ΠΑΝΕΠΙΣΤΗΜΙΟ ΜΑΚΕΔΟΝΙΑΣ

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ

ΤΜΗΜΑΤΟΣ ΕΦΑΡΜΟΣΜΕΝΗΣ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΕΝΑΣ ΑΛΓΟΡΙΘΜΟΣ ΠΟΛΥΕΝΑΡΚΤΗΡΙΑΣ ΤΟΠΙΚΗΣ ΑΝΑΖΗΤΗΣΗΣ ΓΙΑ ΤΟ ΠΡΟΒΛΗΜΑ ΤΗΣ ΤΕΤΡΑΓΩΝΙΚΗΣ ΑΝΤΙΣΤΟΙΧΙΣΗΣ

Διπλωματική Εργασία

του

Κουφάκη Νικολάου

Θεσσαλονίκη , Ιούνιος 2020

ΕΝΑΣ ΑΛΓΟΡΙΘΜΟΣ ΠΟΛΥΕΝΑΡΚΤΗΡΙΑΣ ΤΟΠΙΚΗΣ ΑΝΑΖΗΤΗΣΗΣ ΓΙΑ ΤΟ ΠΡΟΒΛΗΜΑ ΤΗΣ ΤΕΤΡΑΓΩΝΙΚΗΣ ΑΝΤΙΣΤΟΙΧΙΣΗΣ

Κουφάκης Νικόλαος

Πτυχίο Εφαρμοσμένης Πληροφορικής, Πανεπιστήμιο Μακεδονίας, 2018

Διπλωματική Εργασία

υποβαλλόμενη για τη μερική εκπλήρωση των απαιτήσεων του

ΜΕΤΑΠΤΥΧΙΑΚΟΥ ΤΙΤΛΟΥ ΣΠΟΥΔΩΝ ΣΤΗΝ ΕΦΑΡΜΟΣΜΕΝΗ ΠΛΗΡΟΦΟΡΙΚΗ

Επιβλέπων Καθηγητής
Σιφαλέρας Άγγελος

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 25/6/2020

ΣΙΦΑΛΕΡΑΣ ΑΓΓΕΛΟΣ	ΣΑΜΑΡΑΣ ΝΙΚΟΛΑΟΣ	ΧΡΗΣΤΟΥ ΒΑΡΣΑΚΕΛΗΣ ΔΗΜΗΤΡΙΟΣ
.....	.....	.....

Κουφάκης Νικόλαος

.....

Περίληψη

Η παρούσα διπλωματική εργασία έχει ως σκοπό τη σχεδίαση και υλοποίηση ενός αλγορίθμου για την επίλυση του προβλήματος της τετραγωνικής αντιστοίχισης. Ως εργαλεία και τεχνικές χρησιμοποιούνται η μέθοδος αναζήτησης μεταβλητής γειτνίασης και η πολυεναρκτήρια τοπική αναζήτηση. Εφόσον γίνει παράθεση του θεωρητικού υποβάθρου, τόσο για το πρόβλημα όσο και για τις τεχνικές που αναπτύχθηκαν, παρουσιάζεται λεπτομερώς η σχεδίαση και πρακτική υλοποίηση του προγράμματος. Εν συνεχεία αναλύονται τα αποτελέσματα του προγράμματος και γίνεται σύγκριση με άλλες αλγοριθμικές υλοποιήσεις. Η εργασία ολοκληρώνεται με τα συμπεράσματα της έρευνας καθώς και με προτάσεις για μελλοντική βελτίωση.

Λέξεις Κλειδιά: Πρόβλημα τετραγωνικής αντιστοίχισης, αναζήτηση μεταβλητής γειτνίασης, πολυεναρκτήρια τοπική αναζήτηση

Abstract

The aim of this master thesis is the design and implementation of an algorithm for the quadratic assignment problem. The tools and techniques used are the variable neighbourhood search and the multi start local search methods. Having cited the theoretical background, both for the problem and the developed techniques, the design and practical implementation of the program is presented in detail. The results of the program are then analyzed and compared with other algorithmic implementations. The thesis concludes with the findings of the research, as well as suggestions for future improvement.

Keywords: Quadratic assignment problem, variable neighbourhood search, multi start local search

ΕΥΧΑΡΙΣΤΙΕΣ

Θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή μου κ. Άγγελο Σιφαλέρα για την στήριξη και καθοδήγηση που μου πρόσφερε καθ'ολη τη διάρκεια της εκπόνησης της παρούσας διπλωματικής. Επίσης θα ήθελα να ευχαριστήσω την οικογένειά μου, τους φίλους μου και όσους με στήριξαν κατά τη διάρκεια των μεταπτυχιακών μου σπουδών.

ΠΕΡΙΕΧΟΜΕΝΑ

Κατάλογος Σχημάτων

Κατάλογος Πινάκων

ΚΕΦΑΛΑΙΟ 1

Εισαγωγή

1.1 Πρόβλημα - Σημαντικότητα του θέματος

Το πρόβλημα της τετραγωνικής αντιστοίχισης (Quadratic Assignment Problem) αποτελεί ένα κλασικό παράδειγμα προβλήματος συνδυαστικής βελτιστοποίησης. Έτσι ονομάζονται τα προβλήματα στα οποία πρέπει να βρεθεί η βέλτιστη λύση μέσα σε ένα πεπερασμένο σύνολο λύσεων. Προφανώς όσο μεγαλώνει το σύνολο των λύσεων τόσο πιο δύσκολα μπορεί να βρεθεί η βέλτιστη λύση. Για αυτό το λόγο η μέθοδος της εξαντλητικής αναζήτησης δεν επιστρέφει τη βέλτιστη λύση σε ικανοποιητικό χρόνο.

Έτσι λοιπόν, στρεφόμαστε στη χρήση μεθευρετικών αλγορίθμων οι οποίοι μπορεί να μην εγγυώνται ότι η λύση που επιστρέφουν είναι η βέλτιστη, αλλά χρειάζονται λιγότερο, κατά κανόνα, χρόνο για μια άκρως ικανοποιητική λύση. Το πρόβλημα της τετραγωνικής αντιστοίχισης εκτός από το ερευνητικό ενδιαφέρον, έχει και εφαρμογές στην πραγματική ζωή, όπως ο σχεδιασμός εγκαταστάσεων ενός νοσοκομείου ή ενός εμπορικού κέντρου, ακόμα και ο σχεδιασμός κυκλωμάτων.

Όπως και με τα άλλα, δύσκολα υπολογιστικά προβλήματα, έτσι και το πρόβλημα της τετραγωνικής αντιστοίχισης (QAP), χρειάζεται για την ώρα ευρετική προσέγγιση. Σε μεγάλα μεγέθη, μια εξαντλητική αναζήτηση χρειάζεται πολύ χρόνο, ενώ ένας μεθευρετικός αλγόριθμος σύγχρονης τεχνολογίας μπορεί να επιστρέψει σχεδόν άμεσα μια αρκετά ικανοποιητική λύση, αν και είναι απίθανο η λύση αυτή να είναι η βέλτιστη.

1.2 Σκοπός - Στόχοι

Η παρούσα διπλωματική έχει ως στόχο την σχεδίαση και υλοποίηση ενός ευρετικού αλγορίθμου, ο οποίος θα επιχειρεί να επιλύσει στιγμιότυπα του προβλήματος της τετραγωνικής αντιστοίχισης. Ο αλγόριθμος αυτός βασίζεται στην μεθοδολογία της Αναζήτησης Μεταβλητής Γειτνίασης (Variable Neighbourhood Search) και έχει ως αρχικό σκοπό την επιστροφή λύσεων στα στιγμιότυπα του προβλήματος. Έπειτα, αφού ελέγξουμε την απόδοση των επιστρεφόμενων λύσεων, θα δώσουμε

έμφαση στο χρόνο που χρειάζεται για να βρεθεί μια ικανοποιητική λύση. Όταν φτάσουμε στο επιθυμητό επίπεδο ποιότητας και ταχύτητας, τότε θα συγκρίνουμε την απόδοση του αλγορίθμου που αναπτύχθηκε με την απόδοση άλλων μεθόδων που θεωρούνται State Of The Art.

1.3 Βασική Ορολογία

Στη συνέχεια της διπλωματικής εργασίας θα χρησιμοποιηθούν κάποιοι ειδικοί όροι οι οποίοι έχουν να κάνουν με το πρόβλημα της τετραγωνικής αντιστοίχισης και την αναζήτηση μεταβλητής γειτνίασης. Η βασική ορολογία που χρειάζεται να γνωρίζει κάποιος έτσι ώστε να μπορεί να αντιληφθεί τα περιεχόμενα αυτής της εργασίας παρατίθεται εδώ:

• Φάση διατάραξης (Shaking) : είναι μια τεχνική που χρησιμοποιεί ο αλγόριθμος αναζήτησης μεταβλητής γειτνίασης και μας βοηθάει να ξεκολλήσουμε από ένα τοπικό μέγιστο ή ελάχιστο στο οποίο έχουμε κολλήσει.

• Φάση τοπικής αναζήτησης (Local Search) : ο αλγόριθμος επιλέγει μια κατεύθυνση ανάβασης ή κατάβασης από την αρχική λύση και αυτό συνεχίζεται για ένα προκαθορισμένο αριθμό επαναλήψεων.

• Συνάρτηση αξιολόγησης/ Αντικειμενική συνάρτηση : πρόκειται για τη συνάρτηση που καθορίζει την ποιότητα μιας λύσης για ένα στιγμιότυπο του προβλήματος. Σε προβλήματα ελαχιστοποίησης θέλουμε η λύση μας να έχει όσο το δυνατόν μικρότερη τιμή, ενώ σε προβλήματα μεγιστοποίησης στοχεύουμε στη μεγαλύτερη δυνατή τιμή της εκάστοτε λύσης.

• Τελεστής : είναι μια αλγοριθμική τεχνική που χρησιμοποιείται στις φάσεις διατάραξης και τοπικής αναζήτησης. Στο συγκεκριμένο πρόβλημα ο κάθε τελεστής προκαλεί μια μεταβολή στο διάνυσμα της λύσης.

• Γειτονιά λύσεων : πρόκειται για το σύνολο των λύσεων τις οποίες μπορούμε να επισκεφθούμε κάνοντας χρήση ενός συγκεκριμένου τελεστή.

1.4 Διάρθρωση της Εργασίας

Ξεκινώντας από το επόμενο κεφάλαιο, θα γίνει μια περιγραφή του προβλήματος που επιχειρήθηκε να επιλυθεί, αυτό της τετραγωνικής αντιστοίχισης (QAP). Έπειτα θα μελετηθούν κάποιες προϋπάρχουσες τεχνικές για την επίλυσή του και στο τέλος του κεφαλαίου θα δούμε μερικά στιγμιότυπα του προβλήματος.

Στο επόμενο κεφάλαιο θα αναλυθούν οι μέθοδοι επίλυσης που χρησιμοποιήθηκαν στο πρόγραμμα. Αρχικά θα παρουσιαστεί ο αλγόριθμος αναζήτησης μεταβλητής γειτνίασης (Variable Neighborhood Search - VNS), εξετάζοντας κάθε φάση του ξεχωριστά. Έπειτα γίνεται λόγος για τη μέθοδο πολυεναρκτήριας τοπικής αναζήτησης, η οποία αν και δεν ήταν στον αρχικό σχεδιασμό, αποτελεί αναπόσπαστο τμήμα της τελικής υλοποίησης. Εν συνεχεία, παρατίθεται ο σχεδιασμός και η μετέπειτα υλοποίηση της εφαρμογής, καθώς και μια μελέτη των τεχνικών που δοκιμάστηκαν αλλά εν

τέλει απορρίφθηκαν. Στο τέλος του κεφαλαίου υπάρχει η τελική υλοποίηση η οποία όπως θα δούμε διαφέρει από τον αρχικό σχεδιασμό.

Στο επόμενο κεφάλαιο γίνεται παράθεση των αποτελεσμάτων και έπειτα σύγκριση αυτών με άλλους λύτες. Στο τελευταίο κεφάλαιο ολοκληρώνεται η εργασία με την παρουσίαση των συμπερασμάτων αλλά και κάποιων προτάσεων για μελλοντική βελτίωση και επέκταση του προγράμματος.

ΚΕΦΑΛΑΙΟ 2

Quadratic Assignment Problem

2.1 Περιγραφή του Προβλήματος

Όπως ειπώθηκε και προηγουμένως το πρόβλημα της τετραγωνικής αντιστοίχισης ανήκει στα προβλήματα συνδυαστικής βελτιστοποίησης. Αρχικά προτάθηκε από τους Koopmans και Beckmann το 1957 (Koopmans and Beckmann 1957) και παρ'ολο που έχουν περάσει τόσα πολλά χρόνια και είναι σχετικά απλή η δομή του, συνεχίζει ακόμα και σήμερα να αποτελεί ένα πολύ δύσκολο υπολογιστικά πρόβλημα.

Όπως έχει αποδειχθεί, το QAP ανήκει στην κλάση προβλημάτων NP-hard και είναι μάλλον αδύνατον να βρεθεί η βέλτιστη λύση σε πολυωνυμικό χρόνο. Δηλαδή ένας αλγόριθμος ο οποίος προσπαθεί να βρει τη βέλτιστη λύση ψάχνοντας όλους τους πιθανούς συνδυασμούς μεταθέσεων χρειάζεται πολύ περισσότερο χρόνο όσο αυξάνεται το μέγεθος του προβλήματος (Sahni and Gonzalez 1976). Αυτό ουσιαστικά σημαίνει ότι δεν υπάρχει αποδοτικός τρόπος να βρούμε τη βέλτιστη λύση με ακριβή αλγόριθμο άρα αναγκαστικά στρεφόμαστε σε ευρετικές μεθόδους με τις οποίες παίρνουμε μια αρκετά ικανοποιητική λύση (όχι απαραίτητα και βέλτιστη όμως) σε λιγότερο χρόνο.

Το QAP μπορεί να διατυπωθεί ακολούθως: υπάρχει ένα σύνολο n εγκαταστάσεων και ένα σύνολο n θέσεων. Για κάθε ζεύγος θέσεων ορίζεται μια απόσταση και για κάθε ζεύγος εγκαταστάσεων ορίζεται μια ροή. Το πρόβλημα είναι να αναθέσουμε σε κάθε θέση μια εγκατάσταση με στόχο την ελαχιστοποίηση του αθροίσματος των αποστάσεων που πολλαπλασιάζονται με τις αντίστοιχες ροές (Burkard, Pardalos, Du, and Graham 2013).

Αυτό σημαίνει ότι θέλουμε να βρούμε τη σειρά με την οποία θα αναθέσουμε τις εγκαταστάσεις, έτσι ώστε να έχουμε το ελάχιστο κόστος. Ως δεδομένα θεωρούμε ότι έχουμε δύο πίνακες, τον A, ο οποίος περιέχει τις ροές μεταξύ των εγκαταστάσεων και τον B, ο οποίος περιέχει τις αποστάσεις μεταξύ των θέσεων.

Σε μαθηματική μορφή ορίζεται ως εξής: με δεδομένα τον πίνακα $A = (\mathrm{a_{ij}})_{\mathrm{nxn}}$ και τον πίνακα $B = (\mathrm{b_{kl}})_{\mathrm{nxn}}$ και το σύνολο Π που περιέχει όλες τις πιθανές μεταθέσεις των ακεραίων από 1 έως n, πρέπει να βρεθεί εκείνη η μετάθεση η οποία ελαχιστοποιεί την ακόλουθη συνάρτηση (Misevicius

2011):

$$z(\pi) = \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} b_{\pi(i)\pi(j)}$$

Η εξίσωση 2.1 ουσιαστικά είναι υπεύθυνη για την υπολογιστική δυσκολία του QAP. Η εξίσωση αυτή λειτουργεί ως συνάρτηση αξιολόγησης για κάθε μετάθεση και μιας και εμείς θέλουμε να μειώσουμε το κόστος, όσο μικρότερη τιμή έχει αυτή η συνάρτηση για κάποια μετάθεση, τόσο καλύτερη ποιοτικά θεωρείται αυτή η μετάθεση. Το να αλλάξουμε την τρέχουσα λύση σε κάποια άλλη είτε με ανταλλαγή δύο θέσεων είτε με κάποια διαφορετική τεχνική δεν είναι κάτι το υπολογιστικά δύσκολο. Αυτό που επιφέρει τον μεγάλο υπολογιστικό φόρτο είναι η ανάγκη για τον επαναϋπολογισμό αυτής της συνάρτησης κάθε φορά που αλλάζει η λύση. Για να το αποφύγουμε αυτό στρεφόμαστε σε κάποιες τεχνικές που μας επιτρέπουν να υπολογίζουμε κάθε φορά μόνο τη διαφορά που υπάρχει ανάμεσα σε δύο γειτονικές λύσεις όπως θα δούμε σε παρακάτω κεφάλαιο, αυτό της σχεδίασης της εφαρμογής.

2.2 Τεχνικές Επίλυσης του Προβλήματος

Όπως προαναφέρθηκε, είναι ανούσιο να προσπαθούμε να βρούμε τη βέλτιστη λύση σε προβλήματα της κλάσης NP-hard με ακριβείς αλγορίθμους. Για αυτό το λόγο λοιπόν στρεφόμαστε σε ευρετικές μεθοδολογίες οι οποίες σε λιγότερο κατά κανόνα χρόνο, παράγουν μια ικανοποιητική λύση. Το μειονέκτημα αυτών των μεθόδων είναι ότι αδυνατούν να μας εγγυηθούν ότι η λύση που βρήκαν είναι η βέλτιστη. Επίσης, ακόμα και αν καταφέρουμε να βρούμε τη βέλτιστη λύση χρησιμοποιώντας μια ευρετική μεθοδολογία δε μπορούμε να αποδείξουμε ότι όντως αυτή είναι η βέλτιστη λύση, εκτός φυσικά αν γνωρίζουμε την τιμή της από πριν.

Συγκεκριμένα για το QAP, έχουν προταθεί και δοκιμαστεί αρκετές τέτοιες μεθοδολογίες, με ορισμένες από αυτές να αφορούν την μέθοδο της προσομοιωμένης ανόπτησης (Simulated Annealing) (Wilhelm and Ward 1987), τη μέθοδο της αναζήτησης Tabu (Battiti and Tecchiolli 1994) και τη μέθοδο αναζήτησης γειτονιάς μεγάλης κλίμακας (Altner, Ahuja, Ergunm, and Orlin 2011).

Στην παρούσα διπλωματική θα επιχειρήσουμε να αναπτύξουμε έναν αλγόριθμο που βασίζεται στην τεχνική αναζήτησης μεταβλητής γειτνίασης (VNS) με σκοπό την επίλυση των στιγμιοτύπων του προβλήματος που διαθέτουμε. Όπως θα δούμε σε επόμενο κεφάλαιο, υπάρχουν κάποιες τεχνικές οι οποίες επιταχύνουν τη διαδικασία επίλυσης του προβλήματος και κυρίως προσπαθούν να μειώσουν την πολυπλοκότητα του υπολογισμού της αντικειμενικής συνάρτησης, δηλαδή της συνάρτησης που μας βοηθάει να αξιολογήσουμε την κάθε λύση. Αυτές οι τεχνικές για παράδειγμα, επωφελούνται αν το στιγμιότυπο είναι συμμετρικό και αν χρησιμοποιούμε τον τελεστή swap/ 1-opt. Προφανώς θα

αναλυθούν περαιτέρω όταν μεταβούμε στο σημείο ανάλυσης του προγράμματος.

2.3 Στιγμιότυπα του Προβλήματος

Έχοντας μια αρχική ιδέα για το πρόβλημα της τετραγωνικής αντιστοίχισης μπορούμε να υλοποιήσουμε έναν αλγόριθμο που βασίζεται σε κάποια από τις προαναφερθείσες τεχνικές. Για να μπορέσουμε όμως να τον αξιολογήσουμε, χρειαζόμαστε κάποια στιγμιότυπα του προβλήματος τα οποία εκτός από τα αρχικά δεδομένα, διαθέτουν και τη βέλτιστη ή διαφορετικά την καλύτερη μέχρι στιγμής λύση.

Ευτυχώς για εμάς, υπάρχει μια βιβλιοθήκη τέτοιων στιγμιοτύπων η οποία εκτός από τα δεδομένα που χαρακτηρίζουν το πρόβλημα, δηλαδή τους δύο πίνακες με τις ροές και τις αποστάσεις αντίστοιχα, διαθέτει και την αναπαράσταση της καλύτερης λύσης που έχει βρεθεί, η οποία στις περισσότερες περιπτώσεις είναι και η βέλτιστη. Επίσης, αναγράφεται και το συνολικό κόστος της λύσης, το οποίο φυσικά παράγεται σύμφωνα με την εξίσωση 2.1.

Αυτή η βιβλιοθήκη βρίσκεται στο Διαδίκτυο και συγκεκριμένα στον ακόλουθο σύνδεσμο: `http://anjos.mgi.polymtl.ca/qaplib/inst.html`. Κάθε στιγμιότυπο έχει στο όνομά του και το μέγεθός του, δηλαδή το πλήθος των εγκαταστάσεων. Παρακάτω ακολουθεί ένα παράδειγμα του πώς απεικονίζεται κάθε στιγμιότυπο του προβλήματος:

```
12

 0    90    10    23    43     0     0     0     0     0     0     0
90     0     0     0     0    88     0     0     0     0     0     0
10     0     0     0     0     0    26    16     0     0     0     0
23     0     0     0     0     0     0     0     0     0     0     0
43     0     0     0     0     0     0     0     0     0     0     0
 0    88     0     0     0     0     0     0     1     0     0     0
 0     0    26     0     0     0     0     0     0     0     0     0
 0     0    16     0     0     0     0     0     0    96     0     0
 0     0     0     0     0     1     0     0     0     0    29     0
 0     0     0     0     0     0     0    96     0     0     0    37
 0     0     0     0     0     0     0     0    29     0     0     0
 0     0     0     0     0     0     0     0     0    37     0     0

 0    36    54    26    59    72     9    34    79    17    46    95
36     0    73    35    90    58    30    78    35    44    79    36
54    73     0    21    10    97    58    66    69    61    54    63
26    35    21     0    93    12    46    40    37    48    68    85
59    90    10    93     0    64     5    29    76    16     5    76
72    58    97    12    64     0    96    55    38    54     0    34
 9    30    58    46     5    96     0    83    35    11    56    37
34    78    66    40    29    55    83     0    44    12    15    80
79    35    69    37    76    38    35    44     0    64    39    33
17    44    61    48    16    54    11    12    64     0    70    86
46    79    54    68     5     0    56    15    39    70     0    18
95    36    63    85    76    34    37    80    33    86    18     0
```

Σχήμα **2.3.1** Στιγμιότυπο QAP.

Όπως παρατηρούμε, το στιγμιότυπο ξεκινάει δηλώνοντας το μέγεθος του προβλήματος, το οποίο στην προκειμένη περίπτωση είναι το 12. Προφανώς αποτελεί ένα από τα μικρά στιγμιότυπα, αλλά επελέγη διότι ήταν πολύ πιο ευειδής η απεικόνισή του σε σχέση με άλλα στιγμιότυπα μεγαλύτερου μεγέθους. Το στιγμιότυπο συνεχίζει με τα δεδομένα των δύο πινάκων, A και B, οι οποίοι περιέχουν τις ροές και τις αποστάσεις αντίστοιχα. Αυτά τα δεδομένα αποτελούν επαρκείς στοιχεία για να προσπαθήσει κάποιος να επιλύσει το στιγμιότυπο. Παρ'ολα αυτά όμως η βιβλιοθήκη QAPLIB μας παρέχει άλλο ένα αρχείο για το κάθε στιγμιότυπο, αυτό της καλύτερης μέχρι τώρα (και σε πολλές περιπτώσεις της βέλτιστης) λύσης. Το αρχείο που περίεχει τη λύση έχει την ακόλουθη μορφή:

```
 12 9552
7  5 12  2  1  3  9 11 10  6  8  4
```

Σχήμα **2.3.2** Λύση στιγμιοτύπου QAP.

Σε αυτό το αρχείο όπως φαίνεται έχουμε ως αρχικό δεδομένο και πάλι το μέγεθος, ακολουθούμενο από την τιμή κόστους της αντικειμενικής συνάρτησης. Το οποίο κόστος όπως προαναφέρθηκε βγαίνει από την εξίσωση 2.1 έχοντας ως είσοδο τους πίνακες A και B και το διάνυσμα της λύσης.

Μετά από το κόστος ακολουθεί η λύση σε μορφή διανύσματος και επί της ουσίας αποτελεί μια μετάθεση των αριθμών από το 1 έως και το n, το μέγεθος δηλαδή του προβλήματος.

Πρακτικά το διάνυσμα αυτό μας λέει πώς πρέπει να τοποθετηθούν οι εγκαταστάσεις έτσι ώστε να έχουμε το κόστος που αναγράφεται στη λύση και υποτίθεται είναι το χαμηλότερο γνωστό κόστος. Στο παράδειγμά μας, με αναφορά την εικόνα **2.3.2**, ισχύει ότι αν βάλουμε τις εγκαταστάσεις στην παρακάτω σειρά: *7 5 12 2 1 3 9 11 10 6 8 4* (δηλαδή στην πρώτη τοποθεσία την έβδομη εγκατάσταση, στη δεύτερη τοποθεσία την πέμπτη εγκατάσταση και ούτω καθεξής), θα έχουμε κόστος *9552*.

