τους φοιτητές που εξετάζονται στο μάθημα δεν εξετάζονται παράλληλα και σε άλλο. Μία άλλη παρατήρηση που μπορεί να γίνει είναι ότι το ημερολόγιο, όπως είναι

αναμενόμενο λόγω της λειτουργίας των γενετικών αλγορίθμων ,παρόλο που είναι αποδεκτό δεν είναι βέλτιστο. Έτσι για παράδειγμα στο μάθημα ΑΑΑ51 στην Αθήνα ο αλγόριθμος προτιμά να χρησιμοποιήσει δύο αίθουσες την 115 και την 304 για να εξεταστούν δύο τμήματα φοιτητών, παρά να χρησιμοποιήσει μόνο μία στην οποία χωρούν όλοι οι φοιτητές και των δύο τμημάτων

ΜΑΘΗΜΑ	ΤΜΗΜΑ	ΦΟΙΤΗΤΕΣ	ΑΙΘΟΥΣΑ	Αρ. Φοιτητών / Χωρ. Αίθουσας
AAA50	**BBB50ATHINA10**	**9**	**113 (ΑΘΗ)**	**9/19**
Χρονοθυρίδα : 1	BBB50THESSALONIKI10	10	602 (ΘΕΣ)	10/20
	BBB50PAT1	15	11 (ΠΑΤ)	15/15
	BBB50ATHINA11	9	ΑΜΦ2 (ΑΘΗ)	9/60
	BBB50ATHINA12	9	115 (ΑΘΗ)	9/19
	BBB50ATHINA20	9	108 (ΑΘΗ)	9/20
	BBB50THESSALONIKI11	10	602 (ΘΕΣ)	20/20
	BBB50ATHINA21	9	116 (ΑΘΗ)	9/20
	BBB50ATHINA13	8	116 (ΑΘΗ)	17/20
	BBB50THESSALONIKI12	5	611 (ΘΕΣ)	5/45
	BBB50THESSALONIKI22	10	603 (ΘΕΣ)	10/28
	BBB50ATHINA22	9	112 (ΑΘΗ)	9/19
	BBB50ATHINA31	6	108 (ΑΘΗ)	15/20
	BBB50ATHINA23	6	ΑΜΦ1 (ΑΘΗ)	6/110
	BBB50THESSALONIKI20	10	610 (ΘΕΣ)	10/45
	BBB50ATHINA32	9	ΑΜΦ2 (ΑΘΗ)	18/60
	BBB50ATHINA33	9	113 (ΑΘΗ)	18/19
	BBB50THESSALONIKI21	4	615 (ΘΕΣ)	4/25
	BBB50ATHINA34	5	111 (ΑΤΗ)	5/16
	BBB50PAT10	2	21 (ΠΑΤ)	2/15
	BBB50ATHINA3	1	110 (ΑΤΗ)	1/17
	BBB50THESSALONIKI2	1	610 (ΘΕΣ)	11/45
AAA51	**BBB51ATHINA10**	**9**	**ΑΜΦ1 (ΑΘΗ)**	**9/110**
Χρονοθυρίδα : 4	BBB51THESSALONIKI10	10	611 (ΘΕΣ)	10/45
	BBB51ATHINA11	9	116 (ΑΘΗ)	9/20
	BBB51ATHINA20	9	ΑΜΦ2 (ΑΘΗ)	9/60
	BBB51ATHINA30	5	113 (ΑΘΗ)	9/19
	BBB51THESSALONIKI11	10	610 (ΘΕΣ)	10/45
	BBB51ATHINA3	20	108 (ΑΘΗ)	20/20
	BBB51ATHINA21	9	ΑΜΦ2 (ΑΘΗ)	18/60
	BBB51ATHINA12	9	115 (ΑΘΗ)	9/19
	BBB51THESSALONIKI12	8	611 (ΘΕΣ)	18/45
	BBB51ATHINA22	9	304 (ΑΤΗ)	9/20
AAA60	**BBB60THESSALONIKI12**	**2**	**602 (ΘΕΣ)**	**2/20**

