ΔΙΑΤΜΗΜΑΤΙΚΟ ΠΡΟΓΡΑΜΜΑ
ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ ΣΤΗ
ΔΙΟΙΚΗΣΗ ΕΠΙΧΕΙΡΗΣΕΩΝ

MBA 2016 - 2018

Με επιβλέποντα:
Καπάρης Κωνσταντίνος,
Επίκουρος Καθηγητής

Διπλωματική Εργασία

ROSTERING OPTIMIZATION FOR AN ONCOLOGY CLINIC

του

Μυστακίδη Αριστείδη του Ηλία

Υποβλήθηκε ως απαιτούμενο για την απόκτηση του διπλώματος μεταπτυχιακών σπουδών στη Διοίκηση Επιχειρήσεων

Σεπτέμβριος 2019

Θέλω να ευχαριστήσω την οικογένεια μου και τους πολύ κοντινούς μου ανθρώπους για τη βοήθεια τους κατά τη περίοδο της εκπόνησης της εργασίας. Επίσης, θέλω να δώσω ευχαριστίες στον επιβλέπων της εργασίας Επίκουρο Καθηγητή Καπάρη Κωνσταντίνο και στη νοσοκόμα της Κλινική Παθολογικής Ογκολογίας του Γενικού Νοσοκομείου «Παπαγεωργίου» Αγάπη Συμεωνίδου.

Table of Contents

Περίληψη

Η παρούσα διπλωματική εργασία είναι μία παρουσίαση, ανάλυση και επίλυση του προβλήματος οργάνωσης ανθρώπινου δυναμικού που αντιμετωπίζει η ογκολογική κλινική του νοσοκομείου Παπαγεωργίου Θεσσαλονίκης. Το πρόβλημα που προκύπτει είναι ένα πρόβλημα Ακέραιου Προγραμματισμού.

Στο παρόν πρόγραμμα χρονοδιαγράμματος βαρδιών των νοσοκόμων παρατηρείται ανισορροπία στον αριθμό αλλά και στο είδος βάρδιας (πρωί, μεσημέρι, βράδυ) που καλύπτει το ανθρώπινο δυναμικό της κλινικής.

Η διπλωματική εργασία στοχεύει στην επίλυση του προβλήματος μέσω της επίτευξης ισορροπίας στον αριθμό βαρδιών που καλύπτουν οι νοσοκόμες του τμήματος, καθώς και του συγκεκριμένου τμήματος της ημέρας που καλύπτει η καθεμία.

Αφού γίνεται μια μικρή βιβλιογραφική ανασκόπηση, ακολουθεί η περιγραφή και η μοντελοποίηση του προβλήματος. Στη συνέχεια για την επίλυση χρησιμοποιείται Java μαζί με το πρόγραμμα της IBM, Cplex, μέσω μιας βιβλιοθήκης Java, την Iloplex.

Ο λόγος που επιλέχθηκε η συγκεκριμένη εργασία είναι για να γίνει εμφανές στον αναγνώστη πόσο εύκολα σε ένα αληθινό case study σε Ελληνικό Νοσοκομείο μπορεί να εμφανιστούν προβλήματα επιχειρησιακής έρευνας και πως μπορούν αυτά να μοντελοποιηθούν και να επιλυθούν.

Το προαναφερθέν πρόβλημα επιλύθηκε δύο φορές. Η πρώτη επίλυση έγινε χρησιμοποιώντας όλους του περιορισμούς από τη περιγραφή που προήλθε από το προσωπικό της κλινικής ενώ η δεύτερη αλλάζοντας κάποιους περιορισμούς, αναλύοντας τις αλλαγές στη λύση του.

Abstract

The current thesis is a presentation, analysis and solution of the rostering optimization problem faced by the oncology clinic of Papageorgiou Hospital of Thessaloniki. The problem that arises is an Integer Programming problem.

In this scheduling problem, there is an imbalance in the number and type of shifts (morning, noon, evening) that is covered by the human resources of the clinic.

The thesis aims at solving the problem by finding a balance in the number of shifts covered by the oncology's department nurses and the specific part of the day that they cover.

After a brief literature review, the problem is presented and modeled. What is more, Java programming is used for finding the optimized solution, along with IBM's Cplex program, through a Java library, Iloplex.

The main reason that this thesis was created, is to clearly present to the reader how a real-life case study at a Greek Hospital can hide operational research problems and how these problems can be modeled and solved.

The above-mentioned problem has been solved twice. The first solution was made using all the constraints from the description provided by the clinic staff while the second by relaxing some constraints, analyzing the changes in the solution.

Εισαγωγή

«Ο σκοπός της Επιχειρησιακής Έρευνας είναι να εξετάσει ποσοτικά αν ένας οργανισμός λαμβάνει τα μέγιστα από τον εξοπλισμό που διαθέτει σχετικά με τον στόχο του, ποιοι είναι οι κυρίαρχοι παράγοντες για να επιτευχθεί αυτό στον ελάχιστο χρόνο, με το ελάχιστο κόστος, και σε ποιο βαθμό μεταβολές σε επιμέρους στόχους θα συνεισφέρουν σε μία οικονομικότερη αλλά εξίσου έγκαιρη επίτευξη του ολικού στρατηγικού στόχου». (Sir Robert Watson-Watt 1892-1973).

Αρχικά, στο Κεφάλαιο 1, εκτός των απαραίτητων ιστορικών στοιχείων γίνεται μία εισαγωγή στην επιστήμη της Επιχειρησιακής Έρευνας, παρουσιάζοντας βασικές έννοιες της.
Έπειτα, το Κεφάλαιο 2 επικεντρώνεται στο αντικείμενο που θα μελετηθεί, αυτό δηλαδή του Ακέραιου Προγραμματισμού (Integer Programming) και Γραμμικού Προγραμματισμού (Linear Programming).
Στο Κεφάλαιο 3, παρουσιάζεται το πρόβλημα που έχει κληθεί η συγκεκριμένη εργασία να επιλύσει με μια λεπτομερή περιγραφή, ενώ στο κεφάλαιο 4 αναπτύσσεται το μαθηματικό μοντέλο που χρησιμοποιείται για την επίλυση του προβλήματος.
Στο 5ο κεφάλαιο δίνεται η επίλυση του προβλήματος και η βέλτιστη λύση, ενώ στο 6ο κεφάλαιο γίνεται ανάλυση ευαισθησίας χαλαρώνοντας κάποιο περιορισμό.
Τέλος στο 7ο κεφάλαιο γίνονται προτάσεις για περαιτέρω έρευνα και χρήση του κώδικα.

1 Βιβλιογραφική Ανασκόπηση Επιχειρησιακής Έρευνας

1.1 Ορισμοί

Η Επιχειρησιακή Έρευνα (Operational ή Operations Research, Management Science, Quantitative Analysis) αφορά τις μεθοδολογίες ανάπτυξης μαθηματικών μοντέλων που συνδράμουν στη λήψη βέλτιστων αποφάσεων σε πολύπλοκα συστήματα στα οποία αλληλοεπιδρούν πρώτες ύλες, ανθρώπινο δυναμικό, κεφάλαια, μεθοδολογίες και γενικότερα οντότητες που έχουν κάποιο στόχο. Αποτελεί το κυρίαρχο γνωστικό αντικείμενο της συστημικής ανάλυσης στη λήψη αποφάσεων (Γεωργίου & Οικονόμου, 2011).

Η Επιχειρησιακή Έρευνα επιδιώκει τον προσδιορισμό της βέλτιστης πορείας δράσης ενός προβλήματος απόφασης, υπό τον περιορισμό συγκεκριμένων πόρων. Τα μαθηματικά μοντέλα αποτελούν ακρογωνιαίο λίθο για την επιστήμη της επιχειρησιακής έρευνας, αλλά υπάρχουν και προβλήματα που δεν μπορούν να εκφραστούν σε μαθηματικούς όρους (Taha, 2007).

Εφαρμόζεται σε προβλήματα που αφορούν την διαχείριση των εργασιών και ασχολιών (operations) εντός ενός οργανισμού/επιχείρησης. Η φύση αυτών των οργανισμών ποικίλλει και επεκτείνεται σε κάθε πιθανό τομέα της εποχής μας: στην βιομηχανία, στις μεταφορές, στις τηλεπικοινωνίες, στην υγειονομική περίθαλψη, σε θέματα οικονομικού σχεδιασμού και διαχείρισης, στον δημόσιο τομέα κ.λπ.

Μέσω της Επιχειρησιακής Έρευνας δεν έχουμε απλά μια θεωρητική μελέτη ενός προβλήματος, αλλά μια ουσιώδη και καθ' όλα πρακτική επίλυσή του. Επομένως, για να θεωρείται επιτυχημένη η συμμετοχή της, θα πρέπει εν τέλει να δίνει στους φορείς λήψης αποφάσεων σαφή και κατανοητά αποτελέσματα και συμπεράσματα.

Ένα ακόμη χαρακτηριστικό της Επιχειρησιακής Έρευνας είναι το γεγονός ότι οι όποιες λύσεις δίνονται, προκύπτουν πάντα με το σκεπτικό της επιχείρησης ως σύνολο. Μπορεί να λαμβάνονται υπόψη όλες οι πιθανές μεταβλητές, αλλά αυτό γίνεται αποκλειστικά με σκοπό το μέγιστο όφελος του οργανισμού και με απώτερο στόχο όχι κάτι λιγότερο από τη βέλτιστη λύση.

Οι έννοιες που ενδέχεται να συναντήσει κάποιος ασχολούμενος με την Επιχειρησιακή Έρευνα είναι ποικίλες. Ενδεικτικά αναφέρονται, ο γραμμικός και μη γραμμικός προγραμματισμός, ο ακέραιος προγραμματισμός, ο δυναμικός προγραμματισμός, η πολυκριτηριακή ανάλυση αποφάσεων, η θεωρία ουρών αναμονής, η στοχαστική μοντελοποίηση και η θεωρία παιγνίων .

1.2 Εξέλιξη της Επιχειρησιακής Έρευνας

Το πρώτο βήμα της Επιχειρησιακής Έρευνας (Ε.Ε.) έγινε το 1665 με την μέθοδο του Newton, που στόχευε στην επίλυση διαφορικής εξίσωσης με τον ελάχιστο αριθμό λύσεων. Τον 20ο αιώνα ως πρώτες προσπάθειες εφαρμογής της Επιχειρησιακής Έρευνας μπορούν να αναφερθούν, η μελέτη του Δανού μαθηματικού A.K. Ernald ο οποίος εξέτασε διάφορα προβλήματα που είχαν σχέση με το χρόνο απασχόλησης τηλεφωνικών γραμμών το 1927, καθώς και διάφορες μελέτες του Horace C. Levelson τη δεκαετία 1920-1930 (Κιόχος, Θάνος & Σαλαμούρης, 2002).

Επισήμως όμως, θεωρείται πως η Επιχειρησιακή Έρευνα ξεκίνησε λίγο πριν και κατά την διάρκεια του δευτέρου παγκοσμίου πολέμου. Τότε, δημιουργήθηκαν οι πρώτες ομάδες επιχειρησιακής έρευνας από τις βρετανικές και αμερικανικές δυνάμεις, αποτελούμενες από επιστήμονες διαφόρων τομέων, με σκοπό την αντιμετώπιση και επίλυση των στρατηγικών, τακτικών και προβλημάτων που προεκύπταν κατά την διάρκεια του πολέμου. Ο όρος Operational Research γεννήθηκε αυτή την περίοδο, καθώς ουσιαστικά σήμαινε Research in Military Operations. Συγκεκριμένα, σημείο αναφοράς αποτέλεσαν οι ομάδες που ασχολήθηκαν με την αποδοτικότερη χρήση ενός νέου ραντάρ (τότε «σύστημα έγκαιρης επισήμανσης αεροσκαφών»). Πραγματοποιήθηκε μελέτη, όχι μόνο του συστήματος αυτού καθαυτού, αλλά των λειτουργιών του, της ορθότερης χρήσης του εξοπλισμού, καθώς και της συμπεριφοράς του ανθρώπινου δυναμικού που το χειρίζεται. Οι συγκεκριμένες μελέτες, θεωρείται ότι έπαιξαν καθοριστικό ρόλο στην έκβαση της Μάχης της Αγγλίας (Battle of Britain, 1940)[1].

Η επιτυχημένη αυτή είσοδος της επιστημονικής προσέγγισης σε στρατιωτικά θέματα επέτρεψε το άνοιγμά της και σε άλλους τομείς. Η γενικότερη βιομηχανική (και όχι μόνο) ανάπτυξη που ακολούθησε τον Δεύτερο Παγκόσμιο Πόλεμο δημιούργησε νέες ανάγκες, καθώς η πολυπλοκότητα των προβλημάτων που προκύπταν σε διάφορους οργανισμούς, επιχειρήσεις αλλά ακόμη και στις κυβερνήσεις, κατέστησαν αναγκαία την εξεύρεση νέων τρόπων αντιμετώπισής τους. Δεν άργησαν να συνειδητοποιήσουν, ότι τα προβλήματα που, επιτυχώς, αντιμετωπίστηκαν κατά την διάρκεια του πολέμου είναι παρόμοια, απλά σε άλλο πλαίσιο. Αυτό είχε ως αποτέλεσμα να δημιουργηθούν επιστημονικές κοινότητες και κέντρα για την Ε.Ε., αλλά και ακαδημαϊκά προγράμματα. Γεννήθηκαν έννοιες όπως γραμμικός προγραμματισμός (Linear Programming), μεθοδολογίες και εργαλεία για επίλυση διαφόρων προβλημάτων επιχειρησιακής έρευνας. Μάλιστα, πολλά εργαλεία από αυτά που χρησιμοποιούνται ευρέως και σήμερα, αναπτύχθηκαν επαρκώς εκείνη την εποχή (Παπαδόπουλος, 2013).

Επιστήμονες που εξέταζαν τα προβλήματα γραμμικού προγραμματισμού (LP) έφθασαν γρήγορα στο συμπέρασμα ότι θα ήταν εφικτό να λυθούν τα προβλήματα που είχαν με μερικές μεταβλητές ακέραιων αριθμών (Dantzig, οπ.

[1] Ανακτήθηκε από en.wikipedia.org.

αναφ. στο Δήλια, 2011). Με αυτά τα συμπεράσματα, ο μαθηματικός κόσμος οδηγήθηκε στους αλγόριθμους για τη λύση των αμιγών προβλημάτων ακέραιου προγραμματισμού (Integer Programming - IP). Οι πρώτοι αλγόριθμοι επίλυσης IP προβλημάτων αναπτύχθηκαν από τους Danzig, Fulkerson και Jonshon το 1954 που ασχολήθηκαν με το πρόβλημα του πλανόδιου πωλητή και από τον Gomory τη δεκαετία του '50 και '60, ο οποίος ανέπτυξε τον αλγόριθμο cutting-planes. Οι Land και Doig εισήγαγαν στη συνέχεια τον Branch and Bound αλγόριθμο το 1960. Πιο πρόσφατα έχουν χρησιμοποιηθεί: η απαρίθμηση (implicit enumeration) από τους Balas, Ceria, Cornuéjols & Natra (1996), η αποσύνθεση (decomposition) από τον Benders, η Lagrangian relaxation από τον Geoffrion το 1974 και οι προσεγγίσεις heuristics από τους Zanakis και Evans (Δήλια, 2011). Τέλος, το 1991 αναπτύχθηκε ένας συνδυασμός των αλγορίθμων cutting planes και Branch and Bound για το πρόβλημα του πλανώδιου πωλητή από τους Padberg και Rinaldi, ο αλγόριθμος branch and cut.

Τα προβλήματα ακέραιου προγραμματισμού κατατάσσονται στα προβλήματα εκείνα με υψηλό βαθμό πολυπλοκότητας. Η πολυπλοκότητα αυτή συνίσταται στον συνδυαστικό τους χαρακτήρα. Σε πολλά προβλήματα ακέραιου προγραμματισμού, η εύρεση της βέλτιστης λύσης δεν είναι δυνατόν να βρεθεί σε εύλογο χρονικό διάστημα – ο αλγόριθμος ακόμα και σε ισχυρούς υπολογιστές απαιτεί πολύ χρόνο εξαιτίας των πολλών βημάτων που απαιτούνται. Γι' αυτό το λόγο, αρκεί μια λύση η οποία είναι ενδεχομένως υποδεέστερη της βέλτιστης αλλά πιο συμφέρουσα όσον αφορά το κόστος που θα ήταν απαραίτητο για την επίτευξη της άριστης (Κιόχος, Θάνος & Σαλαμούρης, 2002).

Η επιχειρησιακή έρευνα συνέχισε να αναπτύσσεται τις επόμενες δεκαετίες, με την ανάπτυξη της τεχνολογίας και τη δημιουργία των ηλεκτρονικών υπολογιστών η εξέλιξή της ήταν ραγδαία. Ο λόγος είναι προφανής, η επιχειρησιακή έρευνα προϋποθέτει μεγάλη ποσότητα υπολογισμών, κάτι που οι υπολογιστές μπορούν να κάνουν τάχιστα, σε αντίθεση με τον άνθρωπο. Από το 1980 και μετά, αναπτύχθηκε και το πρώτο αντίστοιχο software (Spreadsheet OR add - in software)[2], γεγονός, που σε συνδυασμό με την εξάπλωση των προσωπικών ηλεκτρονικών υπολογιστών, επέτρεψε σε κάθε άνθρωπο να έχει πρόσβαση στην επιστήμη αυτή. Σήμερα, έχουν αναπτυχθεί πάρα πολλά προγράμματα (software) (POM-QM, WINQSB, LINDO κτλ), γλώσσες προγραμματισμού (R, matlab) ή βιβλιοθήκες (Gurobi, CPLEX) σε ήδη υπάρχουσες γλώσσες (java, python, C#, C++) που βοηθούν στην επίλυση προβλημάτων επιχειρησιακής έρευνας και υπάρχουν σε κάθε υπολογιστή, παρόλο που ενδέχεται να μην είναι ευρέως γνωστά ή διαδεδομένα.

[2] Ανακτήθηκε από www.iit.comillas.edu.

2 Ακέραιος Προγραμματισμός

2.1 Ορισμός και Προβλήματα Ακέραιου Προγραμματισμού

Είναι απαραίτητο να αποσαφηνιστούν οι έννοιες γραμμικός, ακέραιος και ακέραιος γραμμικός προγραμματισμός, πριν γίνει παρουσίαση των βασικών τύπων προβλημάτων ακεραίου προγραμματισμού.

Ο γραμμικός προγραμματισμός (LP) είναι μια μαθηματική μέθοδος για τη λύση προβλημάτων στα οποία γίνεται προσπάθεια να βρεθεί η άριστη χρήση περιορισμένων πόρων μιας επιχείρησης, με σκοπό να επιτευχθεί η μεγιστοποίηση του κέρδους ή η ελαχιστοποίηση του κόστους, μέσα στα όρια συγκεκριμένων περιορισμών και δυνατοτήτων της επιχείρησης. (Σαπουτζής, 1993).

Ένα πρόβλημα ακέραιου (γραμμικού) προγραμματισμού, είναι εκείνο το πρόβλημα γραμμικού προγραμματισμού (LP) όπου τουλάχιστον μία μεταβλητή παίρνει αποκλειστικά ακέραιες τιμές. Η έννοια αμιγώς ακέραιος προγραμματισμός (PIP) χρησιμοποιείται για προβλήματα όπου όλες οι μεταβλητές παίρνουν μόνο ακέραιες τιμές (Chen, Batson & Dang, 2010).

Μια εκπληκτικά ευρεία κατηγορία πρακτικών προβλημάτων μπορεί να διαμορφωθεί χρησιμοποιώντας ακέραιες μεταβλητές και γραμμικούς περιορισμούς. Μερικές φορές ένα τέτοιο μοντέλο αποτελείται αποκλειστικά από ακέραιες μεταβλητές. Αυτό, όπως αναφέρθηκε, είναι ένα μοντέλο αμιγού ακέραιου προγραμματισμού (PIP), ωστόσο συχνό φαινόμενο αποτελεί η ύπαρξη συνεχών μεταβλητών μαζί με ακέραιες μεταβλητές. Ένα τέτοιο μοντέλο ονομάζεται και μοντέλο μικτού ακέραιου προγραμματισμού (MIP) (Pandian & Jayalakshmi, 2012).