Επιπρόσθετα, αν κοιτάξουμε τη δομή του στιγμιοτύπου, παρατηρούμε ότι έχει συμμετρική μορφή. Αυτή η ιδιότητα μας επιτρέπει να υπολογίζουμε την αντικειμενική συνάρτηση αξιολόγησης πιο γρήγορα όπως θα δούμε σε επόμενο κεφάλαιο. Εφόσον όλο το υπολογιστικό βάρος ουσιαστικά συγκεντρώνεται στον υπολογισμό της αντικειμενικής συνάρτησης, ό,τι μπορούμε να κάνουμε για να το μειώσουμε είναι κέρδος και μας επιτρέπει να επιταχύνουμε σημαντικά τη διαδικασία επίλυσης.

ΚΕΦΑΛΑΙΟ 3

Μέθοδοι επίλυσης του προβλήματος

3.1 Εισαγωγή

Σε αυτό το κεφάλαιο θα παρουσιαστούν οι δύο βασικότερες μέθοδοι που επιλέχθηκαν με σκοπό την επίλυση του προβλήματος της τετραγωνικής αντιστοίχισης. Θα μελετηθεί αρχικά η μέθοδος της αναζήτησης μεταβλητής γειτνίασης, που αποτελεί το μεγαλύτερο μέρος της προσπάθειας κατά τη διάρκεια της υλοποίησης του προγράμματος και έπειτα θα παρουσιαστεί η μέθοδος πολυεναρκτήριας τοπικής αναζήτησης. Η τελευταία εφαρμόστηκε προς το τέλος της ανάπτυξης του προγράμματος, αλλά όπως θα δούμε στη συνέχεια, ήταν αυτή που προσέφερε τις μεγαλύτερες βελτιώσεις και εν τέλει οδήγησε το πρόγραμμα στην τελική μορφή του.

3.2 Αναζήτηση Μεταβλητής Γειτνίασης

3.2.1 Εισαγωγή

Η αναζήτηση μεταβλητής γειτνίασης αποτελεί μια μεθευρετική μέθοδο η οποία χρησιμοποιείται κυρίως για την επίλυση προβλημάτων συνδυαστικής βελτιστοποίησης (όπως είναι και το πρόβλημα της τετραγωνικής αντιστοίχισης που μελετάμε) και βασίζεται στην ιδέα των γειτονικών λύσεων. Αυτό που την κάνει αρκετά αποδοτική μέθοδο είναι η ικανότητα να ξεφεύγει από τοπικά βέλτιστα όταν παγιδεύεται με στόχο την εύρεση του καθολικού βέλτιστου(Mladenovic and Hansen 1997). Ένα ακόμα χαρακτηριστικό της αναζήτησης μεταβλητής γειτνίασης είναι ο συνδυασμός της τοπικής αναζήτησης με δυναμικές δομές γειτονιάς οι οποίες μεταβάλλονται κατά τη διάρκεια της αναζήτησης.

Πρακτικά, αυτό που κάνει η VNS είναι να μεταπηδά σε απομακρυσμένες γειτονιές της τρέχουσας λύσης όταν βρίσκει βελτίωση σε κάποια από αυτές. Όσον αφορά την τοπική αναζήτηση, αυτή χρησιμοποιείται σε κάθε γειτονιά επανειλημμένα μέχρι να βρεθεί το τοπικό βέλτιστο, όπου στο πρόβλημά μας είναι το τοπικό ελάχιστο. Σε πρώτη φάση δηλαδή προσπαθεί να βρει το τοπικό βέλτιστο στην τρέχουσα γειτονιά και μετά αλλάζει γειτονιά με χρήση κάποιων τελεστών που θα μελετήσουμε αργότερα. Οι τελεστές αυτοί ουσιαστικά ορίζουν τις γειτονικές λύσεις στο κάθε

πρόβλημα. Η αναζήτηση στις άλλες γειτονιές συνεχίζεται μέχρι να υπάρξει κάποια βελτίωση (να ξεκολλήσουμε δηλαδή από το τοπικό βέλτιστο της προηγούμενης γειτονιάς) και αν βρεθεί τότε μεταπηδάει στη γειτονιά που επίφερε τη βελτίωση.

Υπάρχουν τρεις βασικές παρατηρήσεις που χαρακτηρίζουν την αναζήτηση μεταβλητής γειτνίασης (Mladenovic, Hansen, Perez, and Brimberg 2010):

• Ένα τοπικό βέλτιστο (ελάχιστο ή μέγιστο) για μια δομή γειτονιάς δεν είναι απαραίτητα και τοπικό βέλτιστο για μια άλλη δομή γειτονιάς.

• Το καθολικό βέλτιστο (η βέλτιστη δηλαδή λύση του προβλήματος) αποτελεί και τοπικό βέλτιστο για όλες τις δομές γειτονιών.

• Για πολλά προβλήματα τα τοπικά βέλτιστα για διαφορετικές γειτονιές βρίσκονται σχετικά κοντά μεταξύ τους.

Όπως αρκετές άλλες μεθευρετικές μεθοδολογίες, έτσι και η VNS έχουν κάποια κύρια και βασικά βήματα. Στην προκειμένη μέθοδο αυτά τα βήματα είναι τα εξής: η φάση διατάραξης, όπου έχει ως σκοπό την απεγκλώβιση από τοπικά βέλτιστα, η φάση τοπικής αναζήτησης, όπου έχει ως σκοπό την εύρεση του τοπικού βέλτιστου στη συγκεκριμένη γειτονιά και η φάση αλλαγής γειτονιάς, χάρη στην οποία ισχυροποιείται η διαφοροποίηση των λύσεων. Θα ακολουθήσει ανάλυση της κάθε φάσης καθώς επίσης και αναφορά των πιο διαδεδομένων εκδοχών της μεθόδου VNS.

3.2.2 Φάση Διατάραξης

Το πρώτο πράγμα που κάνει αυτή η μέθοδος, πριν από τα βασικά βήματα, είναι να δημιουργήσει τις δομές γειτνίασης των λύσεων. Αρχικά λοιπόν, ορίζουμε το πεπερασμένο σύνολο των προεπιλεγμένων γειτονιών, N_k, $(k = 1, ..., k = k_{max})$ και επίσης ορίζουμε N_x το σύνολο των γειτονικών λύσεων στην k-ιοστή γειτονιά του x (Mladenovic and Hansen 2001). Οι γειτονιές ορίζονται με βάση τον τελεστή με τον οποίο υπολογίζονται. Όπως θα δούμε παρακάτω υπάρχουν αρκετοί τέτοιοι τελεστές που χρησιμοποιούνται είτε για τη φάση διατάραξης είτε για τη φάση τοπικής αναζήτησης και κάποιες φορές και για τις δύο φάσεις.

Όταν λοιπόν είμαστε έτοιμοι με το σύνολο των δομών γειτνίασης, τότε πρέπει να βρούμε μια αρχική λύση. Αυτή βρίσκεται είτε τυχαία, δηλαδή μια τυχαία μετάθεση των ακεραίων 1 έως n που αποτελούν την απεικόνιση της λύσης, είτε χρησιμοποιείται κάποιος αλγόριθμος αρχικοποίησης ο οποίος υπόσχεται μια καλύτερη λύση απο την τυχαία. Προφανώς όμως με χρήση αλγορίθμου αρχικοποίησης, χάνουμε χρόνο γιατί όπως είναι λογικό, η τυχαία μετάθεση απαιτεί μηδενικό χρόνο υπολογισμού.

Αφού ξεκινήσουμε λοιπόν για $k = 1$, παράγουμε τυχαία μια γειτονική λύση από την k-ιοστή γειτονιά του x. Δηλαδή εφαρμόζουμε στην τρέχουσα λύση τον τελεστή διατάραξης που έχουμε επιλέξει k φορές. Στην πορεία αυτό το βήμα βοηθάει να απεγκλωβιστούμε από τυχόν τοπικά βέλτιστα στα οποία έχουμε παγιδευτεί. Όπως γίνεται κατανοητό, όσο αυξάνεται ο αριθμός των επαναλήψεων

και μεγαλώνει το k, τόσο πιο έντονη γίνεται και η φάση διατάραξης, άρα μετακινούμαστε σε πιο μακρινή γειτονιά της τρέχουσας λύσης και οι πιθανότητες να ξεφύγουμε από κάποιο τοπικό βέλτιστο αυξάνονται.

Παρακάτω ακολουθεί η διαδικασία της διατάραξης σε αλγοριθμική μορφή(Mladenovic, Hansen, Todosijevic, and Hanafi 2016):

Algorithm 1 Shaking

```
1: procedure SHAKE(x, k, N)
2:     Choose x' ∈ N_k(x) randomly
3:     return x'
4: end procedure
```

Ήδη έχει γίνει λόγος για τους τελεστές με τους οποίους πραγματοποιείται η φάση διατάραξης και όπως θα δούμε αργότερα και η φάση τοπικής αναζήτησης. Αυτοί δεν είναι κάτι άλλο παρά τρόποι με τους οποίους μεταβάλλουμε την τρέχουσα λύση έτσι ώστε να προσπαθήσουμε να βρούμε κάποια άλλη με καλύτερη συνάρτηση αξιολόγησης. Στο συγκεκριμένο πρόβλημα, εφ'οσον η λύση του παρουσιάζεται ως ένα διάνυσμα των αριθμών από το ένα ως το n σε κάποια συγκεκριμένη σειρά, οι πράξεις που μπορούμε να κάνουμε σχετίζονται με τη μεταβολή αυτού του διανύσματος.

Ας δούμε παρακάτω τις πιο διαδεδομένες τεχνικές τροποποίησης του διανύσματος ακολουθούμενες από ένα παράδειγμα έκαστες.

• 1-opt / swap : Με τη χρήση του swap επιλέγουμε δύο στοιχεία της λύσης, που στην περίπτωσή μας απεικονίζουν τις εγκαταστάσεις, και μεταφέρουμε το ένα στη θέση του άλλου. Αν επίσης τυχαίνει να βρίσκονται σε διπλανές θέσεις τότε αυτό μπορεί να ονομαστεί και 1-opt. Για παράδειγμα με αρχική λύση: [2,3,5,4,1] και χρήση του swap στις θέσεις 2 και 5 παίρνουμε τη γειτονική λύση: [2,1,5,4,3]. Η χρήση του τελεστή 1-opt στις θέσεις 3 και 4 στην αρχική λύση θα είχε ως αποτέλεσμα τη γειτονική λύση: [2,3,4,5,1].

• 2-opt : Αυτός ο τελεστής επιλέγει δύο σημεία του διανύσματος και αντιστρέφει τα στοιχεία που βρίσκονται ανάμεσά τους(Croes 1958). Προφανώς τα υπόλοιπα στοιχεία παραμένουν ανεπηρέαστα. Ως παράδειγμα ας θεωρήσουμε την αρχική λύση: [7,5,2,1,3,6,4]. Ο τελεστής 2-opt με επιλεγμένες θέσεις την πρώτη και την έκτη επιστρέφει ως αποτέλεσμα το νέο διάνυσμα: [7,3,1,2,5,6,4].

• relocate : Ένας πιο απλός τελεστής στην πράξη, επιλέγει ένα στοιχείο του διανύσματος και το μεταφέρει σε κάποια άλλη θέση. Για παράδειγμα έστω η αρχική λύση: [2,3,5,4,1] και το επιλεγμένο στοιχείο βρίσκεται στην τέταρη θέση. Αν επιλέξουμε ως νέα του θέση τη δεύτερη τότε έχουμε τη νέα λύση: [2,4,3,5,1].

• Shift left / Shift right : Αυτός δεν είναι ιδιαίτερα συχνά χρησιμοποιούμενος τελεστής, αλλά επειδή έγινε δοκιμή του στο πρόγραμμα που αναπτύχθηκε, θεωρώ σημαντική την αναφορά του. Ουσιαστικά μεταφέρει όλα τα στοιχεία του διανύσματος έναν αριθμό θέσεων αριστερά ή δεξιά, ανάλογα με τις επιλογές μας. Με αρχική λύση: [2,3,5,4,1] και μεταφορά όλων των στοιχείων κατά

μια θέση δεξιά έχουμε το νέο διάνυσμα: [1,2,3,5,4].

Όπως προαναφέρθηκε αυτοί οι τελεστές μπορούν να χρησιμοποιηθούν στη διαδικασία διατάραξης και ο καθένας τους ουσιαστικά ορίζει και μια δομή γειτονιάς. Άλλες γειτονικές λύσεις παίρνουμε με χρήση του swap και άλλες με χρήση του 2-opt. Επιπρόσθετα όμως, με χρήση αυτών των τελεστών γίνεται και το επόμενο βήμα, η διαδικασία τοπικής αναζήτησης. Όπως γίνεται κατανοητό, το λογικό είναι να μη χρησιμοποιούμε τον ίδιο τελεστή και για τα δύο βήματα γιατί τότε αδυνατούμε να εξερευνήσουμε διαφορετικές γειτονιές λύσεων.

Αυτό σημαίνει ότι αν παγιδευτούμε σε κάποιο τοπικό βέλτιστο με κάποιον συγκεκριμένο τελεστή, δε θα μπορέσουμε να απεμπλακούμε στη διαδικασία διατάραξης αν χρησιμοποιούμε και εκεί τον ίδιο τελεστή. Για αυτό μια καλή προσέγγιση είναι να έχουμε παραπάνω τελεστές για τα βήματα της VNS έτσι ώστε να μπορούμε να εξερευνούμε μεγαλύτερο πλήθος γειτονιών.

3.2.3 Φάση Τοπικής Αναζήτησης

Σε αυτή τη φάση η μέθοδος αναζήτησης μεταβλητής γειτνίασης επικεντρώνεται σε μια συγκεκριμένη γειτονιά και τη ψάχνει εξαντλητικά μέχρι να βρει το τοπικό βέλτιστό της. Δηλαδή χρησιμοποιεί έναν συγκεκριμένο τελεστή, κατά προτίμηση διαφορετικό από αυτόν της φάσης διατάραξης, και προσπαθεί να βρει την καλύτερη ποιοτικά λύση που υπάρχει σε αυτή τη γειτονιά.

Υπάρχουν δύο κύριες τεχνικές, αυτή του first improvement και αυτή του best improvement. Η χρήση του best improvement σημαίνει ότι σαρώνεται ολόκληρη η γειτονιά και αν η καλύτερη ποιοτικά λύση που βρέθηκε είναι καλύτερη από την τρέχουσα, τότε γίνεται αποδεκτή. Αντίθετα, αν γίνει χρήση της τεχνικής first improvement, τότε με το που βρεθεί βελτιωμένη λύση της τρέχουσας, γίνεται αμέσως αποδεκτή. Αυτό κάνει την τεχνική first improvement σαφώς πιο γρήγορη από την best improvement, αλλά μπορεί αν συνεχίζαμε την αναζήτηση στη συγκεκριμένη γειτονιά να βρίσκαμε μια λύση αρκετά καλύτερη από αυτή που τελικά αποδεχτήκαμε.

Σύμφωνα με την εμπειρική μελέτη των Hansen, Mladenovic(Mladenovic and Hansen 2006), αν ξεκινήσουμε με μία τυχαία αρχική λύση τότε παίρνουμε καλύτερα αποτελέσματα με τη χρήση της τεχνικής first improvement. Αν όμως έχουμε πάρει την αρχική λύση ως αποτέλεσμα κάποιου αλγορίθμου αρχικοποίησης (constructive heuristic), τότε έχουμε γενικά καλύτερα και γρηγορότερα αποτελέσματα με τη χρήση της best improvement.

Ο Charles Heider(Heider 1972) πρότεινε το 1972 τον εξής κανόνα: η σάρωση στη γειτονιά της αρχικής λύσης να γίνεται με έναν κυκλικό και προκαθορισμένο τρόπο. Η τρέχουσα λύση ανανεώνεται αμέσως μόλις βρεθεί μια βελτιωμένη γειτονική λύση, άρα η τεχνική ανήκει στην κατηγορία του first improvement. Τέλος, ξεκινάμε να σαρώνουμε τη γειτονιά της νέας λύσης από τη θέση όπου διακόπηκε η προηγούμενη σάρωση.

Παρακάτω ακολουθούν οι αναπαραστάσεις των τεχνικών best improvement και first improvement σε μορφή ψευδοκώδικα(Mladenovic, Hansen, and Perez 2010) :

Algorithm 2 Best improvement local search

```
1: procedure BEST IMPROVEMENT(x)
2:     repeat
3:         x' ← x;
4:         x ← argmin_{y∈N(x)} f(y)
5:     until(f(x) ≥ f(x'));
6: end procedure
```

Algorithm 3 First improvement local search

```
1: procedure FIRST IMPROVEMENT(x)
2:     repeat
3:         x' ← x; i ← 0;
4:         repeat
5:             i ← i + 1;
6:             x ← argmin[f(x), f(x_i)], x_i ∈ N(x)
7:         until(f(x) < f(x_i)   or   i = |N(x)|);
8:     until(f(x) ≥ f(x'));
9: end procedure
```

Όπως φαίνεται και από την αλγοριθμική αναπαράσταση, αυτό που ξεχωρίζει τις δύο τεχνικές είναι το βάθος στο οποίο πραγματοποιούν την αναζήτηση. Από τη μία η τεχνική first improvement αποδέχεται αμέσως μια λύση που είναι έστω και λίγο καλύτερη της τρέχουσας, πράγμα που την κάνει γρηγορότερη και πιο αποτελεσματική, ειδικά όταν η αρχική λύση είναι τυχαία. Από την άλλη η τεχνική best improvement εξαντλεί όλη τη γειτονιά λύσεων και ουσιαστικά επιστρέφει το τοπικό βέλτιστο της συγκεκριμένης γειτονιάς λύσεων. Αν αυτό το τοπικό βέλτιστο είναι καλύτερο από την τρέχουσα λύση, τότε το αποδεχόμαστε και την αντικαθιστά. Η ολική αναζήτηση της γειτονιάς όμως σημαίνει ότι αυτή η τεχνική είναι μεν πιο χρονοβόρα, αλλά και ότι επιστρέφει κατά κανόνα καλύτερα αποτελέσματα όταν η αρχική λύση είναι προϊόν κάποιου αλγορίθμου αρχικοποίησης.

3.2.4 Αλλαγή Γειτονιάς

Αφού έχουμε δει τα βήματα διατάραξης και τοπικής αναζήτησης, ήρθε η ώρα να μελετήσουμε το βήμα αλλαγής γειτονιάς. Το τμήμα δηλαδή της VNS που αποφασίζει αν πρέπει να μεταβούμε για αναζήτηση σε διαφορετική γειτονιά λύσεων και το πραγματοποιεί αν χρειάζεται. Όπως είπαμε, οι γειτονιές αναγνωρίζονται ανάλογα με τον τελεστή που χρησιμοποιούμε για αναζήτηση, δηλαδή το σύνολο λύσεων με τον τελεστή swap αποτελεί διαφορετική γειτονιά από το σύνολο λύσεων με τον τελεστή 2-opt για παράδειγμα.

Υπάρχουν αρκετές προτάσεις και εκδοχές για το πώς μπορούμε να αλλαζουμε γειτονιές. Εκτός όμως από τον τρόπο που αλλάζουμε γειτονιές, κάτι ακόμα που έχει σημασία είναι και η σειρά με την οποία εξερευνούμε κάθε γειτονιά. Μπορεί για παράδειγμα να έχουμε καλύτερα αποτελέσματα

αν έχουμε ως πρώτη γειτονιά αναζήτησης μια γειτονιά λύσεων με τον τελεστή swap και ως δεύτερη γειτονιά χρησιμοποιούμε τον τελεστή 2-opt. Φυσικά κάτι τέτοιο μπορούμε να το γνωρίζουμε μόνο πειραματικά και διαφέρει ανάλογα με το πρόβλημα, οπότε δεν υπάρχει κάποιος σίγουρος κανόνας για το ποια είναι η σωστή σειρά των γειτονιών αναζήτησης.

Όταν η αλλαγή γειτονιάς γίνεται με ντετερμινιστικό τρόπο, δηλαδή δεν υπάρχει η έννοια της τυχαιότητας ή της πιθανότητας, τότε αυτή η μέθοδος ονομάζεται Variable Neighbourhood Descent-VND(Mladenovic, Hansen, Perez, and Brimberg 2010). Αυτή η μέθοδος έχει κάποιες παραλλαγές οι οποίες έχουν να κάνουν με την επιλογή της επόμενης γειτονιάς, αφού έχουμε βρει κάποιο τοπικό βέλτιστο. Όσον αφορά τον τρόπο με τον οποίο εκτελούνται αυτές οι αλλαγές, υπάρχουν αρκετές τεχνικές, αλλά τέσσερις είναι οι πιο διαδεδομένες και δοκιμασμένες. Παρουσιάζονται παρακάτω σύμφωνα με την έρευνα των Mladenovic, Hansen, Todosijevic, Hanafi (Mladenovic, Hansen, Todosijevic, and Hanafi 2016):

• **Basic VND**: Ξεκινάμε από τη βασική εκδοχή αλλαγής γειτονιάς του VND, όπου αρχικά επιλέγουμε τη σειρά των δομών γειτνίασης. Έπειτα αρχίζει η αναζήτηση και αν βρούμε κάποια καλύτερη ποιοτικά λύση από την τρέχουσα σε κάποια γειτονιά, τότε συνεχίζουμε την αναζήτηση στην πρώτη γειτονιά που έχουμε καθορίσει. Σε περίπτωση που δε βρεθεί καλύτερη λύση από την τρέχουσα τότε συνεχίζουμε την αναζήτηση στην επόμενη γειτονιά. Στην αλγοριθμική αναπαράσταση που ακολουθεί βλέπουμε ότι η μεταβλητή k είναι υπεύθυνη για την επιλογή της γειτονιάς.

Algorithm 4 NeighbourhoodChangeSequential

```
1: procedure NeighbourhoodChangeSequential(x, x', k)
2:     if f(x') < f(x) then
3:         x ← x';
4:         k ← 1;
5:     else
6:         k ← k + 1;
7:     end if
8: end procedure
```

• **Cyclic VND**: Σε αυτήν την εκδοχή δεν έχει σημασία αν βρέθηκε καλύτερη λύση της τρέχουσας. Η αναζήτηση συνεχίζεται πάντα από την επόμενη γειτονιά ανεξάρτητα με το αν υπήρξε βελτίωση ή όχι.

Algorithm 5 NeighbourhoodChangeCyclic

```
1: procedure NeighbourhoodChangeCyclic(x, x', k)
2:     k ← k + 1;
3:     if f(x') < f(x) then
4:         x ← x';
5:     end if
6: end procedure
```

• **Pipe VND**: Σύμφωνα με αυτήν την εκδοχή, αν βρεθεί βελτίωση στην τρέχουσα λύση σε μια συγκεκριμένη γειτονιά, τότε συνεχίζουμε την αναζήτηση σε αυτήν την ίδια γειτονιά. Δηλαδή παραμένουμε στην τρέχουσα γειτονιά λύσεων και δεν μετακινούμαστε.

Algorithm 6 NeighbourhoodChangePipe

```
1: procedure NeighbourhoodChangePipe(x, x', k)
2:     if f(x') < f(x) then
3:         x ← x';
4:     else
5:         k ← k + 1;
6:     end if
7: end procedure
```

• **Skewed VND**: Αυτή η παραλλαγή διαφέρει από τις προηγούμενες διότι έχει την εξής ιδιότητα: υπάρχει περίπτωση να επισκεφθούμε λύση η οποία δεν αποτελεί βελτίωση της τρέχουσας λύσης αλλά είναι χειρότερη. Ουσιαστικά γίνεται χρήση μιας συνάρτησης η οποία επιστρέφει τη διαφορά μεταξύ της τρέχουσας λύσης και της τοπικά βέλτιστης λύσης. Ουσιαστικά πρόκειται για την απόσταση μεταξύ αυτών των δύο λύσεων και επιτρέπει την κίνηση σε περιοχές λύσεων που βρίσκονται μακριά από την τρέχουσα λύση x. Με αυτόν τον τρόπο μπορούμε να αποφύγουμε την παγίδευση σε τοπικά βέλτιστα και να καταφέρουμε εν τέλει να βελτιώσουμε την τρέχουσα λύση, ακόμα και αν στην πορεία έχουμε δεχθεί κάποια χειρότερη ποιοτικά λύση.

Algorithm 7 NeighbourhoodChangeSkewed

```
1: procedure NeighbourhoodChangeSkewed(x, x', k, a)
2:     if f(x') − f(x) < ad(x', x) then
3:         x ← x';
4:         k ← 1;
5:     else
6:         k ← k + 1;
7:     end if
8: end procedure
```

Αυτό που κάνει όλες αυτές τις μεθόδους να διαφέρουν είναι το τι συμβαίνει στην μεταβλητή k όταν βρίσκουμε καλύτερη λύση της τρέχουσας, ή όταν τελειώνει η αναζήτηση. Αυτή η μεταβλητή στην πράξη επιλέγει ποια θα είναι η επόμενη γειτονιά αναζήτησης με σκοπό την αποφυγή παγίδευσης σε κάποιο τοπικό βέλτιστο. Ανάλογα με το κάθε πρόβλημα έχουμε διαφορετικά αποτελέσματα με τη χρήση κάθε μεθόδου αλλαγής γειτονιάς. Προφανώς δεν υπάρχει κάποια μέθοδος που είναι καλύτερη από τις υπόλοιπες σε όλες τις περιπτώσεις, αλλά σε διαφορετικά προβλήματα πρέπει να πειραματιζόμαστε μέχρι να βρούμε εμπειρικά ποια από αυτές τις μεθόδους επιστρέφει τα καλύτερα αποτελέσματα.