Χρονοθυρίδα : 0	BBB60ATHINA10	9	110 (ΑΤΗ)	9/17
	BBB60ATHINA20	9	112 (ΑΘΗ)	9/19
	BBB60THESSALONIKI13	10	615 (ΘΕΣ)	10/25
	BBB60ATHINA11	6	112 (ΑΘΗ)	15/19
	BBB60ATHINA21	4	109 (ΑΤΗ)	4/20
	BBB60THESSALONIKI14	4	603 (ΘΕΣ)	4/28
	BBB60ATHINA22	2	116 (ΑΘΗ)	2/20
	BBB60ATHINA3	1	112 (ΑΘΗ)	10/19
	BBB60ATHINA30	9	109 (ΑΤΗ)	13/19
	BBB60ATHINA31	9	112 (ΑΘΗ)	19/19
AAA61	**BBB61THESSALONIKI1**	**11**	**615 (ΘΕΣ)**	**11/25**
Χρονοθυρίδα : 2	BBB61ATHINA11	3	112 (ΑΘΗ)	3/19
	BBB61ATHINA22	7	115 (ΑΘΗ)	7/19
	BBB61ATHINA12	9	ΑΜΦ1 (ΑΘΗ)	9/110
	BBB61ATHINA23	9	111 (ΑΤΗ)	9/16
	BBB61THESSALONIKI10	10	603 (ΘΕΣ)	10/28
	BBB61ATHINA13	9	116 (ΑΘΗ)	9/20
	BBB61THESSALONIKI11	10	602 (ΘΕΣ)	10/20
	BBB61ATHINA24	3	116 (ΑΤΗ)	12/20
	BBB61ATHINA2	1	116 (ΑΤΗ)	13/20
AAA62	**BBB62THESSALONIKI1**	**10**	**611 (ΘΕΣ)**	**10/45**
Χρονοθυρίδα : 5	BBB62THESSALONIKI10	2	611 (ΘΕΣ)	12/45

Πίνακας 5.5 : Εξεταστικό Ημερολόγιο παραδείγματος

5.4 Αποτελέσματα εκτέλεσης για τα εναλλακτικά πρόβλημα

Το πρώτο εναλλακτικό πρόβλημα περιέχει τα παρακάτω δεδομένα: 2916 φοιτητές, 181 τμήματα μαθημάτων και 277 εξεταστικά δωμάτια.

Στον πίνακα 5.6 φαίνεται ο χρόνος εκτέλεσης μίας γενιάς για διάφορους πληθυσμούς του γενετικού αλγόριθμου. Παρατηρούμε ότι με τον δεκαπλασιασμό των φοιτητών και των μαθημάτων σε σχέση με το εξεταστικό ημερολόγιο της προηγούμενης παραγράφου, ο χρόνος εκτέλεσης του αλγόριθμου πολλαπλασιάζεται με ένα παράγοντα 400.

Διάφορες δοκιμές που έγιναν έφταναν, για την συνάρτηση καταλληλότητας, σε ένα αποτέλεσμα 2,8 x 10^{-4}, στη γενιά 4500 του αλγορίθμου μετά από χρόνο εκτέλεσης 5 περίπου ημερών, χρησιμοποιώντας όλα τα εξεταστικά δωμάτια. Στο ίδιο πρόβλημα με την χρήση του 10% των εξεταστικών δωματίων, στην ίδια γενιά, η συνάρτηση καταλληλότητας είναι ίση με 2,9 x 10^{-4} . Το αποτέλεσμα στην δεύτερη περίπτωση είναι ελαφρά καλύτερο.

Πληθυσμός	***Χρόνος Γενιάς (sec)***
50	*25*
100	*55*
150	*78*
200	*162*
300	*260*

Πίνακας 5.6 : Χρόνος εκτέλεσης μίας γενιάς του γενετικού αλγόριθμου σε σχέση με τον πληθυσμό

Το τελευταίο δείγμα είχε 23953 φοιτητές και 1416 τμήματα μαθημάτων. Για το συγκεκριμένο δείγμα ο χρόνος εκτέλεσης μίας γενιάς δεν μπορεί να προσδιοριστεί με τον συγκεκριμένο Η/Υ που χρησιμοποιήθηκε μιας και κατά την διάρκεια εκτέλεσης του αλγορίθμου για 48 ώρες, δεν έχει παραχθεί αποτέλεσμα, στον υπολογιστικό σύστημα που χρησιμοποιήθηκε.