Η εφαρμογή του ακέραιου προγραμματισμού (IP), γνωστή και ως διακριτός προγραμματισμός, ως μέθοδος μοντελοποίησης δεν είναι τόσο ευρεία όσο ο γραμμικός προγραμματισμός. Όμως, σε περιπτώσεις όπου απαιτείται η παραγωγή ολοκληρωμένων ποσότητων συγκεκριμένων αγαθών, όπως αυτοκίνητα, αεροπλάνα ή σπίτια, αλλά και σε περιπτώσεις που χρησιμοποιούνται ολοκληρωμένες ποσότητες κάποιου πόρου, όπως για παράδειγμα εργαζομένων, μπορούμε να χρησιμοποιήσουμε ένα μοντέλο ακέραιου ή μικτού ακέραιου προγραμματισμού αντί για ένα μοντέλο γραμμικού προγραμματισμού. Η συχνότητα εμφάνισης εφαρμογών ακέραιου προγραμματισμού είναι μεγάλη, ωστόσο σε τέτοιες περιπτώσεις, είναι συχνότερη η χρήση συμβατικών μοντέλων γραμμικού προγραμματισμού και η στρογγυλοποίηση των βέλτιστων τιμών λύσεων στους πλησιέστερους ακέραιους αριθμούς.

Ο τύπος εφαρμογής που περιγράφηκε στη προηγούμενη παράγραφο αποκρύπτει την πραγματική ισχύ του ακέραιου προγραμματισμού σαν μέθοδο μοντελοποίησης. Τα πιο πρακτικά IP μαθηματικά μοντέλα περιορίζουν τις

ακέραιες μεταβλητές σε δύο τιμές, 0 ή 1. Τέτοιες μεταβλητές 0 - 1 χρησιμοποιούνται για να αντιπροσωπεύσουν τις αποφάσεις «ναι ή όχι» (Padberg & Rijal, 1996). Οι λογικές συνδέσεις μεταξύ τέτοιων αποφάσεων μπορούν συχνά να επιβληθούν χρησιμοποιώντας γραμμικούς περιορισμούς.

Είναι σκόπιμο να αναφερθεί η δυσκολία στον τρόπο επίλυσης των μοντέλων ακέραιου και μικτού ακέραιου όσο και ο τρόπος κατασκευής τους. Τα μοντέλα μικτού ακέραιου προγραμματισμού, από μαθηματικής άποψης, θυμίζουν πολλές φορές, όσον αφορά τον υπολογισμό λύσεων, τα συχνά απλούστερα και πιο κατανοητά μοντέλα γραμμικού προγραμματισμού παρόμοιου μεγέθους. Η δυσκολία των ακέραιων προβλημάτων σε σύγκριση με τα συνεχή και γραμμικά προβλήματα έγκειται στο σημείο που γίνεται ο υπολογισμός ακέραιων μεταβλητών. Από τη μία πλευρά, ένα μοντέλο γραμμικού προγραμματισμού που περιλαμβάνει χιλιάδες περιορισμούς και μεταβλητές μπορεί σχεδόν σίγουρα να λυθεί σε εύλογο χρονικό διάστημα χρησιμοποιώντας ένα σύγχρονο πρόγραμμα υπολογιστή και λογισμικό, ενώ κάτι παρόμοιο δεν ισχύει για τα μοντέλα ακέραιου προγραμματισμού. Η οικοδόμηση ενός μοντέλου IP ενέχει τον κίνδυνο μη παροχής λύσης σε εύλογο χρονικό διάστημα όπως αναφέρθηκε προηγουμένως (Κιόχος, Θάνος & Σαλαμούρης, 2002).

Γίνεται μια προσπάθεια από τον Williams (2013) ώστε να ταξινομηθούν τα διάφορα είδη προβλημάτων για τα οποία μπορούν να κατασκευαστούν μοντέλα IP. Αναπόφευκτα, υπάρχει μια ορισμένη αλληλοεπικάλυψη σε αυτήν την ταξινόμηση, όπου συγκεκριμένες εφαρμογές δεν καταλήγουν σε μία μόνο κατηγορία. Πολλά πρακτικά προβλήματα συνδυάζουν πτυχές από διάφορες κατηγορίες. Τα είδη προβλημάτων είναι τα παρακάτω:

- Προβλήματα με διακριτά δεδομένα και λύσεις
- Προβλήματα με λογικές συνθήκες
- Συνδυαστικά προβλήματα
- Μη γραμμικά προβλήματα
- Προβλήματα δικτύου

Με αυτή τη χαλαρή ταξινόμηση, είναι επιθυμητό να μεταφερθεί στον αναγνώστη μια αίσθηση για τα είδη των προβλημάτων στα οποία εφαρμόζεται ο γραμμικός ή ο ακέραιος προγραμματισμός.

2.1.1 Προβλήματα με διακριτά δεδομένα και λύσεις

Αυτή η κατηγορία προβλημάτων περιλαμβάνει τις πιο προφανείς εφαρμογές ακέραιου προγραμματισμού όπου είναι απαραίτητη, η δημιουργία ή η χρήση αποκλειστικά ακέραιων μονάδων. Στις οικονομικές επιστήμες, οι οικονομολόγοι μερικές φορές αναφέρονται σε τέτοια προβλήματα, που έχουν «άγονες» εισόδους ή εξόδους.

Οι μεταβλητές αντιπροσωπεύουν τις ποσότητες δύο διαφορετικών αγαθών που πρέπει να υπολογιστούν και είναι σημαντικό να γίνει σαφές ότι τα αποτελέσματα αυτά πρέπει να είναι αναπόσπαστα, παραδείγματος χάριν, να αντιπροσωπεύουν αδιαίρετα αγαθά, όπως τα αυτοκίνητα. Σε περίπτωση που αυτό δεν ισχύει και αντιπροσωπεύουν διακριτά αγαθά, όπως για παράδειγμα

λίτρα νερού, θα είναι ικανοποιητική η αντιμετώπιση του προβλήματος ως μοντέλου LP.

Εν αντιθέσει, αν ο περιορισμός της φύσης των μεταβλητών τους υποχρεώνει να είναι ακέραιοι, τότε η αμέσως επόμενη κίνηση που προκύπτει είναι η στρογγυλοποίηση των τιμών αυτών. Η στρογγυλοποίηση όμως των τιμών στους πλησιέστερους ακέραιους αριθμούς δίνει πολλές φορές μια μη εφικτή λύση ή λύση που δίνει δεν είναι η βέλτιστη. Σε τέτοιες περιπτώσεις είναι απαραίτητο να επιλυθεί ένα τέτοιο πρόβλημα ως μοντέλο IP (Williams, 2013).

Γενικότερα, και στις δύο αυτές περιπτώσεις παρατηρούνται δυσκολίες για τα περισσότερα προβλήματα ακέραιου προγραμματισμού που αντιμετωπίζονται ως προβλήματα γραμμικού προγραμματισμού, ενώ το πλήθος των μεταβλητών είναι πιθανό να είναι πολύ μεγαλύτερο και τα σφάλματα που σχετίζονται με τη στρογγυλοποίηση της κλασματικής λύσης LP να είναι σοβαρά. Αντίθετα, η επίλυση ενός τέτοιου προβλήματος ως ενός μοντέλου IP θα μπορούσε να πάρει πολύ χρόνο λόγω των πολλών συνδυασμών ολοκληρωμένων λύσεων που θα μπορούσαν να ληφθούν υπόψη (Meyer, 1975).

Παρόμοιες θεωρήσεις με αυτές που προαναφέρθηκαν ισχύουν για προβλήματα όπου οι είσοδοι και όχι οι έξοδοι είναι διακριτές. Όπως και στο θέμα της παρούσας μεταπτυχιακής εργασίας, μια τέτοια είσοδος θα είναι προβλήματα διαχείρισης του ανθρώπινου δυναμικού, η οποία, αν είναι επαρκώς μεγάλη, μπορεί να αντιμετωπιστεί ως συνεχής.

2.1.2 Προβλήματα με λογικές συνθήκες

Συχνό φαινόμενο είναι η επιβολή επιπλέον περιορισμών σε ένα μοντέλο γραμμικού προγραμματισμού. Αυτοί οι περιορισμοί είναι μερικές φορές λογικού χαρακτήρα, που δεν μπορούν να διαμορφωθούν από συμβατικά προβλήματα γραμμικού προγραμματισμού. Για παράδειγμα, ένα μοντέλο LP μπορεί να χρησιμοποιηθεί για να αποφασιστεί η ποσότητα παραγωγής καθενός από τα πιθανά προϊόντα ενός εργοστασίου που υπόκεινται σε περιορισμούς παραγωγικής ικανότητας (εφαρμογή συνδυασμού προϊόντων). Θα μπορούσε ακόμα να προστεθεί μια επιπλέον προϋπόθεση, όπως «Εάν παραχθεί το προϊόν Α τότε πρέπει να παραχθεί και το προϊόν Β ή Γ». Η εισαγωγή ορισμένων επιπλέον ακέραιων μεταβλητών στο μοντέλο μαζί με επιπλέον μαθηματικούς περιορισμούς, μπορεί εύκολα να δημιουργήσει τέτοιους περιορισμούς. Το μοντέλο που προκύπτει είναι ένα ακέραιο γραμμικό πρόβλημα. Οποιαδήποτε τέτοια λογική συνθήκη όπως η παραπάνω μπορεί να επιβληθεί σε μοντέλο LP χρησιμοποιώντας τεχνικές IP (Jeroslow,1989).

2.1.3 Συνδυαστικά προβλήματα

Πολλά επιχειρησιακά ερευνητικά προβλήματα έχουν ως χαρακτηριστικό τον πολύ μεγάλο αριθμό εφικτών λύσεων. Αυτά προκύπτουν από διαφορετικές επιχειρησιακές εντολές (ή περιορισμούς) όπως από την κατανομή πόρων ή ανθρώπων σε διαφορετικές θέσεις. Τέτοια προβλήματα αναφέρονται ως συνδυαστικά.

Αυτή η κατηγορία είναι πιθανόν να υποδιαιρεθεί περαιτέρω, σε προβλήματα αλληλουχίας και προβλήματα κατανομής. Ένας ιδιαίτερα χαρακτηριστικός και δυσχερής τύπος προβλήματος αλληλουχίας προκύπτει στον προγραμματισμό ενός εργοστασίου, όπου είναι επιθυμητή η βέλτιστη κατανομή εργασιών σε διαφορετικές μηχανές σε ένα τομέα εργασίας (μοντέλο αλληλουχίας εργασιών) (Taha, 2007). Ο ακέραιος προγραμματισμός δίνει μια μέθοδο μοντελοποίησης αυτού του τύπου, όμως όπως αναφέρθηκε, σε πολλές περιπτώσεις δεν έχει αποδειχθεί πολύ επιτυχημένος τρόπος αντιμετώπισης από πλευράς χρόνου και πόρων αυτού του προβλήματος μέχρι σήμερα.

Ένα άλλο πολύ γνωστό πρόβλημα αλληλουχίας είναι το προαναφερθέν *πρόβλημα του πλανόδιου πωλητή* (Danzig, Fulkerson & Jonshon, 1954). Αυτό το πρόβλημα έχει στόχο την ανεύρεση της βέλτιστης διαδρομής ώστε ο πωλητής να επισκεφτεί ένα σύνολο πόλεων και να επιστρέψει στην αρχική του πόλη καλύπτοντας την ελάχιστη δυνατή απόσταση.

Κλασσικό πρόβλημα κατανομής αποτελεί αυτό του μεριδίου αγοράς. Το πρόβλημα περιλαμβάνει την κατανομή των πελατών σε τμήματα μιας εταιρείας ώστε να τους προσφερθούν οι απαραίτητες υπηρεσίες. Παρόλο που η σύνθεση είναι συγκριτικά απλή, προβλήματα αυτού του τύπου δεν είναι πάντα εύκολο να λυθούν. Προβλήματα παρόμοιας μορφής προκύπτουν κατά την επιλογή των έργων και την κατανομή του προϋπολογισμού του κεφαλαίου.
Τέλος, ένα ακόμα πρόβλημα κατανομής είναι το πρόβλημα προγραμματισμού του αεροσκάφους. Αυτό το πρόβλημα έχει ως στόχο την εκχώρηση αεροσκαφών σε σύνολα πτήσεων με το βέλτιστο τρόπο (Williams, 2013).

2.1.4 Μη γραμμικά προβλήματα

Μερικές φορές, μη γραμμικά προβλήματα μπορούν να αντιμετωπίζονται ως προβλήματα ακέραιου προγραμματισμού. Ενδεικτικά αναφέρεται ότι, εάν το πρόβλημα μπορεί να εκφραστεί σε μια ακέραιη μορφή προγραμματισμού, μπορεί να λυθεί χρησιμοποιώντας είτε μεθόδους ακέραιου προγραμματισμού IP είτε γραμμικού LP. Η κατηγορία αυτή περιλαμβάνει προβλήματα που περιέχουν οικονομίες κλίμακας, προβλήματα τετραγωνικού προγραμματισμού και προβλήματα γεωμετρικού προγραμματισμού, αλλά και γενικότερα μη γραμμικά προβλήματα προγραμματισμού (Jeroslow, 1987).

2.1.5 Προβλήματα δικτύου

Ένα δίκτυο είναι μια συλλογή από κόμβους (nodes) οι οποίοι συνδέονται με γραμμές, τις οποίες ονομάζουμε ακμές (arcs, branches) (Γεωργίου & Οικονόμου, 2011).

Στο παράδειγμα των Cook, Cunningham, Pulleyblank & Schrijver (1998) παρουσιάζεται το πρόβλημα δικτύου μιας εταιρείας πετρελαίου που προσπαθεί να πάει σε όλες τις πλατφόρμες εξόρυξης πετρελαίου με ελικόπτερο καλύπτοντας την ελάχιστη δυνατή διαδρομή. Το παραπάνω παράδειγμα αντιμετωπίζεται ως μια εφαρμογή του προαναφερθέντος *προβλήματος του πλανόδιου πωλητή*.

Ενδεικτικά, παρατηρούνται περιπτώσεις όπου ο γραμμικός και ο ακέραιος προγραμματισμός μπορούν να μοντελοποιήσουν και προβλήματα δικτύων. Όπως είναι κατανοητό, η κατηγορία τέτοιων προβλημάτων είναι ευρεία και θα μπορούσε να αναλυθεί σε βάθος, όμως δεν κρίνεται απαραίτητο στα πλαίσια της συγκεκριμένης εργασίας.

Είναι χαρακτηριστικό ότι από τις κατηγορίες εφαρμογών ακέραιου προγραμματισμού που προαναφέρθηκαν, υπάρχει αλληλοεπικάλυψη στην ταξινόμηση του Williams με αποτέλεσμα, όπως και στο *πρόβλημα του πλανόδιου πωλητή*, συγκεκριμένα προβλήματα να μην εντάσσονται σε μία μόνο κατηγορία.

2.2 Αλγόριθμοι ακέραιου γραμμικού προγραμματισμού

Οι αλγόριθμοι ακέραιου γραμμικού προγραμματισμού (ILP) βασίζονται στη δυναμική της μεγάλης υπολογιστικής επιτυχίας του γραμμικού προγραμματισμού. Η στρατηγική αυτών των αλγορίθμων, σύμφωνα με τον Taha (2007) περιλαμβάνει τρία βήματα:

1. Γίνεται ελάφρυνση (relaxation) του χώρου λύσεων του ILP, διαγράφοντας τους ακέραιους περιορισμούς όλων των ακέραιων μεταβλητών και αντικαθιστώντας κάθε δυαδική μεταβλητή y με τη συνεχή περιοχή 0 - 1. Το αποτέλεσμα αυτής της ελάφρυνσης είναι ένα απλό μοντέλο Γραμμικού Προγραμματισμού.
2. Επιλύεται το γραμμικό μοντέλο και προσδιορίζεται η συνεχή βέλτιστη λύση του.
3. Ξεκινώντας από το συνεχές βέλτιστο σημείο, προστίθενται ειδικοί περιορισμοί που τροποποιούν επαναληπτικά το χώρο των λύσεων του γραμμικού προβλήματος με ένα τρόπο που τελικά οδηγεί σε ένα βέλτιστο ακρότατο σημείο που ικανοποιεί τις ακέραιες απαιτήσεις.

Έχουν αναπτυχθεί δύο γενικές μέθοδοι για τη δημιουργία των ειδικών περιορισμών στο Βήμα 3.

1. Η μέθοδος διακλάδωσης και φραγής (Branch – and – bound - B&B).

2. Η μέθοδος των τεμνόντων επιπέδων (Cutting - plane).

Είναι κοινή διαπίστωση, ότι καμία από τις δύο μεθόδους δεν είναι σίγουρα αποτελεσματική για όλα τα προβλήματα ακέραιου προγραμματισμού. Όμως, η εμπειρία δείχνει ότι η μέθοδος διακλάδωσης και φραγής είναι με διαφορά πιο επιτυχημένη σε σχέση με τη μέθοδο των τεμνόντων επιπέδων.

Το 1991 αναπτύχθηκε ένας συνδυασμός των αλγορίθμων cutting planes και Branch and Bound για το *πρόβλημα του πλανώδιου πωλητή* από τους Padberg & Rinaldi, ο αλγόριθμος «Branch and Cut», ο οποίος χρησιμοποιείται και από πολλά εργαλεία της σημερινής εποχής για την επίλυση προβλημάτων ακέραιου γραμμικού προγραμματισμού. Ένα από αυτά τα εργαλεία είναι και η το λογισμικό CPLEX της IBM που παρέχει και την βιβλιοθήκη σε γλώσσα Java.[3]

2.3 Προβλήματα εργατικού δυναμικού ακέραιου προγραμματισμού

Στο παρελθόν, οι επιχειρήσεις αντιμετώπιζαν το πρόβλημα του προγραμματισμού εργατικού δυναμικού, όσο το δυνατόν καλύτερα, προχωρώντας σε συνεχείς διακυμάνσεις και μεταλλάξεις στο εργατικό τους δυναμικό με απολύσεις και προσλήψεις. Αυτή η πραγματικότητα παρήγαγε ένα μεγάλο αριθμό νέων διατάξεων και ρυθμίσεων εργασίας (ετήσιες ώρες εργασίας, κατανομή θέσεων εργασίας, εναλλαγή θέσεων εργασίας, κ.λπ.) και προώθησαν νέες ιδέες και ρυθμίσεις όπως εργασία με μερική απασχόληση ή προσωρινή εργασία (Ellis & Stredwick 1998). Κατά συνέπεια αναπτύχθηκαν διάφορες ερευνητικές μελέτες που έχουν επικεντρωθεί στον προγραμματισμό του εργατικού δυναμικού.

Ο μικτός προγραμματισμός ακέραιων αριθμών (MIP), καθώς και ο ακέραιος προγραμματισμός (IP), είναι από τις πιο επιτυχημένες μεθόδους για την μοντελοποίηση και επίλυση προβλημάτων προγραμματισμού (scheduling). Από τα πρώτα έργα του Dantzig (1954) μέχρι τις μέρες μας, έχει δημοσιευθεί μεγάλος αριθμός επιστημονικών έργων. Επιπλέον, ο μικτός ακέραιος προγραμματισμός και ο ακέραιος προγραμματισμός χρησιμοποιήθηκαν για την επίλυση πραγματικών προβλημάτων στις επιχειρήσεις (Azmat, Hürlimann & Widmer, 2004), όπως το ιεραρχικό πρόβλημα του εργατικού δυναμικού (Billionnet, 1999) ή το πρόβλημα της ετήσιας ώρας (Corominas, Garcia & Pastor, 2002).