Εφόσον είδαμε τους βασικότερους τρόπους με τους οποίους αλλάζουμε γειτονιά στη μέθοδο

VND ας δούμε την ίδια τη μέθοδο VND η οποία όπως είπαμε προκύπτει όταν η αλλαγή γειτονιάς γίνεται με ντετερμινιστικό τρόπο.

Algorithm 8 VND

```
1: procedure VND(x, k'_max)
2:     repeat
3:         k ← 1;
4:         repeat
5:             x' ← argmin_{y∈N'_k(x)} f(x);
6:             NeighbourhoodChange(x, x', k);
7:         until k = k'_max;
8:     until no improvement found
9: end procedure
```

3.2.5 Διαφορετικές εκδοχές της VNS

Σε αυτό το σημείο, μιας και έχουμε δει όλα τα βασικά βήματα της μεθόδου VNS, θα δούμε ποιο είναι το κλασικό σχήμα της, καθώς και ποιες είναι οι πιο δημοφιλείς παραλλαγές της. Η βασική ιδέα όπως είπαμε και σε προηγούμενο μέρος της εργασίας είναι να υπάρχει μια φάση διατάραξης, μια φάση τοπικής αναζήτησης και μια φάση αλλαγής γειτονιάς. Παρακάτω παρουσιάζεται η βασική εκδοχή της VNS ακολουθούμενη από τις πιο δημοφιλείς παραλλαγές της.

• **Basic VNS**(Mladenovic and Hansen 1997): Η βασική εκδοχή λοιπόν έχει την ιδιότητα ότι συνδυάζει και ντετερμινιστικές αλλά και στοχαστικές, τυχαίες δηλαδή, μεθόδους για την αλλαγή της δομής γειτνίασης. Στο σημείο της τοπικής αναζήτησης χρησιμοποιούνται αιτιοκρατικές τεχνικές με τη μέθοδο best improvement. Όσον αφορά το στοχαστικό κομάτι, αυτό γίνεται αντιληπτό στο σημείο της μεθόδου που γίνεται η φάση διατάραξης. Εκεί επιλέγεται τυχαία ένα σημείο από τη k-ιοστή γειτονιά της τρέχουσας λύσης, με απώτερο σκοπό την αποφυγή του εγκλωβισμού σε τοπικό βέλτιστο. Σε μορφή ψευδοκώδικα φαίνεται το πώς συνυδάζονται οι αιτιοκρατικές με τις στοχαστικές τεχνικές:

Algorithm 9 Basic VNS

```
1: procedure BASIC VNS(x, k_max, t_max)
2:     t ← 0
3:     while t < t_max do
4:         k ← 1
5:         repeat
6:             x' ← Shake(x, k)
7:             x'' ← BestImprovement(x')
8:             x, k ← NeighbourhoodChange(x, x'', k)
9:         until k = k_max
10:        t ← CPUTime()
11:    end while
12:    return x
13: end procedure
```

• **General VNS**: Στη γενική εκδοχή αυτό που κάνουμε είναι να αντικαταστήσουμε το βήμα της τοπικής αναζήτησης με τη μέθοδο VND που είδαμε προηγουμένως. Έτσι προκύπτει ο παρακάτω αλγόριθμος ο οποίος έχει οδηγήσει σε πολύ καλά υπολογιστικά αποτελέσματα όπως φαίνεται για παράδειγμα στην έρευνα των Andreatta, Ribeiro(Andreatta and Ribeiro 2002). Όπως είναι φανερό και από το ψευδοκώδικα, η κυριότερη διαφορά είναι ότι έχουμε αντικαταστήσει το βήμα της τοπικής αναζήτησης με τη μέθοδο VND η οποία υπάρχει στον αλγόριθμο 8.

Algorithm 10 General VNS

```
1: procedure GENERAL VNS(x, k'_max, k_max, t_max)
2:     repeat
3:         k ← 1;
4:         repeat
5:             x' ← Shake(x, k);
6:             x'' ← VND(x', k'_max);
7:             NeighbourhoodChange(x, x'', k);
8:         until k = k_max;
9:         t ← CPUTime();
10:    until(t > t_max);
11: end procedure
```

• **Reduced VNS**: Αυτή η μέθοδος επιτυγχάνεται όταν δε χρησιμοποιούμε αιτιοκρατικές τεχνικές όπως η VND, αλλά αντιθέτως επιλέγουμε τυχαία σημεία από τη γειτονιά $N_k(x)$. Στη συνέχεια συγκρίνουμε τις τυχαίες τιμές με την τιμή της τρέχουσας λύσης και όταν υπάρχει βελτίωση, η τρέχουσα λύση ανανεώνεται. Σε αυτή τη μέθοδο η συνθήκη τερματισμού συνήθως καθορίζεται από το πόσος χρόνος έχει περάσει.

Algorithm 11 Reduced VNS

```
1: procedure REDUCED VNS(x, k_max, t_max)
2:     repeat
3:         k ← 1;
4:         repeat
5:             x' ← Shake(x, k);
6:             NeighbourhoodChange(x, x', k);
7:         until k = k_max;
8:         t ← CPUTime();
9:     until(t > t_max);
10: end procedure
```

• **Skewed VNS**(Mladenovic and Hansen 2001): Τέλος, η τελευταία παραλλαγή που θα μελετήσουμε είναι η Skewed VNS. Αυτή στο τμήμα της αλλαγής γειτονιάς χρησιμοποιεί την τεχνική της Skewed VND που είδαμε στην προηγούμενη ενότητα, στον αλγόριθμο 7. Έτσι προφανώς υπάρχει η περίπτωση επίσκεψης μιας λύσης που είναι χειρότερη της τρέχουσας αλλά επαρκώς διαφορετική, ανάλογα φυσικά με τις παραμέτρους που έχουμε θέσει στη συνάρτηση που επιστρέφει τη διαφορά μεταξύ της τρέχουσας και της τοπικά βέλτιστης λύσης.

Algorithm 12 Skewed VNS

```
1: procedure SKEWED VNS(x, k_max, t_max, a)
2:     repeat
3:         k ← 1; x_best ← x;
4:         repeat
5:             x' ← Shake(x, k);
6:             x'' ← FirstImprovement(x');
7:             KeepBest(x_best, x);
8:             NeighbourhoodChange(x, x'', k, a);
9:         until k = k_max;
10:        x ← x_best;
11:        t ← CPUTime();
12:    until(t > t_max);
13: end procedure
```

Με αυτήν την τελευταία παραλλαγή κλείνει το κεφάλαιο της VNS. Είδαμε τα βασικά βήματα της μεθόδου καθώς και τις σημαντικότερες εκδοχές της. Συνεχίζουμε με την πολυεναρκτήρια τοπική αναζήτηση.

3.3 Πολυεναρκτήρια τοπική αναζήτηση

Οι περισσότερες μέθοδοι τοπικής αναζήτησης χρειάζονται κάποιο εργαλείο με τη βοήθεια του οποίου ξεφεύγουν από τοπικά βέλτιστα. Ένας τρόπος για να γίνει αυτό είναι και η φάση διατάραξης που είδαμε προηγουμένως. Άλλος ένας τρόπος είναι να κάνουμε πολλαπλές επανεκκινήσεις όταν φτάνουμε σε ένα σημείο που δε βελτιώνεται για πολλές επαναλήψεις. Αυτή η τεχνική ονομάζεται πολυεναρκτήρια τοπική αναζήτηση (Levente and Andras 2009) και ουσιαστικά επαναλαμβάνει

πολλές φορές τον αλγόριθμο. Σε περίπτωση που η αρχική μας λύση παράγεται τυχαία, όπως στο πρόγραμμα που υλοποιήθηκε, τότε κάθε νέα έναρξη μπορεί να αποφέρει διαφορετικά αποτελέσματα. Με δεδομένο το γεγονός πως γίνονται εκατοντάδες επανεκκινήσεις το δευτερόλεπτο, η πιθανότητα να βρεθεί κάποια λύση, που η βελτίωσή της θα οδηγήσει σε καλύτερο τοπικό βέλτιστο από το ήδη υπάρχον, αυξάνεται σημαντικά. Αυτό σημαίνει ότι είναι πολύ πιθανόν σε μια επανεκκίνηση να βρεθεί ένα μονοπάτι λύσεων που μπορεί να οδηγήσει ακόμα και στο ολικό βέλτιστο.

Στο συγκεκριμένο πρόγραμμα, όπως θα φανεί και στο κεφάλαιο της υλοποίησης, η προσθήκη της πολυεναρκτήριας τοπικής αναζήτησης έγινε προς το τέλος, αλλά τα αποτελέσματα ήταν τόσο καλύτερα από τα ήδη υπάρχοντα, με αποτέλεσμα να αλλάξει η δομή του αλγορίθμου. Από εκεί που βασιζόταν καθαρά σε αναζήτηση μεταβλητής γειτνίασης, η κύρια μέθοδος που επιστρέφει τις καλύτερες ποιοτικά λύσεις στην τελική υλοποίηση του προγράμματος είναι η πολυεναρκτήρια τοπική αναζήτηση. Σε συνδυασμό μάλιστα με τεχνικές από την αναζήτηση μεταβλητής γειτνίασης, όπως για παράδειγμα η χρήση δύο γειτονιών αντί για μία, έχουμε τα καλύτερα ποιοτικά αποτελέσματα. Το πώς υλοποιήθηκε και λειτουργεί η πολυεναρκτήρια τοπική αναζήτηση μελετάται στο επόμενο κεφάλαιο.

Στη συνέχεια θα περάσουμε στα πρακτικά της εργασίας, όπου θα αναλυθεί ο σχεδιασμός και η υλοποίησή της.

ΚΕΦΑΛΑΙΟ 4

Σχεδιασμός και Υλοποίηση της Εφαρμογής

4.1 Σχεδιασμός της Εφαρμογής

4.1.1 Εργαλεία που χρησιμοποιήθηκαν

Για την υλοποίηση της εφαρμογής χρησιμοποιήθηκε το Visual Studio και η γλώσσα προγραμματισμού που επιλέχθηκε είναι η C Sharp. Ο λόγος που επιλέχθηκε αυτή η γλώσσα είναι ότι έχω μεγάλη οικειότητα μαζί της και εκτός αυτού αποτελεί μια άκρως ανερχόμενη και εξελισσόμενη γλώσσα η οποία δέχεται συνεχώς αναβαθμίσεις. Συν τοις άλλοις, το Visual Studio επιτρέπει την εύκολη υλοποίηση GUI (graphical user interface), με κουμπιά, λίστες και άλλα εργαλεία, που διευκολύνουν τον χρήστη στο χειρισμό του προγράμματος. Αυτός ήταν ακόμα ένας λόγος που προτίμησα τα συγκεκριμένα εργαλεία, έτσι ώστε αν ήθελε κάποιος άλλος χρήστης να τρέξει το πρόγραμμα να μην αντιμετώπιζε κάποια δυσκολία.

4.1.2 Βασική ιδέα

Η βασική μεθοδολογία που επιλέχθηκε για την επίλυση των στιγμιοτύπων του QAP ήταν η αναζήτηση μεταβλητής γειτνίασης που μελετήθηκε στο προηγούμενο κεφάλαιο. Ως εκ τούτου το πρόγραμμα ξεκίνησε με σκοπό την εκτέλεση των βημάτων της μεθόδου VNS. Η επίλυση λοιπόν των στιγμιοτύπων έπερεπε να επιτευχθεί με την εξής σειρά: αρχικά θα υπήρχε η φάση διατάραξης όπου το διάνυσμα της λύσης υποβάλλεται σε κάποιες αλλαγές με βάση συγκεκριμένους τελεστές. Έπειτα έρχεται η φάση της τοπικής αναζήτησης, όπου προσπαθούμε να βρούμε κάποια καλύτερη λύση από την τρέχουσα. Τέλος, υπάρχει η φάση αλλαγής γειτονιάς, όπου αποφασίζουμε αν πρέπει να μεταβούμε σε άλλη γειτονιά λύσεων καθώς και ποια πρέπει να είναι η γειτονιά αυτή.

Στην αρχή, για λόγους απλότητας αποφασίστηκε να υπάρχει μια μόνο γειτονιά για τη φάση τοπικής αναζήτησης, έτσι ώστε να μη γίνει το πρόγραμμα πιο πολύπλοκο από όσο είναι απαραίτητο. Η φάση διατάραξης από την άλλη, πρέπει να έχει παραπάνω τελεστές και κατά προτίμηση διαφορετικούς από τον τελεστή της τοπικής αναζήτησης. Αυτό γίνεται για να μην επισκεφτόμαστε τις ίδιες γειτονιές και στις δύο φάσεις, με απώτερο σκοπό την αποφυγή της κύκλωσης και της παγίδευσης

σε τοπικά βέλτιστα.

Το ότι δουλεύουμε μόνο με μια γειτονιά σημαίνει ότι η φάση της αλλαγής γειτονιάς δεν έχει νόημα ύπαρξης σε αυτό το σημείο του προγράμματος. Προφανώς αργότερα θα προστεθούν παραπάνω γειτονιές μέχρι να αποφασιστεί πειραματικά ποια είναι η πιο αποτελεσματική προσέγγιση.

Όσον αφορά τον υπολογισμό της αρχικής λύσης, η μέθοδος που επιστρέφει τα πιο γρήγορα αποτελέσματα είναι αυτή της τυχαίας αρχικής λύσης. Αυτό σημαίνει ότι το διάνυσμα της λύσης ανακατεύεται τυχαία και έτσι ο χρόνος που χρειάζεται για τον υπολογισμό είναι μηδενικός. Η επιλογή όμως τυχαίας αρχικής λύσης μας οδηγεί στην εξής απόφαση: η τοπική αναζήτηση είναι καλύτερο να υλοποιηθεί με βάση τη μέθοδο first improvement αντί για αυτή της best improvement. Όπως επισημάνθηκε σε προηγούμενο κεφάλαιο, με τυχαία αρχική λύση έχουμε πιο γρήγορα και πιο καλά ποιοτικά αποτελέσματα με τη μέθοδο first improvement. Αντίθετα αν χρησιμοποιηθεί κάποιος αλγόριθμος που υπολογίζει μια σχετικά καλή αρχική λύση, με κάποιο χρονικό κόστος όμως, τότε η μέθοδος best improvement εμφανίζεται πιο αποτελεσματική.

4.1.3 Βοηθητικές συναρτήσεις

Προτού περάσουμε στο ευρετικό μέρος του προγράμματος, πρέπει να αναφερθούν ορισμένες συναρτήσεις οι οποίες είναι βασικές για τη λειτουργία της εφαρμογής αλλά δε σχετίζονται με τη μεθοδολογία. Τέτοιες είναι οι συναρτήσεις που αποθηκεύουν τα δεδομένα του προβλήματος, όπως για παράδειγμα ένας parser. Παρακάτω παρουσιάζονται οι συναρτήσεις που σχεδιάστηκαν αρχικά ανεξάρτητα από τη μεθοδολογία επίλυσης των στιγμιοτύπων.

• parser : Αυτή η συνάρτηση ανοίγει το αρχείο με τα δεδομένα από τη βιβλιοθήκη QAPLIB και αποθηκεύει τους δύο πίνακες με τις ροές και τις αποστάσεις σε δύο μεταβλητές flows και distances αντίστοιχα. Επίσης χρειάζεται να αποθηκευτεί και το μέγεθος καθώς και το διάνυσμα της λύσης του κάθε στιγμιότυπου. Ακολουθεί ενδεικτικά το τμήμα αυτής της συνάρτησης που αποθηκεύει τις αποστάσεις σε μορφή ψευδοκώδικα. Αντιστοίχως αποθηκεύονται και οι ροές.

Algorithm 13 parser

```
1: procedure PARSER(file, distances)
2:     for i = 0; i < size; i++ do
3:         for j = 0; j < size; j++ do
4:             distances[i, j] = Parse(s[j + size * i]);
5:         end for
6:     end for
7: end procedure
```

Όπως φαίνεται και από τον ψευδοκώδικα, γίνεται χρήση της έτοιμης συνάρτησης που προσφέρει η C Sharp, Parse. Με αυτόν τον τρόπο δημιουργείται ένας πίνακας ο οποίος περιέχει τις αποστάσεις μεταξύ των εγκαταστάσεων.

• randomizer : Αυτή η συνάρτηση πρακτικά δέχεται το διάνυσμα της λύσης του στιγμιότυπου και το επιστρέφει τυχαιοποιημένο. Επειδή δεν υπήρχε έτοιμη συνάρτηση της C Sharp, δανείστηκα την υλοποίησή της από τον ακόλουθο σύνδεσμο: `http://csharphelper.com/blog/2014/07/randomize-arrays-in-c/`. Εκτός από την ίδια την υλοποίηση, ο σύνδεσμος προσφέρει και μια ανάλυση της συνάρτησης, η οποία μας αποδεικνύει ότι η συνάρτηση διαλέγει με ομοιόμορφη τυχαιότητα το κάθε στοιχείο. Ουσιαστικά υπάρχει πιθανότητα $1/N$ για οποιοδήποτε στοιχείο να καταλήξει σε οποιαδήποτε θέση στο νέο, τυχαίο διάνυσμα, με το N να αντιπροσωπεύει το μέγεθος του διανύσματος. Σε μορφή ψευδοκώδικα η συνάρτηση τυχαιοποίησης έχει την εξής μορφή:

Algorithm 14 randomizer

```
1: procedure RANDOMIZER(vector)
2:     rand = newRandom();
3:     for i = 0; i < vector.length; i++ do
4:         j = rand.Next(i, items.Length);
5:         temp = items[i];
6:         items[i] = items[j];
7:         items[j] = temp;
8:     end for
9: end procedure
```

• calculate fitness : Η συνάρτηση αξιολόγησης, πρακτικά η συνάρτηση υπολογισμού του κόστους. Στη γενική της μορφή είναι η υλοποίηση της εξίσωσης 2.1. Αργότερα θα δούμε πώς μπορούμε να τη μεταβάλλουμε έτσι ώστε να επιταχυνθεί η διαδικασία υπολογισμού της, όταν το στιγμιότυπό μας έχει κάποιες συγκεκριμένες ιδιότητες. Για την ώρα όμως, η βασική της λειτουργία αποτυπώνεται ως εξής:

Algorithm 15 calculateFitness

```
1: procedure CALCULATEFITNESS(distances, flows, solution)
2:     fitness = 0;
3:     for i = 0; i < size; i++ do
4:         for j = 0; j < size; j++ do
5:             a = flows[i, j];
6:             b = distances[solution[i] − 1, solution[j] − 1];
7:             fitness+ = a ∗ b;
8:         end for
9:     end for
10: end procedure
```

4.2 Τεχνικές που Δοκιμάστηκαν

Όπως είδαμε και πριν, ο κύριος κορμός του προγράμματος είχε τα εξής βήματα: αρχικά γινόταν η μεταφορά των δεδομένων απο τα αρχεία της βιβλιοθήκης στγμιοτύπων, σε μεταβλητές πινάκων.

Έπειτα ξεκινούσαμε με μια τυχαία λύση και άρχιζε η λειτουργία της μεθόδου VNS. Αυτή αποτελούνταν από τα βασικά βήματα, της διατάραξης και της τοπικής αναζήτησης. Υπενθυμίζω ότι στην αρχή υπήρχε μόνο μια γειτονιά, δηλαδή μόνο ένας τελεστής τοπικής αναζήτησης, ο swap, οπότε το βήμα αλλαγής γειτονιάς ήταν ανούσιο. Όσον αφορά τη φάση της διατάραξης, αυτή γινόταν με τον τελεστή 2-opt. Η χρήση διαφορετικών τελεστών βοηθάει στην αποφυγή κύκλωσης και διευκολύνει την απεγκλώβιση από τοπικά βέλτιστα.

Αυτή λοιπόν ήταν η αρχική μορφή του προγράμματος, η οποία όπως είναι λογικό δέχεται εμπλουτισμό και βελτιώσεις. Μια πολύ σημαντική τεχνική που χρησιμοποιείται στο QAP είναι αυτή της γρήγορης ανανέωσης της τιμής της συνάρτησης αξιολόγησης. Είχαμε δει σε προηγούμενο κεφάλαιο ότι ο υπολογισμός αυτής της συνάρτησης αποτελεί ουσιαστικά το δυσκολότερο υπολογιστικό κομμάτι και πως ο περισσότερος χρόνος σπαταλάται εκεί. Οτιδήποτε λοιπόν μπορούμε να κάνουμε για να επιταχύνουμε αυτή τη διαδικασία είναι προς όφελός μας.

Συγκεκριμένα, όταν χρησιμοποιείται ο τελεστής swap στο διάνυσμα της λύσης, δηλαδή όταν απλά δύο στοιχεία ανταλλάζουν θέση, τότε υπάρχει ένας μαθηματικός τύπος ο οποίος μας επιτρέπει να υπολογίζουμε τη διαφορά που προκαλείται στην τιμή της συνάρτησης αξιολόγησης. Αυτό πρακτικά σημαίνει ότι δε χρειάζεται να υπολογίσουμε εξαρχής ολόκληρη τη συνάρτηση, αλλά πολύ πιο γρήγορα, να βρούμε τη διαφορά και να την αφαιρέσουμε από την τρέχουσα τιμή. Αυτή η τεχνική μειώνει σημαντικά το χρόνο υπολογισμού και αποτελεί επίσης το λόγο που ο τελεστής τοπικής αναζήτησης επιλέχθηκε να είναι ο τελεστής swap. Προφανώς επειδή στη φάση τοπικής αναζήτησης γίνονται εντατικά οι αλλαγές στο διάνυσμα της λύσης, είναι επιθυμητό να χρησιμοποιούμε εκεί τον τελεστή για τον οποίο διαθέτουμε ένα είδος γρήγορης ανανέωσης (fast update).

Αυτό που θέλουμε είναι να επιταχύνουμε τη διαδικασία εύρεσης γειτονικών λύσεων στο βήμα της τοπικής αναζήτησης. Αντί λοιπόν να βαίνουμε σε επαναϋπολογισμό της τιμής της εξίσωσης 2.1, υπολογίζουμε τη διαφορά με βάση τον ακόλουθο τύπο (Misevicius 2011):

$$\Delta_z(\pi, i, j) = (a_{ii} - a_{jj})(b_{\pi(j)\pi(j)} - b_{\pi(i)\pi(j)}) + (a_{ij} - a_{ji})(b_{\pi(j)\pi(i)} - b_{\pi(i)\pi(j)})$$
$$+ \sum_{k=1, k\neq i,j}^{n} [(a_{ik} - a_{jk})(b_{\pi(j)\pi(k)} - b_{\pi(i)\pi(k)}) + (a_{ki} - a_{kj})(b_{\pi(k)\pi(j)} - b_{\pi(k)\pi(i)})]$$

Χάρη σε αυτόν τον τύπο καταφέρνουμε και κερδίζουμε πολύτιμο χρόνο στον υπολογισμό της τιμής της συνάρτησης αξιολόγησης. Το μόνο που μένει να κάνουμε είναι να αφαιρέσουμε τη διαφορά που θα βρούμε από την τιμή αξιολόγησης της τρέχουσας λύσης και έτσι επιταχύνεται η φάση τοπικής αναζήτησης.

Δυστυχώς όμως, αυτός ο τύπος είναι αποκλειστικά για τον τελεστή swap και δεν υπάρχει παρόμοια τεχνική για τους άλλους τελεστές. Έτσι όταν χρειάζεται να χρησιμοποιήσουμε διαφορετικό τελεστή, για παράδειγμα τον 2-opt, είμαστε αναγκασμένοι να υπολογίσουμε τη συνάρτηση

αξιολόγησης από την αρχή.

Επιπρόσθετα, υπάρχει άλλη μια τεχνική που μπορούμε να εφαρμόσουμε η οποία θα μας επιφέρει επιτάχυσνη στον υπολογισμό της τιμής της συνάρτησης αξιολόγησης. Όταν ένας από τους δύο πίνακες A, B ή και οι δύο είναι συμμετρικοί, τότε ο τύπος 4.2 μπορεί να γίνει ακόμα αποδοτικότερος (Fischetti, Monaci, and Salvagnin 2012). Αν για παράδειγμα μόνο ο πίνακας A είναι συμμετρικός, τότε μπορούμε να μετατρέψουμε και τον πίνακα B σε συμμετρικό και έτσι να είμαστε σε θέση να χρησιμοποιήσουμε τον ακόλουθο, πιο γρήγορο σε υπολογισμό τύπο (Merz and Freisleben 2000):

$$\Delta_z(\pi, i, j) = \sum_{k=1, k\neq i,j}^{n} (a'_{ik} - a'_{jk})(b_{\pi(j)\pi(k)} - b_{\pi(i)\pi(k)})$$

Αν όμως κανένας από τους δύο πίνακες δεν είναι συμμετρικός τότε δεν μπορούμε να τους μετατρέψουμε, διότι πρέπει αναγκαστικά ο ένας από τους δύο να πληροί τις προϋποθέσεις συμμετρικότητας. Από τη βιβλιοθήκη στιγμιοτύπων υπάρχουν στιγμιότυπα από όλες τις κατηγορίες, δηλαδή κάποια στιγμιότυπα έχουν και τους δύο πίνακες συμμετρικούς, άλλα μόνο τον έναν από τους δύο και άλλα κανέναν από τους δύο. Για αυτό το λόγο χρειάστηκε να χρησιμοποιηθεί ένα σύνολο συναρτήσεων το οποίο είχε ως σκοπό τον έλεγχο συμμετρικότητας και όπου ήταν αναγκαίο και δυνατό, την μετατροπή κάποιου πίνακα σε συμμετρικό.