Παρατηρείται ότι όσο πιο πολύ μεγαλώνει το γονιδίωμα (αριθμός μαθημάτων) ή ο αριθμός των μαθημάτων τόσο πιο πολύ μεγαλώνει ο χρόνος εκτέλεσης . Προκειμένου να μειωθεί ο χρόνος εκτέλεσης του αλγόριθμου θα πρέπει να αυξηθεί η υπολογιστική ισχύ του επεξεργαστή ή να χρησιμοποιηθεί παράλληλος προγραμματισμός για τον γενετικό αλγόριθμο. Προκειμένου να γίνει ο παραλληλισμός θα πρέπει το τμήμα, στο οποίο υπολογίζεται η συνάρτηση καταλληλότητας, να διασπαστεί σε τμήματα τα οποία θα μπορούν να εκτελούνται ανεξάρτητα μεταξύ τους.

Επίλογος

Η μεταπτυχιακή αυτή διατριβή ασχολήθηκε με την δημιουργία αποδεκτών εξεταστικών ημερολογίων με την χρήση γενετικών αλγορίθμων. Δόθηκε βάρος κυρίως στους περιορισμούς οι οποίοι διέπουν τον εξεταζόμενο φοιτητή και το γεγονός ότι η εξέταση ενός μαθήματος θα πρέπει να μπορεί να γίνει σε συγκεκριμένες αίθουσες και σε συγκεκριμένα χρονικά διαστήματα. Δημιουργήθηκε μία συνάρτηση καταλληλότητας, η οποία ανταποκρίνεται στην ικανοποίηση των περιορισμών που είχαν τεθεί από τους υπεύθυνους.

Κατά την διάρκεια της διατριβής αυτής δύο ήταν τα κύρια προβλήματα τα οποία αντιμετωπίστηκαν:

1. Η αναπαράσταση του γονιδιώματος για την οποία αναλώθηκε ο περισσότερος χρόνος προκειμένου να αποφασιστεί ποια θα ήταν η καλύτερη, ώστε να καλύπτονται και πιθανές επεκτάσεις του γενετικού αλγόριθμου.

2. Ο χρόνος εκτέλεσης του γενετικού αλγόριθμου, ο οποίος αυξάνεται εκθετικά με την αύξηση των μαθημάτων και των φοιτητών.

Μελλοντικά θέματα που θα πρέπει να εξεταστούν είναι:

- Πιθανές επεκτάσεις του προγράμματος, λαμβάνοντας υπ' όψιν τους ανελαστικούς περιορισμούς που αφορούν τον διδάσκοντα και τους επιτηρητές των εξετάσεων

- Ελαστικοί περιορισμοί που αφορούν το μέγεθος της εξεταστικής περιόδου

- Εκτέλεση του γενετικού αλγόριθμου που παρουσιάστηκε στην διπλωματική αυτή διατριβή για το σύνολο των μαθημάτων ενός πανεπιστημίου και σύγκριση αυτών με το ημερολόγιο μιας συγκεκριμένης εξεταστικής περιόδου που έχει παραχθεί με τον συμβατικό για το πανεπιστήμιο τρόπο. Για την παραγωγή των αποτελεσμάτων θα πρέπει να χρησιμοποιηθεί περισσότερη υπολογιστική ισχύς.[17]

Όσον αφορά τα δύο πρώτα θέματα, αυτά μπορούν να υλοποιηθούν αρκετά εύκολα με μικρές προσθήκες στην συνάρτηση καταλληλότητας, για το τρίτο θέμα λόγω του γεγονότος ότι τα εργαλεία που χρησιμοποιήθηκαν μπορούν να εκτελεστούν σε όλα τα λειτουργικά συστήματα, θα χρειαστούν λίγες επεμβάσεις για να μπορέσουν να δώσουν ένα εξεταστικό ημερολόγιο με όλα τα μαθήματα.