Σε πολλά προβλήματα workforce scheduling όπως του Billionnet (1999), αναπτύχθηκαν μαθηματικά μοντέλα με σκοπό την καλύτερη δυνατή κατανόηση

[3] Ανακτήθηκε από www.ibm.com

και περιγραφή του προβλήματος. Ο στόχος αυτού του μοντέλου είναι να προσδιορίσει ένα βέλτιστο ιεραρχικό εργατικό δυναμικό στο οποίο μπορεί να γίνονται εύκολα και ευέλικτα ανακατατάξεις, με διαφορετικές καθημερινές απαιτήσεις εργασίας εντός μιας εβδομάδας και συγκεκριμένο αριθμό ημερών ρεπό.

Στο αντίστοιχο πρόβλημα εργατικού δυναμικού των Seckiner, Gokcen & Kurt (2006), οι οποίοι βασίστηκαν στη μοντελοποίηση του προβλήματος του Billionnet, οι εργαζόμενοι δουλεύουν σε διαφορετικές βάρδιες καθημερινά, με διαφορετική χρονική διάρκεια.

Ένα ακόμη παράδειγμα είναι το πρόβλημα των Burke, Curtois, Qu & Vanden Berghe, του οποίου σκοπός είναι η αυτοματοποίηση της διαδικασίας προγραμματισμού των βαρδιών και η μείωση του φόρτου εργασίας. Πρέπει να τονιστεί ότι το συγκεκριμένο πρόβλημα αναλύει σε μεγαλύτερο βάθος και άλλα θέματα, για παράδειγμα ότι τα καλύτερα και ποιοτικότερα δρομολόγια σε μια εργασία μειώνουν την κόπωση και το άγχος των εργαζομένων. Επιπρόσθετα η μείωση της υπερβολικής εργασίας και του φτωχού προγραμματισμού, συμβάλλει στη μεγιστοποίηση της χρήσης του ελεύθερου χρόνου των εργαζομένων, ικανοποιώντας ταυτόχρονα περισσότερες απαιτήσεις τους. Ένα πιο ευχαριστημένο εργατικό δυναμικό θα οδηγεί σε υψηλότερη παραγωγικότητα, αυξημένη ποιότητα υπηρεσιών ασθενών και καλύτερο επίπεδο υγειονομικής περίθαλψης.

2.4 Αντίστοιχα μαθηματικά μοντέλα προγραμματισμού εργατικού δυναμικού

Σε περιπτώσεις όπως των Jaumard, Semet, Vovor το πρόβλημα *'nurse scheduling'* παρουσιάζεται σαν ένα πρόβλημα δικτύου 0 - 1 '*shortest path'* (Jaumard, Semet & Vovor, 1996). Αντίστοιχα, όπως αναφέρθηκε, σε παρόμοια προβλήματα workforce scheduling όπως του Billionnet (1999), αναπτύχθηκαν μαθηματικά μοντέλα με σκοπό την καλύτερη δυνατή κατανόηση και περιγραφή του προβλήματος, αλλά και τον προσδιορισμό ενός βέλτιστου ιεραρχικού εργατικού δυναμικού στο οποίο ο υψηλότερα ειδικευμένος εργαζόμενος μπορεί να υποκαταστήσει ένα άτομο με χαμηλότερη εξειδίκευση, αλλά όχι το αντίστροφο. Οι καθημερινές απαιτήσεις εργασίας εντός μιας εβδομάδας μπορεί να διαφέρουν, αλλά κάθε εργαζόμενος πρέπει να λαμβάνει συγκεκριμένο αριθμό «Ν» ημέρες ρεπό την εβδομάδα. Η αντικειμενική συνάρτηση ήταν ως εξής:

$$Min \sum_{k=1}^{m} C_k W_k$$

Όπου W_k είναι ο αριθμός εργαζόμενων τύπου k και C_k το κόστος αυτού του τύπου εργαζόμενου.

Σε άλλο πρόβλημα εργατικού δυναμικού όπως αυτό των Seckiner, Gokcen και Kurt (2006), το οποίο βασίστηκε στη μοντελοποίηση του Billionnet, το εργατικό δυναμικό δουλεύει σε 3 διαφορετικές βάρδιες καθημερινά οι οποίες έχουν διαφορετική διάρκεια η κάθε μία (8, 10, 12 ώρες). Κάποιοι εργαζόμενοι δουλεύουν λιγότερες μέρες, με περισσότερες ώρες ενώ άλλοι περισσότερες μέρες με λιγότερες ώρες. Η αντικειμενική συνάρτηση σε αυτό το πρόβλημα ήταν:

$$Min \sum_{b=1}^{B} \sum_{k=1}^{m} C_{bk} W_{bk},$$

όπου η μεταβλητή b αντιπροσωπεύει το τύπο βάρδιας. Οι υπόλοιπες μεταβλητές είναι όπως στο προαναφερθέν πρόβλημα του Billionnet.

Αυτό που γίνεται αντιληπτό, είναι πως σε προβλήματα workforce scheduling υπάρχει ο παράγοντας κόστος (σε προβλήματα ελαχιστοποίησης) ή ο παράγοντας παραγωγικότητας (σε προβλήματα μεγιστοποίησης), οι οποίοι δίνουν τη ευκολία να παραχθεί η αντίστοιχη μεταβλητή C_{bk} ώστε να είναι δυνατό να γίνει στο εκάστοτε πρόβλημα από ακέραιου προγραμματισμού ελαχιστοποίηση ή μεγιστοποίηση. Στο πρόβλημα που θα παρουσιαστεί, η μεταβλητή αυτή θα έχει στόχο να υπάρχει ισορροπία σε πρωινές, απογευματινές και νυχτερινές βάρδιες που θα καλύψουν οι εργαζόμενοι κάθε εβδομάδα.

Ο παράγοντας C λοιπόν του προβλήματος της συγκεκριμένης μεταπτυχιακής εργασίας, θα προκύψει από εισροή (input) δεδομένων της προηγούμενης εβδομάδας. Πιο συγκεκριμένα θα χρησιμοποιηθεί σαν βάρος, το είδος της οκτάωρης βάρδιας που κάλυψαν (πρωινή, μεσημεριανή, νυχτερινή) οι νοσοκόμες τη προηγούμενη εβδομάδα.

3 Η Κλινική

Ταυτότητα

Η Κλινική Παθολογικής Ογκολογίας, ΑΠΘ, ιδρύθηκε τον Ιούλιο του 2003 (Αρ. ΦΕΚ 901) και εντάχθηκε στον Οργανισμό του Γενικού Νοσοκομείου «Παπαγεωργίου» με απόφαση του Διοικητικού Συμβουλίου στη συνεδρίασή του 162/12-07-2004. Η επίσημη έναρξη της λειτουργίας της Κλινικής έγινε τον Σεπτέμβριο του 2005.

Η Πανεπιστημιακή Κλινική Παθολογικής Ογκολογίας διευθύνεται από τον Καθηγητή κ. Χρήστο Ν.Παπανδρέου.[4]

3.1 Το Πρόβλημα Nurse Restoring – Περιγραφή

Η περιγραφή του προβλήματος που πάρθηκε από το προσωπικό του νοσοκομείου, ήταν η παρακάτω:

ΠΡΟΓΡΑΜΜΑ ΔΕΚΕΜΒΡΙΟΥ

STAFF

Η κλινική απασχολεί τις παρακάτω νοσηλεύτριες:

1. Ιωάννα (προϊσταμένη)
2. Σοφία (αντικαθιστών προϊσταμένη)
   (Οι νοσοκόμες 1 και 2 δεν μπορούν να λείπουν μαζί.)
3. Αγάπη (νοσηλεύτρια ΤΕ)
4. Μαρία (νοσηλεύτρια ΤΕ)
5. Βασιλική (νοσηλεύτρια ΤΕ)
6. Κατερίνα (νοσηλεύτρια ΤΕ)
7. Βαλασία (νοσηλεύτρια ΤΕ)
8. Αναστασία (νοσηλεύτρια ΤΕ)
9. Ευγενία (νοσηλεύτρια ΤΕ)
10. Χαρούλα (νοσηλεύτρια ΤΕ)
11. Παρθένα (νοσηλεύτρια ΔΕ)
12. Ελένη (νοσηλεύτρια ΔΕ)
13. Μαρία Τ. (νοσηλεύτρια ΔΕ)

[4] Ανακτήθηκε από www.papageorgiou-hospital.gr

(Οι νοσοκόμες ΔΕ 11-12-13 δεν μπορούν να είναι μόνες τους χωρίς νοσηλεύτρια ΤΕ μαζί τους)

Υποσημείωση:

Το νοσηλευτικό προσωπικό της Ογκολογικής κλινικής απαρτίζεται από 13 νοσηλευτές: την προϊσταμένη, την αντικαθιστώσα προϊσταμένη, 8 νοσηλευτές ΤΕ και 3 νοσηλευτές ΔΕ.

ΤΕ είναι νοσηλεύτριες τεχνολογικής εκπαίδευσης, ενώ ΔΕ είναι νοσηλεύτριες δευτεροβάθμιας εκπαίδευσης.

Εξαιρουμένων της προϊσταμένης και της αντικαθιστώσας προϊσταμένης που εργάζονται καθημερινές πενθήμερο και ωράριο 07.00 – 15.00, το πρόγραμμα για το υπόλοιπο προσωπικό είναι μηνιαίο σε 24ωρη βάση των 3 οκταώρων (07.00 – 15.00, 15.00 – 23.00 και 23.00 – 07.00). Με βάση τον κανονισμό, η ελάχιστη χρονική διαφορά ανάμεσα σε 2 βάρδιες που μπορεί να καλύψει μία νοσηλεύτρια είναι 11 ώρες, ουσιαστικά όμως αυτό ορίζεται μετά από 2 βάρδιες δηλαδή 16 ώρες. Επομένως είναι επιτρεπτό μια νοσοκόμα μετά το τέλος της βάρδιας να ξαναεργαστεί μετά από 16 ώρες ή 2 οκτάωρες βάρδιες.

ΣΤΟΧΟΣ

Ο στόχος του προβλήματος είναι η επίτευξη ισορροπίας σε πρωινές, απογευματινές και νυχτερινές βάρδιες, η αποφυγή διπλών βαρδιών, καθώς και η ισορροπία στον αριθμό των εργάσιμων ημερών μέσα στην εβδομάδα μεταξύ των νοσοκόμων (για παράδειγμα η κάλυψη 6 βαρδιών από μια νοσοκόμα και μόλις 2 από μια άλλη). Παρατηρείται συχνά νοσοκόμες να καλύπτουν συνεχόμενες βάρδιες χωρίς να περάσουν τουλάχιστον 16 ώρες μεταξύ του τέλους της προηγούμενης και της αρχής της επόμενης. Ακόμα, παρατηρείται πολλές φορές να μένει μια νοσοκόμα σε βάρδια αντί για δύο, λόγω μη σωστού προγραμματισμού ενώ σε άλλες περιπτώσεις ενδέχεται να εργαστούν και 4 νοσοκόμες για την κάλυψη μιας βάρδιας.

ΚΑΘΗΜΕΡΙΝΕΣ

- Στο πρωινό (07.00 – 15.00) ωράριο απασχολούνται η προϊσταμένη, η αντικαθιστώσα προϊσταμένη και το ελάχιστο 2 νοσηλευτές.
- Στο απογευματινό (15.00 – 23.00) ωράριο απασχολούνται πάντα 2 νοσηλευτές, εκ των οποίων η μία τουλάχιστον πρέπει να είναι ΤΕ.
- Στο βραδινό (23.00 – 07.00) ωράριο απασχολούνται πάντα 2 νοσηλευτές εκ των οποίων η μία πρέπει να είναι ΤΕ.

ΑΡΓΙΕΣ

- Αργία θεωρείται από τις 12.00 το βράδυ της προηγουμένης ημέρας (00.00 ή 00.00 π.μ.) μέχρι τις 12 το βράδυ της επόμενης (00.00 ή 00.00 π.μ.), δηλαδή 24 ώρες.
- Στο πρωινό ωράριο (7:00-15:00) απασχολούνται πάντα 2 νοσηλεύτριες εκ των οποίων η μία τουλάχιστον πρέπει να είναι ΤΕ.
- Στο απογευματινό ωράριο (15:00-23:00) απασχολούνται πάντα 2 νοσηλεύτριες εκ των οποίων η μία τουλάχιστον πρέπει να είναι ΤΕ.
- Στο βραδινό ωράριο (23:00-7:00) απασχολούνται 2 νοσηλεύτριες εκ των οποίων η μία τουλάχιστον είναι ΤΕ.
- Στο χρονικό διάστημα του ενός μήνα (Δεκέμβριος 31 μέρες) πρέπει να καλυφθούν 62 ωράρια πρωινά, 62 ωράρια απογευματινά και 62 ωράρια βραδινά.
- Σε ακραίες περιπτώσεις μπορεί να βρίσκεται 1 ΤΕ νοσηλεύτρια το βράδυ.
- Σε γενική εφημερία πρέπει να παρευρίσκονται πάντα 2 νοσηλεύτριες το βραδύ.
- Σε μηνιαία βάση αν κάποιος δουλέψει ρεπό, Σάββατο ή Κυριακή δεν θα δουλέψει καθημερινή.
- Κάθε νοσοκόμα δεν μπορεί να εργαστεί παραπάνω από 5 φορές την εβδομάδα.
- Αργίες θεωρούνται η Δευτέρα 25 και Τρίτη 26 Δεκέμβρη και η Δευτέρα 1 Ιανουαρίου.
- Γενική εφημερία έχει οριστεί στις 1,5,9,13,17,21,25,29 Δεκέμβρη.
- Αν κάποια είναι εφημερεύουσα δουλεύει στο τμήμα αλλά σαν 3ο άτομο.
- Κάθε νοσοκόμα μπορεί να εργαστεί το μέγιστο 3 απογεύματα την εβδομάδα.
- Κάθε νοσοκόμα μπορεί να εργαστεί το μέγιστο 2 βράδια την εβδομάδα.

3.2 Περιγραφή Περιορισμών

1. Η Ιωάννα και η Σοφία απασχολούνται πάντα σε ωράριο πρωινό (7:00-15:00) τις καθημερινές και τα Σαββατοκύριακα έχουν ρεπό. Στις 25/12 και 26/12 και οι δύο έχουν Αργία. Επίσης η Ιωάννα έχει άδεια από τις 11/12 έως και 15/12.

2. Το λοιπό προσωπικό δεν πρέπει να εργαστεί σε τουλάχιστον σε μία εκ των δύο εορτών (Χριστούγεννα - Πρωτοχρονιά).

3. Η Αγάπη θα εργαστεί ως εφημερεύουσα στις 4/12 και σε βραδινό ωράριο (23:00-7:00), άρα μπορεί να δουλέψει και σε πρωινό ωράριο (7:00-15:00) ή να έχει ρεπό. Στις 7/12 ζήτησε γονική άδεια, στις 9/12 και 10/12 ζήτησε να έχει ρεπό. Επίσης λόγω εορτών πρέπει να έχει ρεπό 23/12 και 24/12 καθώς και Αργία στις 25/12 και 26/12.

4. Η Μαρία ζήτησε ρεπό στις 5/12, 6/12 και 7/12. Στις 10/12 ζήτησε ωράριο πρωινό (7:00-15:00) και στις 11/12 θα είναι εφημερεύουσα σε απογευματινό ωράριο (15:00-23:00),άρα δεν μπορεί να απασχοληθεί στο τμήμα της. Και τέλος ζήτησε να απασχοληθεί σε ωράριο πρωινό (7:00-15:00) στις 23,25 και 26/12.

5. Η Βασιλική θα έχει άδεια από 4/12 έως και 8/12 ,άρα 2/12 , 3/12, 9/12 και 10/12 θα έχει ρεπό. Στις 20/12 θα είναι εφημερεύουσα σε απογευματινό ωράριο (15:00-23:00) και δεν θα μπορεί να απασχοληθεί στο τμήμα της. Επίσης ζήτησε στις 30/12 και στις 31/12 να έχει ρεπό.

6. Η Κατερίνα ζητάει ειδικό πρόγραμμα το οποίο προσαρμόζεται βάσει του προγράμματος του άντρα της. (Σταθερό πρόγραμμα που δίνει η ίδια)

7. Η Βαλασία ζήτησε 1/12 να έχει ρεπό, μαζεμένα ρεπό από 14/12 έως και 20/12. Επίσης ζήτησε τις καθημερινές να μην έχει πολλά απογευματινά ωράρια (15:00-23:00).

8. Η Αναστασία έχει άδεια στις 1/12 και 4/12, άρα στις 2/12 και 3/12 πρέπει να έχει ρεπό.

9. Η Ευγενία ζήτησε 24/12 να εργαστεί στο πρωινό ωράριο (7:00-15:00).

10. Η Χαρούλα είναι συμβασιούχα και μπορεί να απασχοληθεί σε βραδινό ωράριο (23:00-7:00) μόνο 4 φορές μέσα στον μήνα και να εργαστεί σε 2 μόνο αργίες μέσα στο μήνα. Επίσης ζητάει Πέμπτη και Παρασκευή να μην απασχολείται σε απογευματινό ωράριο (15:00-23:00).

11. Η Παρθένα ζήτησε ρεπό στις 23/12 και 24/12. Στις 25/12 επιθυμεί να εργαστεί σε πρωινό ωράριο (7:00-15:00) και βραδινό ωράριο (23:00-7:00), στις 26/12 ζήτησε βραδινό ωράριο (23:00-7:00). Επίσης στις 31/12 ζήτησε να εργαστεί σε απογευματινό ωράριο (15:00-23:00).

12. Η Ελένη ζητάει κάθε Δευτέρα και Τρίτη να μην εργάζεται σε απογευματινό ωράριο (15:00-23:00) και κατά προτίμηση να εργαστεί σε πρωινό (7:00-15:00) ή να έχει ρεπό.

13. Η Μαρία Τ. είναι συμβασιούχα και μπορεί να εργάζεται μέσα στο μήνα 4 φορές σε βραδινό ωράριο (23:00-7:00) και 2 βάρδιες σε αργίες.

Επιπλέον πληροφορίες

- Οι μέρες Κανονικής άδειας του μόνιμου προσωπικού είναι 25.
- Το προσωπικό δικαιούται 4 μέρες Υπεύθυνης δήλωσης - απουσίας μέσα στον χρόνο
- Οι εκπαιδευτικές άδειες δεν είναι συγκεκριμένες.
- Η διπλή βάρδια 7:00-15:00 και 23:00-7:00 ακολουθείται την άλλη μέρα από βάρδια 23:00-7:00 ή από παύλα/κενό.
- Νυχτερινή βάρδια θεωρείται από τις 22:00 έως και τις 6:00 το πρωί.
- Αργία θεωρείται από τις 24:00 (00.00 ή 00.00 π.μ) το βράδυ της προηγούμενης εώς και τις 24:00 (00.00 ή 00.00 π.μ) της επόμενης μέρας.
- Οι μέρες κανονικής άδειας αριθμώνται από την πρώτη εργάσιμη μέρα και αυτό προϋποθέτει τα 2 Σαββατοκύριακα πριν και μετά την κανονική άδεια να έχεις ρεπό (επιθυμητό).
- Η πρώτη εργάσιμη μέρα μετά την κανονική άδεια πρέπει να είναι στο πρωινό ωράριο (7:00-15:00) ή διπλοβάρδια (7:00-15:00 και 23:00-7:00).
- Τα ρεπό που αναλογούν σε κάθε εβδομάδα είναι 2 αλλά αυτό χωρίς να δίνονται απαραίτητα μέσα στην ίδια εβδομάδα. (Μπορει να δουλέψει κάποιος και 10 μέρες συνεχόμενες)
- Όσοι εργαζόμενοι εργάζονται ημέρες αργιών δικαιούνται ένα επιπλέον ρεπό.
- Μετά από απογευματινό ωράριο (15:00-23:00), την επόμενη μέρα δεν ακολουθεί διπλοβάρδια (7:00-15:00 και 23:00-7:00).