Πρώτα απ'ολα, το βασικό είναι να δούμε αν ένας πίνακας είναι συμμετρικός. Αυτό πραγματοποιείται στο πρόγραμμα με τη συνάρτηση isSymmetric η οποία δέχεται ως δεδομένα έναν πίνακα και επιστρέφει μια τιμή true ή false. Παρακάτω ακολουθεί σε μορφή ψευδοκώδικα:

Algorithm 16 isSymmetric

```
1: procedure ISSYMMETRIC(matrix)
2:     transMatrix ← transpose(matrix);
3:     for i = 0; i < size; i++ do
4:         for j = 0; j < size; j++ do
5:             if matrix[i, j] ≠ transMatrix[i, j] then
6:                 return false;
7:             end if
8:         end for
9:     end for
10: end procedure
```

Παρατηρούμε ότι γίνεται χρήση της συνάρτησης transpose, η οποία αντιμεταθέτει τα στοιχεία ενός πίνακα με βάση τη διαγώνιό του. Η συνάρτηση isSymmetric ουσιαστικά ελέγχει αν μετά από αυτή τη μετάθεση ο πίνακας παραμένει ίδιος. Αν δεν έχει αλλάξει κάτι, τότε προφανώς ο πίνακας ήταν συμμετρικός. Το πώς δουλεύει η συνάρτηση transpose φαίνεται στο ακόλουθο τμήμα ψευδοκώδικα:

Algorithm 17 transpose

```
1: procedure TRANSPOSE(matrix)
2:     for i = 0; i < size; i++ do
3:         for j = 0; j < size; j++ do
4:             transMatrix[i, j] ← matrix[j, i];
5:         end for
6:     end for
7:     return transMatrix;
8: end procedure
```

Αν ο πίνακας μετά τη μετάθεση είναι ο ίδιος, τότε πρόκειται για συμμετρικό πίνακα. Τι γίνεται όμως αν ο πίνακας δεν είναι εξαρχής συμμετρικός· Όπως είπαμε, αν κανένας από τους δύο πίνακες δεν είναι συμμετρικός, τότε δεν μπορούμε να κάνουμε κάτι. Αν όμως ο ένας από τους δύο πληροί τις προϋποθέσεις συμμετρικότητας σύμφωνα με τη συνάρτηση 16, τότε μπορούμε να μετατρέψουμε και τον άλλο, μη συμμετρικό πίνακα, σε συμμετρικό. Για την επίτευξη αυτής της μετατροπής χρησιμοποιούμε τη συνάρτηση makeSymmetric. Οι λειτουργίες της φαίνονται στο επόμενο σχήμα:

Algorithm 18 makeSymmetric

```
1: procedure MAKESYMMETRIC(matrix)
2:     transMatrix ← transpose(matrix);
3:     for i = 0; i < size; i++ do
4:         for j = 0; j < size; j++ do
5:             symMatrix[i, j] = matrix[i, j] + transMatrix[i, j];
6:         end for
7:     end for
8:     return symMatrix;
9: end procedure
```

Με αυτόν τον τρόπο λοιπόν γίνεται εφικτή η μετατροπή ενός πίνακα από μη συμμετρικό σε συμμετρικό. Όταν δουλεύουμε με συμμετρικούς πίνακες ο υπολογισμός της συνάρτησης αξιολόγησης γίνεται πολύ πιο γρήγορα όταν χρησιμοποιούμε τον τελεστή swap αφού χρειάζεται απλά να υπολογιστεί η διαφορά της νέας λύσης από την τρέχουσα με βάση τον τύπο 4.2

Αυτές οι δύο τεχνικές, η γρήγορα ανανέωση με τον τελεστή swap και η εκμετάλλευση της συμμετρικής ιδιότητας, απέδωσαν πολύ καλύτερα από την αρχική εκδοχή του προγράμματος που έκανε απλά τα βασικά βήματα της μεθόδου VNS. Χάρη σε αυτές επιτεύχθηκαν καλύτεροι χρόνοι στην επίλυση των στιγμιοτύπων και ουσιαστικά αναβάθμισαν την αποτελεσματικότητα του προγράμματος.

Υπήρχε όμως άλλη μια ενέργεια η οποία απέδωσε πολύ σημαντική βελτίωση και υπάρχει ακόμα και στην τελική έκδοση του προγράμματος. Το ιδανικό θα ήταν να μπορούμε να μετατρέψουμε τη σειριακή εκτέλεση του προγράμματος σε παράλληλη, έτσι ώστε να έχουμε περισσότερες εκτελέσεις στον ίδιο χρόνο και ως αποτέλεσμα καλύτερη αξιοποίηση του χρόνου. Επειδή όμως στη C Sharp δεν υπάρχει αντίστοιχο πακέτο όπως στη C ++, που έχει το OPENMP για παράδειγμα, αναγκαστικά

στρεφόμαστε σε άλλες τεχνικές.

Αυτό που προσφέρει το Visual Studio είναι η επιλογή να γίνει αυτόματα βελτιστοποίηση του κώδικα. Αυτή η επιλογή υπάρχει στο μενού για τη μεταγλώττιση του προγράμματος και έχει κάποιες συνέπειες στο τελικό πρόγραμμα. Αρχικά δε συνιστάται για την αποσφαλμάτωση καθώς δεν επιτρέπει στο χρήστη να σταματάει το πρόγραμμα στα σημεία που θέλει με break points. Όσον αφορά όμως την ταχύτητα του προγράμματος, αυτή παρουσιάζει πολύ σημαντική βελτίωση. Αυτό συμβαίνει διότι οι περισσότερες δομές επαναλήψεων παραλληλοποιούνται αυτόματα. Εκτός από αυτό οι μεταβλητές παραμένουν αποθηκευμένες στα μητρώα καθ'όλη τη διάρκεια εκτέλεσης του προγράμματος, πράγμα που σημαίνει πως η εκτέλεση γίνεται γρηγορότερη. Αυτά και μερικά ακόμα αποτελέσματα της βελτιστοποίησης περιγράφονται στον επίσημο σύνδεσμο της Microsoft: `https://docs.microsoft.com/en-us/cpp/build/reference/o-options-optimize-code?view=vs-2019`.

Συγκεκριμένα στο παρόν πρόγραμμα όταν γινόταν εκτέλεση με την επιλογή βελτιστοποίησης ενεργή, παρατηρούνταν ακόμα και διπλάσιος αριθμός επαναλήψεων στο τμήμα τοπικής αναζήτησης. Αυτό σημαίνει ότι στον ίδιο χρόνο είχαμε τη δυνατότητα να ελέγξουμε πάνω από τις διπλάσιες γειτονικές λύσεις σε σχέση με την εκτέλεση του προγράμματος χωρις την επιλογή της βελτιστοποίησης. Το γεγονός αυτό κάνει αυτή την επιλογή τουλάχιστον το ίδιο σημαντική με τις τεχνικές που είδαμε παραπάνω. Αν και δεν αποτελεί κάποιο αλγοριθμικό τέχνασμα, μας επιτρέπει να αυξήσουμε πολύ σημαντικά την αποτελεσματικότητα του προγράμματός μας.

Αυτές ήταν οι αλλαγές των οποίων η υιοθέτηση στο πρόγραμμα επέφερε τις σημαντικότερες βελτιώσεις. Προφανώς όμως υπήρχαν και άλλες αλλαγές οι οποίες είχαν από μικρό έως και αρνητικό αντίκτυπο. Στη συνέχεια θα μελετήσουμε αυτές τις αλλαγές και θα δούμε ποιες τελικά απέμειναν στην τελική εκδοχή του προγράμματος.

Ξεκινώντας, μια αξιοσημείωτη τεχνική είναι αυτή της αυτόματης προσαρμογής του προγράμματος. Όπως έχουμε ήδη πει αρκετές φορές, όταν ξεκινάμε από τυχαία λύση είναι προτιμότερη η χρήση της μεθόδου first improvement και όταν έχουμε μια σχετικά καλή αρχική λύση προτιμούμε τη χρήση της μεθόδου best improvement. Αυτό που επιχειρήθηκε ήταν να υπάρχει στο πρόγραμμα η ιδιότητα εναλλαγής σε αυτές τις δύο τεχνικές ανάλογα με το πόσο τοις εκατό απέχει η τρέχουσα λύση από την καλύτερη γνωστή λύση. Έτσι λοιπόν ενώ ξεκινάμε με τυχαία αρχική λύση και αναζητούμε βελτίωση με first improvement, στη συνέχεια αν αποκτήσουμε μια καλή τρέχουσα λύση, γίνεται εναλλαγή της αναζήτησης με best improvement.

Επειδή όμως τα στιγμιότυπα διαφέρουν μεταξύ τους και δεν έχει νόημα να διαλέξουμε αυθαίρετα μια τιμή ποσοστιαίας διαφοράς από την καλύτερη γνωστή λύση, μπορεί ο χρήστης να διαλέξει ένα ποσοστό λάθους, για παράδειγμα μια τιμή του τριάντα τοις εκατό διαφορά από τη βέλτιστη και όταν η τρέχουσα λύση πέσει κάτω από αυτό το κατώφλι, τότε η αναζήτηση πάυει να γίνεται με first improvement και συνεχίζει με best improvement.

Δυστυχώς αν και σε θεωρητικό επίπεδο φαίνεται να είναι μια τεχνική που προκαλεί βελτίωση,

στην πράξη τα αποτελέσματα δεν ήταν ενθαρρυντικά για την ενσωμάτωσή της στην τελική εκδοχή του προγράμματος. Τα αποτελέσματα που έβγαιναν δεν ήταν χειρότερα αλλά ούτε και καλύτερα, οπότε χάριν απλότητας η τεχνική αυτή δε θεωρήθηκε ότι απέδιδε.

Η μεγαλύτερη πρόκληση σε αυτό το πρόβλημα ήταν η επίτευξη απεγκλωβισμού από τα τοπικά βέλτιστα. Αυτό υποτίθεται ότι επιτυγχάνεται στη φάση διατάραξης αλλά σε ορισμένα στιγμιότυπα, η βασική εκδοχή της VNS που χρησιμοποιούνταν δεν κατάφερνε να ξεφύγει και κατέληγε να παγιδεύεται σε κάποιο τοπικό βέλτιστο. Για να μπορέσουμε να ξεφύγουμε από τοπικό βέλτιστο το πρώτο πράγμα που πρέπει να γίνει είναι η σωστή ανίχνευση της παγίδευσής μας. Πρέπει δηλαδή να γνωρίζουμε ότι έχουμε κολλήσει σε τοπικό βέλτιστο έτσι ώστε να εφαρμόσουμε τις κατάλληλες τεχνικές για να ξεφύγουμε.

Σε αυτό βοήθησε η ακόλουθη τεχνική, των αποτυχημένων επαναλήψεων. Εάν μετά από κάποιο συγκεκριμένο αριθμό επαναλήψεων δεν έχει βρεθεί καλύτερη λύση της τρέχουσας, τότε αυτό σημαίνει ότι υπάρχει μεγάλη πιθανότητα να έχουμε παγιδευτεί σε τοπικό βέλτιστο. Το ίδιο μπορεί να ανιχνευτεί και με άλλο τρόπο, αντί δηλαδή να μετράμε αποτυχημένες επαναλήψεις, να έχουμε ως κριτήριο το χρόνο. Δηλαδή αν μετά από κάποια δευτερόλεπτα δεν έχουμε βελτιώσει την τρέχουσα λύση τότε πρέπει να εφαρμόσουμε τεχνικές απεγκλώβισης. Όμως η κλήση συναρτήσεων για μέτρηση χρόνου είναι καλό να περιορίζονται καθώς είναι οι ίδιες τους χρονοβόρες. Έτσι λοιπόν επιλέχθηκε να ανιχνεύονται τα τοπικά βέλτιστα με την τεχνική των αποτυχημένων επαναλήψεων.

Όπως είναι προφανές, ο αριθμός αυτών των επαναλήψεων μηδενίζεται αν υπάρξει έστω και μια βελτίωση. Όσον αφορά το μέγεθός του, αυτό είναι άρρηκτα συνδεδεμένο με το πλήθος του κάθε στιγμιότυπου και με τον επιτρεπόμενο χρόνο εκτέλεσης του προγράμματος. Τα μικρά στιγμιότυπα με μέγεθος δώδεκα για παράδειγμα επιτρέπουν πολύ λιγότερες επαναλήψεις από τα μεγάλα στιγμιότυπα. Συγκεκριμένα στο πρόγραμμα ήταν τα χιλιοστά των δευτερολέπτων του επιτρεπόμενου χρόνου εκτέλεσης επί πέντε, ενώ για στιγμιότυπα με μέγεθος μεγαλύτερο του εκατό τα χιλιοστά πολλαπλασιαζόντουσαν επί του δύο. Αρχικά στο πρόγραμμα μπορούσε ο χρήστης να αποφασίσει για τον αριθμό των αποτυχημένων επαναλήψεων, αλλά αργότερα αποφασίστηκε αυτό να γίνεται αυτόματα έτσι ώστε να μη χρειάζεται ο χρήστης να επεμβαίνει σε κάθε παράμετρο.

Όσον αφορά την επίδοση αυτής της τεχνικής, εξαρτάται άμεσα από το τι κάνουμε αφού εντοπίσουμε ότι έχουμε παγιδευτεί. Η τεχνική των αποτυχημένων επαναλήψεων είναι ουσιαστικά ένα εργαλείο ανίχνευσης των τοπικών βέλτιστων. Το τι ενέργεια κάνουμε εφόσον το εντοπίσουμε αυτό είναι άλλο θέμα. Μπορούμε για πράδειγμα να αλλάξουμε τελεστή στη φάση διατάραξης ή και στη φάση τοπικής αναζήτησης, αν και τότε χάνουμε τη γρήγορη ανανέωση που υπάρχει μόνο για τον τελεστή swap. Δοκιμάστηκαν και οι δύο αυτές παραλλαγές όμως δεν υπήρξε αισθητή βελτίωση με καμία από αυτές. Η μεγάλη βελτίωση εμφανίστηκε όταν κυρίως λόγω ερευνητικής περιέργειας αποφάσισα να ξαναξεκινώ με νέα, τυχαία λύση κάθε φορά που παγιδευόταν το πρόγραμμα σε κάποιο τοπικό βέλτιστο. Το πώς ακριβώς συνέβη αυτό θα αναλυθεί περαιτέρω στο επόμενο υποκεφάλαιο

που έχει να κάνει με την τελική εκδοχή του προγράμματος.

Συνεχίζοντας με τις τεχνικές που δοκιμάστηκαν, μια ακόμα προσέγγιση ήταν η εναλλαγή του τελεστή στη φάση διατάραξης σε κάθε επανάληψη. Όπως έχουμε δει στη φάση διατάραξης υπάρχει μια παράμετρος k, η οποία επιλέγει μια τυχαία γειτονική λύση με βάση κάποιο τελεστή. Σε αυτή τη φάση του προγράμματος επιλέχθηκε αυτός ο τελεστής να είναι ο 2-opt. Άρα σε κάθε φάση διατάραξης η γειτονιά που επιλεγόταν έβγαινε με βάση αυτόν τον τελεστή. Σε μια προσπάθεια να ενταντικοποιηθεί η φάση διατάραξης δοκιμάστηκε η ακόλουθη τεχνική: όσο άλλαζε η παράμετρος k, άλλαζε και ο τελεστής στη φάση διατάραξης. Για παράδειγμα για $k = 1$ γινόταν χρήση του τελεστή 2-opt, για $k = 2$ του τελεστή swap και για $k = 3$ του τελεστή relocate ή shift right. Όλο αυτό έγινε με σκοπό τη διευκόλυνση αποφυγής των τοπικών βέλτιστων και τα αποτελέσματα ήταν λίγο πιο ενθαρρυντικά από πριν. Δεν υπήρξε μεγάλη βελτίωση αλλά φαινόταν ότι το πρόγραμμα δεν κολλούσε τόσο εύκολα σε τοπικά βέλτιστα.

Στην προγραμματιστική της μορφή η τεχνική αυτή είναι πολύ εύκολο να υλοποιηθεί. Έστω ότι έχουμε τέσσερις τελεστές για τη φάση διατάραξης και πρέπει να επιλέγουμε ποιος θα χρησιμοποιηθεί αναλόγως με την παράμετρο k. Αυτό που έχουμε να κάνουμε είναι να περνάμε ως παράμετρο στη συνάρτηση shaking έναν ακέραιο και αυτή να αποφασίζει ποιος τελεστής πρέπει να χρησιμοποιηθεί. Αν στείλουμε ως παράμετρο τη μονάδα θα χρησιμοποιηθεί ο πρώτος τελστής, αν στείλουμε το δύο ο δεύτερος και ούτω καθεξής. Επειδή όμως το k παίρνει τιμές μεγαλύτερες από το πλήθος των τελεστών, χρησιμοποιούμε το υπόλοιπο της διαίρεσης του k με το πλήθος των τελεστών ως την παράμετρο που παίρνει η συνάρτηση shaking. Για παράδειγμα αν το k φτάσει να έχει τιμή 15 και το πλήθος τελεστών είναι το 4, τότε στέλνουμε στη συνάρτηση shaking την τιμή $15 mod(4) + 1 = 4$. Ο λόγος που προσθέτουμε τη μονάδα και το αποτέλεσμα βγαίνει 4, δηλαδή ο τέταρτος τελεστής, είναι επειδή όταν βγαίνει μηδέν υπονοείται ότι πρόκειται για τον πρώτο τελεστή. Έτσι προσθέτουμε $+1$ στο αποτέλεσμα του mod για να μην υπάρχει πρόβλημα.

Συνεχίζοντας με τις αλλαγές στη φάση της διατάραξης και στην προσπάθεια να γίνεται πιο έντονα για να απελευθερωνόμαστε από τα τοπικά βέλτιστα, δοκιμάστηκε να χρησιμοποιούνται περισσότεροι τελεστές σε αυτή τη φάση. Δηλαδή αντί να έχουμε μόνο έναν τελεστή, να υπάρχει μεγαλύτερο πλήθος και να χρησιμοποιούνται ο ένας μετά τον άλλον προτού τελειώσει η φάση διατάραξης. Για παράδειγμα ξεκινάμε με τον 2-opt και αμέσως μόλις επιστραφεί το διάνυσμα της λύσης εφαρμόζουμε και έναν ακόμα τελεστή, έστω τον shift right. Αν και σε κάποια μεμονωμένα στιγμιότυπα φάνηκε να λειτουργεί, σε γενικές γραμμές τα αποτελέσματα δεν ήταν ικανοποιητικά. Σίγουρα ευθυνόταν και το γεγονός ότι μετά από τη χρήση του κάθε τελεστή, εφόσον δεν πρόκειται για τον τελεστή swap, έπρεπε να γίνεται υπολογισμός της τιμής της αντικειμενικής συνάρτησης εξαρχής. Αυτό προκαλούσε επιβράδυνση στην εκτέλεση του προγράμματος, έχοντας ως αποτέλεσμα τη μείωση των λύσεων που προλάβαινε να ελέγξει. Οπότε συμπερασματικά δεν πρόκειται για μια αλλαγή που βοήθησε.

Μια ακόμα ενδιαφέρουσα θεωρητικά προσέγγιση ήταν να γίνει εναλλαγή του τελεστή τοπικής

αναζήτησης από swap σε 2-opt. Προφανώς τότε άλλαζε και ο τελεστής της φάσης διατάραξης για να μην υπάρχει το ενδεχόμενο κύκλωσης αφού θα ήταν και οι δύο τελεστές ίδιοι. Αυτή η αλλαγή δυστυχώς δεν προκάλεσε βελτίωση, τουναντίον μάλιστα, επέφερε επιδείνωση στην ποιότητα των λύσεων. Το γεγονός αυτό βέβαια δεν προκαλεί μεγάλη έκπληξη και ο λόγος είναι απλός. Επειδή στη φάση τοπικής αναζήτησης γίνονται χιλιάδες επαναλήψεις, είναι αναγκαίο να χρησιμοποιηθεί τελεστής για τον οποίο υπάρχει γρήγορη ανανέωση της τιμής της συνάρτησης αξιολόγησης. Όπως είδαμε και προηγουμένως ο μόνος τελεστής που διαθέτει τέτοια ανανέωση είναι ο swap. Έτσι λοιπόν η χρήση άλλου τελεστή μπορεί να εξερευνά διαφορετικές γειτονιές, αλλά αυτό έχει αρνητική επίπτωση στο χρόνο εκτέλεσης. Κατά συνέπεια πραγματοποιούνται λιγότερες επαναλήψεις και έτσι το πλήθος των λύσεων που ελέγχεται είναι μικρότερο. Όπως είναι φανερό η αλλαγή αυτή δεν προτείνεται για το συγκεκριμένο πρόβλημα, εκτός φυσικά αν στο μέλλον βρεθεί τρόπος για να γίνεται γρήγορη ανανέωση στην τιμή της συνάρτησης αξιολόγησης όταν χρησιμοποιείται άλλος τελεστής.

Έχοντας φτάσει σε ένα σημείο όπου έχουν δοκιμαστεί όλες οι προαναφερθείσες τεχνικές, μια επιπλέον ενέργεια στην οποία θα μπορούσαμε να προβούμε είναι η προσθήκη επιπρόσθετων γειτονιών στη φάση τοπικής αναζήτησης. Αυτό πρακτικά σημαίνει ότι από εκεί που είχαμε μόνο έναν τελεστή, συγκεκριμένα τον swap λόγω της γρήγορης ανανέωσης, τώρα θα προστεθούν μερικοί ακόμα τελεστές για τη διεύρυνση των λύσεων. Προφανώς αυτό θα έχει επίπτωση στο πλήθος των επαναλήψεων καθώς στενεύουν τα χρονικά περιθώρια όταν το συνολικό πλήθος λύσεων που πρέπει να ελεγχθεί αυξάνεται. Επίσης, οι υπόλοιποι τελεστές απαιτούν τον επαναϋπολογισμό της τιμής της αντικειμενικής συνάρτησης.

Το θετικό αυτής της αλλαγής είναι ότι με περισσότερες γειτονιές αυξάνονται οι πιθανότητες αποφυγής των τοπικών βελτίστων, αφού κάθε γειτονιά συνήθως έχει διαφορετικά τοπικά βέλτιστα, αλλά όλες έχουν το ίδιο ολικό βέλτιστο. Αυτό που υλοποιήθηκε και δοκιμάστηκε είναι να υπάρχουν συνολικά τρεις δομές γειτονικών λύσεων, μια με τον τελεστή swap, μια με τον τελεστή 2-opt και μια με τον τελεστή shift right. Εννοείται πως η σειρά έχει σημασία και επιλέχθηκε αυτή η συγκεκριμένη σειρά έτσι ώστε να εκεμταλλευόμαστε πλήρως το πλεονέκτημα της γρήγορης ανανέωσης του τελεστή swap, προτού περάσουμε στη χρήση των άλλων δύο τελεστών.

Επεξηγηματικά, ο αλγόριθμος τοπικής αναζήτησης έψαχνε όλο το πλήθος των λύσεων με swap και μετά χρησιμοποιούσε τον επόμενο τελεστή αν δεν έβρισκε βελτίωση. Σε περίπτωση όμως που βρισκόταν κάποια καλύτερη ποιοτικά λύση της τρέχουσας, έπρεπε να επιλέξουμε ποια θα ήταν η επόμενη κίνηση. Όπως είδαμε στις παραλλαγές της μεθόδου VND, υπάρχουν αρκετοί τρόποι επιλογής της επόμενης γειτονιάς λύσεων. Στο πρόγραμμα επιλέχθηκε να χρησιμοποιηθούν τρεις από αυτές, με την πρώτη να είναι η βασική εκδοχή της VND όπως αυτή περιγράφεται στον αλγόριθμο 4. Σύμφωνα με την πρώτη εκδοχή όταν συμβεί κάποια βελτίωση η αναζήτηση επιστρέφει και πάλι στην πρώτη γειτονιά. Η παράμετρος k δηλαδή παίρνει την τιμή της μονάδας. Συνεχίζοντας, η επόμενη τακτική είναι η Cyclic VND κατά την οποία ακόμα και αν βρεθεί βελτίωση, η αναζήτηση

θα συνεχιστεί από την επόμενη γειτονιά όπως φαίνεται και στο τμήμα ψευδοκώδικα 5. Τέλος, η τελευταία παραλλαγή ονομάζεται Pipe VND και ορίζει πως αν βρεθεί κάποια βελτίωση τότε η αναζήτηση θα συνεχιστεί στη γειτονιά στην οποία βρέθηκε αυτή η βελτίωση. Συνεχίζουμε δηλαδή στην ίδια γειτονιά, όπως είδαμε και στον αλγόριθμο 6.

Στη γραφική διεπιφάνεια του προγράμματος, ο χρήστης είχε αρχικά τη δυνατότητα να επιλέξει ανάμεσα σε αυτές τις τρεις τεχνικές. Επειδή όμως τα υπολογιστικά αποτελέσματα των τριών αυτών τεχνικών δεν παρουσίασαν κάποια διαφορά μεταξύ τους, αποφασίστηκε να χρησιμοποιηθεί μόνο η βασική εκδοχή κατά την οποία όταν βρεθεί βελτίωση επιστρέφουμε για αναζήτηση στην πρώτη γειτονιά. Δυστυχώς η ενσωμάτωση των τριών αυτών παραλλαγών δεν απέδωσε πράγμα που σημαίνει πως επιστρέψαμε σε μια πιο απλοϊκή εκδοχή του προγράμματος.