17 Έχει γίνει η απαραίτητη αίτηση στο Hellas Grid

Βιβλιογραφία

[01] D. Abramson & J. Abela (1991): "A parallel genetic algorithms for solving the school timetabling problem". *Technical report, Division of Information Technology, C.S.I.R.O*.

[02] Garey M.R. And Johnson D.S. (1979): "Computers and Intractability: a Guide to the Theory of NP - Completeness". *Freeman*

[03] Bartsch T, Drexl A,Kroeger S. "Scheduling the professional soccer leagues of Austria and Germany." (2006): *Computers & Operations Research* 33:1907–37.

[04] Brezulianu A., Fira M. and Fira L. (2010): "A genetic algorithm approach for a constrained employee scheduling problem as applied to employees at mall type shops", *International Journal of Advanced Science and Technology* Vol. 14

[05] Briskorn D. (2008): "Sport leagues scheduling: models, combinatorial properties, and optimization algorithms". In: Lecture notes in eco- nomics and mathematical systems, vol. 603. Berlin: *Springer*.

[06] Burke, E.K., Bykov, Y., &, Petrovic, S. (2001): "A multicriteria approach to examination timetabling". In: E.K. Burke and P.D. Causmaecker (eds.) (2003). Practice *and Theory of Automated Timetabling: Selected Papers from the 3rd International Conference. Springer Lecture Notes in Computer Science* **2079** 118 – 131

[07] Burke, E.K., Elliman, D.G., Ford, P.H., &, Weare, R.F. (1996): "Examination timetabling in British Universities: A survey". In : Burke, E.K., & Ross, P. (eds). (1996). Practice and Theory of Automated Timetabling: Selected Papers from the 1st International Conference. *Springer Lecture Notes in Computer Science* **1153**, 76 – 90

[08] Burke, E.K., Kingston, J.H., &, de Wera, D. (2004): "Applications to timetabling", In: J. Gross and J. Yellen (eds.) (2004): *The handbook of Graph Theory, Chapman Hall/CRC Press,* **2004**, 445 – 474

[09] Burke, E.K., Newall, J.P., &, Weare, R.F. (1994): "A Memetic Algorithm for University Exam Timetabling", Springer – Verlag p.p 241-250

[10] Burke, E.K., Newall, J.P., &, Weare, R.F. (1998): "A simple heuristically guided search for the timetable problem." In: *Proceedings of the International ICSCS Symposium on Engineering of Intelligence Systems (EIS98)*, 574 – 579

[11] Carter, M.W., Laporte, G. (1996): "Recent Developments in Practical Examination Timetabling"

[12] Carter, M.W., Laporte, G., &, Lee, S.Y. (1996): "Examination Timetabling: Algorithmic strategies and applications". *Journal of Operational Research Society,* **47** (3): 373 – 383

[13] J. Chen, T. Yeung (1993): "*Hybrid Expert System Approach to Nurse Scheduling". Computers in Nursing*, 11(4): 183-90.

[14] Colorni, A., Dorigo, M., Maniezzo, V. (1992): "Distributed optimization by ant colonies". In: Proceedings of the 1st European Conference on Artificial Life, pp. 134-142. *Elsevier*, Amsterdam

[15] Corne D., Fang H.-L. Mellish C. (1993): "Solving the module exams scheduling problem with genetic algorithms". In Paul W.H., Chung, Gilian Lovegrove and Moonis Ali, editors, *Proceedings of the Sixth International Conference in Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*, pages 370 – 373. Gordon and Breach Science Publishers

[16] Darwin C. (1859): "The Origin of Species by Means of Natural Selection or The Preservation of Favoured Races in the Struggle for Life", *John Murray, London*

[17] David, P. (1998): A constraint-based approach for examination timetabling using local repair techniques. In E.K. Burke and M.W. Carter (eds.) (1998). Practice *and Theory of Automated Timetabling: Selected Papers from the 2nd International Conference. Springer Lecture Notes in Computer Science* ***1408*** 169 - 186

[18] Davis, Martin; Logemann, George, and Loveland, Donald (1962): "A Machine Program for Theorem Proving". *Communications of the ACM* **5** (7): 394–397.