4 Μοντελοποίηση

4.1 Εισαγωγή Προβλήματος

Αρχικά, από τη περιγραφή γίνεται κατανοητό ότι οι 2 προϊστάμενες νοσοκόμες δεν ειναι μέρος του προβλήματος, καθώς έχουν σταθερό ωράριο (πρωινό) και δεν χρειάζεται να προσθεθούν σαν παράμετροι στο μοντέλο.

Επομένως το πρόβλημα περιλαμβάνει 11 νοσοκόμες. Το μοντέλο θα έχει ως στόχο να επιλύει το πρόβλημα σε εβδομαδιαία βάση, παίρνοντας υπόψη δεδομένα από την προηγούμενη εβδομάδα. Δεδομένου ότι το πρόβλημα μας θα είναι ελαχιστοποίησης και ότι δεν υπάρχει κάποια μεταβλητή (π.χ. κόστος εργασίας) θα πρέπει να χρησιμοποιήσουμε σαν εξτρά περιορισμό τις φορές που έκαναν βάρδια οι νοσοκόμες την προηγούμενη εβδομάδα καθώς και σαν μεταβλητή το είδος της βάρδιας που κάλυψαν (πρωινή, μεσημεριανή, νυχτερινή). Οι νοσοκόμες είναι:

1. Αγάπη (νοσηλεύτρια ΤΕ)
2. Μαρία (νοσηλεύτρια ΤΕ)
3. Βασιλική (νοσηλεύτρια ΤΕ)
4. Κατερίνα (νοσηλεύτρια ΤΕ)
5. Βαλασία (νοσηλεύτρια ΤΕ)
6. Αναστασία (νοσηλεύτρια ΤΕ)
7. Ευγενία (νοσηλεύτρια ΤΕ)
8. Χαρούλα (νοσηλεύτρια ΤΕ)
9. Παρθένα (νοσηλεύτρια ΔΕ)
10. Ελένη (νοσηλεύτρια ΔΕ)
11. Μαρία Τ. (νοσηλεύτρια ΔΕ)

Επιπλέον, οι γενικές εφημερίες μπορούν να καλυφθούν και από τις 2 προϊστάμενες νοσοκόμες και είναι σταθερός ο αριθμός που γίνονται μέσα σε κάθε μήνα.

Τέλος, η μοντελοποίηση θα γίνει σε 2 στάδια, αρχικά με το γενικό μοντέλο και τους περιορισμούς που ισχύουν κάθε εβδομάδα, και στη συνέχεια με τους περιορισμούς της εκάστοτε εβδομάδας. Από τα δεδομένα προκύπτουν 4 εβδομαδιαία προβλήματα με ίδιο μαθηματικό μοντέλο αλλά διαφορετικούς περιορισμούς. Προς αποφυγή επαναληψιμότητας, θα λυθεί το πρόβλημα της πρώτης εβδομάδας. Το πρόβλημα προς επίλυση είναι ένα πρόβλημα ακέραιου προγραμματισμού.

4.2 Εργαλείο επίλυσης

Για την επίλυση του προβλήματος θα χρησιμοποιήσουμε ένα συνδιασμό λογισμικού επίλυσης – γλώσσας προγραμματισμού. Το λογισμικό είναι το ILOG CPLEX Optimization Studio της IBM (academic licence) το οποίο υποστηρίζεται μέσω βιβλιοθήκης (IloCplex) και jar (cplex.jar – παρέχεται με την εγκατάσταση

του ILOG CPLEX Optimization Studio) και σε γλώσσα Java (όπως και σε python, C κτλ). Το IDE που θα χρησιμοποιηθεί είναι το Apache NetBeans 8.2 και η έκδοση της Java είναι η 8. Η μαθηματική μέθοδος επίλυσης που χρησιμοποιεί το λογισμικό είναι η branch and cut.

4.3 Γενικό μοντέλο

Αρχικά πρέπει να καλυφθούν 42 8ωρα από 11 υπαλλήλους. Έχουμε ότι i → ο κάθε εργαζόμενος και j → η κάθε βάρδια ορίζοντας ότι η πρώτες 2 είναι οι πρωινές στις 1 του μήνα (κάθε μέρα έχει 2 πρωινές, 2 απογευματινές και 2 βραδινές). Επιπλέον, καθώς i, j είναι αδιαίρετοι, $i, j \in Z$. Οι αριθμός κάθε βάρδιας αντικατοπτρίζεται στον πίνακα (Πίνακας 1):

ΔΕΚΕΜΒΡΙΟΣ														
1Η ΕΒΔΟΜΑΔΑ														
	1		2		3		4		5		6		7	
	ΠΑΡΑΣΚΕΥΗ		ΣΑΒΒΑΤΟ		ΚΥΡΙΑΚΗ		ΔΕΥΤΕΡΑ		ΤΡΙΤΗ		ΤΕΤΑΡΤΗ		ΠΕΜΠΤΗ	
ΠΡΩΙ	1	2	7	8	13	14	19	20	25	26	31	32	37	38
ΑΠΟΓΕΥΜΑ	3	4	9	10	15	16	21	22	27	28	33	34	39	40
ΒΡΑΔΥ	5	6	11	12	17	18	23	24	29	30	35	36	41	42

Πίνακας 1

$$42 = \sum_{i=1}^{11} \sum_{j=1}^{42} X_{ij},$$

$$X_{ij} \in \{0,1\},$$

$$i \in \{1,11\},$$

$$j \in \{1,42\}$$

Ο παραπάνω αριθμός ενδέχεται να τροποποιηθεί αν υπάρχουν εφημερίες οπού χρειάζεται μία επιπλέον νοσοκόμα την ημέρα της εφημερίας στο βραδινό ωράριο.

Σε αυτό το σημείο είναι σημαντικό να αναφέρουμε ότι για χάρη κατανόησης του κώδικα και για ευκολία επίλυσης προγραμματιστικά, θεωρούμε ότι η αρίθμηση των νοσοκόμων και των βάρδιων ξεκινάει από το 0 και όχι από το 1. Έτσι η

νοσοκόμα 1 θα γίνει η νοσοκόμα 0, η νοσοκόμα 2 θα γίνει η νοσοκόμα 1... κτλ. Αντίστοιχα, το ίδιο θα συμβεί με τις βάρδιες.

Επομένως έχουμε ότι

$$42 = \sum_{i=0}^{10} \sum_{j=0}^{41} \mathrm{Xij}$$

$$X_{ij} \in \{0,1\},$$

$$i \in \{0,10\},$$

$$j \in \{0,41\}$$

Και

ΔΕΚΕΜΒΡΙΟΣ														
1Η ΕΒΔΟΜΑΔΑ														
	1		2		3		4		5		6		7	
	ΠΑΡΑΣΚΕΥΗ		ΣΑΒΒΑΤΟ		ΚΥΡΙΑΚΗ		ΔΕΥΤΕΡΑ		ΤΡΙΤΗ		ΤΕΤΑΡΤΗ		ΠΕΜΠΤΗ	
ΠΡΩΙ	0	1	6	7	12	13	18	19	24	25	30	31	36	37
ΑΠΟΓΕΥΜΑ	2	3	8	9	14	15	20	21	26	27	32	33	38	39
ΒΡΑΔΥ	4	5	10	11	16	17	22	23	28	29	34	35	40	41

Πίνακας 2

Κάθε νοσοκόμα μπορεί να εργαστεί ξανά αφού περάσουν τουλάχιστον 11 ώρες (δηλαδή 2 οκτάωρες βάρδιες) μετά το τέλος της βάρδιας της.

$$X_{ij} + X_{i(j+1)} + X_{i(j+2)} + X_{i(j+3)} + X_{i(j+4)} + X_{i(j+5)} = 1,$$

$$X_{ij} \in \{0,1\},$$

$$i,j \in \{0,10\},$$

$$j \in \{0,41\}$$

Έτσι, οι νοσοκόμες είναι:

0. Αγάπη (νοσηλεύτρια ΤΕ)
1. Μαρία (νοσηλεύτρια ΤΕ)
2. Βασιλική (νοσηλεύτρια ΤΕ)

3. Κατερίνα (νοσηλεύτρια ΤΕ)
4. Βαλασία (νοσηλεύτρια ΤΕ)
5. Αναστασία (νοσηλεύτρια ΤΕ)
6. Ευγενία (νοσηλεύτρια ΤΕ)
7. Χαρούλα (νοσηλεύτρια ΤΕ)
8. Παρθένα (νοσηλεύτρια ΔΕ)
9. Ελένη (νοσηλεύτρια ΔΕ)
10. Μαρία Τ. (νοσηλεύτρια ΔΕ)

Κάθε νοσοκόμα δεν μπορεί να εργαστεί παραπάνω από 5 φορές την εβδομάδα. (Αν συμβεί αυτό θα έχει εξτρά ρεπό την επόμενη)

$$\sum_{j=0}^{41} X_{ij} \leq 5$$

Κάθε βάρδια γίνεται από μία νοσοκόμα.

$$\sum_{i=0}^{10} X_{ij} = 1$$

$$X_{ij} \in \{0,1\},$$

$$i \in \{0,10\},$$

$$j \in \{0,41\}$$

Έστω ότι οι πρωινές βάρδιες {0,1,...,37} ανήκουν στο σύνολο Π και απογευματινές {2,3,…39} στο σύνολο Α, ενώ οι Βραδινές {4,5…41} στο σύνολο Β.

Οι νοσοκόμες ΔΕ (8,9,10) δεν μπορούν να έχουν ίδια ώρα βάρδια χωρίς νοσοκόμες ΤΕ. Επομένως 2 νοσοκόμες ΤΕ δεν μπορούν να είναι μαζί στην ίδια βάρδια.

$$X_{9j} + X_{10(j+1)} \leq 1 \ ,$$

$$X_{9j} + X_{8(j+1)} \leq 1 \ ,$$

$$X_{10j} + X_{8(j+1)} \leq 1,$$

$$X_{9j} + X_{10(j-1)} \leq 1,$$

$$X_{9j} + X_{8(j-1)} \leq 1,$$

$$X_{10j} + X_{8(j-1)} \leq 1 \ , \ j \in B$$

Επίσης οι συμβασιούχες 7,10 μπορούν να εργαστούν 4 νυχτερινά το μήνα (1 νυχτερινή βάρδια/εβδομάδα). Άρα

$$\sum_{j \in B} X_{7j} \leq 1 \, , \sum_{j \in B} X_{10j} \leq 1$$

Επιπλέον για τις υπόλοιπες νοσοκόμες, εκτός της Κατερίνας που έχει δικό της πρόγραμμα, κάθε μια μπορεί να καλύψει μέγιστο αριθμό 2 νυχτερινών βαρδιών την εβδομάδα. Έτσι έχουμε ότι:

$$\sum_{j \in B} X_{0j} \leq 2 \, ,$$

$$\sum_{j \in B} X_{1j} \leq 2 \, ,$$

$$\sum_{j \in B} X_{2j} \leq 2 \, ,$$

$$\sum_{j \in B} X_{4j} \leq 2 \, ,$$

$$\sum_{j \in B} X_{5j} \leq 2 \, ,$$

$$\sum_{j \in B} X_{6j} \leq 2 \, ,$$

$$\sum_{j \in B} X_{8j} \leq 2 \, ,$$

$$\sum_{j \in B} X_{9j} \leq 2 \, ,$$

$$\sum_{j \in B} X_{10j} \leq 2 \, ,$$

Ακόμα, ο μέγιστος αριθμός απογευματινών βαρδιών που μπορεί η κάθε νοσοκόμα να καλύψει μέσα στην εβδομάδα είναι 3. Δηλαδή για κάθε νοσοκόμα i ισχύει ότι:

$$\sum_{\mathrm{j} \in \mathrm{A}} X_{ij} \leq 3 \ ,$$

Το μοντέλο θα είναι ελαχιστοποίησης, με σκοπό να υπάρχει ισορροπία σε βάρδιες μεταξύ των υπαλλήλων. Αυτό γίνεται με τον δείκτη βάρους του κάθε υπαλλήλου για κάθε βάρδια $C_{iΠ}, C_{iA}, C_{iB}$(πρωί, απόγευμα, βράδυ). Ο εκτιμητής - δείκτης αυτός προσδιορίστηκε εμπειρικά μέσα από δοκιμές που αναφέρονται στη συνέχεια και από το feedback των εργαζομένων του νοσοκομείου.

Κάνοντας μια εκτίμηση βάρους για κάθε βάρδια της προηγούμενης εβδομάδας, θα θεωρηθεί πως η πρωινή, όντας η καλύτερη και αυτή με τη μεγαλύτερη ζήτηση, δεν θα δώσει επιπλέον βάρος αν μια νοσοκόμα καλύψει το πρωινό οκτάωρο. Επομένως $C_{iΠ} = 1$.

Αντίστοιχα, παράλληλη εκτίμηση θα γίνει για τις βραδινές βάρδιες. Θεωρώντας ότι το οκτάωρο (23:00-7:00) είναι το δυσκολότερο να καλυφθεί, με βάση το feedback των νοσοκόμων και θα πρέπει να έχει το μεγαλύτερο βάρος. Αρχικά με δοκιμή $C_{iB} = 1.4$, $C_{iB} = 1.3$ ή $C_{iB} = 1.2$ για κάθε βραδινή βάρδια, παρατηρήθηκε ότι αν μια νοσοκόμα καλύψει πολλές νυχτερινές βάρδιες την προηγούμενη εβδομάδα υπάρχει ο κίνδυνος το βάρος των νυχτερινών βαρδιών της επόμενης να γίνει πολύ μεγάλο και κατά συνέπεια μια νυχτερινή βάρδια να ισούται με 2 ή 3 πρωινές. Έτσι, εκτιμήθηκε ότι για κάθε βραδινή βάρδια $C_{iB} = 1.1$, με το συνολικό βάρος των βαρδιών της προηγούμενης εβδομάδας:

$$C_{iB} = 1 + 0.1(\sum_{j \in B} X_{ij})$$

Τέλος, για το απογευματινό ωράριο ο εκτιμητής βάρους θα πρέπει να κυμαίνεται ανάμεσα στο βάρος του πρωϊνού ($C_{iΠ} = 1$) και του νυχτερινού ($C_{iB} = 1.1$) οκτάωρου. Όπως και με τη νυχτερινή βάρδια, μέσα από δοκιμές ($C_{iA} = 1.04$, $C_{iA} = 1.03$ ή $C_{iA} = 1.02$) γίνεται η προσπάθεια εύρεσης του κατάλληλου εκτιμητή ώστε σε περιπτώσεις που μια νοσοκόμα καλύψει πολλές απογευματινές βάρδιες να μην θεωρηθεί πως το βάρος τριών η τεσσάρων απογευματινών βαρδιών είναι παρόμοιο αριθμητικά με το βάρος μιας νυχτερινής. Έτσι, εκτιμήθηκε ότι για κάθε βραδινή βάρδια $C_{iA} = 1.01$, με το συνολικό βάρος των βαρδιών της προηγούμενης εβδομάδας:

$$C_{iA} = 1 + 0.01(\sum_{\mathrm{j} \in \mathrm{A}} X_{ij})$$

και

$$Min\ X_{11} + X_{12} + C_{1A}X_{13} + C_{1A}X_{14} + C_{1B}X_{15} + C_{1B}X_{16} + X_{17} + \cdots + C_{1B}X_{1(42)} + X_{21} + \cdots + C_{11B}X_{11(42)}$$

Δηλαδή, αν η νοσοκόμα 2 είχε κάνει 2 απογευματινές και 3 βραδινές βάρδιες τη προηγούμενη βδομάδα, για την τωρινή θα είχαμε $C_{2A} = 1.02$ και $C_{2B} = 1.3$, όπου $C_{2A} = C_{2\_3} = C_{2\_4} = \cdots$

Σύμφωνα με το πρόγραμμα της προηγούμενης εβδομάδας - τελευταίας του Νοεμβρίου (Πίνακας 3)

ID ΕΡΓΑΖΟΜΕΝΟΥ ΑΝΑ ΒΑΡΔΙΑ														
ΝΟΕΜΒΡΙΟΣ														
ΤΕΛΕΥΤΑΙΑ ΕΒΔΟΜΑΔΑ														
	24		25		26		27		28		29		30	
	ΠΑΡΑΣΚΕΥΗ		ΣΑΒΒΑΤΟ		ΚΥΡΙΑΚΗ		ΔΕΥΤΕΡΑ		ΤΡΙΤΗ		ΤΕΤΑΡΤΗ		ΠΕΜΠΤΗ	
ΠΡΩΙ	2,8	3,7	2	7	0	1,7	0,2	4,6	2	6,9	1	6,10	0,7	2,6,8
ΑΠΟΓΕΥΜΑ	5	9	6	9	10	6	1	7	3	7	3	7	10	4
ΒΡΑΔΥ	1	10	2	3	0	3,10	0	4		4	1	9	8	6

Πίνακας 3

Έχουμε ότι:

$C_{0Π} = 1, C_{0A} = 1\ ,\ C_{0B} = 1.2$

$C_{1Π} = 1, C_{1A} = 1.01\ ,\ C_{1B} = 1.2$

$C_{2Π} = 1, C_{2A} = 1\ ,\ C_{2B} = 1.1$

$C_{3Π} = 1, C_{3A} = 1.02\ ,\ C_{3B} = 1.2$

$C_{4Π} = 1, C_{4A} = 1.01\ ,\ C_{4B} = 1.2$

$C_{5Π} = 1, C_{5A} = 1.01\ ,\ C_{5B} = 1$

$C_{6Π} = 1, C_{6A} = 1.02\ ,\ C_{6B} = 1.1$

$C_{7Π} = 1, C_{7A} = 1.03\ ,\ C_{7B} = 1$

$C_{8Π} = 1, C_{8A} = 1\ ,\ C_{8B} = 1$

$C_{9Π} = 1, C_{9A} = 1.02\ ,\ C_{9B} = 1.1$

$C_{10Π} = 1, C_{10A} = 1.01\ ,\ C_{10B} = 1.2$

Έτσι το μοντέλο ελαχιστοποίησης θα γίνει:

$$Min \sum_{i=0}^{10} \sum_{j=0}^{41} C_{ij} X_{ij} ,$$

$$X_{ij} \in \{0,1\},$$

$$i \in \{0,10\},$$

$$j \in \{0,41\}$$

Τέλος, μετά από το γενικό μοντέλο πάμε στους περιορισμούς της κάθε εβδομάδας.

4.4 Εβδομαδιαίο μοντέλο

Είναι δεδομένο ότι θα πρέπει να υπάρχει ένας ελάχιστος αριθμός βαρδιων που θα πρέπει κάθε νοσοκόμα να καλύψει μέσα στην εβδομάδα. Δεδομένου ότι οι 11 νοσοκόμες πραγματοποιώντας 4 βάρδιες την εβδομάδα καλύπτουν 44 βάρδιες>42 θεωρούμε ότι ο ελάχιστος αριθμός βαρδιων θα είναι 3. Όμως, θα θεωρήσουμε ότι αν μια νοσοκόμα είναι εφημερευουσα θα πρέπει να δουλέψει για μια βάρδια λιγότερη στη κλινική καθώς το ίδιο θα σύμβεί σε περίπτωση αδειών. Επίσης θεωρούμε ότι αν τη προηγούμενη εβδομάδα κάποιος έκανε λιγότερες από 5 βάρδιες την επόμενη θα κάνει σίγουρα 5 (εξαιρούνται οι περιπτώσεις αδειών – εφημερειών). Επομένως ο ελάχιστος αριθμός βαρδιων θα μπει σαν τελευταίος περιορισμός.