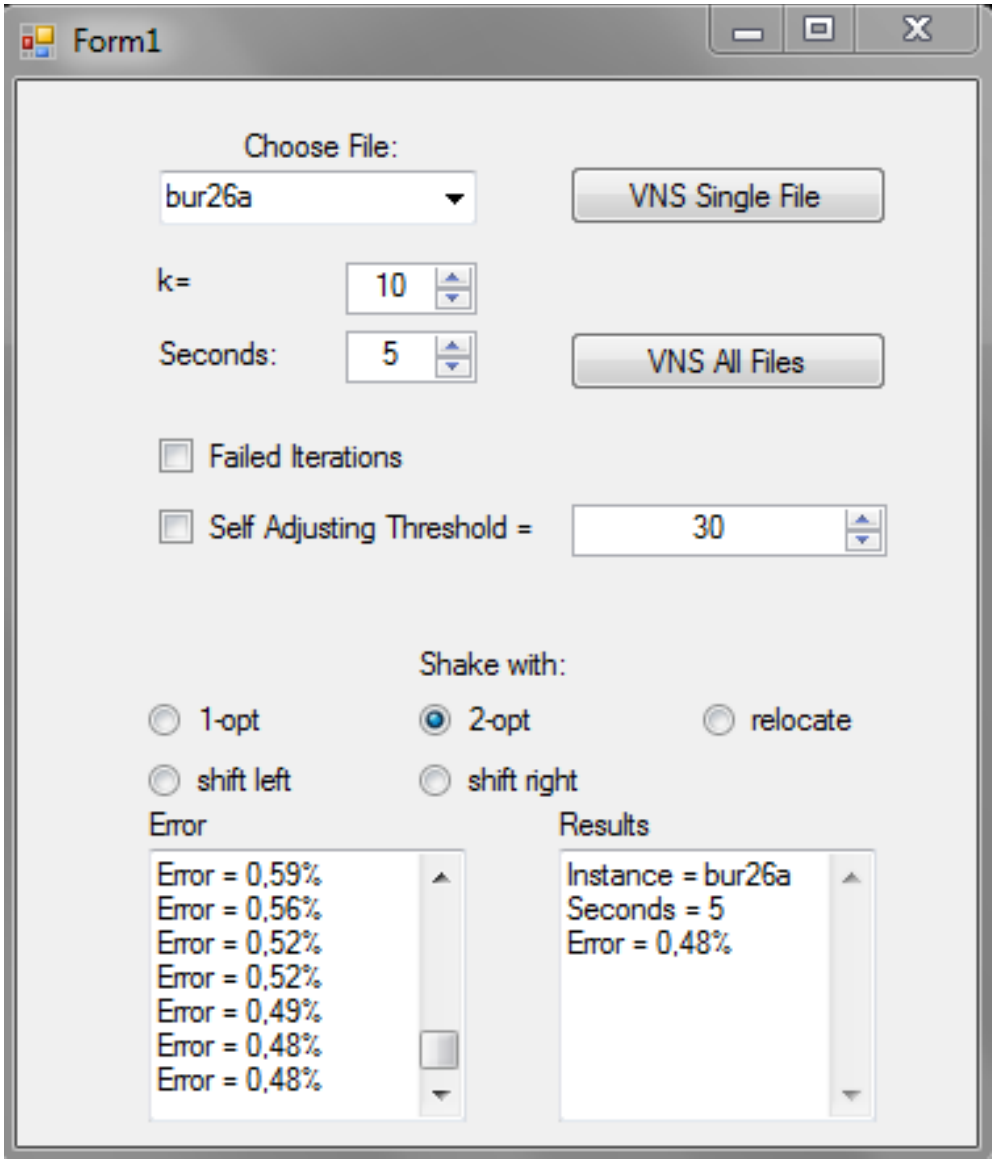
Φτάνοντας στο τέλος των τεχνικών που δοκιμάστηκαν, μένει ακόμα μια τεχνική η οποία ενώ θεωρητικά φάνταζε αρκετά υποσχόμενη, στην πράξη δεν παρατηρήθηκε κάποια βελτίωση. Πρόκειται για την τεχνική don't look bits η οποία είχε αρχικά προταθεί για την επιτάχυνση της αναζήτησης στο πρόβλημα του πλανόδιου πωλητή (Bentley 1992). Η επεξήγηση της λειτουργίας της στο πρόβλημα της τετραγωνικής αντιστοίχισης θα γίνει με βάση την έρευνα του T. Stutzle (Stutzle 1999). Σύμφωνα με αυτή, όταν ξεκινάμε τη φάση τοπικής αναζήτησης, κάθε εγκατάσταση, δηλαδή πρακτικά κάθε ακέραιος στο διάνυσμα της λύσης συσχετίζεται με ένα bit το οποίο είναι απενεργοποιημένο, έχει δηλαδή τιμή μηδέν. Αν κατά τη διάρκεια της αναζήτησης μια εγκατάσταση δεν προσφέρει βελτίωση για καμία διαφορετική θέση, τότε το bit παίρνει την τιμή ένα, θεωρείται δηλαδή ενεργοποιημένο. Όταν κάποιος ακέραιος της λύσης έχει ενεργοποιημένο bit τότε δεν ελέγχεται σε επόμενη αναζήτηση. Αν βρεθεί όμως βελτίωση και το στοιχείο αυτό αλλάξει θέση τότε και το bit που το συνοδεύει μετατρέπεται ξανά σε μηδέν. Με αυτήν τη τεχνική αποφεύγονται οι άσκοποι έλεγχοι σε στοιχεία που δε προκαλούν βελτίωση και έτσι μπορούμε να επικεντρώσουμε την αναζήτηση στα άλλα στοιχεία τα οποία εμφανίζονται πιο πολλά υποσχόμενα.

Αυτή η τεχνική υποτίθεται ότι μειώνει το πλήθος των λύσεων που πρέπει να ελεγχθούν και συμπερασματικά αυξάνει την αποτελεσματικότητα του προγράμματος στον ίδιο χρόνο. Παρ'ολα αυτά, ενώ σε θεωρητικό επίπεδο φαίνεται πως όντως θα υπάρξει βελτίωση, στην πρακτική υλοποίησή της αυτή η τεχνική δεν επέφερε κάποια αισθητή βελτίωση. Ίσως ευθύνεται κάποια άλλη τεχνική που χρησιμοποιήθηκε και για αυτό η τεχνική των don't look bits δεν απέδωσε. Άρα εν κατακλείδι αυτή η τεχνική απορρίφθηκε από το πρόγραμμα καθώς όταν δεν παρατηρείται βελτίωση τείνουμε να προτιμάμε την απλοϊκότερη προσέγγιση.

Εδώ ολοκληρώνεται το σύνολο όλων των τεχνικών που δοκιμάστηκαν κατά την ανάπτυξη του προγράμματος. Στη συνέχεια θα δούμε την αρχική μορφή του και θα αναλυθεί η γραφική διεπιφάνειά του μαζί με τις λειτουργίες που το απαρτίζουν.

4.3 Αρχική Υλοποίηση της Εφαρμογής

Σε αυτό το σημείο θα αναλυθούν οι λειτουργίες του προγράμματος που είχε υλοποιηθεί σε αυτό το στάδιο. Αφού δούμε τη γραφική διεπιφάνεια, θα μελετηθούν οι μέθοδοι που χρησιμοποιούνται καθώς και ο ρόλος κάθε στοιχείου στο παράθυρο. Ακολουθεί η φόρμα που εμφανίζεται στο χρήστη όταν τρέχει το πρόγραμμα:



Σχήμα **4.3.1** Αρχικό GUI.

Ξεκινώντας, παρατηρούμε ότι υπάρχει μια αναδυόμενη λίστα τέρμα πάνω αριστερά, η οποία περιέχει όλα τα στιγμιότυπα από τη Διαδικτυακή βιβλιοθήκη QAPLIB και μας βοηθάει να επιλέξουμε ποιο θέλουμε να επιχειρήσουμε να επιλύσουμε μεμονωμένα. Αφού επιλέξουμε το στιγμιότυπο που

επιθυμούμε, μπορούμε να πατήσουμε το κουμπί VNS Single File που βρίσκεται στα δεξιά της λίστας. Με το που ενεργοποιηθεί το κουμπί, το πρόγραμμα επιχειρεί να επιλύσει όποιο στιγμιότυπο αναγράφεται στην αναδυόμενη λίστα.

Αν όμως επιθυμούμε να επιχειρήσουμε να επιλύσουμε όλα τα στιγμιότυπα που βρίσκονται στο φάκελο, τότε πρέπει να πατήσουμε το αμέσως από κάτω κουμπί, το VNS All Files. Το συγκεκριμένο κουμπί ξεκινάει από το πρώτο στιγμιότυπο και συνεχίζει μέχρι και το τελευταίο. Υπάρχει φυσικά για λόγους ευκολίας, όταν θέλουμε να πειραματιστούμε με όλα τα στιγμιότυπα και όχι με κάποιο μεμονωμένο.

Συνεχίζοντας, βλέπουμε δύο αριθμητικά πεδία εισαγωγής, τα οποία είναι για τις παραμέτρους k και *seconds* αντίστοιχα. Προφανώς η παράμετρος *seconds* αναφέρεται στο ανώτατο χρονικό όριο που επιτρέπουμε να εκτελεστεί κάθε στιγμιότυπο. Η παράμετρος k όπως είπαμε χρησιμοποιείται στη φάση διατάραξης και βοηθάει στην παραγωγή μιας γειτονικής λύσης από την k -οστή δομή γειτνίασης της τρέχουσας λύσης. Η τιμή της συνήθως κυμαίνεται από 10 μέχρι 20, αλλά χωρίς να υπάρχει κάποιος ακριβής κανόνας, την επιλέγουμε εμπειρικά κατά βάση.

Λίγο παρακάτω, υπάρχει ένα checkbox ονόματι failed iterations, το οποίο όταν είναι ενεργοποιημένο, εφαρμόζει την τεχνική των αποτυχημένων επαναλήψεων. Ως υπενθύμιση, να σημειωθεί ότι ενώ αρχικά το μέγεθος των επαναλήψεων βρισκόταν στην κρίση του χρήστη, αργότερα αποφασίστηκε να υπολογίζεται αναλόγως με το μέγεθος του στιγμιοτύπου και το επιτρεπόμενο όριο χρόνου. Άρα λοιπόν, αν μετά από κάποιο ορισμένο αριθμό επαναλήψεων δεν παραουσιαστεί κάποια βελτίωση, τότε το πρόγραμμα υποθέτει ότι έχει εγκλωβιστεί και ξεκινάει την προσπάθεια απεγκλώβισης. Η ενέργεια που επιλέχθηκε ως απάντηση στο πρόβλημα των τοπικών βέλτιστων ήταν να αλλάζει ο τελεστής της φάσης διατάραξης. Όταν λοιπόν κολλήσει κάπου το πρόγραμμα, επιλέγει τυχαία κάποιον άλλον τελεστή από αυτούς που είναι διαθέσιμοι. Το ποιοι είναι αυτοί θα το δούμε παρακάτω. Με αυτήν την ενέργεια υποτίθεται πως το πρόγραμμα θα έβρισκε τρόπο να συνεχίσει την αναζήτηση σε άλλη δομή γειτονικών λύσεων με απώτερο σκοπό την απεγκλώβιση από το τοπικό βέλτιστο στο οποίο είχε παγιδευτεί.

Ακριβώς κάτω από το πρώτο checkbox υπάρχει ακόμα ένα με όνομα Self adjusting threshold το οποίο στα δεξιά του έχει ένα πεδίο αριθμητικής εισαγωγής. Ο ρόλος αυτού του checkbox είναι να κάνει το πρόγραμμα να προσαρμόζεται αυτόματα ανάλογα με το πόσο καλή είναι η ποιότητα της λύσης που έχει βρει. Αν για παράδειγμα ο αριθμός στο πεδίο εισαγωγής είναι το 30 όπως ακριβώς και στην εικόνα, τότε όσο η λύση έχει τιμή χειρότερη από τη γνωστή βέλτιστη κατά τριάντα τοις εκατό, η μέθοδος που ακολουθείται είναι αυτή της first improvement. Σε περίπτωση όμως που ξεπεράσουμε αυτό το σκαλοπάτι και βρούμε για παράδειγμα λύση με διαφορά μικρότερη του τριάντα τοις εκατό από τη βέλτιστη, τότε υιοθετείται η τεχνική best improvement. Αυτό γίνεται επειδή όπως είπαμε η τεχνική best improvement επιστρέφει πιο ποιοτικά αποτελέσματα όταν χρησιμοποιείται σε μια σχετικά καλή λύση.

Φυσικά ο αριθμός 30 είναι απλά ενδεικτικός και αλλάζει ανάλογα με το κάθε στιγμιότυπο. Για αυτό το λόγο δε γίνεται να τον υπολογίζουμε αυτόματα όπως τον αριθμό των αποτυχημένων επαναλήψεων. Αντίθετα η επιλογή του βασίζεται στην κρίση του χρήστη και έχει να κάνει με τη δυσκολία του εκάστοτε στιγμιοτύπου. Προφανώς για τα πιο εύκολα στιγμιότυπα ο αριθμός αυτός μικραίνει, ενώ για τα αρκετά δύσκολα παίρνει μεγαλύτερες τιμές.

Μετά από τα δύο checkboxes είναι το σημείο όπου γίνεται η επιλογή του τελεστή για τη φάση διατάραξης. Όπως είναι κατανοητό μπορεί να επιλεγεί μόνο ένας κάθε φορά και ως προεπιλεγμένος αποφασίστηκε να είναι ο τελεστής 2-opt. Υπενθυμίζεται ότι ο τελεστής 1-opt είναι υποκατηγορία του τελεστή swap και μαζί με το relocate και τα δύο shift, ολοκληρώνεται το σύνολο των διαθέσιμων τελεστών. Ανάμεσα σε αυτούς λοιπόν γίνεται η τυχαία επιλογή όταν ενεργοποιούνται οι αποτυχημένες επαναλήψεις για τις οποίες έγινε λόγος πιο πάνω. Η ευκολία εναλλαγής τελεστή βοηθάει σημαντικά στη δημιουργία συμπερασμάτων σχετικά με το ποιος είναι πιο κατάλληλος για κάθε στιγμιότυπο. Είναι σημαντικό να σημειωθεί ξανά πως γρήγορη ανανέωση της τιμής της συνάρτησης αξιολόγησης έχουμε μόνο για τον τελεστή 1-opt (προφανώς και για το swap). Μπορεί δηλαδή με τους υπόλοιπους να έχουμε την πολυτέλεια επίσκεψης διαφορετικών γειτονιών, αλλά χάνουμε το σημαντικό πλεονέκτημα της γρήγορης ανανέωσης.

Τέλος, περνάμε στο σημείο εμφάνισης των αποτελεσμάτων, το οποίο αποτελείται από δύο πεδία εμφάνισης κειμένου. Στο αριστερό τμήμα γίνεται συνεχής ανανέωση του ποσοστιαίου σφάλματος της καλύτερης λύσης που έχει βρει το πρόγραμμα σε σχέση με την καλύτερη γνωστή λύση (βέλτιστη). Έτσι μπορούμε να καταλάβουμε σε ποια σημεία κολλάει το πρόγραμμα σε τοπικό βέλτιστο, αν για παράδειγμα φτάσει στην τιμή 5 τοις εκατό και δεν παρουσιάσει ξανά βελτίωση, τότε αυτό σημαίνει ότι παγιδεύτηκε. Στα δεξιά εμφανίζεται αποτέλεσμα όταν τελειώσει η εκτέλεση ενός στιγμιότυπου. Εκεί εμφανίζεται το όνομα του στιγμιότυπου, ο χρόνος σε δευτερόλεπτα που χρειάστηκε για να φτάσει στη συγκεκριμένη λύση και το πόσο απέχει αυτή η λύση από την καλύτερη γνωστή λύση. Στο παράδειγμά μας βλέπουμε ότι χρησιμοποιήθηκαν και τα πέντε δευτερόλεπτα και πως η λύση που βρέθηκε εν τέλει είχε απόκλιση 0.48 τοις εκατό από τη βέλτιστη.

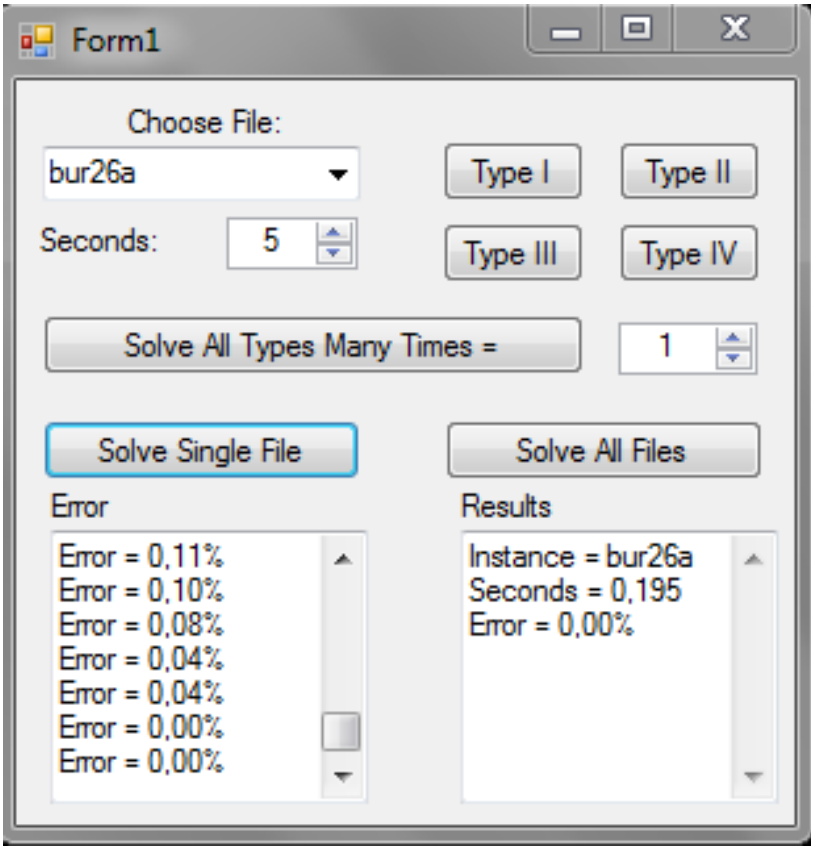
Απλά και μόνο για σκοπούς σύγκρισης με την τελική εκδοχή, αξίζει να σημειωθούν κάποια αρχικά αποτελέσματα στα οποία φτάσαμε κάνοντας χρήση αυτής της αρχικής μορφής του προγράμματος. Έχοντας ως ανώτατο χρονικό όριο τα τριάντα δευτερόλεπτα για κάθε στιγμιότυπο, επιχειρήθηκε η επίλυση όλων των στιγμιοτύπων από τη Διαδικτυακή βιβλιοθήκη QAPLIB. Το μέσο σφάλμα λοιπόν μετά το πέρας της εκτέλεσης του προγράμματος ήταν $7,56$ τοις εκατό, με στιγμιότυπα της κλάσης chr να έχουν μέσο σφάλμα μετά από πέντε εκτελέσεις μέχρι και 55 τοις εκατό (!). Όπως γίνεται αντιληπτό, αυτά τα αποτελέσματα δεν είναι καθόλου ικανοποιητικά, διαφέρουν πολύ από τα αναμενόμενα και πρέπει να γίνει κάποιου είδους μεγάλης ανακάλυψης για να υπάρξει αισθητή βελτίωση. Επίσης αυτά τα αποτελέσματα ήταν ό,τι καλύτερο μπορούσε να εξάγει το πρόγραμμα χρησιμοποιώντας τις τεχνικές που παρουσιάστηκαν πριν. Προφανώς με τη χρήση των όχι και τόσο

αποδοτικών τεχνικών, τα αποτελέσματα ήταν ακόμα χειρότερα.

Παρακάτω θα παρουσιαστεί η τελική εκδοχή του προγράμματος, αυτή δηλαδή που χρησιμοποιηθήκε για την εξαγωγή των αποτελεσμάτων που θα μελετηθούν στο επόμενο κεφάλαιο. Επιπρόσθετα θα αναλυθεί η τεχνική που ακολουθήθηκε και εκτόξευσε την ποιότητα των αποτελεσμάτων.

4.4 Τελική Υλοποίηση της Εφαρμογής

Εκτός από τη γραφική διεπιφάνεια, άλλαξαν και κάποιες λειτουργίες και προσεγγίσεις του προγράμματος. Αυτές θα αναλυθούν περαιτέρω, αλλά ας δούμε αρχικά ποια είναι η τελική εμφάνιση του προγράμματος:



Σχήμα **4.4.2** Τελικό GUI.

Ήδη παρατηρούμε το πόσο έχει αλλάξει η εμφάνιση του προγράμματος από την αρχική. Ουσιαστικά η μόνη παράμετρος που επηρρεάζεται από το χρήστη είναι τα δευτερόλεπτα εκτέλεσης του κάθε στιγμιοτύπου και όπως γνωρίζουμε αυτή δεν έχει να κάνει με τις αλγοριθμικές τεχνικές όπως η επιλογή τελεστή για τη φάση διατάραξης. Ξεκινώντας λοιπόν την ανάλυση των λειτουργιών, βλέπουμε ότι το πεδίο με την επιλογή στιγμιοτύπου από την αναδυόμενη λίστα και το πεδίο με την επιλογή του χρονικού ορίου έμειναν απαράλλαχτα.

Προχωράμε στην ανάλυση και παρατηρούμε ότι δεξιά αυτών των επιλογών υπάρχουν τέσσερα νέα κουμπιά τα οποία τα οποία αναφέρονται στους τέσσερις διαφορετικούς τύπους των στιγμιοτύπων. Το πώς ορίζονται αυτοί οι τύποι θα μελετηθεί στο κεφάλαιο με τα αποτελέσματα. Για την ώρα αρκεί να αναφερθεί πως η ύπαραξη αυτών των κουμπιών είναι καθαρά για τη διευκόλυνση του χρήστη και δεν επηρρεάζει την αλγοριθμική προσέγγιση στο πρόβλημα.

Το ίδιο ισχύει και για το κουμπί εν ονόματι Solve All Types Many Times, το οποίο ξεκινάει και επιχειρεί να επιλύσει όλα τα στιγμιότυπα ξεχωρίζοντας τα αποτελέσματα βάσει των τύπων οπου είπαμε προηγουμένως. Το πεδίο αριθμητικής εισαγωγής στα δεξιά του καθορίζει το πόσες φορές θα γίνει αυτή η ολική εκτέλεση. Όπως και πριν, αυτή η λειτουργία βοηθάει πολύ στην καταγραφή των αποτελεσμάτων, καθώς μπορεί ο χρήστης να βάλει το πρόγραμμα να δουλεύει ολη τη νύχτα με ένα μεγάλο χρονικό όριο και να μη χρειάζεται να επαναλαμβάνει την εκτέλση κάθε φορά.

Από κάτω υπάρχουν δύο γνώριμα κουμπιά με τίτλους Solve Single File και Solve All Files, τα οποία κάνουν ακριβώς αυτό που υπόσχονται. Τέλος, από κάτω είναι τα πεδία αποτελεσμάτων που έχουν την ίδια λειτουργία με το αρχικό πρόγραμμα. Η μεγάλη διαφορά είναι ότι ενώ προηγουμένως σε πέντε δευτερόλεπτα δεν είχε κατορθώσει το πρόγραμμα να επιλύσει το συγκεκριμένο στιγμιότυπο και είχε σφάλμα 0.48 τοις εκατό, τώρα επιτυγχάνει την πλήρη επίλυσή του σε χρόνο μάλιστα που ισούται με 0.195 δευτερόλεπτα. Πρόκειται προφανώς για πολύ σημαντική βελτίωση, αλλά δεν είναι τίποτα μπροστά στη συνολική βελτίωση που παρουσιάστηκε σε όλα τα στιγμιότυπα.

Για λόγους σύγκρισης με την αρχική εκδοχή, να θυμίσουμε ότι είχαμε μέσο σφάλμα ίσο με 7,56 τοις εκατό για όλα τα στιγμιότυπα με ανώτατο χρονικό όριο τα τριάντα δευτερόλεπτα. Στην τελική εκδοχή όμως αυτό το μέσο σφάλμα πέφτει κατακόρυφα στο 0.68. τοις εκατό! Πρόκειται κυριολεκτικά για τεράστια βελτίωση, η οποία αλλάζει οριστικά την αποτελεσματικότητα του προγράμματος. Ας δούμε λοιπόν τι συνέβη και φτάσαμε σε τόσο γιγάντια βελτίωση.

Έχοντας δοκιμάσει όλες τις προαναφερθείσες τεχνικές, είχαμε φτάσει σε ερευνητικό αδιέξοδο. Κάποιες από τις νέες τεχνικές επέφεραν βελτίωση, η οποία όμως δεν ήταν αρκετά ουσιώδης έτσι ώστε να χαρακτηριστεί το πρόγραμμα αποτελεσματικό. Αυτό που παρατηρήθηκε είναι ότι ορισμένες φορές το πρόγραμμα παγιδευόταν σε πολύ κακές λύσεις της τάξης του 50+ τοις εκατό και αδυνατούσε να ξεκολλήσει. Στο ίδιο στιγμιότυπο όμως σε επόμενη εκτέλεση μπορεί να έφτανε σε λύση της τάξης του 1 τοις εκατό χειρότερη από τη βέλτιστη λύση. Αυτό μας υποδεικνύει εμμέσως πλην σαφώς δύο πράγματα. Πρώτον ότι οι μέθοδοι απεγκλώβισής μας από τοπικά βέλτιστα, όπως η φάση διατάραξης, δεν είναι αρκετά αποδοτικές. Πρέπει λοιπόν να υλοποιηθεί κάποια διαφορετικη μέθοδος. Δεύτερον, ότι στο ίδιο στιγμιότυπο του προβλήματος με τις ίδιες παραμέτρους, κάποια τρεξίματα προχωράνε πολύ καλά, ενώ άλλα δυστυχώς παγιδεύονται σχετικά νωρίς και αδυνατούνε να ξεφύγουν.