[19] DellaCroce F, Oliveri D. "Scheduling the Italian football league: an ILP-based approach." (2006) *Computers & Operations Research* 33:1963–74.

[20] Fang Hsiao – Lan, (1991): "Genetic Algorithm in Timetabling and Scheduling". *Ph.D. University of Edimburg*

[21] Di Gaspero, L. and Schaerf, A. (2000): "Tabu Search Techniques for Examination Timetabling". *Lecture Notes In Computer Science, selected papers from the Third International Conference on Practice and Theory of Automated Timetabling* (2000) 104 - 117.

[22] Dueck, Gunter (1993): "New Optimization Heuristics The Great Deluge Algorithm and the Record-to-Record Travel", *Journal of Computational Physics*, Volume 104, Issue 1, p. 86-92.

[23] Eley M. (2007): "Ant algorithms for the exam timetabling problem", Proceedings of the 6th international conference on Practice and theory of automated timetabling VI, *Springer-Verlag Berlin*, Heidelberg

[24] Erben, W. (1995): "Timetabling Using Genetic Algorithms", in: Pearson, D.W.; Steele,N.C.; Albrecht, R.F. (eds.): Artificial Neural Nets and Genetic Algorithms. *Proceedings of the International Conference in Alès/France. Springer*, Wien: 30-32

[25] Frausto - Solis, J., Alonso – Pecina, F. : "A Hybrid Simulated Annealing – Tabu Search Algorithm for Post Enrollment Course Timetabling"

[26] Holland J. H. (1970): Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence. *The University of Michingan*

[27] Lawer, EL (1990): "A dynamic programming algorithm from preemptive scheduling of a single machine to minimize the number of late jobs", *Springer*, Volume 26, Number 1, 125-133, DOI: 10.1007/BF02248588

[28] Loevinsohn H. (1992): "A New Perspective on Scheduling: Freedom and Cost Control". *Nursing Management*, 23(7): 56-61.

[29] Merlot, L.T.G., Boland, N., Hughes, B.D., &, Stuckey, P.J. (2003): "A hybrid algorithm for the examination timetabling problem". In: E.K. Burke and P.D. Causmaecker (eds.) (2003). Practice *and Theory of Automated Timetabling: Selected Papers from the 3rd International Conference. Springer Lecture Notes in Computer Science* **2740** 207 – 231

[30] Moscato, P. (1989): "On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts: Towards Memetic Algorithms". *Caltech Concurrent Computation Program* (report 826).

[31] Okada M. (1992): "An Approach to the Generalized Nurse Scheduling Problem-Generation of a Declarative Program to Represent Institution specific knowledge. *" Computers and Biomedical Research*, 1992, 25: 417-34.

[32] Parejo J. A., Ruiz-Cortés A., Lozano S. and Fernandez P. (2012): "Metaheuristic optimization frameworks: a survey and benchmarking", *Soft Computing* - A Fusion of Foundations, Methodologies and Applications Volume 16, Number 3 , 527-561, DOI: 10.1007/s00500-011-0754-8

[33] Petrovic, S., &, Bykov, Y. (2003): "A multiobjective optimization technique for exam timetabling based on trajectories" In: E.K. Burke and P.D. Causmaecker (eds.) (2003). Practice *and Theory of Automated Timetabling: Selected Papers from the 4th International Conference. Springer Lecture Notes in Computer Science* **2740** 179 – 192

[34] Qu, R., &, Burke, E.K. (2006): "Hybridisations within a Graph Based Hyper heuristic Framework for University Timetabling Problems", *Technical Report NOTTCS-TR-2006-1, School of CsiT, University of Nottingham.*

[35] Russel S., Norvig P. (2003): "Artificial Intelligence – A modern approach", *Pentice Hall,* International

[36] Ritchie, G. and Levine, J. (2004): "A Hybrid Ant Algorithm for scheduling independent jobs in heterogeneous computing environments" In: *Proceedings of the 23rd Workshop of the UK Planning and Scheduling Special Interest Group*.