Επιπλέον, σε περιπτώσεις όπως η νοσοκόμα 2 (Βασιλική) ή η νοσοκόμα 0 (Αγάπη) που έχει άδεια και ρεπό πολλές μέρες αυτή την εβδομάδα, δεν εξαλείφονται οι μεταβλητές που αντιπροσωπεύουν τις βάρδιες που θα καλύψει αυτή την εβδομάδα, αλλά μηδενίζονται για χάρη ευελιξίας, με σκοπό το μοντέλο να είναι ευέλικτο ως προς τις αλλαγές κάθε εβδομάδας.

Συνεχίζοντας, έχουμε:

- Η νοσοκόμα (3) Κατερίνα ορίζει το πρόγραμμα της με βάση του άνδρα της. Θα δουλέψει βραδινό 1 και 2 του μήνα επομένως:

$$X_{3\_5} = 1 ,$$
$$X_{3\_11} = 1,$$
$$X_{3\_21} = 1 ,$$
$$X_{3\_27} = 1,$$
$$X_{3\_21} = 1$$

- Η νοσοκόμα (2) Βασιλική έχει ζητήσει άδεια 4-8 (4-7 για την πρώτη εβδομάδα) και ρεπό 2-3. Ακόμα δεν μπορεί να εργαστεί το προηγούμενο βράδυ. Επομένως:

$$\sum_{j=4}^{41} X_{2\_j} = 0$$

Τέλος, θα πρέπει οι ελάχιστες φορές που θα εργαστεί αυτή την εβδομάδα να είναι 7-2(ρεπό)- 4(άδεια)=1. Άρα:

$$\sum_{j=0}^{41} X_{2\_j} \geq 1$$

- Η νοσοκόμα (0) Αγάπη στις 4/12 σε ωράριο 23-7 θα απασχοληθεί σαν εφημερεύουσα αρα δεν μπορεί να εργαστεί 16 ωρες πριν και μετά. Επίσης στις 7/12 έχει ζητήσει γονική αδεια. Άρα δεν μπορεί να εργαστεί το προηγούμενο βράδυ.Επομένως, έχουμε:

$$\sum_{j=16}^{27} X_{0\_j} = 0$$

και $$\sum_{j=34}^{41} X_{0\_j} = 0$$

Τέλος θα πρέπει οι ελάχιστες φορές που θα εργαστεί αυτή την εβδομάδα να είναι 7-2(ρεπό)-1(εφημερία)-1(άδεια)=3. Άρα:

$$\sum_{j=0}^{41} X_{0\_j} \geq 3$$

- Η νοσοκόμα (1) Μαρία ζήτησε 5/12, 6/12 και 7/12 Ρεπό. Θεωρώντας ότι μπορεί τα ρεπό μεσα στην εβδομάδα να είναι και 3 θα δοκιμάσουμε αρχικά να λύσουμε το πρόβλημα με 3 ρεπό. Έτσι Άρα δεν μπορεί να εργαστεί το προηγούμενο βράδυ από τις 5/12. Επομένως:

$$\sum_{j=22}^{41} X_{1\_j} = 0$$

- Η νοσοκόμα (4) Βαλασία ζήτησε 1/12 να έχει Ρεπό, οπότε:

$$\sum_{j=0}^{5} X_{4\_j} = 0$$

- Η νοσοκόμα (5) Αναστασία ζήτησε Άδεια στις 1/12 και 4/12, ενώ στις 2/12 και 3/12 ζήτησε να έχει ρεπό. Επομένως:

$$\sum_{j=0}^{25} X_{5\_j} = 0$$

Τέλος θα πρέπει οι ελάχιστες φορές που θα εργαστεί αυτή την εβδομάδα να είναι 7-2(ρεπό)-2(άδεια)=3. Άρα:

$$\sum_{j=0}^{41} X_{5\_j} \geq 3$$

- Η νοσοκόμα (6) Ευγενία δεν έχει περιορισμό αυτή την εβδομαδα.
- Η νοσοκόμα (7) Χαρούλα ζητάει Πέμπτη και Παρασκευή να μην απασχολείται σε ωράριο 15-23. Άρα:

$$X_{7\_2} = 1\,,$$
$$X_{7\_3} = 1,$$
$$X_{7\_38} = 1\,,$$
$$X_{7\_39} = 1$$

- Η νοσοκόμα (8) Παρθένα δεν έχει περιορισμό αυτή την εβδομαδα.
- Η νοσοκόμα (9) Ελένη ζητάει Δευτέρα και Τρίτη να έχει ρεπό. Άρα δεν μπορεί να εργαστεί το βράδυ της Κυριακής. Επομένως:

$$\sum_{j=16}^{29} X_{9\_j} = 0$$

- Η νοσοκόμα (10) Μαρία Τ εργάστηκε την Πέμπτη 3-11 τη προηγούμενη εβδομαδα έτσι δεν μπορεί να εργαστεί πρωί Παρασκευής. Άρα:

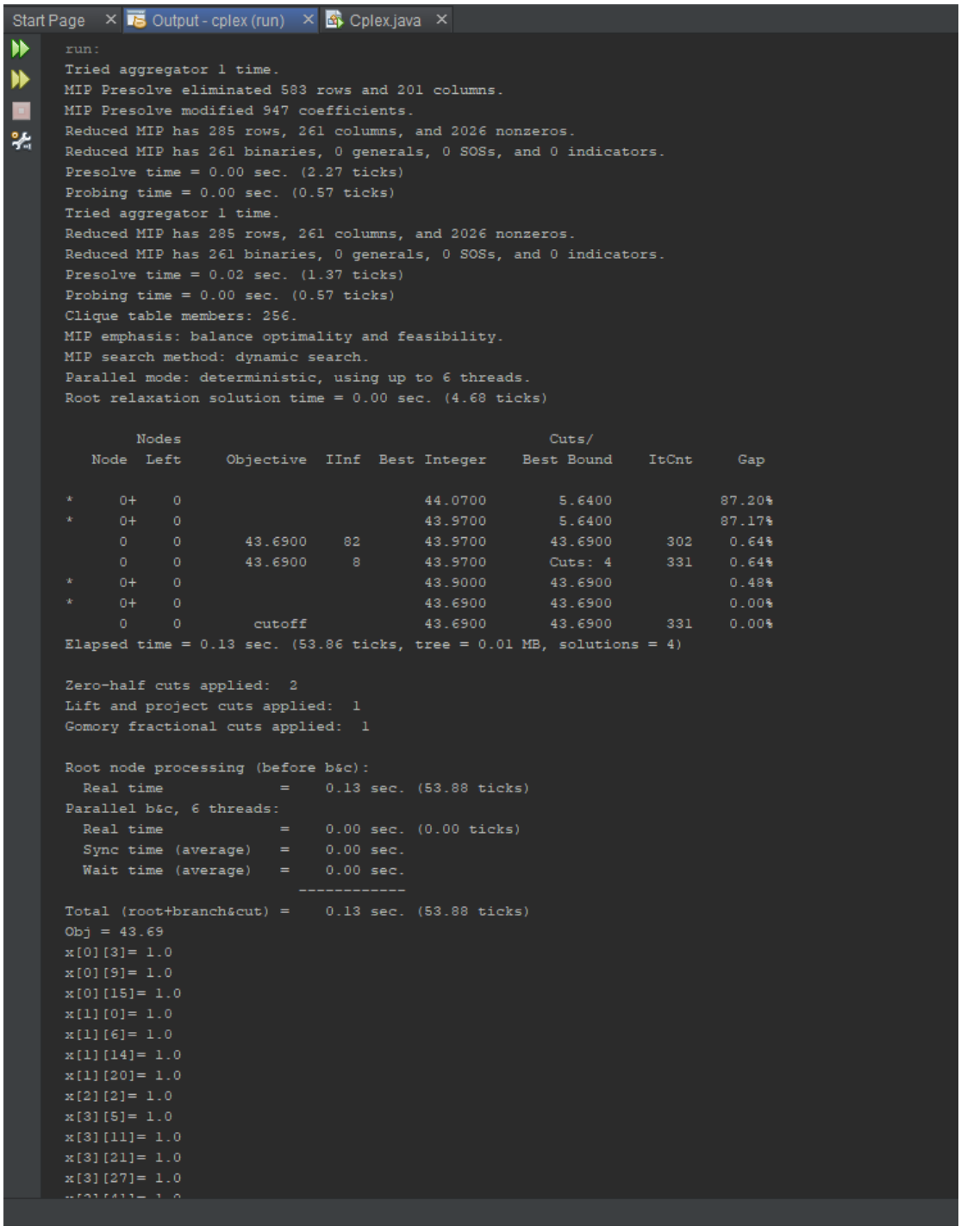
$$X_{10\_0} = 0\,,$$
$$X_{10\_1} = 0$$

- Τέλος, εκτός από τις νοσοκόμες 0 (Αγάπη), 2 (Βασιλική) και 5 (Αναστασία), οι νοσοκόμες θα πρέπει να κάνουν τουλάχιστον 4 βάρδιες την εβδομάδα. Επομένως για κάθε νοσοκόμα $i \rightarrow i \in \{1, 3, 4, 6, 7, 8, 9, 10\}$

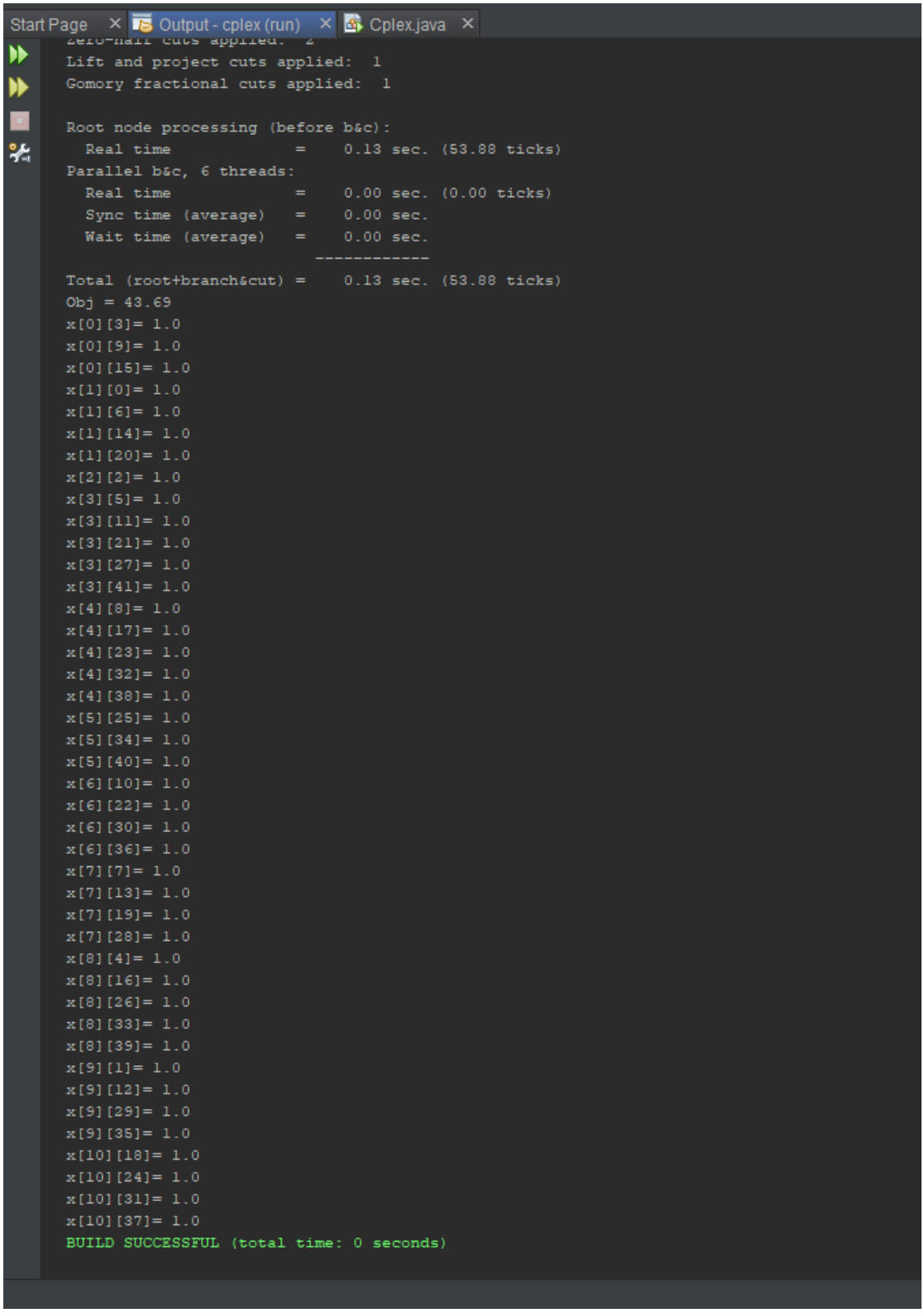
$$\sum_{j=0}^{41} X_{ij} \geq 4,$$

5 Επίλυση

Αναλυτικά ο κώδικας και οι εντολές που χρησιμοποιήθηκαν παρουσιάζονται στο παράρτημα 1. Το compile του προγράμματος έδωσε την παρακάτω λύση (Εικόνα 4, Εικόνα 5):



Εικόνα 4



Εικόνα 5

Παρατηρείται ότι υπάρχουν 4 λύσεις. Το αποτέλεσμα του compile έδωσε βέλτιστη λύση 43.69. Πιο αναλυτικά, το output έδωσε τα παρακάτω αποτελέσματα:

Η βέλτιστη ελάχιστη τιμή που δίνει η αντικειμενική συνάρτηση είναι 43.6900.

Η λύση σε μορφή πίνακα που περιέχει τα ID της κάθε νοσοκόμας είναι η εξής:

ID ΕΡΓΑΖΟΜΕΝΟΥ ΑΝΑ ΒΑΡΔΙΑ														
ΔΕΚΕΜΒΡΙΟΣ														
1Η ΕΒΔΟΜΑΔΑ														
	1		2		3		4		5		6		7	
	ΠΑΡΑΣΚΕΥΗ		ΣΑΒΒΑΤΟ		ΚΥΡΙΑΚΗ		ΔΕΥΤΕΡΑ		ΤΡΙΤΗ		ΤΕΤΑΡΤΗ		ΠΕΜΠΤΗ	
ΠΡΩΙ	1	9	1	7	9	7	10	7	10	5	6	10	6	10
ΑΠΟΓΕΥΜΑ	2	0	4	0	1	0	1	3	8	3	4	8	4	8
ΒΡΑΔΥ	8	3	6	3	8	4	6	4	7	9	5	9	5	3

Πίνακας 6

Σε σχέση με το πραγματικό πρόγραμμα, το οποίο είχε τη μορφή

ID ΕΡΓΑΖΟΜΕΝΟΥ ΑΝΑ ΒΑΡΔΙΑ														
ΔΕΚΕΜΒΡΙΟΣ														
1Η ΕΒΔΟΜΑΔΑ														
	1		2		3		4		5		6		7	
	ΠΑΡΑΣΚΕΥΗ		ΣΑΒΒΑΤΟ		ΚΥΡΙΑΚΗ		ΔΕΥΤΕΡΑ		ΤΡΙΤΗ		ΤΕΤΑΡΤΗ		ΠΕΜΠΤΗ	
ΠΡΩΙ	1	2,7	9		4	8	0,10	9	5	6,7	3,8	5,6	4	8
ΑΠΟΓΕΥΜΑ	0	9	4	10	6	9	3	8	3	8	4	9	9	5
ΒΡΑΔΥ	1	3,8	1	3	4		0,10	4	ΟΓΕ	7	7	6	3	

Πίνακας 7

γίνεται έυκολα αντιληπτό ότι όλες οι βάρδιες καλύφθηκαν σωστά, ενώ φυσικά δεν υπάρχουν νοσοκόμες που καλύπτουν συνεχόμενες βάρδιες. Ακόμα δεν υπάρχει περίπτωση να καλυφθεί βάρδια με περισσότερο από το απαραίτητο προσωπικό όπως βλέπουμε στο πραγματικό πρόγραμμα.

Πιο συγκεκριμένα, ονομαστικά το πρόγραμμα είναι το παρακάτω:

ID ΕΡΓΑΖΟΜΕΝΟΥ ΑΝΑ ΒΑΡΔΙΑ														
ΔΕΚΕΜΒΡΙΟΣ														
1Η ΕΒΔΟΜΑΔΑ														
	1		2		3		4		5		6		7	
	ΠΑΡΑΣΚΕΥΗ		ΣΑΒΒΑΤΟ		ΚΥΡΙΑΚΗ		ΔΕΥΤΕΡΑ		ΤΡΙΤΗ		ΤΕΤΑΡΤΗ		ΠΕΜΠΤΗ	
ΠΡΩΙ	Μαρία	Ελένη	Μαρία	Χαρούλα	Ελένη	Χαρούλα	Μαρία Τ	Χαρούλα	Μαρία Τ	Αναστασία	Ευγενία	Μαρία Τ	Ευγενία	Μαρία Τ
ΑΠΟΓΕΥΜΑ	Βασιλική	Αγάπη	Βαλασία	Αγάπη	Μαρία	Αγάπη	Μαρία	Κατερίνα	Παρθένα	Κατερίνα	Βαλασία	Παρθένα	Βαλασία	Παρθένα
ΒΡΑΔΥ	Παρθένα	Κατερίνα	Ευγενία	Κατερίνα	Παρθένα	Βαλασία	Ευγενία	Βαλασία	Χαρούλα	Ελένη	Αναστασία	Ελένη	Αναστασία	Κατερίνα

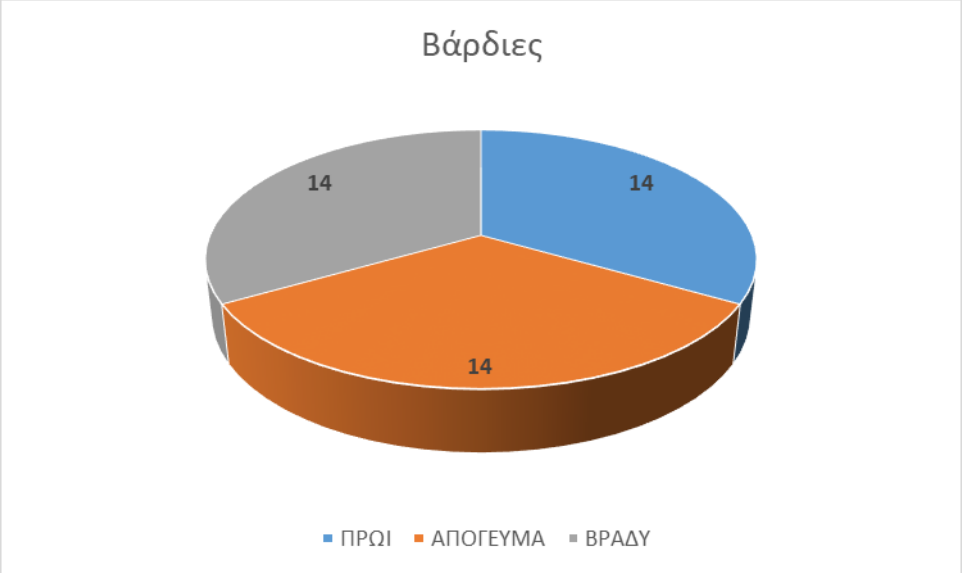
Πίνακας 8

Επιπλέον, ο αριθμός βαρδιών που θα καλύψει η κάθε νοσοκόμα παρουσιάζεται στον παρακάτω πίνακα:

ID	Νοσοκόμες	Αριθμός βαρδιών εβδομάδας
0	Αγάπη (νοσηλεύτρια ΤΕ)	3
1	Μαρία (νοσηλεύτρια ΤΕ)	4
2	Βασιλική (νοσηλεύτρια ΤΕ)	1
3	Κατερίνα (νοσηλεύτρια ΤΕ)	5
4	Βαλασία (νοσηλεύτρια ΤΕ)	5
5	Αναστασία (νοσηλεύτρια ΤΕ)	3
6	Ευγενία (νοσηλεύτρια ΤΕ)	4
7	Χαρούλα (νοσηλεύτρια ΤΕ)	4
8	Παρθένα (νοσηλεύτρια ΔΕ)	5
9	Ελένη (νοσηλεύτρια ΔΕ)	4
10	Μαρία Τ. (νοσηλεύτρια ΔΕ)	4
	Σύνολο	42

Πίνακας 9

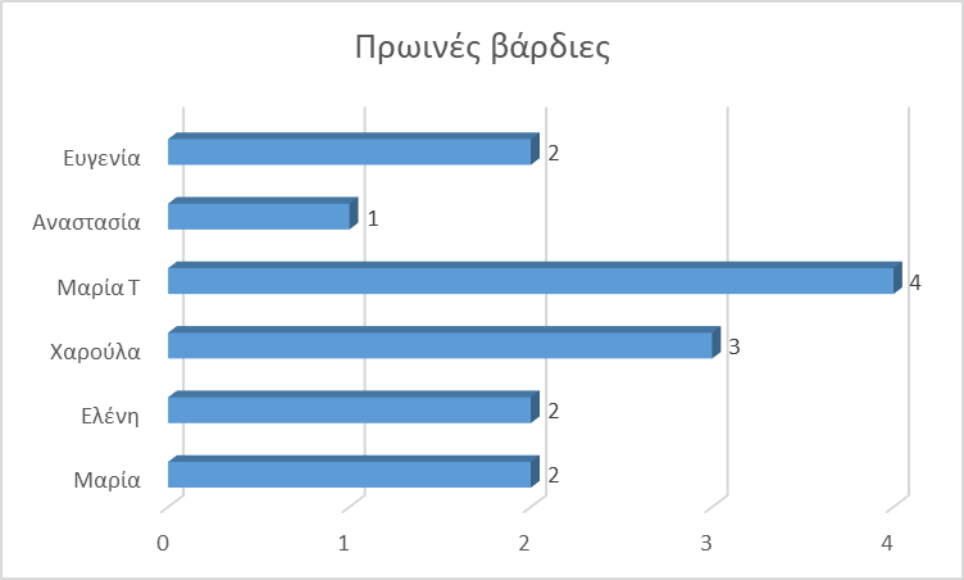
Πιο αναλυτικά, αρχικά παρατηρείται ότι όλες οι βάρδιες έχουν καλυφθεί με συνολικό αριθμό 42 και δεν υπάρχει κάποιο ακάλυπτο 8ωρο.