Έτσι λοιπόν σε μια απέλπιδα προσπάθεια να βρεθεί σημαντική βελτίωση και κυρίως από ερευνητική περιέργεια αποφάσισα ότι έπρεπε η φάση διατάραξης να γίνει τόσο εντατική ώστε να εγγυάται

την εξερεύνηση πλήρως διαφορετικών γειτονιών λύσεων. Όσο και να προσπάθησα να την εντατικοποιήσω, τα αποτελέσματα δεν ήταν ικανοποιητικά. Ως έσχατη λύση, αποφασίστηκε κάθε φορά που εξερευνάται πλήρως μια γειτονιά δίχως βελτίωση, η τρέχουσα λύση να τυχαιοποιείται (προφανώς μένει αποθηκευμένη στη μνήμη η καλύτερη τιμή που έχει βρεθεί ως εκείνη τη στιγμή). Με αυτόν τον τρόπο είναι σαν να τρέχει πολλές φορές το πρόγραμμα πάνω στο ίδιο στιγμιότυπο εξερευνώντας έτσι έναν πολύ μεγάλο αριθμό λύσεων. Τα αρχικά αποτελέσματα ήταν πάρα πολλά υποσχόμενα, καθώς παρατηρήθηκε άμεση βελτίωση.

Ύστερα από πειραματισμούς αποφασίστηκε να μην υπάρχει καθόλου φάση διατάραξης, αλλά τη θέση της να καταλαμβάνει πλέον η τεχνική τυχαιοποίησης της λύσης. Δηλαδή όταν έχει εξερευνηθεί πλήρως μια συγκεκριμένη γειτονιά και δεν έχει παρουσιαστεί βελτίωση, πράγμα που σημαίνει ότι έχουμε παγιδευτεί σε τοπικό βέλτιστο, τότε πρακτικά ξεκινάμε με μια καινούρια, τυχαία λύση. Η τεχνική αυτή ονομάζεται Multi start και βασίζεται στις πολλαπλές επανεκκινήσεις, εξ ου και η ονομασία της (Mustafa and Topaloglu 2017). Στα ελληνικά συνηθίζεται να αναφέρεται ως μέθοδος πολυεναρκτήριας τοπικής αναζήτησης.

Η ολική απουσία της φάσης διατάραξης σήμαινε πως αρκετές παράμετροι δεν είχαν νόημα ύπαρξης, άρα καταλήγουμε σε ένα πιο απλό και εύκολο στο χειρισμό πρόγραμμα. Όπως φαίνεται και στη γραφική διεπιφάνεια της εικόνας **4.4.2**, ο χρήστης επιλέγει μόνο το χρονικό όριο και το πλήθος των στιγμιοτύπων που επιθυμεί να επιλύσει. Περνώντας στα τεχνικά σημεία και όσον αφορά τις μεθόδους που έμειναν απαράλλακτες, συνεχίζουμε να χρησιμοποιούμε τις συναρτήσεις που ασχολούνται με τη συμμετρικότητα των πινάκων όπως είπαμε πρωτύτερα. Προφανώς παραμένει η μέθοδος της γρήγορης ανανέωσης και φυσικά η πλέον σημαντικότερη μέθοδος τυχαιοποίησης διανύσματος. Επίσης το πρόγραμμα χρησιμοποιεί δύο συγκεκριμένες γειτονιές στη φάση τοπικής αναζήτησης με αυτές να εξερευνούνται με βάσει τους τελεστές swap και 2-opt με αυτή τη σειρά.

Η ροή του προγράμματος έχει την εξής μορφή: αφού αποκτήσουμε μια αρχική, τυχαία λύση ξεκινάμε τη φάση τοπικής αναζήτησης. Εκεί χρησιμοποιώντας ως πρώτο τον τελεστή swap εξερευνούμε τη γειτονιά μέχρι να μην μπορεί να βρεθεί κάποια περαιτέρω βελτίωση. Τότε καλούμε τη συνάρτηση που παράγει ένα τυχαίο διάνυσμα λύσης αλλά μετά συνεχίζουμε την αναζήτηση με το δεύτερο τελεστή, τον 2-opt. Όταν παγιδευτούμε και με το δεύτερο τελεστή, γίνεται ξανά τυχαιοποίηση του διανύσματος της λύσης και η επόμενη αναζήτηση γίνεται ξανά με το swap. Σε περίπτωση εύρεσης βελτιωμένης λύσης σε οποιαδήποτε από τις δύο γειτονιές, η αναζήτηση συνεχίζει στην πρώτη γειτονιά. Προφανώς κατά τη διάρκεια εκτέλεσης, υπάρχει αποθηκευμένη η καλύτερη τιμή που έχει βρεθεί. Επίσης σε περίπτωση που βρεθεί η βέλτιστη λύση, η αναζήτηση τερματίζει άμεσα.

Χρησιμοποιώντας την τεχνική της πολυεναρκτήριας τοπικής αναζήτησης, μπορούμε ουσιαστικά να εξερευνήσουμε πλήρως ένα πολύ μεγαλύτερο πλήθος γειτονιών και εν τέλει να οδηγηθούμε σε πολύ καλές ποιοτικά λύσεις. Αυτό γίνεται ξεκάθαρο και από τη σύγκριση ανάμεσα στην αρχική και την τελική εκδοχή του προγράμματος όπου παρουσιάζεται τουλάχιστον δεκαπλάσια βελτίωση στα

σφάλματα των λύσεων.

Θα ακολουθήσει η αναπαράσταση σε μορφή ψευδοκώδικα της συνάρτησης που είναι υπεύθυνη για την τοπική αναζήτηση στο πρόγραμμα.

Algorithm 19 localSearch

```
1: procedure localSearch(solution, tmax)
2:     while Neighbourhood < 3 do                ▷ while there are more neighbourhoods to be explored
3:         for i = 0; i < size; i++ do
4:             for i = 0; i < size; i++ do
5:                 if elapsedMilliSeconds > tmax then                          ▷ time's up
6:                     return;
7:                 end if
8:                 if Neighbourhood == 1 then                                  ▷ 1st neighbourhood
9:                     Swap(solution, i, j);        ▷ fast update is implemented inside the swap function
10:                else if Neighbourhood == 2 then                             ▷ 2nd neighbourhood
11:                    TwoOpt(solution, i, j);
12:                    calculateFitness(solution);
13:                end if
14:                if currentFitness > bestFitness then                        ▷ improvement
15:                    Neighbourhood ← 1;                                      ▷ back to 1st neighbourhood
16:                    return;
17:                end if
18:            end for
19:        end for
20:        Neighbourhood++;                                                    ▷ next neighbourhood
21:        Randomize(solution);
22:        calculateFitness(solution);
23:    end while
24:    Neighbourhood ← 1                                                       ▷ back to 1st neighbourhood
25:    return;
26: end procedure
```

Μια αξιοσημείωτη παρατήρηση είναι ότι η συνάρτηση αξιολόγησης $calculateFitness$ καλείται μόνο όταν γίνεται χρήση του 2-opt και όταν τυχαιοποιείται το διάνυσμα. Σε όλες τις άλλες περιπτώσεις εκμεταλλευόμαστε τη γρήγορη ανανέωση και έτσι κερδίζουμε πολύτιμο χρόνο. Για αυτό το λόγο άλλωστε επιλέχθηκε ως πρώτη γειτονιά αυτή με τον τελεστή swap.

Το αλγοριθμικό αυτό τμήμα του προγράμματος, που είναι υπεύθυνο για την τοπική αναζήτηση, έχει ενσωματώσει και τη συνάρτηση $Randomize$ την οποία είδαμε και προηγουμένως. Η κλήση της πραγματοποιείται όταν παγιδευτούμε και έχουμε εξερευνήσει τις λύσεις σε μια γειτονιά. Προτού συνεχίσουμε με το δεύτερο τελεστή, γίνεται η τυχαιοποίηση του διανύσματος, κάτι που ουσιαστικά σημαίνει πως γίνεται επανεκκίνηση. Το πρόγραμμα λοιπόν χρησιμοποιεί τοπική αναζήτηση δύο γειτονιών με πολυεναρκτήρια τοπική αναζήτηση. Η τελική αυτή προσέγγιση απέφερε τα καλύτερα αποτελέσματα και δεν υπάρχει καν λόγος σύγκρισης με τις προηγούμενες εκδοχές. Στιγμιότυπα τα οποία όπως είπαμε είχαν μέσο σφάλμα έως και 55 τοις εκατό, πλέον κυμαίνονται στα επίπεδα σφάλματος της τάξης του 2 τοις εκατό, με χρονικό όριο πάντα τα τριάντα δευτερόλεπτα.

Συν τοις άλλοις, ένα ακόμα πλεονέκτημα αυτής της εκδοχής είναι ότι αποτελεί μια απλότερη προ-

σέγγιση σχετικά με την προγενέστερή της. Πλέον δεν υπάρχει νόημα ύπαρξης τόσων παραμέτρων. Με την παράλειψη της φάσης διατάραξης, αποφεύγουμε τη χρήση μεγάλου πλήθους τελεστών αφού καταλήγουμε να χρησιμοποιούμε τους δύο πιο απλούς και διαδεδομένους. Ο χρήστης δε χρειάζεται να παρεμβαίνει στην εισαγωγή παραμέτρων, παρά μόνο στο χρονικό όριο. Έτσι το πρόγραμμα εκτός από πιο γρήγορο και πιο αποτελεσματικό, έγινε και πιο εύχρηστο και απλοϊκό στην εκτέλεσή του.

Στο επόμενο κεφάλαιο θα παρουσιαστούν αναλύτικά τα αποτελέσματα του προγράμματος σε διάφορες εκτελέσεις με διαφορετικά χρονικά όρια. Εκτός από αυτό θα δούμε και σε ποιους διαφορετικούς τύπους χωρίζονται τα στιγμιότυπα, καθώς και σε ποιους από αυτούς τους τύπους υπάρχει η καλύτερη επίδοση. Αφού μελετηθούν τα αποτελέσματα αυτά καθάυτά, θα προχωρήσουμε σε σύγκριση με άλλους λύτες.

ΚΕΦΑΛΑΙΟ 5

Ανάλυση Αποτελεσμάτων

5.1 Αποτελέσματα

Σε αυτό το κεφάλαιο θα παρουσιαστούν και θα σχολιαστούν τα αποτελέσματα για όλα τα στιγμιότυπα σε διαφορετικά χρονικά όρια. Οι πίνακες που θα ακολουθήσουν απαρτίζονται από το όνομα του κάθε στιγμιοτύπου, την καλύτερη γνωστή λύση που τις περισσότερες φορές αποτελεί και τη βέλτιστη λύση, τη μέση λύση του προγράμματος μετά από 5 επαναλήψεις, το μέσο σφάλμα επί τοις εκατό μετά από 5 επαναλήψεις και το μέσο χρόνο που χρειάστηκε μετά από 5 επαναλήψεις. Όπως θα δούμε τα στιγμιότυπα χωρίζονται σε τέσσερις κατηγορίες αναλόγως με τον τύπο τους. Ο διαχωρισμός έγινε σύμφωνα με την έρευνα των Burkard, Karisch, Rendl για τη βιβλιοθήκη στιγμιοτύπων QAPLIB από όπου δανειστήκαμε και εμείς τα στιγμιότυπα (Burkard, Karisch, and Rendl 1997).

Υπάρχουν λοιπόν τέσσερις κατηγορίες οι οποίες αναφέρονται παρακάτω:

• Τύπος 1: Αποτελείται από τυχαία, μη δομημένα στιγμιότυπα. Η τυχαία παραγωγή τους βασίζεται σε μια ομοιόμορφη κατανομή και είναι εμπειρικά τα πιο δύσκολα στιγμιότυπα προς επίλυση. Μερικά από αυτά είναι τα στιγμιότυπα που αρχίζουν με tai και μετά το μέγεθός τους υπάρχει το a που τα διαχωρίζει από τα tai*b.

• Τύπος 2: Αποτελείται επίσης από τυχαία στιγμιότυπα τα οποία όμως βασίζονται σε πλέγμα. Δηλαδή οι ροές παράγονται τυχαία και οι αποστάσεις ορίζονται από την απόσταση Manhattan. Επεξηγηματικά, η απόσταση Manhattan δύο σημείων σε κάποιο πλέγμα ορίζεται ως το άθροισμα της οριζόντιας και κάθετης απόστασής τους. Σε αντίθεση με την ευκλίδεια απόσταση η οποία θα ήταν η διαγώνιος που ενώνει τα δυο σημεία στο πλέγμα. Χαρακτηριστικά στιγμιότυπα τύπου 2 είναι τα nug* και sko*.

• Τύπος 3: Ο τύπος αυτός απαρτίζεται από στιγμιότυπα με δεδομένα από την πραγματική ζωή. Ως παράδειγμα έχουμε τη διάταξη εγκαταστάσεων νοσοκομείου ή εμπορικού κέντρου και σχεδίαση κυκλωμάτων. Όπως θα δούμε και στα αριθμητικά αποτελέσματα αυτός ο τύπος στιγμιοτύπων είναι και ο πιο εύκολος προς επίλυση. Περιλαμβάνει στιγμιότυπα όπως chr* και esc*.

• Τύπος 4: Ο τελευταίος τύπος περιλαμβάνει τυχαία μεν στιγμιότυπα, τα οποία όμως παράγονται

με τέτοιο τρόπο έτσι ώστε να μοιάζουν με αυτά της πραγματικής ζωής. Ουσιαστικά πρόκειται για τυχαία στιγμιότυπα τα οποία όμως θα μπορούσαν να είναι και αληθινά. Τέτοιου τύπου στιγμιότυπα είναι τα tai*b.

Αφού λοιπόν διαχωρίσαμε τα στιγμιότυπα σε τύπους ας δούμε τα αποτελέσματα σε κάθε τύπο με ανώτατο χρονικό όριο το μισό λεπτό. Πρέπει επίσης να αναφερθούν τα χαρακτηριστικά του επεξεργαστή στον οποίο τρέξαμε τα πειράματα. Ο επεξεργαστής ήταν Intel Core i7-4770K @ 3.50GHz.

Πίνακας **5.1** Αποτελέσματα στιγμιοτύπων τύπου 1 με χρονικό όριο το μισό λεπτό

Instance	zOpt	Avg best solution	Avg Error (%)	Average Time
tai20a	703482	703910,00	0,06	0,17
tai25a	1167256	1171908,40	0,40	0,42
tai30a	1818146	1837401,20	1,06	0,50
tai35a	2422002	2464766,40	1,77	0,50
tai40a	3139370	3204598,00	2,08	0,50
tai50a	4938796	5066501,20	2,59	0,50
tai60a	7205962	7396474,40	2,64	0,50
tai80a	13499184	13875797,60	2,79	0,50
tai100a	21052466	21599896,80	2,60	0,50
rou12	235528	235528,00	0,00	0,00
rou15	354210	354210,00	0,00	0,00
rou20	725522	725522,00	0,00	0,07
lipa20a	3683	3683,00	0,00	0,00
lipa20b	27076	27076,00	0,00	0,00
lipa30a	13178	13178,00	0,00	0,05
lipa30b	151426	151426,00	0,00	0,00
lipa40a	31538	31833,20	0,94	0,50
lipa40b	476581	476581,00	0,00	0,00
lipa50a	62093	62732,40	1,03	0,50
lipa50b	1210244	1210244,00	0,00	0,02
lipa60a	107218	108167,80	0,89	0,50
lipa60b	2520135	2520135,00	0,00	0,23
lipa70a	169755	171109,00	0,80	0,50
lipa70b	4603200	4783391,80	3,91	0,22
lipa80a	253195	255009,00	0,72	0,50
lipa80b	7763962	8721249,80	12,33	0,46
lipa90a	360630	363028,40	0,67	0,50
lipa90b	12490441	14052770,20	12,51	0,48
Average			1,78	0,31

Πίνακας **5.2** Αποτελέσματα στιγμιοτύπων τύπου 2 με χρονικό όριο το μισό λεπτό

Instance	zOpt	Avg best solution	Avg Error (%)	Average Time
nug12	578	578,00	0,00	0,00
nug14	1014	1014,00	0,00	0,00
nug15	1150	1150,00	0,00	0,00
nug16a	1610	1610,00	0,00	0,00
nug16b	1240	1240,00	0,00	0,00
nug17	1732	1732,00	0,00	0,01
nug18	1930	1930,00	0,00	0,00
nug20	2570	2570,00	0,00	0,00
nug21	2438	2438,00	0,00	0,00
nug22	3596	3596,00	0,00	0,00
nug24	3488	3488,00	0,00	0,01
nug25	3744	3744,00	0,00	0,06
nug27	5234	5234,00	0,00	0,03
nug28	5166	5166,00	0,00	0,05
nug30	6124	6130,67	0,11	0,42
scr12	31410	31410,00	0,00	0,00
scr15	51140	51140,00	0,00	0,00
scr20	110030	110030,00	0,00	0,01
sko42	15812	15869,00	0,36	0,50
sko49	23386	23508,00	0,52	0,50
sko56	34458	34667,00	0,61	0,50
sko64	48498	48815,67	0,66	0,50
sko72	66256	66658,33	0,61	0,50
sko81	90998	91655,33	0,72	0,50
sko90	115534	116469,67	0,81	0,50
sko100a	152002	152997,33	0,65	0,50
sko100b	153890	154951,00	0,69	0,50
sko100c	147862	148885,00	0,69	0,50
sko100d	149576	150712,67	0,76	0,50
sko100e	149150	150341,00	0,80	0,50
sko100f	149036	150135,33	0,74	0,50
tho30	149936	150187,33	0,17	0,40
tho40	240516	241592,00	0,45	0,50
tho150	8133398	8236097,67	1,26	0,50
wil50	48816	48896,00	0,16	0,50
wil100	273038	274229,33	0,44	0,50
Average			0,31	0,26

Πίνακας **5.3** Αποτελέσματα στιγμιοτύπων τύπου 3 με χρονικό όριο το μισό λεπτό

Instance	zOpt	Avg best solution	Avg Error (%)	Average Time
kra30a	88900	89050,00	0,17	0,21
kra30b	91420	91443,33	0,03	0,32
kra32	88700	88700,00	0,00	0,09
bur26a	5426670	5426670,00	0,00	0,04
bur26b	3817852	3817852,00	0,00	0,02
bur26c	5426795	5426795,00	0,00	0,04
bur26d	3821225	3821225,00	0,00	0,08
bur26e	5386879	5386879,00	0,00	0,07
bur26f	3782044	3782044,00	0,00	0,01
bur26g	10117172	10117172,00	0,00	0,15
bur26h	7098658	7098658,00	0,00	0,03
chr12a	9552	9552,00	0,00	0,00
chr12b	9742	9742,00	0,00	0,00
chr12c	11156	11156,00	0,00	0,01
chr15a	9896	9896,00	0,00	0,01
chr15b	7990	7990,00	0,00	0,00
chr15c	9504	9504,00	0,00	0,03
chr18a	11098	11098,00	0,00	0,10
chr18b	1534	1534,00	0,00	0,00
chr20a	2192	2199,33	0,33	0,48
chr20b	2298	2355,33	2,49	0,50
chr20c	14142	14142,00	0,00	0,01
chr22a	6156	6201,67	0,74	0,50
chr22b	6194	6261,33	1,09	0,50
chr25a	3796	3879,67	2,20	0,47
els19	17212548	17212548,00	0,00	0,03
esc16a	68	68,00	0,00	0,00
esc16b	292	292,00	0,00	0,00
esc16c	160	160,00	0,00	0,00
esc16d	16	16,00	0,00	0,00
esc16e	28	28,00	0,00	0,00
esc16g	26	26,00	0,00	0,00
esc16h	996	996,00	0,00	0,00
esc16i	14	14,00	0,00	0,00
esc16j	8	8,00	0,00	0,00
esc32a	130	131,33	1,03	0,43
esc32b	168	168,00	0,00	0,01
esc32c	642	642,00	0,00	0,00
esc32d	200	200,00	0,00	0,00
esc32e	2	2,00	0,00	0,00
esc32g	6	6,00	0,00	0,00
esc32h	438	438,00	0,00	0,00
esc64a	116	116,00	0,00	0,00
esc128	64	64,00	0,00	0,02
had12	1652	1652,00	0,00	0,00
had14	2724	2724,00	0,00	0,00
had16	3720	3720,00	0,00	0,00
had18	5358	5358,00	0,00	0,00
had20	6922	6922,00	0,00	0,00
ste36a	9526	9618,00	0,97	0,50
ste36b	15852	15936,00	0,53	0,38
ste36c	8239110	8266708,67	0,33	0,50
Average			0,19	0,11

Πίνακας **5.4** Αποτελέσματα στιγμιοτύπων τύπου 4 με χρονικό όριο το μισό λεπτό

Instance	zOpt	Avg best solution	Avg Error (%)	Average Time
tai20b	122455319	122455319,00	0,00	0,00
tai25b	344355646	344355646,00	0,00	0,02
tai30b	637117113	637573321,83	0,07	0,50
tai35b	283315445	283685183,67	0,13	0,40
tai40b	637250948	637438941,33	0,03	0,41
tai50b	458821517	459628024,00	0,18	0,50
tai60b	608215054	610148110,83	0,32	0,50
tai64c	1855928	1855928,00	0,00	0,00
tai80b	818415043	830278537,67	1,45	0,50
tai100b	1185996137	1195218012,83	0,78	0,50
tai150b	498896643	507781252,00	1,78	0,50
Average			0,43	0,35

Πίνακας **5.5** Μέσο σφάλμα και μέσος χρόνος όλων των στιγμιοτύπων για μισό λεπτό

Average Error	0,68
Average Time	0,26

Παρατηρούμε ότι όντως τα στιγμιότυπα τύπου 1 έχουν μεγαλύτερο μέσο σφάλμα σε σχέση με τα υπόλοιπα. Επίσης όπως είπαμε τα στιγμιότυπα τύπου 3. τα οποία βασίζονται σε πραγματικά προβλήματα, είναι και τα πιο εύκολα προς επίλυση. Επίσης έχουμε συνολικό μέσο σφάλμα 0.68 τοις εκατό και μέσο χρόνο επίλυσης 0.26 λεπτά. Ο μέσος χρόνος είναι μικρότερος του μισού λεπτού επειδή όταν βρίσκουμε την βέλτιστη λύση πριν λήξει ο χρόνος συνεχίζουμε αμέσως στο επόμενο στιγμιότυπο. Αξίζει να σημειώσουμε ότι το μεγαλύτερο ποσοστιαίο σφάλμα το έχουν τα στιγμιότυπα lipa80b και lipa90b, τα οποία όπως θα δούμε αργότερα, με μεγαλύτερο χρονικό όριο, μπορούμε να τα επιλύσουμε. Όμως στιγμιότυπα όπως το lipa90a, αν και έχουν ήδη μικρό σφάλμα στο μισό λεπτό, είναι πολύ δύσκολο να βελτιώσουμε αυτήν την τιμή ακόμα και με μεγαλύτερο χρονικό όριο.

Ας περάσουμε τώρα στα αποτελέσματα με ανώτατο χρονικό όριο το 1 λεπτό.