[37] Schaerf A. (1993): "Scheduling sport tournaments using constraint logic programming". *Constraints* 4:43–65.

[38] Selman Bart, Kautz Henry, and Cohen Bram (1996): "Local Search Strategies for Satisfiability Testing." . Final version appears in Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge, October 11-13, 1993. David S. Johnson and Michael A. Trick, ed. DIMACS *Series in Discrete Mathematics and Theoretical Computer Science,* vol. 26, AMS, 1996.

[39] Terrashima – Marin, R., Ross, P., &, Valenzuela – Rendon M. (1999): "Clique – based crossover for solving the timetabling problem with GAs", In: *Schoenauer M. et all (eds.) Proceedings of CEC99 Conference, Washington*, IEEE Press. 1200 – 1206

[40] Van Hentenryck P, Vergados Y. (2006): "Traveling tournament scheduling: a systematic evaluation of simulated annealing". In: In- tegration of AI and OR techniques in constraint programming for combinatorial optimization problems. Lecture notes in computer science, vol. 3990. Berlin: *Springer*. p. 228–43.

[41] Varadharajan T. K., Chandrasekharan Rajendran (2005): "A multi-objective simulated-annealing algorithm for scheduling in flowshops to minimize the makespan and total flowtime of jobs" *European Journal of Operational Research*, Volume 167, Issue3, Pages 772–795

[42] Weise Thomas (2009): "Global Optimization Algorithms – Theory and Application" 2nd Edition, *http://www.it-weise.de/*

[43] Γεωργοπουλος, Λυκοθανάσης (1999): "Εισαγωγή στους γενετικούς αλγορίθμους" *ΕΑΠ.*

Παράρτημα Α
Τμήματα Κώδικα

Α.1 Αρχείο Παραμέτρων ECJ

```
# Copyright 2006 by Sean Luke and George Mason University

# Licensed under the Academic Free License version 3.0
# See the file "LICENSE" for more information

parent.0 = ../ec/simple/simple.params

# The following is a quick-and-dirty example of how to use the vector
# package to create a simple GA.  The GA is as follows:
#
# 1-point crossover over a vector of integers
# ints range from 0 to 100 inclusive.  Fitness = sum(vector)
# 2-tournament selection
# Individuals are selected, crossed over, and then mutated
# mutation probability = 0.1
# mutation is randomization of the integer
```

```
pop.subpop.0.size = 300
generations = 100000
pop.subpop.0.duplicate-retries = 0
# specify species information
pop.subpop.0.species = ec.vector.IntegerVectorSpecies
pop.subpop.0.species.fitness = ec.simple.SimpleFitness
pop.subpop.0.species.ind = ec.vector.IntegerVectorIndividual
pop.subpop.0.species.min-gene = 0
pop.subpop.0.species.max-gene = 3323
pop.subpop.0.species.genome-size = 9194
pop.subpop.0.species.crossover-type = one
pop.subpop.0.species.mutation-prob = 0.03
pop.subpop.0.species.pipe = ec.vector.breed.VectorMutationPipeline
pop.subpop.0.species.pipe.source.0 = ec.vector.breed.VectorCrossoverPipeline

select.tournament.size = 2
pop.subpop.0.species.pipe.source.0.source.0 = ec.select.TournamentSelection
pop.subpop.0.species.pipe.source.0.source.1 =  same
breed.elite.0=1
eval.problem = Exams.examsGA
```

A.2 Κώδικας Γενετικού αλγόριθμου

Ο κώδικας του γενετικού αλγόριθμου επισυνάπτεται στο συμπιεσμένο αρχείο Code.zip που συνοδεύει την μεταπτυχιακή διατριβή.