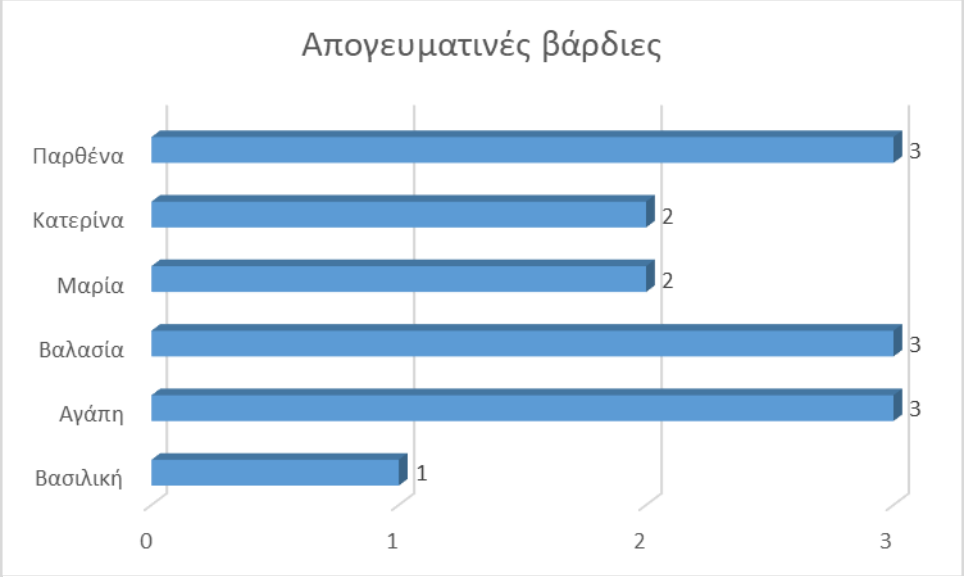
Εικόνα 10

Ακόμα, γίνεται αντιληπτό ότι καμία νοσοκόμα δεν κάνει περισσότερες από 5 βάρδιες μέσα στην εβδομάδα ή λιγότερες από 4 - εκτός βέβαια από αυτές που

έχουν γενική εφημερία ή άδεια όπως η Αγάπη, η Βασιλική και Αναστασία (Εικόνα 9).



Εικόνα 11



Εικόνα 12

Επιπλέον, οι συμβασιούχες καλύπτουν το πολύ ένα βράδυ μέσα την εβδομάδα, ενώ οι υπόλοιπες μέγιστο 2 - εκτός της Κατερίνας (Εικόνα 12). Συνεχίζοντας, παρατηρείται ότι δεν υπάρχει υπάλληλος που να κάνει 2 συνεχόμενες βάρδιες που να μην έχουν χρονική διάρκεια μεταξύ τους τουλάχιστον 16 ώρες από το τέλος της μίας μέχρι την αρχή της επόμενης.

Εικόνα 11

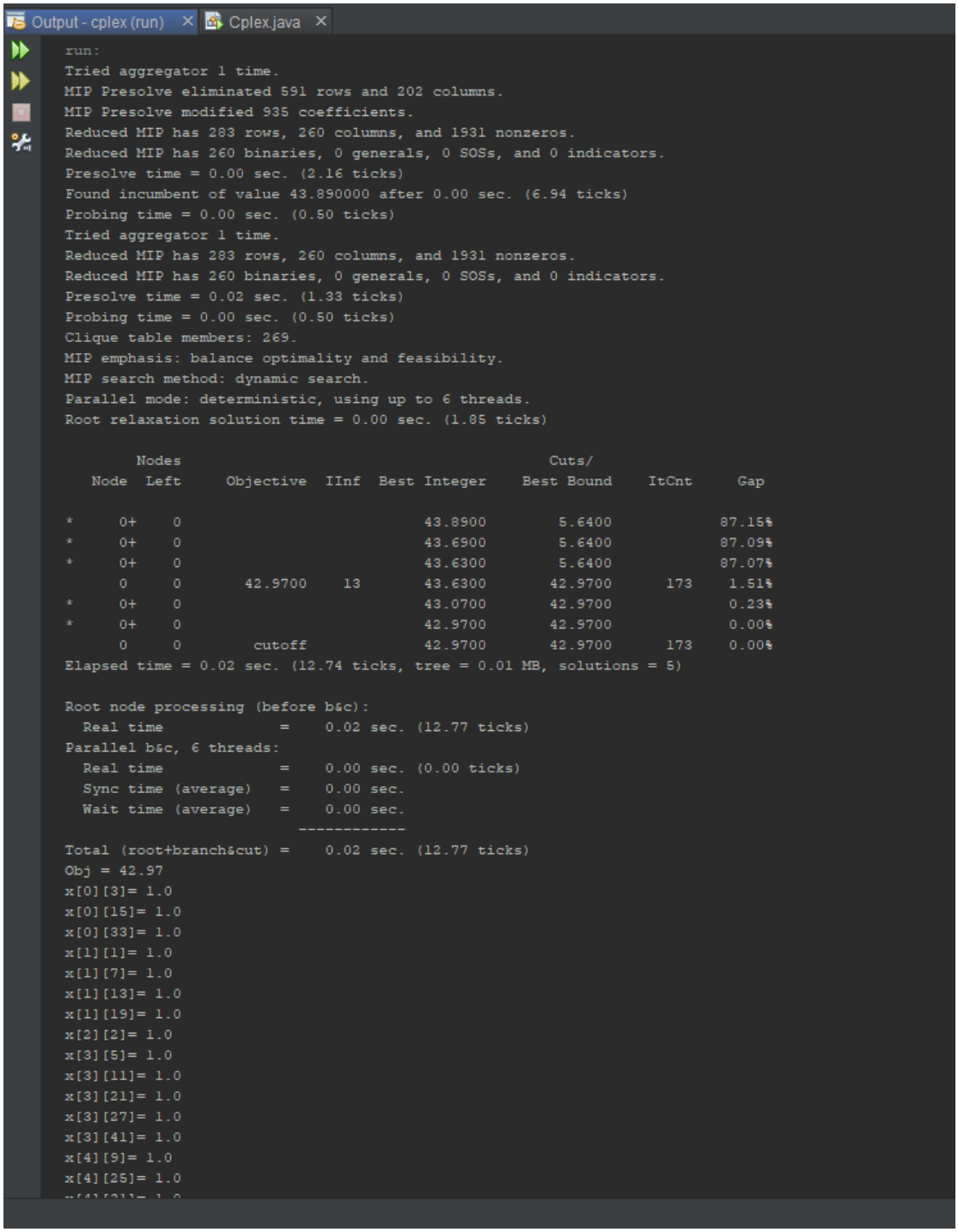
Μία ακόμα συνθήκη που καλύπτεται είναι ότι οι νοσοκόμες ΔΕ δεν δουλεύουν μαζί στο ίδιο 8ωρο και ότι πάντα συμπληρώνονται από νοσοκόμες ΤΕ.
Ακόμα, φαίνεται ότι δεν υπάρχουν περιπτώσεις να δουλεύουν νοσοκόμες σε μέρες ή ωράρια που έχουν ζητήσει να αποφύγουν ή να πάρουν άδεια.
Τέλος, καλύπτονται και οι περιορισμοί που αφορούν τις ζητούμενες άδειες και τα ρεπό από το προσωπικό.

Κλείνοντας, θα πρέπει να γίνει αναφορά και στο feedback που πήρε η συγκεκριμένη λύση από το νοσοκομείο. Μετά από επικοινωνία με το προσωπικό της κλινικής και πιο συγκεκριμένα με τη νοσοκόμα που προσδιόρισε το πρόβλημα και ήρθε σε επικοινωνία με το τμήμα Οργάνωσης και Διοίκησης του Πανεπιστημίου Μακεδονίας, η απάντηση ήταν πως η λύση είναι σωστή και καλύπτει όλες τις ανάγκες του προσωπικού και της κλινικής.

Παρατηρείται, πως όλοι οι αρχικοί περιορισμοί που έχουν τεθεί καλύπτονται και πως το αποτέλεσμα που προκύπτει είναι αποδεκτό σαν λύση και αποδεκτό και από το προσωπικό της κλινικής. Έτσι λοιπόν η λύση θεωρείται αποδεκτή.

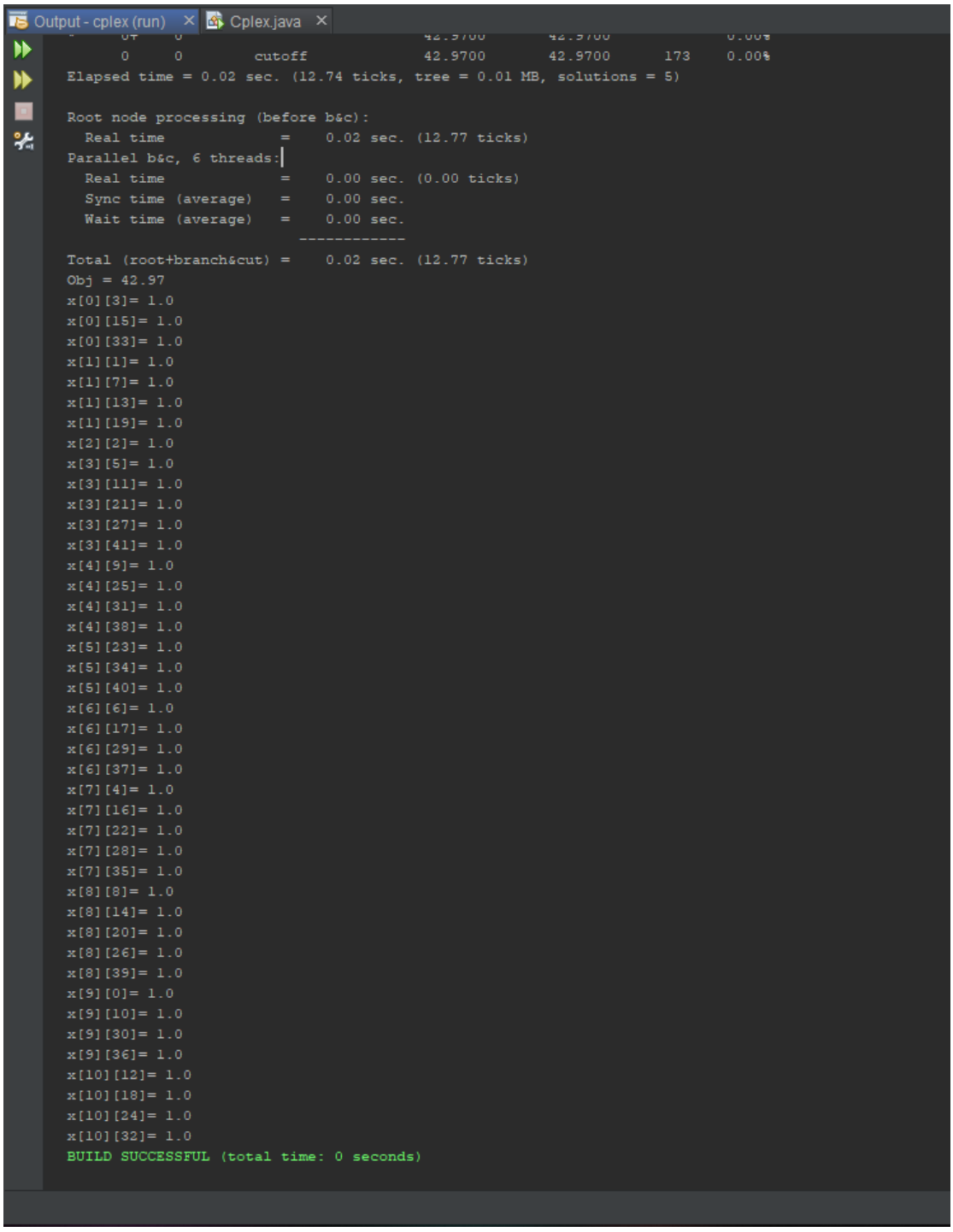
6 Ανάλυση Ευαισθησίας

Σε περίπτωση που χαλαρώσουμε κάποιους περιορισμούς, η λύση πρόκειται να αλλάξει. Ειδικότερα, ένας περιορισμός που θα μπορούσε να χαλαρώσει είναι η αλλαγή του περιορισμού της μέγιστη τιμή των βραδινών βαρδιών που μπορούν να κάνουν οι νοσοκόμες. Αντίστοιχα, το ίδιο θα μπορούσε να θεωρηθεί για τον περιορισμό ως προς τη μέγιστη τιμή για τις απογευματινές βάρδιες.



Εικόνα 12

Αναλυτικά ο κώδικας και οι εντολές που χρησιμοποιήθηκαν παρουσιάζονται στο παράρτημα 2. Η λύση του προβλήματος μετά από το compile του κώδικα είναι η εξής (Εικόνα 14, Εικόνα 15):



Εικόνα 13

Παρατηρούμε ότι υπάρχουν 5 λύσεις. Το αποτέλεσμα του compile μας έδωσε βέλτιστη λύση 42.97. Πιο αναλυτικά, το output έδωσε τα παρακάτω αποτελέσματα:

Η βέλτιστη ελάχιστη τιμή που δίνει η αντικειμενική συνάρτηση είναι 42.9700, ενώ η λύση σε μορφή πίνακα που περιέχει τα ID της κάθε νοσοκόμας φαίνεται στον επόμενο πίνακα (Πίνακας 16).

ID ΕΡΓΑΖΟΜΕΝΟΥ ΑΝΑ ΒΑΡΔΙΑ														
ΔΕΚΕΜΒΡΙΟΣ														
1Η ΕΒΔΟΜΑΔΑ														
	1		2		3		4		5		6		7	
	ΠΑΡΑΣΚΕΥΗ		ΣΑΒΒΑΤΟ		ΚΥΡΙΑΚΗ		ΔΕΥΤΕΡΑ		ΤΡΙΤΗ		ΤΕΤΑΡΤΗ		ΠΕΜΠΤΗ	
ΠΡΩΙ	9	1	6	1	10	1	10	1	10	4	9	4	9	6
ΑΠΟΓΕΥΜΑ	2	0	8	4	8	0	8	3	8	3	10	0	4	8
ΒΡΑΔΥ	7	3	9	3	7	6	7	5	7	6	5	7	5	3

Πίνακας 14

Πιο συγκεκριμένα, ονομαστικά το πρόγραμμα είναι το παρακάτω:

ID ΕΡΓΑΖΟΜΕΝΟΥ ΑΝΑ ΒΑΡΔΙΑ														
ΔΕΚΕΜΒΡΙΟΣ														
1Η ΕΒΔΟΜΑΔΑ														
	1		2		3		4		5		6		7	
	ΠΑΡΑΣΚΕΥΗ		ΣΑΒΒΑΤΟ		ΚΥΡΙΑΚΗ		ΔΕΥΤΕΡΑ		ΤΡΙΤΗ		ΤΕΤΑΡΤΗ		ΠΕΜΠΤΗ	
ΠΡΩΙ	Ελένη	Μαρία	Ευγενία	Μαρία	Μαρία Τ.	Μαρία	Μαρία Τ.	Μαρία	Μαρία Τ.	Βαλασία	Ελένη	Βαλασία	Ελένη	Ευγενία
ΑΠΟΓΕΥΜΑ	Βασιλική	Αγάπη	Παρθένα	Βαλασία	Παρθένα	Αγάπη	Παρθένα	Κατερίνα	Παρθένα	Κατερίνα	Μαρία Τ.	Αγάπη	Βαλασία	Παρθένα
ΒΡΑΔΥ	Χαρούλα	Κατερίνα	Ελένη	Κατερίνα	Χαρούλα	Ευγενία	Χαρούλα	Αναστασία	Χαρούλα	Ευγενία	Αναστασία	Χαρούλα	Αναστασία	Κατερίνα

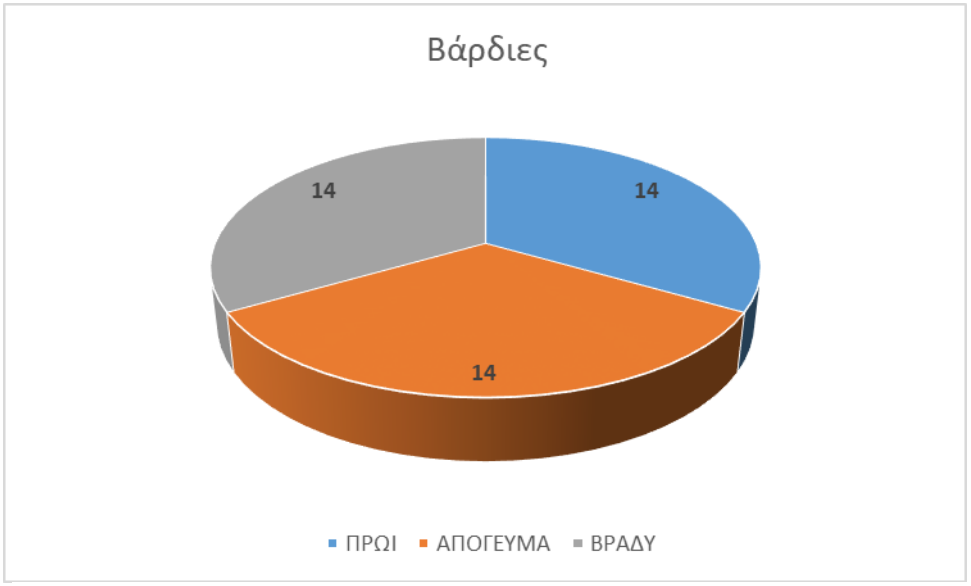
Πίνακας 15

Επιπλέον, ο αριθμός βαρδιών που θα καλύψει η κάθε νοσοκόμα παρουσιάζεται στον παρακάτω πίνακα (Πίνακας 18):