Πίνακας **5.6** Αποτελέσματα στιγμιοτύπων τύπου 1 με χρονικό όριο το 1 λεπτό

Instance	zOpt	Avg best solution	Avg Error (%)	Average Time
tai20a	703482	703482,00	0,00	0,08
tai25a	1167256	1169813,20	0,22	0,67
tai30a	1818146	1835796,80	0,97	1,00
tai35a	2422002	2456524,40	1,43	1,00
tai40a	3139370	3197906,40	1,86	1,00
tai50a	4938796	5057452,40	2,40	1,00
tai60a	7205962	7400014,80	2,69	1,00
tai80a	13499184	13863528,80	2,70	1,00
tai100a	21052466	21592854,80	2,57	1,00
rou12	235528	235528,00	0,00	0,00
rou15	354210	354210,00	0,00	0,00
rou20	725522	725522,00	0,00	0,06
lipa20a	3683	3683,00	0,00	0,00
lipa20b	27076	27076,00	0,00	0,00
lipa30a	13178	13178,00	0,00	0,05
lipa30b	151426	151426,00	0,00	0,00
lipa40a	31538	31710,40	0,55	0,64
lipa40b	476581	476581,00	0,00	0,00
lipa50a	62093	62705,80	0,99	1,00
lipa50b	1210244	1210244,00	0,00	0,02
lipa60a	107218	108152,60	0,87	1,00
lipa60b	2520135	2520135,00	0,00	0,12
lipa70a	169755	171096,80	0,79	1,00
lipa70b	4603200	4603200,00	0,00	0,22
lipa80a	253195	254996,80	0,71	1,00
lipa80b	7763962	8081818,80	4,09	0,63
lipa90a	360630	362999,60	0,66	1,00
lipa90b	12490441	15099526,40	20,89	1,00
Average			1,59	0,55

Πίνακας **5.7** Αποτελέσματα στιγμιοτύπων τύπου 2 με χρονικό όριο το 1 λεπτό

Instance	zOpt	Avg best solution	Avg Error (%)	Average Time
nug12	578	578,00	0,00	0,00
nug14	1014	1014,00	0,00	0,00
nug15	1150	1150,00	0,00	0,00
nug16a	1610	1610,00	0,00	0,00
nug16b	1240	1240,00	0,00	0,00
nug17	1732	1732,00	0,00	0,02
nug18	1930	1930,00	0,00	0,00
nug20	2570	2570,00	0,00	0,00
nug21	2438	2438,00	0,00	0,01
nug22	3596	3596,00	0,00	0,00
nug24	3488	3488,00	0,00	0,01
nug25	3744	3744,00	0,00	0,02
nug27	5234	5234,00	0,00	0,03
nug28	5166	5166,00	0,00	0,19
nug30	6124	6125,33	0,02	0,88
scr12	31410	31410,00	0,00	0,00
scr15	51140	51140,00	0,00	0,00
scr20	110030	110030,00	0,00	0,01
sko42	15812	15852,00	0,25	1,00
sko49	23386	23478,67	0,40	1,00
sko56	34458	34634,00	0,51	1,00
sko64	48498	48740,33	0,50	1,00
sko72	66256	66632,00	0,57	1,00
sko81	90998	91532,33	0,59	1,00
sko90	115534	116251,33	0,62	1,00
sko100a	152002	152925,33	0,61	1,00
sko100b	153890	154846,67	0,62	1,00
sko100c	147862	148940,33	0,73	1,00
sko100d	149576	150651,33	0,72	1,00
sko100e	149150	150331,67	0,79	1,00
sko100f	149036	150144,33	0,74	1,00
tho30	149936	149936,00	0,00	0,25
tho40	240516	241421,00	0,38	1,00
tho150	8133398	8221136,67	1,08	1,00
wil50	48816	48890,00	0,15	1,00
wil100	273038	274017,33	0,36	1,00
Average			0,27	0,51

Πίνακας **5.8** Αποτελέσματα στιγμιοτύπων τύπου 3 με χρονικό όριο το 1 λεπτό

Instance	zOpt	Avg best solution	Avg Error (%)	Average Time
kra30a	88900	89050,00	0,17	0,47
kra30b	91420	91431,67	0,01	0,55
kra32	88700	88700,00	0,00	0,14
bur26a	5426670	5426670,00	0,00	0,01
bur26b	3817852	3817852,00	0,00	0,05
bur26c	5426795	5426795,00	0,00	0,02
bur26d	3821225	3821225,00	0,00	0,14
bur26e	5386879	5386879,00	0,00	0,07
bur26f	3782044	3782044,00	0,00	0,01
bur26g	10117172	10117172,00	0,00	0,24
bur26h	7098658	7098658,00	0,00	0,05
chr12a	9552	9552,00	0,00	0,00
chr12b	9742	9742,00	0,00	0,00
chr12c	11156	11156,00	0,00	0,01
chr15a	9896	9896,00	0,00	0,01
chr15b	7990	7990,00	0,00	0,00
chr15c	9504	9504,00	0,00	0,02
chr18a	11098	11098,00	0,00	0,13
chr18b	1534	1534,00	0,00	0,00
chr20a	2192	2197,67	0,26	0,68
chr20b	2298	2351,00	2,31	1,00
chr20c	14142	14142,00	0,00	0,01
chr22a	6156	6175,33	0,31	0,93
chr22b	6194	6235,33	0,67	0,82
chr25a	3796	3923,00	3,35	1,00
els19	17212548	17212548,00	0,00	0,10
esc16a	68	68,00	0,00	0,00
esc16b	292	292,00	0,00	0,00
esc16c	160	160,00	0,00	0,00
esc16d	16	16,00	0,00	0,00
esc16e	28	28,00	0,00	0,00
esc16g	26	26,00	0,00	0,00
esc16h	996	996,00	0,00	0,00
esc16i	14	14,00	0,00	0,00
esc16j	8	8,00	0,00	0,00
esc32a	130	130,33	0,26	0,64
esc32b	168	168,00	0,00	0,00
esc32c	642	642,00	0,00	0,00
esc32d	200	200,00	0,00	0,00
esc32e	2	2,00	0,00	0,00
esc32g	6	6,00	0,00	0,00
esc32h	438	438,00	0,00	0,00
esc64a	116	116,00	0,00	0,00
esc128	64	64,00	0,00	0,01
had12	1652	1652,00	0,00	0,00
had14	2724	2724,00	0,00	0,00
had16	3720	3720,00	0,00	0,00
had18	5358	5358,00	0,00	0,00
had20	6922	6922,00	0,00	0,00
ste36a	9526	9594,00	0,71	1,00
ste36b	15852	15878,67	0,17	0,88
ste36c	8239110	8279107,67	0,49	1,00
Average			0,17	0,19

Πίνακας **5.9** Αποτελέσματα στιγμιοτύπων τύπου 4 με χρονικό όριο το 1 λεπτό

Instance	zOpt	Avg best solution	Avg Error (%)	Average Time
tai20b	122455319	122455319,00	0,00	0,00
tai25b	344355646	344355646,00	0,00	0,03
tai30b	637117113	637150918,33	0,01	1,00
tai35b	283315445	283492594,00	0,06	0,90
tai40b	637250948	637293599,00	0,01	0,73
tai50b	458821517	459338876,17	0,11	1,00
tai60b	608215054	609101126,00	0,15	1,00
tai64c	1855928	1855928,00	0,00	0,00
tai80b	818415043	826082010,17	0,94	1,00
tai100b	1185996137	1192228213,33	0,53	1,00
tai150b	498896643	504486944,17	1,12	1,00
Average			0,27	0,70

Πίνακας **5.10** Μέσο σφάλμα και μέσος χρόνος όλων των στιγμιοτύπων για 1 λεπτό

Average Error	0,57
Average Time	0,49

Εδώ βλέπουμε ότι με διπλάσιο χρόνο παρατηρείται κάποια βελτίωση καθώς μειώνονται τα μέσα σφάλματα για κάθε τύπο. Δυστυχώς το στιγμιότυπο lipa90b δεν κατάφερε να απεγκλωβιστεί εγκαίρως από κάποιο τοπικό βέλτιστο και έτσι παρουσιάζει μεγαλύτερο σφάλμα. Στη συνέχεια αυξάνουμε το ανώτατο χρονικό όριο στα 2 λεπτά και έχουμε τα παρακάτω αποτελέσματα:

Πίνακας **5.11** Αποτελέσματα στιγμιοτύπων τύπου 1 με χρονικό όριο τα 2 λεπτά

Instance	zOpt	Avg best solution	Avg Error (%)	Average Time
tai20a	703482	703482,00	0,00	0,12
tai25a	1167256	1169697,60	0,21	1,45
tai30a	1818146	1835648,40	0,96	2,00
tai35a	2422002	2456680,40	1,43	2,00
tai40a	3139370	3193514,40	1,72	2,00
tai50a	4938796	5050101,20	2,25	2,00
tai60a	7205962	7386828,00	2,51	2,00
tai80a	13499184	13860228,80	2,67	2,00
tai100a	21052466	21580017,60	2,51	2,00
rou12	235528	235528,00	0,00	0,00
rou15	354210	354210,00	0,00	0,00
rou20	725522	725522,00	0,00	0,10
lipa20a	3683	3683,00	0,00	0,00
lipa20b	27076	27076,00	0,00	0,00
lipa30a	13178	13178,00	0,00	0,05
lipa30b	151426	151426,00	0,00	0,00
lipa40a	31538	31715,40	0,56	1,92
lipa40b	476581	476581,00	0,00	0,00
lipa50a	62093	62713,40	1,00	2,00
lipa50b	1210244	1210244,00	0,00	0,02
lipa60a	107218	108146,60	0,87	2,00
lipa60b	2520135	2520135,00	0,00	0,21
lipa70a	169755	171097,40	0,79	2,00
lipa70b	4603200	4603200,00	0,00	0,24
lipa80a	253195	254945,40	0,69	2,00
lipa80b	7763962	8074717,20	4,00	1,27
lipa90a	360630	362968,00	0,65	2,00
lipa90b	12490441	14557286,80	16,55	1,92
Average			1,41	1,12

Πίνακας **5.12** Αποτελέσματα στιγμιοτύπων τύπου 2 με χρονικό όριο τα 2 λεπτά

Instance	zOpt	Avg best solution	Avg Error (%)	Average Time
nug12	578	578,00	0,00	0,00
nug14	1014	1014,00	0,00	0,00
nug15	1150	1150,00	0,00	0,00
nug16a	1610	1610,00	0,00	0,00
nug16b	1240	1240,00	0,00	0,00
nug17	1732	1732,00	0,00	0,02
nug18	1930	1930,00	0,00	0,00
nug20	2570	2570,00	0,00	0,00
nug21	2438	2438,00	0,00	0,01
nug22	3596	3596,00	0,00	0,01
nug24	3488	3488,00	0,00	0,01
nug25	3744	3744,00	0,00	0,03
nug27	5234	5234,00	0,00	0,02
nug28	5166	5166,00	0,00	0,17
nug30	6124	6125,33	0,02	1,15
scr12	31410	31410,00	0,00	0,00
scr15	51140	51140,00	0,00	0,00
scr20	110030	110030,00	0,00	0,01
sko42	15812	15861,00	0,31	2,00
sko49	23386	23470,00	0,36	2,00
sko56	34458	34602,00	0,42	2,00
sko64	48498	48757,33	0,53	2,00
sko72	66256	66657,00	0,61	2,00
sko81	90998	91415,67	0,46	2,00
sko90	115534	116207,33	0,58	2,00
sko100a	152002	152840,00	0,55	2,00
sko100b	153890	154780,33	0,58	2,00
sko100c	147862	148774,67	0,62	2,00
sko100d	149576	150582,67	0,67	2,00
sko100e	149150	150127,00	0,66	2,00
sko100f	149036	150062,67	0,69	2,00
tho30	149936	149936,00	0,00	0,69
tho40	240516	241222,00	0,29	1,61
tho150	8133398	8209299,67	0,93	2,00
wil50	48816	48888,67	0,15	2,00
wil100	273038	273850,67	0,30	2,00
Average			0,24	0,99

Πίνακας **5.13** Αποτελέσματα στιγμιοτύπων τύπου 3 με χρονικό όριο τα 2 λεπτά

Instance	zOpt	Avg best solution	Avg Error (%)	Average Time
kra30a	88900	88900,00	0,00	0,13
kra30b	91420	91420,00	0,00	0,85
kra32	88700	88700,00	0,00	0,13
bur26a	5426670	5426670,00	0,00	0,02
bur26b	3817852	3817852,00	0,00	0,04
bur26c	5426795	5426795,00	0,00	0,03
bur26d	3821225	3821225,00	0,00	0,09
bur26e	5386879	5386879,00	0,00	0,03
bur26f	3782044	3782044,00	0,00	0,02
bur26g	10117172	10117172,00	0,00	0,29
bur26h	7098658	7098658,00	0,00	0,03
chr12a	9552	9552,00	0,00	0,00
chr12b	9742	9742,00	0,00	0,00
chr12c	11156	11156,00	0,00	0,00
chr15a	9896	9896,00	0,00	0,00
chr15b	7990	7990,00	0,00	0,00
chr15c	9504	9504,00	0,00	0,05
chr18a	11098	11098,00	0,00	0,11
chr18b	1534	1534,00	0,00	0,01
chr20a	2192	2192,00	0,00	0,42
chr20b	2298	2326,33	1,23	1,28
chr20c	14142	14142,00	0,00	0,01
chr22a	6156	6159,33	0,05	1,04
chr22b	6194	6227,00	0,53	1,60
chr25a	3796	3810,67	0,39	0,78
els19	17212548	17212548,00	0,00	0,14
esc16a	68	68,00	0,00	0,00
esc16b	292	292,00	0,00	0,00
esc16c	160	160,00	0,00	0,00
esc16d	16	16,00	0,00	0,00
esc16e	28	28,00	0,00	0,00
esc16g	26	26,00	0,00	0,00
esc16h	996	996,00	0,00	0,00
esc16i	14	14,00	0,00	0,00
esc16j	8	8,00	0,00	0,00
esc32a	130	130,00	0,00	0,39
esc32b	168	168,00	0,00	0,00
esc32c	642	642,00	0,00	0,00
esc32d	200	200,00	0,00	0,00
esc32e	2	2,00	0,00	0,00
esc32g	6	6,00	0,00	0,00
esc32h	438	438,00	0,00	0,00
esc64a	116	116,00	0,00	0,00
esc128	64	64,00	0,00	0,03
had12	1652	1652,00	0,00	0,00
had14	2724	2724,00	0,00	0,00
had16	3720	3720,00	0,00	0,00
had18	5358	5358,00	0,00	0,00
had20	6922	6922,00	0,00	0,00
ste36a	9526	9592,00	0,69	2,00
ste36b	15852	15897,67	0,29	1,93
ste36c	8239110	8261068,00	0,27	2,00
Average			0,07	0,26

Πίνακας **5.14** Αποτελέσματα στιγμιοτύπων τύπου 4 με χρονικό όριο τα 2 λεπτά

Instance	zOpt	Avg best solution	Avg Error (%)	Average Time
tai20b	122455319	122455319,00	0,00	0,00
tai25b	344355646	344355646,00	0,00	0,02
tai30b	637117113	637247245,00	0,02	2,00
tai35b	283315445	283701994,17	0,14	1,26
tai40b	637250948	637320229,00	0,01	1,63
tai50b	458821517	459551888,00	0,16	2,00
tai60b	608215054	609880277,00	0,27	2,00
tai64c	1855928	1855928,00	0,00	0,00
tai80b	818415043	829209994,17	1,32	2,00
tai100b	1185996137	1196095551,67	0,85	2,00
tai150b	498896643	506449184,67	1,51	2,00
Average			0,39	1,36

Πίνακας **5.15** Μέσο σφάλμα και μέσος χρόνος όλων των στιγμιοτύπων για 2 λεπτά

Average Error	0,53
Average Time	0,93

Βλέπουμε ότι υπάρχει και πάλι βελτίωση όσο αυξάνεται το χρονικό όριο αλλά κάποια στιγμή η βελτίωσή αυτή λογικά θα μειωθεί. Όσον αφορά το πιο δύσκολο στιγμιότυπο, το lipa90b, υπάρχει βελτίωση σε σχέση με πριν, αλλά και πάλι βλέπουμε ότι στο μισό λεπτό έχουμε καλύτερο αποτέλεσμα. Αυτό συνέβη επειδή έτυχε να απεγκλωβιστεί πιο εύκολα στο μισό λεπτό, αλλά όσο ανεβαίνει το χρονικό όριο, οι πιθανότητες καλύτερων αποτελεσμάτων αυξάνονται κατά πολύ. Βάζοντας το χρονικό όριο στα 3 λεπτά παίρνουμε τα εξής αποτελέσματα:

Πίνακας **5.16** Αποτελέσματα στιγμιοτύπων τύπου 1 με χρονικό όριο τα 3 λεπτά

Instance	zOpt	Avg best solution	Avg Error (%)	Average Time
tai20a	703482	703482,00	0,00	0,09
tai25a	1167256	1168216,00	0,08	1,06
tai30a	1818146	1829650,00	0,63	3,00
tai35a	2422002	2456604,80	1,43	3,00
tai40a	3139370	3190730,00	1,64	3,00
tai50a	4938796	5056783,60	2,39	3,00
tai60a	7205962	7383344,80	2,46	3,00
tai80a	13499184	13849280,00	2,59	3,00
tai100a	21052466	21570538,40	2,46	3,00
rou12	235528	235528,00	0,00	0,01
rou15	354210	354210,00	0,00	0,00
rou20	725522	725522,00	0,00	0,04
lipa20a	3683	3683,00	0,00	0,00
lipa20b	27076	27076,00	0,00	0,00
lipa30a	13178	13178,00	0,00	0,05
lipa30b	151426	151426,00	0,00	0,00
lipa40a	31538	31714,40	0,56	2,48
lipa40b	476581	476581,00	0,00	0,00
lipa50a	62093	62703,20	0,98	3,00
lipa50b	1210244	1210244,00	0,00	0,00
lipa60a	107218	108146,00	0,87	3,00
lipa60b	2520135	2520135,00	0,00	0,18
lipa70a	169755	171074,60	0,78	3,00
lipa70b	4603200	4603200,00	0,00	0,20
lipa80a	253195	254948,20	0,69	3,00
lipa80b	7763962	7763962,00	0,00	0,91
lipa90a	360630	362983,00	0,65	3,00
lipa90b	12490441	13525882,80	8,29	1,65
Average			0,95	1,52

Πίνακας **5.17** Αποτελέσματα στιγμιοτύπων τύπου 2 με χρονικό όριο τα 3 λεπτά

Instance	zOpt	Avg best solution	Avg Error (%)	Average Time
nug12	578	578,00	0,00	0,00
nug14	1014	1014,00	0,00	0,00
nug15	1150	1150,00	0,00	0,00
nug16a	1610	1610,00	0,00	0,00
nug16b	1240	1240,00	0,00	0,00
nug17	1732	1732,00	0,00	0,01
nug18	1930	1930,00	0,00	0,00
nug20	2570	2570,00	0,00	0,00
nug21	2438	2438,00	0,00	0,01
nug22	3596	3596,00	0,00	0,00
nug24	3488	3488,00	0,00	0,01
nug25	3744	3744,00	0,00	0,02
nug27	5234	5234,00	0,00	0,03
nug28	5166	5166,00	0,00	0,14
nug30	6124	6124,00	0,00	1,00
scr12	31410	31410,00	0,00	0,00
scr15	51140	51140,00	0,00	0,00
scr20	110030	110030,00	0,00	0,01
sko42	15812	15822,67	0,07	2,42
sko49	23386	23464,00	0,33	3,00
sko56	34458	34599,33	0,41	3,00
sko64	48498	48701,00	0,42	3,00
sko72	66256	66615,00	0,54	3,00
sko81	90998	91454,00	0,50	3,00
sko90	115534	116229,00	0,60	3,00
sko100a	152002	152736,67	0,48	3,00
sko100b	153890	154665,00	0,50	3,00
sko100c	147862	148565,67	0,48	3,00
sko100d	149576	150445,00	0,58	3,00
sko100e	149150	149926,67	0,52	3,00
sko100f	149036	149975,00	0,63	3,00
tho30	149936	149936,00	0,00	0,31
tho40	240516	241292,67	0,32	3,00
tho150	8133398	8203571,00	0,86	3,00
wil50	48816	48874,33	0,12	3,00
wil100	273038	273928,67	0,33	3,00
Average			0,21	1,44

Πίνακας **5.18** Αποτελέσματα στιγμιοτύπων τύπου 3 με χρονικό όριο τα 3 λεπτά

Instance	zOpt	Avg best solution	Avg Error (%)	Average Time
kra30a	88900	88900,00	0,00	0,28
kra30b	91420	91420,00	0,00	1,20
kra32	88700	88700,00	0,00	0,10
bur26a	5426670	5426670,00	0,00	0,01
bur26b	3817852	3817852,00	0,00	0,09
bur26c	5426795	5426795,00	0,00	0,02
bur26d	3821225	3821225,00	0,00	0,15
bur26e	5386879	5386879,00	0,00	0,07
bur26f	3782044	3782044,00	0,00	0,01
bur26g	10117172	10117172,00	0,00	0,22
bur26h	7098658	7098658,00	0,00	0,03
chr12a	9552	9552,00	0,00	0,00
chr12b	9742	9742,00	0,00	0,00
chr12c	11156	11156,00	0,00	0,01
chr15a	9896	9896,00	0,00	0,00
chr15b	7990	7990,00	0,00	0,00
chr15c	9504	9504,00	0,00	0,04
chr18a	11098	11098,00	0,00	0,07
chr18b	1534	1534,00	0,00	0,00
chr20a	2192	2192,00	0,00	0,82
chr20b	2298	2322,67	1,07	2,32
chr20c	14142	14142,00	0,00	0,01
chr22a	6156	6166,00	0,16	1,91
chr22b	6194	6235,00	0,66	3,00
chr25a	3796	3796,00	0,00	1,66
els19	17212548	17212548,00	0,00	0,03
esc16a	68	68,00	0,00	0,00
esc16b	292	292,00	0,00	0,00
esc16c	160	160,00	0,00	0,00
esc16d	16	16,00	0,00	0,00
esc16e	28	28,00	0,00	0,00
esc16g	26	26,00	0,00	0,00
esc16h	996	996,00	0,00	0,00
esc16i	14	14,00	0,00	0,00
esc16j	8	8,00	0,00	0,00
esc32a	130	130,00	0,00	0,18
esc32b	168	168,00	0,00	0,00
esc32c	642	642,00	0,00	0,00
esc32d	200	200,00	0,00	0,00
esc32e	2	2,00	0,00	0,00
esc32g	6	6,00	0,00	0,00
esc32h	438	438,00	0,00	0,00
esc64a	116	116,00	0,00	0,00
esc128	64	64,00	0,00	0,03
had12	1652	1652,00	0,00	0,00
had14	2724	2724,00	0,00	0,00
had16	3720	3720,00	0,00	0,00
had18	5358	5358,00	0,00	0,00
had20	6922	6922,00	0,00	0,00
ste36a	9526	9578,33	0,55	3,00
ste36b	15852	15858,67	0,04	2,03
ste36c	8239110	8260284,67	0,26	3,00
Average			0,05	0,39

Πίνακας **5.19** Αποτελέσματα στιγμιοτύπων τύπου 4 με χρονικό όριο τα 3 λεπτά

Instance	zOpt	Avg best solution	Avg Error (%)	Average Time
tai20b	122455319	122455319,00	0,00	0,00
tai25b	344355646	344355646,00	0,00	0,02
tai30b	637117113	637132517,50	0,00	2,57
tai35b	283315445	283477809,33	0,06	2,24
tai40b	637250948	637276428,33	0,00	2,01
tai50b	458821517	459047288,33	0,05	3,00
tai60b	608215054	608951853,67	0,12	3,00
tai64c	1855928	1855928,00	0,00	0,00
tai80b	818415043	824618129,17	0,76	3,00
tai100b	1185996137	1193059964,17	0,60	3,00
tai150b	498896643	505359595,33	1,30	3,00
Average			0,26	1,99

Πίνακας **5.20** Μέσο σφάλμα και μέσος χρόνος όλων των στιγμιοτύπων για 3 λεπτά

Average Error	0,37
Average Time	1,34

Εδώ υπάρχει μια μεγαλύτερη βελτίωση καθώς το μέσο σφάλμα έπεσε από 0.53 σε 0.37 τοις εκατό. Προφανώς αυξάνεται ο μέσος χρόνος υπολογισμού αφού ανεβάζουμε το ανώτατο χρονικό όριο αλλά αυτό είναι απολύτως λογικό. Επίσης εδω παρατηρείται και το μικρότερο μέσο σφάλμα στο δύσκολο έως τώρα στιγμιότυπο lipa90b. Ακολουθούν τα αποτελέσματα με χρονικό όριο τα 4 λεπτά.