A.3 Ρουτίνα Evaluation

Βοηθητικές Μεταβλητές

```
int timeslotsNumber = Timeslots.ListCount();
int arrTimeSlots[] = new int[timeslotsNumber];
long roomCapacity[] = new long[TimeTable2.ListCount()];
```

Για κάθε έναν από τους φοιτητές βρές τα μαθήματα που γράφει την ίδια χρονική στιγμή

```
for (Students std: Students.ListAll()){
    for (StudentLessons item: StudentLessons.StudentLessonsList(std.getId())){
            int timeslot = ind2.genome[item.getCL()] % timeslotsNumber;        //      Low Byte
            arrTimeSlots[timeslot]++;
    }

    for (int i =0; i<timeslotsNumber;i++){
            while (arrTimeSlots[i]>1) {
                    stdSameHour++;
                    arrTimeSlots[i]--;
            }
            arrTimeSlots[i]=0;
    }
```

Για όλα τα μαθήματα έλεγξε :

1. αν το δωμάτιο που επιλέχθηκε βρίσκεται στην ίδια πόλη με την πόλη που διδάσκεται το μάθημα

2. εάν το μάθημα ανήκει στην ίδια ομάδα με άλλα μαθήματα πρέπει να εξετάζεται στην ίδια χρονοθυρίδα με αυτά

```
for (int x=0;x<(ind2.genome.length-1);x++){
    Lessons lesson_x =Lessons.GetLesson(x);
    Rooms room_x = Rooms.GetRoom(ind2.genome[x]/Timeslots.ListCount()); //High Byte
    // Check that the Examination room is in the same city with the lesson that will be examined in
    if (lesson_x.getCityId()!=room_x.getCityID()) wrongLessonRoom ++;
    // Check if two lessons are examined in the same time
    int timeslot_x_id = ind2.genome[x] % Timeslots.ListCount();    //      Low Byte

    for (int y=(x+1);y<ind2.genome.length;y++){
            int timeslot_y_id = ind2.genome[y] % Timeslots.ListCount();    //      Low Byte
            if (LessonGroups.IsInSameGroup(x, y))
```

```
                    if (timeslot_y_id!=timeslot_x_id) lesGrpSameHour++;
            }
    }
```

Έλεγξε εάν η χωρητικότητα της αίθουσας είναι μικρότερη από τον αριθμό των ατόμων που εξετάζονται σε ένα μάθημα

```
    for (int x=0;x<(ind2.genome.length-1);x++){
            Lessons lesson_x =Lessons.GetLesson(x);
            Rooms room_x = Rooms.GetRoom(ind2.genome[x]/timeslotsNumber); //High Byte
            roomCapacity [ind2.genome[x]]+=lesson_x.getStudentNumber();

            if (lesson_x.getCityId()<=0){
                    Cities lesson_city = lesson_x.getCity();
                    if (lesson_city!=null) lesson_x.setCityId(lesson_city.getID());
            }

            if ((lesson_x.getCityId()!=room_x.getCityID())||
            (lesson_x.getStudentNumber()>room_x.getCapacity()))
                    wrongLessonRoom ++;
            }
```

Έλεγξε εάν η χωρητικότητα της αίθουσας είναι μικρότερη από τον αριθμό των ατόμων που εξετάζονται σε μία χρονοθυρίδα

```
    for (int rm_count =0; rm_count<TimeTable2.ListCount();rm_count++){
            Rooms room = Rooms.GetRoom( TimeTable2.GetRoom(rm_count));
            if (room.getCapacity()<roomCapacity[rm_count])
                    wrongRoomCapacity++;
            }
```

Υπολογισμός Συνάρτησης Καταλληλότητας

```
    float realfit = (float) (1.0 + lesGrpSameHour*50 + wrongRoomCapacity*1000 +
wrongLessonRoom*1 + stdSameHour*5000);
```

```
float myfitness = (float) (1.0/realfit);

// assume we're using SimpleFitness

((SimpleFitness)ind2.fitness).setFitness(state,

  /// ...the fitness...

  (float) (myfitness) , myfitness==1.0);
```