ID	Νοσοκόμες	Αριθμός βαρδιών εβδομάδας
0	Αγάπη (νοσηλεύτρια ΤΕ)	3
1	Μαρία (νοσηλεύτρια ΤΕ)	4
2	Βασιλική (νοσηλεύτρια ΤΕ)	1
3	Κατερίνα (νοσηλεύτρια ΤΕ)	5
4	Βαλασία (νοσηλεύτρια ΤΕ)	4
5	Αναστασία (νοσηλεύτρια ΤΕ)	3
6	Ευγενία (νοσηλεύτρια ΤΕ)	4
7	Χαρούλα (νοσηλεύτρια ΤΕ)	5
8	Παρθένα (νοσηλεύτρια ΔΕ)	5
9	Ελένη (νοσηλεύτρια ΔΕ)	4
10	Μαρία Τ. (νοσηλεύτρια ΔΕ)	4
	Σύνολο	42

Πίνακας 16

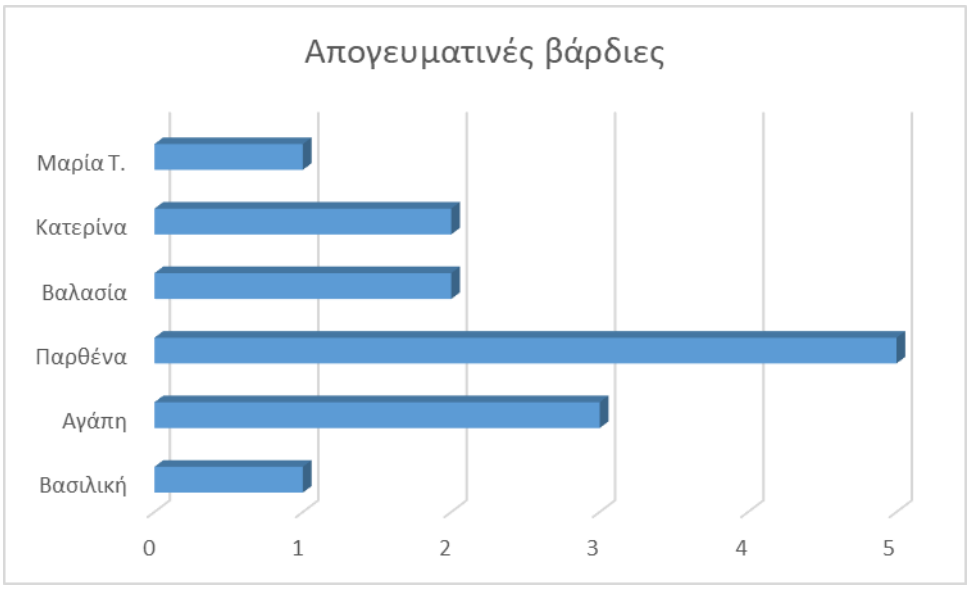
Παρατηρείται ότι έχουμε χαμηλότερη βελτιστη τιμή για την αντικειμενική συνάρτηση. Όμως ταυτόχρονα, εύκολα γίνεται αντιληπτό οτι υπάρχουν περιπτώσεις όπου έχουμε νοσοκόμες που καλύπτουν μόνο βραδυνά ωράρια. Ένα τέτοιο παράδειγμα είναι η νοσοκόμα 7 – Κατερίνα – η οποία είναι όμως συμβασιούχα και δεν επιτρέπεται να καλύψει περισσότερα από ένα βράδυ την εβδομάδα.



Εικόνα 17

Μία άλλη αλλαγή που προέκυψε από την χαλάρωση αυτού του περιορισμού είναι ότι η νοσοκόμα 8 (Παρθένα) καλύπτει 5 απογευματινές βάρδιες κατά τη διάρκεια της εβδομάδας.

Εικόνα 18



Εικόνα 19

Επιπλέον παρατηρείται ότι η νοσοκόμα 8 (Χαρούλα) καλύπτει 8 βραδινά 8ωρα (Εικόνα 22).

Εικόνα 20

Για να προκύψει η συγκεκριμένη βέλτιστη λύση και να υπήρξε η συγκεκριμένη διαφορά σημαίνει ότι οι νοσοκόμες 7 και 8 δεν εργάστηκαν βραδύ και απόγευμα τη προηγούμενη εβδομάδα. Πράγματι, παρατηρείται ότι ο παράγοντας C που

καθορίζει τις απογευματινές βάρδιες της νοσοκόμας 8 και τις βραδινές της νοσοκόμας 8 είναι χαμηλός.

ID ΕΡΓΑΖΟΜΕΝΟΥ ΑΝΑ ΒΑΡΔΙΑ														
ΔΕΚΕΜΒΡΙΟΣ														
1Η ΕΒΔΟΜΑΔΑ														
	1		2		3		4		5		6		7	
	ΠΑΡΑΣΚΕΥΗ		ΣΑΒΒΑΤΟ		ΚΥΡΙΑΚΗ		ΔΕΥΤΕΡΑ		ΤΡΙΤΗ		ΤΕΤΑΡΤΗ		ΠΕΜΠΤΗ	
ΠΡΩΙ	1	9	1	7	9	7	10	7	10	5	6	10	6	10
ΑΠΟΓΕΥΜΑ	2	0	4	0	1	0	1	3	8	3	4	8	4	8
ΒΡΑΔΥ	8	3	6	3	8	4	6	4	7	9	5	9	5	3

Πίνακας 21

Τέλος, στο πίνακα 23 παρατηρούνται οι αλλαγές που προκλήθηκαν στο πρόγραμμα που προέκυψε από την πρώτη λύση μετά τη χαλάρωση των περιορισμών που προαναφέρθηκαν. Με κόκκινο παρουσιάζονται οι διαφοροποιημένες βάρδιες, ενώ οι υπόλοιπες παρέμειναν σταθερές.

7 Προτάσεις και περαιτέρω έρευνα

Έχει αναφερθεί η μεγάλη παρουσία του ακέραιου προγραμματισμού σε διάφορα μοντέλα τη σημερινή εποχή. Το παρόν μοντέλο θα μπορούσε να χρησιμοποιηθεί – εφόσον ο κώδικας γίνει πιο δυναμικός – με περαιτέρω εξέλιξη ως backend κώδικας για software που είναι υπεύθυνα για διαχείριση ανθρώπινου δυναμικού (human resource management scheduling) με σκοπό τη κατανομή ωραρίων των υπαλλήλων μια επιχείρησης.

Επιπρόσθετα, επειδή η γλώσσα που χρησιμοποιήθηκε για τη μοντελοποίηση είναι Java θα ήταν πιθανή η χρησιμοποίηση της σε βάσεις oracle που υποστηρίζουν java procedures είτε ως backend κομμάτι του λογισμικού (όπως προαναφέρθηκε), είτε ως καθαρή predictive ανάλυση δεδομένων.

Μια επιπλέον πιθανή πρόταση είναι να γίνει επέκταση της συγκεκριμένης εφαρμογής και να συνδιαστεί με βασικές τεχνικές μηχανικής μάθησης (machine learning). Διάφορα στοιχεία του αλγορίθμου branch and cut συνεπάγονται με τη λήψη αποφάσεων που επί του παρόντος απευθύνονται ευριστικά (heuristics). Αντ 'αυτού, θα ήταν πολύ ενδιαφέρον να χρησιμοποιηθούν προσεγγίσεις μηχανικής μάθησης (ML), όπως η εποπτευόμενη κατάταξη (supervised ranking) για αποφάσεις που αφορούν συγκεκριμένες εισροές κατά τη διάρκεια των θυγατρικών MIP. Η ιδέα στοχεύει στη βελτίωση της συνολικής απόδοσης των μηχανισμών επίλυσης MIP. Παρόμοιες μελέτες έχουν δείξει ότι υπάρχουν τέτοιες εφαρμογές (Khalil, 2016). Μια πιθανή ιδέα θα ήταν μέσω τεχνικής linear regression να γίνει πρόβλεψη ετήσια για τον αριθμο των βαρδιών κάθε νοσοκόμας. Με τη τεχνική αυτή και χρησιμοποιώντας ένα αρχείο για να αποθηκεύουμε τα δεδομένα κάθε εβδομάδας θα ήταν δυνατό να προστεθεί επιπλέον βάρος ανάλογα με τις βάρδιες που έχει κάνει ο κάθε υπάλληλος με σκοπό το πρόγραμμα να μαθαίνει και πιθανώς να διορθώνει σε βάθος χρόνου το χρονοδιάγραμμα των νοσοκόμων με σκοπό την ισορροπία των βαρδιών που θα καλύψουν σε ένα ολόκληρο έτος.

Ακόμα μια πιθανή πρόταση είναι να χρησιμοποιηθεί ο συνδυασμός CPLEX – Java σε προβλήματα εύρεσης της βέλτιστης διαδρομής σε λογισμικά που χρησιμοποιούν γεωγραφικά δεδομένα. Όπως αναφέρθηκε, τέτοιου είδους προβλήματα μοντελοποιούνται με μικτό ακέραιο προγραμματισμό. Κάθε πιθανή διασταύρωση στο χάρτη θα ήταν και μια νέα μεταβλητή, ενώ ανάλογα με την απόσταση ή την κίνηση θα συμπληρώνονταν τα βάρη C της κάθε μεταβλητής στην πιθανή εξίσωση επίλυσης.

8 Appendix 1

8.1 Κώδικας Java

Ο κώδικας Java επίλυσης είναι:

```java
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package cplex;

import ilog.concert.IloException;
import ilog.concert.IloLinearNumExpr;
import ilog.concert.IloNumVar;
import ilog.cplex.IloCplex;
import java.util.ArrayList;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 *
 * @author Mystakidis Aristeidis
 */
public class Cplex {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) throws IloException {
        // TODO code application logic here
        nursemodel1();
    }

```

```
69      public static void nursemodel1(){
70          try{
71              IloCplex cplex = new IloCplex();
72
73              //nurses
74              int i = 11;
75              //shift
76              int j = 42;
77
78              //variables
79              Double[][] c = new Double[i][j];
80              IloNumVar[][] x = new IloNumVar[i][j];
81
82              for(int k=0; k<i; k++){
83                  for (int q = 0; q<j; q++){
84                      x[k][q] = cplex.boolVar();
85                  }
86              }
87
88              //sunola prwi, mesimeri, vradu
                ArrayList<Integer> prwi = new ArrayList<Integer>();
                ArrayList<Integer> mesimeri = new ArrayList<Integer>();
                ArrayList<Integer> vradu = new ArrayList<Integer>();
92              prwi.add(0);
93              prwi.add(1);
94              prwi.add(6);
95              prwi.add(7);
96              prwi.add(12);
97              prwi.add(13);
98              prwi.add(18);
99              prwi.add(19);
100             prwi.add(24);
101             prwi.add(25);
102             prwi.add(30);
103             prwi.add(31);
104             prwi.add(36);
105             prwi.add(37);
106
                for(int k : prwi){
108                 mesimeri.add(k + 2);
109                 vradu.add(k + 4);
110             }
111
112             //defining the weights
113             for(int p=0; p<i; p++){
114                 for(int k : prwi){
115                     c[p][k] = 1.0;
116                 }
117             }
```

```
117                 }
                    for(int k : mesimeri){
119                     c[0][k] = 1.0;
120                     c[1][k] = 1.01;
121                     c[2][k] = 1.0;
122                     c[3][k] = 1.02;
123                     c[4][k] = 1.01;
124                     c[5][k] = 1.01;
125                     c[6][k] = 1.02;
126                     c[7][k] = 1.03;
127                     c[8][k] = 1.0;
128                     c[9][k] = 1.02;
129                     c[10][k] = 1.01;
130                 }
                    for(int k : vradu){
132                     c[0][k] = 1.2;
133                     c[1][k] = 1.2;
134                     c[2][k] = 1.1;
135                     c[3][k] = 1.2;
136                     c[4][k] = 1.2;
137                     c[5][k] = 1.0;
138                     c[6][k] = 1.1;
139                     c[7][k] = 1.0;
140                     c[8][k] = 1.1;
141                     c[9][k] = 1.1;
142                     c[10][k] = 1.2;
143                 }
144
145
146                 //expressions
147                 IloLinearNumExpr objective = cplex.linearNumExpr();
148                 for (int k = 0; k<i; k++){
149                     for (int p = 0; p<j; p++){
150                         objective.addTerm(c[k][p], x[k][p]);
151                     }
152                 }
153
154                 //define objectve
155                 cplex.addMinimize(objective);
156
157                 //define constrains
158                 //basic
159                 for(int k=0; k<i; k++){
160                     IloLinearNumExpr expr = cplex.linearNumExpr();
161                     for (int p=0; p<j; p++){
162                             expr.addTerm(1.0, x[k][p]);
163                     }
164                     cplex.addLe(expr, 5.0);
165                 }
```

```
165                 }
166                 for(int k=0; k<i; k++){
                        if (k==2){
168                         IloLinearNumExpr expr = cplex.linearNumExpr();
169                         for (int p=0; p<j; p++){
170                                 expr.addTerm(1.0, x[k][p]);
171                         }
172                         cplex.addGe(expr, 1.0);
173                     }
174                     else if (k==0 || k==5){
175                         IloLinearNumExpr expr = cplex.linearNumExpr();
176                         for (int p=0; p<j; p++){
177                                 expr.addTerm(1.0, x[k][p]);
178                         }
179                         cplex.addGe(expr, 3.0);
180                     }else{
181                         IloLinearNumExpr expr = cplex.linearNumExpr();
182                         for (int p=0; p<j; p++){
183                                 expr.addTerm(1.0, x[k][p]);
184                         }
185                         cplex.addGe(expr, 4.0);
186                     }
187                 }
188
189                 for (int p = 0; p<j; p++){
190                     IloLinearNumExpr expr = cplex.linearNumExpr();
191                     expr.addTerm(1.0, x[0][p]);
192                     expr.addTerm(1.0, x[1][p]);
193                     expr.addTerm(1.0, x[2][p]);
194                     expr.addTerm(1.0, x[3][p]);
195                     expr.addTerm(1.0, x[4][p]);
196                     expr.addTerm(1.0, x[5][p]);
197                     expr.addTerm(1.0, x[6][p]);
198                     expr.addTerm(1.0, x[7][p]);
199                     expr.addTerm(1.0, x[8][p]);
200                     expr.addTerm(1.0, x[9][p]);
201                     expr.addTerm(1.0, x[10][p]);
202                     cplex.addEq(expr, 1.0);
203                 }
204                 //DE nurses not without TE
205                 for (int p = 0; p<j-1; p++){
206                     IloLinearNumExpr expr = cplex.linearNumExpr();
207                     expr.addTerm(1.0, x[8][p]);
208                     expr.addTerm(1.0, x[9][p+1]);
209                     cplex.addLe(expr, 1.1);
210                 }
211                 for (int p = 0; p<j-1; p++){
212                     IloLinearNumExpr expr = cplex.linearNumExpr();
213                     expr.addTerm(1.0, x[9][p]);
```

```
                }
                for (int p = 0; p<j-1; p++){
                    IloLinearNumExpr expr = cplex.linearNumExpr();
                    expr.addTerm(1.0, x[9][p]);
                    expr.addTerm(1.0, x[8][p+1]);
                    cplex.addLe(expr, 1.1);
                }
                for (int p = 0; p<j-1; p++){
                    IloLinearNumExpr expr = cplex.linearNumExpr();
                    expr.addTerm(1.0, x[8][p]);
                    expr.addTerm(1.0, x[10][p+1]);
                    cplex.addLe(expr, 1.1);
                }
                for (int p = 0; p<j-1; p++){
                    IloLinearNumExpr expr = cplex.linearNumExpr();
                    expr.addTerm(1.0, x[10][p]);
                    expr.addTerm(1.0, x[8][p+1]);
                    cplex.addLe(expr, 1.1);
                }
                for (int p = 0; p<j-1; p++){
                    IloLinearNumExpr expr = cplex.linearNumExpr();
                    expr.addTerm(1.0, x[9][p]);
                    expr.addTerm(1.0, x[10][p+1]);
                    cplex.addLe(expr, 1.1);
                }
                for (int p = 0; p<j-1; p++){
                    IloLinearNumExpr expr = cplex.linearNumExpr();
                    expr.addTerm(1.0, x[10][p]);
                    expr.addTerm(1.0, x[9][p+1]);
                    cplex.addLe(expr, 1.1);
                }
                //sumvasiouxes max mia vradia tin evdomada..
//                upoloipes max 2 vradies ektos apo katerina
                for (int p=0; p<i; p++){
                    IloLinearNumExpr expr = cplex.linearNumExpr();
                    if(p==7||p==10){
                        for(int k = 0; k<vradu.size(); k++){
                            expr.addTerm(1.0, x[p][vradu.get(k)]);
                        }
                        cplex.addLe(expr, 1.0);
                    }else if(p==3){
                        for(int k = 0; k<vradu.size(); k++){
                            expr.addTerm(1.0, x[p][vradu.get(k)]);
                        }
                        cplex.addLe(expr, 3.0);
                    }
                    else{
                        for(int k = 0; k<vradu.size(); k++){
                            expr.addTerm(1.0, x[p][vradu.get(k)]);
```

```
                }
                else{
                    for(int k = 0; k<vradu.size(); k++){
                        expr.addTerm(1.0, x[p][vradu.get(k)]);
                    }
                    cplex.addLe(expr, 2.0);
                }
            }
            //max 3 apogeumata tin evdomada..
            for (int p=0; p<i; p++){
                IloLinearNumExpr expr = cplex.linearNumExpr();
                    for(int k = 0; k<vradu.size(); k++){
                        expr.addTerm(1.0, x[p][mesimeri.get(k)]);
                    }
                    cplex.addLe(expr, 3.0);
            }
            //oxi parapanw apo mia vardies anamesa se 24 wres
            for (int k = 0; k<i; k++){
                for (int p = 0; p<j-5; p++){
                    cplex.addLe(cplex.sum(cplex.prod(1, x[k][p]),
                            cplex.prod(1, x[k][p+1]),
                            cplex.prod(1, x[k][p+2]),
                            cplex.prod(1, x[k][p+3]),
                            cplex.prod(1, x[k][p+4]),
                            cplex.prod(1, x[k][p+5])), 1);
                }
            }

            //week constrains
            //katerina
            cplex.addEq(x[3][5], 1.0);
            cplex.addEq(x[3][11], 1.0);
            cplex.addEq(x[3][21], 1.0);
            cplex.addEq(x[3][27], 1.0);
            cplex.addEq(x[3][41], 1.0);

            //vasiliki
            for (int p = 4; p<j; p++){
                cplex.addEq(x[2][p], 0.0);
            }


            //agapi
            for (int p = 18; p<28; p++){
                cplex.addEq(x[0][p], 0.0);
            }
            for (int p = 34; p<j; p++){
                cplex.addEq(x[0][p], 0.0);
            }
```

```
282
283             //week constrains
284             //katerina
285             cplex.addEq(x[3][5], 1.0);
286             cplex.addEq(x[3][11], 1.0);
287             cplex.addEq(x[3][21], 1.0);
288             cplex.addEq(x[3][27], 1.0);
289             cplex.addEq(x[3][41], 1.0);
290
291             //vasiliki
292             for (int p = 4; p<j; p++){
293                 cplex.addEq(x[2][p], 0.0);
294             }
295
296             //agapi
297             for (int p = 18; p<28; p++){
298                 cplex.addEq(x[0][p], 0.0);
299             }
300             for (int p = 34; p<j; p++){
301                 cplex.addEq(x[0][p], 0.0);
302             }
303
304             //maria gk
305             for (int p = 22; p<j; p++){
306                 cplex.addEq(x[1][p], 0.0);
307             }
308
309             //valasia
310             for (int p = 0; p<5; p++){
311                 cplex.addEq(x[4][p], 0.0);
312             }
313
314             //anastasia
315             for (int p = 0; p<24; p++){
316                 cplex.addEq(x[5][p], 0.0);
317             }
318
319             //evgenia
320             //no constrains for the 1st week
321
322             //xaroula
323             cplex.addEq(x[7][2], 0.0);
324             cplex.addEq(x[7][3], 0.0);
325             cplex.addEq(x[7][38], 0.0);
326             cplex.addEq(x[7][39], 0.0);
327
328             //parthena
329             //no constrains for the 1st week
330
```

```
312                 }
313
314                 //anastasia
315                 for (int p = 0; p<24; p++){
                        cplex.addEq(x[5][p], 0.0);
317                 }
318
319                 //evgenia
320                 //no constrains for the lst week
321
322                 //xaroula
323                 cplex.addEq(x[7][2], 0.0);
324                 cplex.addEq(x[7][3], 0.0);
325                 cplex.addEq(x[7][38], 0.0);
326                 cplex.addEq(x[7][39], 0.0);
327
328                 //parthena
329                 //no constrains for the lst week
330
331                 //eleni
332                 for (int p = 16; p<29; p++){
333                     cplex.addEq(x[9][p], 0.0);
334                 }
335
336                 //mariaT
337                 cplex.addEq(x[10][0], 0.0);
338                 cplex.addEq(x[10][1], 0.0);
339
340                 //solve
341                 if (cplex.solve()){
342                     System.out.println("Obj = " + cplex.getObjValue());
343                     for (int k = 0; k<i; k++){
344                         for (int p = 0; p<j; p++){
345                             if (cplex.getValue(x[k][p])==1){
346                                 System.out.println("x["+k+"]["+p+"]" +"= " +
347                                         cplex.getValue(x[k][p]));
348                             }
349                         }
350                     }
351                 }
352                 else{
353                     System.out.println("Model not solved!");
354                 }
355             }
356             catch(IloException exc){
                    exc.printStackTrace();
358             }
359         }
360
```

9 Appendix 2

9.1 Κώδικας Java

Ο κώδικας Java είναι:

```
69      public static void nursemodel1(){
70          try{
71              IloCplex cplex = new IloCplex();
72
73              //nurses
74              int i = 11;
75              //shift
76              int j = 42;
77
78              //variables
79              Double[][] c = new Double[i][j];
80              IloNumVar[][] x = new IloNumVar[i][j];
81
82              for(int k=0; k<i; k++){
83                  for (int q = 0; q<j; q++){
84                      x[k][q] = cplex.boolVar();
85                  }
86              }
87
88              //sunola prwi, mesimeri, vradu
89              ArrayList<Integer> prwi = new ArrayList<Integer>();
90              ArrayList<Integer> mesimeri = new ArrayList<Integer>();
91              ArrayList<Integer> vradu = new ArrayList<Integer>();
92              prwi.add(0);
93              prwi.add(1);
94              prwi.add(6);
95              prwi.add(7);
96              prwi.add(12);
97              prwi.add(13);
98              prwi.add(18);
99              prwi.add(19);
100             prwi.add(24);
101             prwi.add(25);
102             prwi.add(30);
103             prwi.add(31);
104             prwi.add(36);
105             prwi.add(37);
106
107             for(int k : prwi){
108                 mesimeri.add(k + 2);
109                 vradu.add(k + 4);
110             }
111
112             //defining the weights
113             for(int p=0; p<i; p++){
114                 for(int k : prwi){
115                     c[p][k] = 1.0;
116                 }
```

```
112                 //defining the weights
113                 for(int p=0; p<i; p++){
114                     for(int k : prwi){
115                         c[p][k] = 1.0;
116                     }
117                 }
                    for(int k : mesimeri){
119                     c[0][k] = 1.0;
120                     c[1][k] = 1.01;
121                     c[2][k] = 1.0;
122                     c[3][k] = 1.02;
123                     c[4][k] = 1.01;
124                     c[5][k] = 1.01;
125                     c[6][k] = 1.02;
126                     c[7][k] = 1.03;
127                     c[8][k] = 1.0;
128                     c[9][k] = 1.02;
129                     c[10][k] = 1.01;
130                 }
                    for(int k : vradu){
132                     c[0][k] = 1.2;
133                     c[1][k] = 1.2;
134                     c[2][k] = 1.1;
135                     c[3][k] = 1.2;
136                     c[4][k] = 1.2;
137                     c[5][k] = 1.0;
138                     c[6][k] = 1.1;
139                     c[7][k] = 1.0;
140                     c[8][k] = 1.1;
141                     c[9][k] = 1.1;
142                     c[10][k] = 1.2;
143                 }
144
145
146                 //expressions
147                 IloLinearNumExpr objective = cplex.linearNumExpr();
148                 for (int k = 0; k<i; k++){
149                     for (int p = 0; p<j; p++){
150                         objective.addTerm(c[k][p], x[k][p]);
151                     }
152                 }
153
154                 //define objectve
155                 cplex.addMinimize(objective);
156
157                 //define constrains
158                 //basic
159                 for(int k=0; k<i; k++){
```

```
            //define constrains
            //basic
            for(int k=0; k<i; k++){
                IloLinearNumExpr expr = cplex.linearNumExpr();
                for (int p=0; p<j; p++){
                        expr.addTerm(1.0, x[k][p]);
                }
                cplex.addLe(expr, 5.0);
            }
            for(int k=0; k<i; k++){
                if (k==2){
                    IloLinearNumExpr expr = cplex.linearNumExpr();
                    for (int p=0; p<j; p++){
                            expr.addTerm(1.0, x[k][p]);
                    }
                    cplex.addGe(expr, 1.0);
                }
                else if (k==0 || k==5){
                    IloLinearNumExpr expr = cplex.linearNumExpr();
                    for (int p=0; p<j; p++){
                            expr.addTerm(1.0, x[k][p]);
                    }
                    cplex.addGe(expr, 3.0);
                }else{
                    IloLinearNumExpr expr = cplex.linearNumExpr();
                    for (int p=0; p<j; p++){
                            expr.addTerm(1.0, x[k][p]);
                    }
                    cplex.addGe(expr, 4.0);
                }
            }

            for (int p = 0; p<j; p++){
                IloLinearNumExpr expr = cplex.linearNumExpr();
                expr.addTerm(1.0, x[0][p]);
                expr.addTerm(1.0, x[1][p]);
                expr.addTerm(1.0, x[2][p]);
                expr.addTerm(1.0, x[3][p]);
                expr.addTerm(1.0, x[4][p]);
                expr.addTerm(1.0, x[5][p]);
                expr.addTerm(1.0, x[6][p]);
                expr.addTerm(1.0, x[7][p]);
                expr.addTerm(1.0, x[8][p]);
                expr.addTerm(1.0, x[9][p]);
                expr.addTerm(1.0, x[10][p]);
                cplex.addEq(expr, 1.0);
            }
            //DE nurses not without TE
```

```
204                 //DE nurses not without TE
205                 for (int p = 0; p<j-1; p++){
206                     IloLinearNumExpr expr = cplex.linearNumExpr();
207                     expr.addTerm(1.0, x[8][p]);
208                     expr.addTerm(1.0, x[9][p+1]);
209                     cplex.addLe(expr, 1.1);
210                 }
211                 for (int p = 0; p<j-1; p++){
212                     IloLinearNumExpr expr = cplex.linearNumExpr();
213                     expr.addTerm(1.0, x[9][p]);
214                     expr.addTerm(1.0, x[8][p+1]);
215                     cplex.addLe(expr, 1.1);
216                 }
217                 for (int p = 0; p<j-1; p++){
218                     IloLinearNumExpr expr = cplex.linearNumExpr();
219                     expr.addTerm(1.0, x[8][p]);
220                     expr.addTerm(1.0, x[10][p+1]);
221                     cplex.addLe(expr, 1.1);
222                 }
223                 for (int p = 0; p<j-1; p++){
224                     IloLinearNumExpr expr = cplex.linearNumExpr();
225                     expr.addTerm(1.0, x[10][p]);
226                     expr.addTerm(1.0, x[8][p+1]);
227                     cplex.addLe(expr, 1.1);
228                 }
229                 for (int p = 0; p<j-1; p++){
230                     IloLinearNumExpr expr = cplex.linearNumExpr();
231                     expr.addTerm(1.0, x[9][p]);
232                     expr.addTerm(1.0, x[10][p+1]);
233                     cplex.addLe(expr, 1.1);
234                 }
235                 for (int p = 0; p<j-1; p++){
236                     IloLinearNumExpr expr = cplex.linearNumExpr();
237                     expr.addTerm(1.0, x[10][p]);
238                     expr.addTerm(1.0, x[9][p+1]);
239                     cplex.addLe(expr, 1.1);
240                 }
241                 //sumvasiouxes mia vradia tin evdomada..
242                 for(int k = 0; k<vradu.size(); k++){
243                     IloLinearNumExpr expr = cplex.linearNumExpr();
244                     expr.addTerm(1.0, x[7][vradu.get(k)]);
245                     cplex.addLe(expr, 1.1);
246                 }
247                 for(int k = 0; k<vradu.size(); k++){
248                     IloLinearNumExpr expr = cplex.linearNumExpr();
249                     expr.addTerm(1.0, x[10][vradu.get(k)]);
250                     cplex.addLe(expr, 1.1);
251                 }
252
```

```
253                 for (int k = 0; k<i; k++){
254                     for (int p = 0; p<j-5; p++){
255                         cplex.addLe(cplex.sum(cplex.prod(1, x[k][p]),
256                                 cplex.prod(1, x[k][p+1]),
257                                 cplex.prod(1, x[k][p+2]),
258                                 cplex.prod(1, x[k][p+3]),
259                                 cplex.prod(1, x[k][p+4]),
260                                 cplex.prod(1, x[k][p+5])), 1);
261                     }
262                 }
263 
264                 //week constrains
265                 //katerina
266                 cplex.addEq(x[3][5], 1.0);
267                 cplex.addEq(x[3][11], 1.0);
268                 cplex.addEq(x[3][21], 1.0);
269                 cplex.addEq(x[3][27], 1.0);
270                 cplex.addEq(x[3][41], 1.0);
271 
272                 //vasiliki
273                 for (int p = 4; p<j; p++){
274                     cplex.addEq(x[2][p], 0.0);
275                 }
276 
277 
278                 //agapi
279                 for (int p = 16; p<27; p++){
280                     cplex.addEq(x[0][p], 0.0);
281                 }
282                 for (int p = 34; p<j; p++){
283                     cplex.addEq(x[0][p], 0.0);
284                 }
285 
286                 //maria gk
287                 for (int p = 22; p<j; p++){
288                     cplex.addEq(x[1][p], 0.0);
289                 }
290 
291                 //valasia
292                 for (int p = 0; p<5; p++){
293                     cplex.addEq(x[4][p], 0.0);
294                 }
295 
296                 //anastasia
297                 for (int p = 0; p<23; p++){
298                     cplex.addEq(x[5][p], 0.0);
299                 }
300 
301                 //evgenia
```

```
            //evgenia
            //no constrains for the 1st week

            //xaroula
            cplex.addEq(x[7][2], 0.0);
            cplex.addEq(x[7][3], 0.0);
            cplex.addEq(x[7][38], 0.0);
            cplex.addEq(x[7][39], 0.0);

            //parthena
            //no constrains for the 1st week

            //eleni
            for (int p = 16; p<29; p++){
                cplex.addEq(x[9][p], 0.0);
            }

            //mariaT
            cplex.addEq(x[10][0], 0.0);
            cplex.addEq(x[10][1], 0.0);

            //solve
            if (cplex.solve()){
                System.out.println("Obj = " + cplex.getObjValue());
                for (int k = 0; k<i; k++){
                    for (int p = 0; p<j; p++){
                        if (cplex.getValue(x[k][p])==1){
                            System.out.println("x["+k+"]["+p+"]" +"= " +
                                    cplex.getValue(x[k][p]));
                        }

                    }
                }
            }
            else{
                System.out.println("Model not solved!");
            }
        }
        catch(IloException exc){
            exc.printStackTrace();
        }
    }

```

```java
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package cplex;

import ilog.concert.IloException;
import ilog.concert.IloLinearNumExpr;
import ilog.concert.IloNumVar;
import ilog.cplex.IloCplex;
import java.util.ArrayList;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 *
 * @author Mystakidis Aristeidis
 */
public class Cplex {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) throws IloException {
        // TODO code application logic here
        nursemodel1();
    }
```

10 Βιβλιογραφία

Azmat S. C., Hürlimann T. & Widmer M. (2004) Mixed Integer Programming to Schedule a Single-Shift Workforce under Annualized Hours. University of Fribourg, Switzerland. Retrieved from

https://link.springer.com/article/10.1023/B:ANOR.0000019105.54898.a4

Balas, E., Ceria, E., Cornuejols, G., & Natraj, N., (1996) *Gomory cuts revisited*. Retrieved from

http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=F10FEB49DB023C325B09544E9A39A721?doi=10.1.1.52.8589&rep=rep1&type=pdf&fbclid=IwAR0QtbzcC5P97eBGG5u-hu-VBH0_HpbtPRdcjx168qItOpaaai_uLFITBaM

Billionnet, A. (1999). Integer Programming to Schedule a Hierarchical Workforce with Variable Demands. European Journal of Operational Research. Retrieved from

https://www.sciencedirect.com/science/article/abs/pii/S0377221798001829

Burke, E.K., Curtois, T., Rong, Q., & Berghe, G.V. (2010) *A Scatter Search Approach to the Nurse Rostering Problem.* Retrieved from

https://www.researchgate.net/publication/255591722_A_Scatter_Search_Approach_to_the_Nurse_Rostering_Problem?fbclid=IwAR2xtvIorbv-O9YrXBDndbzyNgFhWgi_OIKCDAA6Rui8EGHRVDUFsou_Z_s

Γεωργίου, Α. & Οικονόμου, Γ. (2011) *Επιχειρησιακή Έρευνα για τη Λήψη Διοικητικών Αποφάσεων.* Αθήνα**:** Γ. Μπένου.

Corominas A., García A.L., & Pastor R., (2002) Planning production and working time within an annualized hours scheme framework. Annals of Operations Research. Retrieved from

https://www.researchgate.net/publication/225692411_Planning_production_and_working_time_within_an_annualised_hours_scheme_framework

Chen, DS., Batson, R., & Dang Y. (2010) *Applied integer programming: modeling and solution.* Hoboken, N.J.: John Wiley & Sons.

Cook W. J., Cunningham W. H., Pulleyblank W. R., Schrijver A., (1998). Combinatorial Optimization, Canada, John Wiley & Sons.

Dantzig, G., Fulkerson, R. & Johnson, S. (1954, Nov 1) Solution of a Large-Scale Traveling-Salesman Problem. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.468.6812&rep=rep1&type=pdf&fbclid=IwAR2NNXnLnu-BqWqDgSnDAXdhXAOBoELRhKO-DhZ9JB5EmjU159CXVU4uMyI

Δήλια, Κ., (2011) *Ακέραιος Προγραμματισμός* (Μεταπτυχιακή διατριβή, Πανεπιστήμιο Πειραιώς). Ανακτήθηκε από

http://dione.lib.unipi.gr/xmlui/bitstream/handle/unipi/4834/Dilia.pdf?sequence=2&isAllowed=y&fbclid=IwAR3hLpmwUPNwlyODhIpy-dlikJdtNIDM3wJCjTBfdggg6A9ptsRVOmXnbOA .

Ellis S., Stredwick J. (1998). Flexible working practices: techniques and innovations. London: Institute of Personnel and Development.

Jaumard B., Semet F., Vovor T. (16 May 1996) *A generalized linear programming model for nurse scheduling, Montréal, European Journal of Operational Research Volume 107, Issue 1, 16 May 1998, Pages 1-18.* Retrieved from

https://www.sciencedirect.com/science/article/abs/pii/S0377221797003305#aep-abstract-id4

Jeroslow, R. (1987) Representability in mixed integer programming, North-Holland, Amsterdam, Discrete Applied Mathematics, 17, 223–243. Retrieved from

https://core.ac.uk/download/pdf/81196027.pdf

Jeroslow, R. (1989) Logic-Based Decision Support: Mixed Integer Model Formulation, North-Holland, Amsterdam, Annals of Discrete Mathematics, Vol. 40. Retrieved from

https://epdf.pub/queue/logic-based-decision-support-mixed-integer-model-formulation-annals-of-discrete-c6eae6c991a504a81b245f881b3abdef20915.html

Khalil E. B. (2016) Machine Learning for Integer Programming. Georgia Institute of Technology: IJCAI. Retrieved from:

https://www.ijcai.org/Proceedings/16/Papers/576.pdf

Κιόχος, Π., Θάνος, Γ., Σαλαμούρης, Δ. & Κιόχος, Α. (2002) *Επιχειρησιακή έρευνα : μέθοδοι και τεχνικές λήψης επιχειρηματικών αποφάσεων.* Αθήνα: Σύγχρονη Εκδοτική.

Land, A. H., Doig A. G. (July 1960). An automatic method of solving discrete programming problems. Econometrica 28, London School of Economics and Political Science

Retrieved from

http://jmvidal.cse.sc.edu/library/land60a.pdf

Meyer, R.R. (August 1975) Integer and mixed-integer programming models: general properties. Journal of Optimization Theory and Applications, 16, 191–206. Retrieved from

https://link.springer.com/article/10.1007/BF01262932

Παπαδόπουλος, Α. (2013). *Ακέραιος Προγραμματισμός Μέθοδοι Και Εφαρμογές (*διπλωματική εργασία*).* Ανακτήθηκε από

http://dspace.lib.ntua.gr/dspace2/bitstream/handle/123456789/8452/papadopoulosa_simplex.pdf?sequence=3

Padberg, M., & Rijal, M.P., (1996) *Location, scheduling, design, and integer programming.* Boston/London/ Dordrecht: Kluwer academic publishers.

Padberg, M. & Rinaldi, G. (1991, March) *A Branch-and-Cut Algorithm for the Resolution of Large-Scale Symmetric Traveling Salesman Problems.* Retrieved from
https://pdfs.semanticscholar.org/9665/80e91b06f55605869b7155784c2fc0c444bb.pdf?fbclid=IwAR2Yd3K6KxlA9qdDnRdMhaPhvJe1-XqdDykGmkz5LVHq1PcY38USBXHojUs

Pandian P. & Jayalakshmi M. (January – March, 2012) A New Approach For Solving A Class Of Pure Integer Linear Programming Problems, Department Of

Mathematics, School Of Science And Humanities, V It University India, International Journal Of Advanced Engineering Technology. Retrieved from:

https://pdfs.semanticscholar.org/0cde/106248c928e9de298b2571b36d78f5ca2e2c.pdf

Ringel, G, (1974) *Map Color Theorem.* Santa Cruz USA, University of California. Retrieved from

https://www.springer.com/gp/book/9783642657610

Σαπουντζής, Κ.Ι., (1993) *Τεχνικές επιχειρησιακής έρευνας.* Πειραιάς: Σταμουλής**.**

Seckiner, S.U., Gokcen, H., & Kurt, M., (1981, Jan) *An integer programming model for hierarchical workforce scheduling problem.* Retrieved from https://www.researchgate.net/publication/200035380_An_Integer_Programming_Approach_to_Scheduling?fbclid=IwAR2D8OAl5rx0ojMM80mru9Y6RWPzFuka7IcDjV3EMj0Xm0R3rxv_ZBKWcNs

Stredwick, J. (2000). An Introduction to Human Resource Management. UK: Butterworth Heinemann.

Taha, H.A., (2007). *Operations research: An introduction* (8th edition). New Jersey : Pearson.

Williams, H.P., (2013). Model Building in Mathematical Programming (Fifth Edition). UK: John Willey & sons.

IBM ILOG CPLEX Optimization Studio CPLEX User's Manual, Version 12 Release 7