Πίνακας **5.21** Αποτελέσματα στιγμιοτύπων τύπου 1 με χρονικό όριο τα 4 λεπτά

Instance	zOpt	Avg best solution	Avg Error (%)	Average Time
tai20a	703482	703482,00	0,00	0,09
tai25a	1167256	1168355,60	0,09	2,23
tai30a	1818146	1829632,80	0,63	3,66
tai35a	2422002	2451543,60	1,22	4,00
tai40a	3139370	3181086,00	1,33	4,00
tai50a	4938796	5052828,40	2,31	4,00
tai60a	7205962	7381951,20	2,44	4,00
tai80a	13499184	13843024,00	2,55	4,00
tai100a	21052466	21557829,20	2,40	4,00
rou12	235528	235528,00	0,00	0,00
rou15	354210	354210,00	0,00	0,00
rou20	725522	725522,00	0,00	0,01
lipa20a	3683	3683,00	0,00	0,00
lipa20b	27076	27076,00	0,00	0,00
lipa30a	13178	13178,00	0,00	0,07
lipa30b	151426	151426,00	0,00	0,00
lipa40a	31538	31731,40	0,61	3,76
lipa40b	476581	476581,00	0,00	0,00
lipa50a	62093	62676,40	0,94	4,00
lipa50b	1210244	1210244,00	0,00	0,01
lipa60a	107218	108143,80	0,86	4,00
lipa60b	2520135	2520135,00	0,00	0,08
lipa70a	169755	171062,60	0,77	4,00
lipa70b	4603200	4603200,00	0,00	0,21
lipa80a	253195	254934,80	0,69	4,00
lipa80b	7763962	7763962,00	0,00	1,08
lipa90a	360630	362956,80	0,65	4,00
lipa90b	12490441	12490441,00	0,00	1,49
Average			0,62	2,02

Πίνακας **5.22** Αποτελέσματα στιγμιοτύπων τύπου 2 με χρονικό όριο τα 4 λεπτά

Instance	zOpt	Avg best solution	Avg Error (%)	Average Time
nug12	578	578,00	0,00	0,00
nug14	1014	1014,00	0,00	0,00
nug15	1150	1150,00	0,00	0,00
nug16a	1610	1610,00	0,00	0,00
nug16b	1240	1240,00	0,00	0,00
nug17	1732	1732,00	0,00	0,01
nug18	1930	1930,00	0,00	0,00
nug20	2570	2570,00	0,00	0,00
nug21	2438	2438,00	0,00	0,00
nug22	3596	3596,00	0,00	0,00
nug24	3488	3488,00	0,00	0,03
nug25	3744	3744,00	0,00	0,02
nug27	5234	5234,00	0,00	0,06
nug28	5166	5166,00	0,00	0,15
nug30	6124	6124,67	0,01	1,77
scr12	31410	31410,00	0,00	0,00
scr15	51140	51140,00	0,00	0,00
scr20	110030	110030,00	0,00	0,01
sko42	15812	15839,00	0,17	4,00
sko49	23386	23459,00	0,31	4,00
sko56	34458	34568,00	0,32	4,00
sko64	48498	48696,67	0,41	4,00
sko72	66256	66598,67	0,52	4,00
sko81	90998	91451,33	0,50	4,00
sko90	115534	116236,33	0,61	4,00
sko100a	152002	152712,00	0,47	4,00
sko100b	153890	154745,00	0,56	4,00
sko100c	147862	148541,67	0,46	4,00
sko100d	149576	150396,00	0,55	4,00
sko100e	149150	150061,67	0,61	4,00
sko100f	149036	149838,33	0,54	4,00
tho30	149936	149936,00	0,00	0,42
tho40	240516	241237,33	0,30	4,00
tho150	8133398	8195595,67	0,76	4,00
wil50	48816	48874,00	0,12	4,00
wil100	273038	273892,00	0,31	4,00
Average			0,21	1,96

Πίνακας **5.23** Αποτελέσματα στιγμιοτύπων τύπου 3 με χρονικό όριο τα 4 λεπτά

Instance	zOpt	Avg best solution	Avg Error (%)	Average Time
kra30a	88900	88900,00	0,00	0,17
kra30b	91420	91420,00	0,00	1,11
kra32	88700	88700,00	0,00	0,09
bur26a	5426670	5426670,00	0,00	0,02
bur26b	3817852	3817852,00	0,00	0,05
bur26c	5426795	5426795,00	0,00	0,02
bur26d	3821225	3821225,00	0,00	0,07
bur26e	5386879	5386879,00	0,00	0,05
bur26f	3782044	3782044,00	0,00	0,00
bur26g	10117172	10117172,00	0,00	0,21
bur26h	7098658	7098658,00	0,00	0,02
chr12a	9552	9552,00	0,00	0,00
chr12b	9742	9742,00	0,00	0,00
chr12c	11156	11156,00	0,00	0,01
chr15a	9896	9896,00	0,00	0,01
chr15b	7990	7990,00	0,00	0,00
chr15c	9504	9504,00	0,00	0,05
chr18a	11098	11098,00	0,00	0,07
chr18b	1534	1534,00	0,00	0,01
chr20a	2192	2192,00	0,00	0,98
chr20b	2298	2342,00	1,91	4,00
chr20c	14142	14142,00	0,00	0,01
chr22a	6156	6162,33	0,10	1,61
chr22b	6194	6229,33	0,57	4,00
chr25a	3796	3820,67	0,65	2,32
els19	17212548	17212548,00	0,00	0,02
esc16a	68	68,00	0,00	0,00
esc16b	292	292,00	0,00	0,00
esc16c	160	160,00	0,00	0,00
esc16d	16	16,00	0,00	0,00
esc16e	28	28,00	0,00	0,00
esc16g	26	26,00	0,00	0,00
esc16h	996	996,00	0,00	0,00
esc16i	14	14,00	0,00	0,00
esc16j	8	8,00	0,00	0,00
esc32a	130	130,00	0,00	0,43
esc32b	168	168,00	0,00	0,01
esc32c	642	642,00	0,00	0,00
esc32d	200	200,00	0,00	0,00
esc32e	2	2,00	0,00	0,00
esc32g	6	6,00	0,00	0,00
esc32h	438	438,00	0,00	0,00
esc64a	116	116,00	0,00	0,00
esc128	64	64,00	0,00	0,04
had12	1652	1652,00	0,00	0,00
had14	2724	2724,00	0,00	0,00
had16	3720	3720,00	0,00	0,00
had18	5358	5358,00	0,00	0,00
had20	6922	6922,00	0,00	0,00
ste36a	9526	9563,67	0,40	4,00
ste36b	15852	15858,67	0,04	2,76
ste36c	8239110	8262873,67	0,29	4,00
Average			0,08	0,50

Πίνακας **5.24** Αποτελέσματα στιγμιοτύπων τύπου 4 με χρονικό όριο τα 4 λεπτά

Instance	zOpt	Avg best solution	Avg Error (%)	Average Time
tai20b	122455319	122455319,00	0,00	0,00
tai25b	344355646	344355646,00	0,00	0,01
tai30b	637117113	637128540,33	0,00	3,14
tai35b	283315445	283508931,67	0,07	3,58
tai40b	637250948	637320158,33	0,01	2,74
tai50b	458821517	459093885,50	0,06	4,00
tai60b	608215054	608773437,00	0,09	4,00
tai64c	1855928	1855928,00	0,00	0,00
tai80b	818415043	822468750,83	0,50	4,00
tai100b	1185996137	1192866745,50	0,58	4,00
tai150b	498896643	504963542,33	1,22	4,00
Average			0,23	2,68

Πίνακας **5.25** Μέσο σφάλμα και μέσος χρόνος όλων των στιγμιοτύπων για 4 λεπτά

Average Error	0,28
Average Time	1,79

Εδώ παρατηρείται επίσης βελτίωση αλλά μετά από αυτό το σημείο όπως θα δούμε και στη συνέχεια η βελτίωση αυξάνεται ελάχιστα όσο αυξάνουμε το χρονικό όριο. Θα δούμε και τα αποτελέσματα με χρονικό όριο τα 60 λεπτά, αλλά η διαφορά δεν είναι αρκετά μεγάλη έτσι ώστε να ελεγχθούν και τα ενδιάμεσα χρονικά όρια. Το μέχρι πρότεινος μεγαλύτερο σφάλμα το είχε το στιγμιότυπο lipa90b, το οποίο τώρα έχει μηδενικό μέσο σφάλμα, δηλαδή επιλύθηκε όλες τις φορές που επιχειρήθηκε. Σε αντίθεση το στιγμιότυπο lipa90a που είχε ήδη μικρό σφάλμα σε όλα τα υπόλοιπα χρονικά όρια, παραμένει με μικρό σφάλμα, αλλά η ολική του επίλυση είναι πάρα πολύ δύσκολη. Ας δούμε και τα τελευταία αποτελέσματα με χρονικό όριο τα 60 λεπτά για να μπορούμε να έχουμε μια συνολική εικόνα.

Πίνακας **5.26** Αποτελέσματα στιγμιοτύπων τύπου 1 με χρονικό όριο τα 60 λεπτά

Instance	zOpt	Avg best solution	Avg Error (%)	Average Time
tai20a	703482	703482,00	0,00	0,23
tai25a	1167256	1167256,00	0,00	3,32
tai30a	1818146	1818907,60	0,04	47,95
tai35a	2422002	2442744,40	0,86	60,00
tai40a	3139370	3172620,00	1,06	60,00
tai50a	4938796	5022371,60	1,69	60,00
tai60a	7205962	7364148,00	2,20	60,00
tai80a	13499184	13809834,80	2,30	60,00
tai100a	21052466	21524117,60	2,24	60,00
rou12	235528	235528,00	0,00	0,00
rou15	354210	354210,00	0,00	0,00
rou20	725522	725522,00	0,00	0,05
lipa20a	3683	3683,00	0,00	0,00
lipa20b	27076	27076,00	0,00	0,00
lipa30a	13178	13178,00	0,00	0,08
lipa30b	151426	151426,00	0,00	0,00
lipa40a	31538	31538,00	0,00	12,47
lipa40b	476581	476581,00	0,00	0,00
lipa50a	62093	62601,20	0,82	60,00
lipa50b	1210244	1210244,00	0,00	0,01
lipa60a	107218	108097,60	0,82	60,00
lipa60b	2520135	2520135,00	0,00	0,06
lipa70a	169755	171011,40	0,74	60,00
lipa70b	4603200	4603200,00	0,00	0,13
lipa80a	253195	254889,60	0,67	60,00
lipa80b	7763962	7763962,00	0,00	0,88
lipa90a	360630	362879,00	0,62	60,00
lipa90b	12490441	12490441,00	0,00	3,99
Average			0,50	26,04

Πίνακας **5.27** Αποτελέσματα στιγμιοτύπων τύπου 2 με χρονικό όριο τα 60 λεπτά

Instance	zOpt	Avg best solution	Avg Error (%)	Average Time
nug12	578	578,00	0,00	0,00
nug14	1014	1014,00	0,00	0,00
nug15	1150	1150,00	0,00	0,00
nug16a	1610	1610,00	0,00	0,01
nug16b	1240	1240,00	0,00	0,00
nug17	1732	1732,00	0,00	0,02
nug18	1930	1930,00	0,00	0,00
nug20	2570	2570,00	0,00	0,00
nug21	2438	2438,00	0,00	0,00
nug22	3596	3596,00	0,00	0,00
nug24	3488	3488,00	0,00	0,01
nug25	3744	3744,00	0,00	0,01
nug27	5234	5234,00	0,00	0,09
nug28	5166	5166,00	0,00	0,09
nug30	6124	6124,00	0,00	1,73
scr12	31410	31410,00	0,00	0,00
scr15	51140	51140,00	0,00	0,00
scr20	110030	110030,00	0,00	0,01
sko42	15812	15812,67	0,00	30,79
sko49	23386	23419,67	0,14	52,32
sko56	34458	34530,00	0,21	60,00
sko64	48498	48634,33	0,28	60,00
sko72	66256	66485,67	0,35	60,00
sko81	90998	91238,33	0,26	60,00
sko90	115534	115978,00	0,38	60,00
sko100a	152002	152577,00	0,38	60,00
sko100b	153890	154482,67	0,39	60,00
sko100c	147862	148381,33	0,35	60,00
sko100d	149576	150142,00	0,38	60,00
sko100e	149150	149794,67	0,43	60,00
sko100f	149036	149757,00	0,48	60,00
tho30	149936	149936,00	0,00	0,91
tho40	240516	240675,00	0,07	60,00
tho150	8133398	8185753,33	0,64	60,00
wil50	48816	48846,33	0,06	60,00
wil100	273038	273670,00	0,23	60,00
Average			0,14	27,39

Πίνακας **5.28** Αποτελέσματα στιγμιοτύπων τύπου 3 με χρονικό όριο τα 60 λεπτά

Instance	zOpt	Avg best solution	Avg Error (%)	Average Time
kra30a	88900	88900,00	0,00	0,27
kra30b	91420	91420,00	0,00	0,64
kra32	88700	88700,00	0,00	0,07
bur26a	5426670	5426670,00	0,00	0,02
bur26b	3817852	3817852,00	0,00	0,09
bur26c	5426795	5426795,00	0,00	0,05
bur26d	3821225	3821225,00	0,00	0,07
bur26e	5386879	5386879,00	0,00	0,02
bur26f	3782044	3782044,00	0,00	0,02
bur26g	10117172	10117172,00	0,00	0,14
bur26h	7098658	7098658,00	0,00	0,05
chr12a	9552	9552,00	0,00	0,00
chr12b	9742	9742,00	0,00	0,00
chr12c	11156	11156,00	0,00	0,01
chr15a	9896	9896,00	0,00	0,01
chr15b	7990	7990,00	0,00	0,00
chr15c	9504	9504,00	0,00	0,04
chr18a	11098	11098,00	0,00	0,05
chr18b	1534	1534,00	0,00	0,00
chr20a	2192	2192,00	0,00	0,58
chr20b	2298	2298,00	0,00	16,50
chr20c	14142	14142,00	0,00	0,01
chr22a	6156	6156,00	0,00	1,09
chr22b	6194	6206,00	0,19	36,56
chr25a	3796	3796,00	0,00	1,56
els19	17212548	17212548,00	0,00	0,04
esc16a	68	68,00	0,00	0,00
esc16b	292	292,00	0,00	0,00
esc16c	160	160,00	0,00	0,00
esc16d	16	16,00	0,00	0,00
esc16e	28	28,00	0,00	0,00
esc16g	26	26,00	0,00	0,00
esc16h	996	996,00	0,00	0,00
esc16i	14	14,00	0,00	0,00
esc16j	8	8,00	0,00	0,00
esc32a	130	130,00	0,00	0,73
esc32b	168	168,00	0,00	0,00
esc32c	642	642,00	0,00	0,00
esc32d	200	200,00	0,00	0,00
esc32e	2	2,00	0,00	0,00
esc32g	6	6,00	0,00	0,00
esc32h	438	438,00	0,00	0,00
esc64a	116	116,00	0,00	0,00
esc128	64	64,00	0,00	0,02
had12	1652	1652,00	0,00	0,00
had14	2724	2724,00	0,00	0,00
had16	3720	3720,00	0,00	0,00
had18	5358	5358,00	0,00	0,00
had20	6922	6922,00	0,00	0,00
ste36a	9526	9534,00	0,08	56,14
ste36b	15852	15852,00	0,00	3,29
ste36c	8239110	8244696,00	0,07	49,69
Average			0,01	3,23

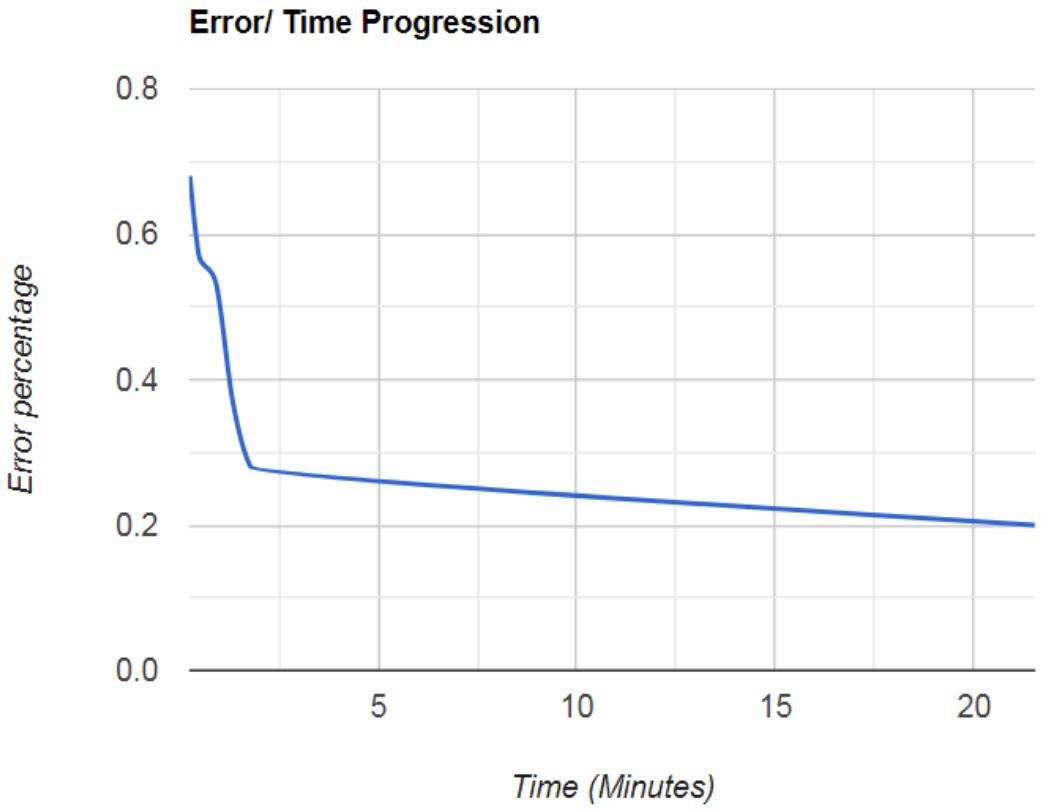
Πίνακας **5.29** Αποτελέσματα στιγμιοτύπων τύπου 4 με χρονικό όριο τα 60 λεπτά

Instance	zOpt	Avg best solution	Avg Error (%)	Average Time
tai20b	122455319	122455319,00	0,00	0,00
tai25b	344355646	344355646,00	0,00	0,02
tai30b	637117113	637117113,00	0,00	21,13
tai35b	283315445	283315445,00	0,00	4,97
tai40b	637250948	637250948,00	0,00	1,04
tai50b	458821517	458848313,67	0,01	57,00
tai60b	608215054	608544226,83	0,05	60,00
tai64c	1855928	1855928,00	0,00	0,00
tai80b	818415043	819874923,67	0,18	60,00
tai100b	1185996137	1190138950,33	0,35	60,00
tai150b	498896643	503632829,83	0,95	60,00
Average			0,14	29,47

Πίνακας **5.30** Μέσο σφάλμα και μέσος χρόνος όλων των στιγμιοτύπων για 60 λεπτά

Average Error	0,20
Average Time	21,53

Παρατηρούμε ότι αν και αυξήθηκε το όριο από τα 4 λεπτά στα 60, το μέσο σφάλμα πήγε από 0.28 σε 0.20 ενώ σαφώς ο μέσος χρόνος πήγε από τα 1.79 λεπτά στα 21.53 λεπτά. Οπότε το πρόγραμμα πραγματοποιεί αρκετές βελτιώσεις μέχρι και τα 4 λεπτά, αλλά παραπάνω χρόνος εκτέλεσης επιφέρει μικρές βελτιώσεις. Το ιδανικό χρονικό όριο λοιπόν είναι στα 4 λεπτά, εκτός αν είμαστε διατεθειμένοι να το ανεβάσουμε κοντά στο δεκάλεπτο για μια μικρή βελτίωση, αν δεν μας πιέζει ο χρόνος. Αξίζει επίσης να σημειωθεί ότι το στιγμιότυπο lipa90a, ακόμα και μετά από 60 λεπτά, έχει μηδαμινή βελτίωση σε σχέση με μικρότερα χρονικά όρια. Αυτό έχει κυρίως να κάνει με την τυχαία δομή του, όπως και τα υπόλοιπα lipa*a, τα οποία μπορεί εξαρχής να είχαν πολύ μικρό σφάλμα σε σχέση με τα lipa*b, αλλά δεν παρουσιάζουν σε καμία περίπτωση τη βελτίωση των στιγμιοτύπων lipa*b, η οποία καταλήγει εν τέλει και στην επίλυσή τους. Ακολουθεί σχετικό γράφημα που μας δείχνει πώς μεταβάλλεται το μέσο σφάλμα επί τοις εκατό σε σχέση με το μέσο χρόνο επίλυσης.



Σχήμα **5.1.1** Γράφημα μέσου σφάλματος

Εδώ φαίνεται ξεκάθαρα ότι ενώ αρχικά υπάρχει αισθητή βελτίωση όσο αυξάνουμε το χρονικό όριο, από ένα σημείο και μετά αυτή η αύξηση δεν επιφέρει τόσο σημαντική βελτίωση. Συγκεκριμένα, όταν ο μέσος χρόνος είναι στα 1.79 λεπτά, που είναι η τιμή που έχουμε για ανώτατο όριο τα 4 λεπτά, βρισκόμαστε στο σημείο που σταματάει η γρήγορη αύξηση της βελτίωσης. Αντιθέτως, όταν ο μέσος χρόνος είναι στα 21.53 λεπτά, δηλαδή όταν έχουμε ανώτατο όριο τα 60 λεπτά, υπάρχει μεν βελτίωση, αλλά δεν είναι τόσο μεγάλη έτσι ώστε να δικαιολογηθεί η ανάθεση των 60 λεπτών ως ανώτατο χρονικό όριο. Όπως είπαμε και πριν, με ανώτατο όριο τα 4 λεπτά, έχουμε την καλύτερη αναλογία χρόνου-σφάλματος.

Στο επόμενο κεφάλαιο θα παρουσιαστεί η σύγκριση ανάμεσα στο πρόγραμμα που αναπτύχθηκε και άλλους λύτες που θεωρούνται State of the art.

5.2 Σύγκριση με άλλους State of the Art λύτες

Σε αυτό το υποκεφάλαιο θα συγκρίνουμε τα αποτελέσματα του προγράμματος με άλλους λύτες που θεωρούνται State of the Art, δηλαδή οι καλύτεροι που έχουν υλοποιηθεί στο υψηλό επίπεδο. Αφού παρουσιαστούν τα αριθμητικά δεδομένα και αποτελέσματα του προγράμματος και κάποιων

άλλων προσεγγίσεων, θα ακολουθήσει η ανάλυση αυτών των δεδομένων.

Ο παρακάτω πίνακας δείχνει τη συγκριτική μελέτη του προγράμματος που αναπτύχθηκε με άλλους State of the Art λύτες. Οι στήλες είναι κατά σειρά: Το όνομα του στιγμιότυπου, η καλύτερη γνωστή λύση, το σφάλμα σε ποσοστό επί τοις εκατό και ο χρόνος επίλυσης σε λεπτά. Οι άλλοι λύτες που χρησιμοποιήθηκαν για τη σύγκριση είναι οι εξής: ITS: Iterated Tabu Search (Misevicius 2011), PHA: Parallel Hybrid Algorithm (Tosun 2015), BLS: parallel Breakout Local Search (Aksan, Dokeroglu, and Cosar 2017), MS-ITS: parallel Multi-Start Iterated Tabu Search (Silva, Coelho, and Darvish 2019). Ο λύτης της τελευταίας στήλης, ο MS-VNS: Multi-Start Variable Neighbourhood Search, είναι αυτός που αναπτύχθηκε. Είναι αξιοσημείωτο και πρέπει να τονιστεί, ότι οι υπόλοιποι λύτες εκτός του πρώτου, χρησιμοποιούν παράλληλες τεχνικές και έτσι αποκτούν προβάδισμα σε σχέση με τις σειριακές υλοποιήσεις. Μια πρόταση για βελτίωση του προγράμματος δηλαδή θα ήταν η παραλληλοποίησή του χρησιμοποιώντας όμως διαφορετικά εργαλεία.

Όσον αφορά τώρα την επιλογή των στιγμιοτύπων αυτών, πρέπει να δικαιολογηθεί το γιατί επιλέχθηκαν αυτά τα συγκεκριμένα στιγμιότυπα. Όπως θα δούμε και στον πίνακα, τα στιγμιότυπα περιλαμβάνουν ένα μέρος τύπου 1, ένα μέρος τύπου 2, κανένα στιγμιότυπο τύπου 3 και όλα του τύπου 4. Αυτά τα στιγμιότυπα που επιλέχθηκαν, θεωρούνται ευρέως τα πιο δύσκολα, σε αντίθεση με τα υπόλοιπα που η επίλυσή τους πραγματοποιείται ευκολότερα. Είδαμε και πριν ότι ο τύπος 3 που περιέχει στιγμιότυπα από την πραγματική ζωή είναι ο ευκολότερος με μέσο σφάλμα 0.08 τοις εκατό και μέσο χρόνο επίλυσης 0.5 λεπτά, όταν το ανώτατο όριο τίθεται στα 4 λεπτά. Προφανώς με τέτοια αποτελέσματα δε μπορεί να θεωρηθεί δύσκολη η επίλυση των στιγμιοτύπων που περιλαμβάνει.

Βλέπουμε ότι τα μόνα στιγμιότυπα που επιλέχθηκε να συγκριθούν είναι της μορφής tai* και sko*. Από τον τύπο 1 παραλείφθηκαν τα rou* και lipa* που και στα αποτελέσματα του προγράμματος έχουν πολύ χαμηλό σφάλμα. Τέλος, τα στιγμιότυπα που παραλείφθηκαν από τον τύπο 2 έχουν επίσης χαμηλές τιμές σφάλματος. Σύμφωνα και με την έρευνα των Benlic και Hao (Benlic and Hao 2015), τα στιγμιότυπα που επιλέχθηκαν είναι τα πιο δύσκολα της βιβλιοθήκης QAPLIB. Ας δούμε λοιπόν παρακάτω τον πίνακα σύγκρισης:

Πίνακας **5.31** Σύγκριση με άλλους State of the Art λύτες

Instance	BKS	ITS		PHA		BLS		MS-ITS		MS-VNS	
		Gap	Time	Gap	Time	Gap	Time	Gap	Time	Gap	Time
tai20a	703482	0	0	0	0,4	0	0,1	0	0	0,00	0,09
tai25a	1167256	0	0,1	0	0,5	0	0,1	0	0	0,09	2,23
tai30a	1818146	0	0,2	0	1	0	0,2	0	0,1	0,63	3,66
tai35a	2422002	0	0,5	0	1,3	0	0,3	0	0,4	1,22	4,00
tai40a	3139370	0,22	1,3	0	10,6	0	32,2	0	30,5	1,33	4,00
tai50a	4938796	0,41	5,5	0	12,7	0	68,2	0	60	2,31	4,00
tai60a	7205962	0,45	12,5	0	19,6	0	108	0,2	60	2,44	4,00
tai80a	13499184	0,36	60	0,64	40	0,5	236	0,4	60	2,55	4,00
tai100a	21052466	0,3	200	0,54	71,9	0,6	449	0,3	60	2,40	4,00
tai20b	122455319	0	0	0	0,4	0	0,1	0	0	0,00	0,00
tai25b	344355646	0	0	0	0,6	0	0	0	0	0,00	0,01
tai30b	637117113	0,01	0	0	0,8	0	0,2	0	0,2	0,00	3,14
tai35b	283315445	0,02	0,1	0	1,1	0	0,3	0	0,2	0,07	3,58
tai40b	637250948	0,01	0,2	0	1,6	0	0,3	0	0,1	0,01	2,74
tai50b	458821517	0,02	0,5	0	5,8	0	0,7	0	0,9	0,06	4,00
tai60b	608215054	0,04	1,1	0	9,5	0	18,6	0	7,6	0,09	4,00
tai80b	818415043	0,23	3	0	27,7	0	218	0	60	0,50	4,00
tai100b	1185996137	0,14	6,7	0	42,5	0	161	0	60	0,58	4,00
sko42	15812	0	0,2	0	1,6	0	0,4	0	0,1	0,17	4,00
sko49	23386	0,01	0,3	0	4	0	0,6	0	9,2	0,31	4,00
sko56	34458	0,02	0,7	0	16,2	0	0,8	0	9,7	0,32	4,00
sko64	48498	0,01	1,5	0	23,1	0	1,2	0	3,1	0,41	4,00
sko72	66256	0,06	3	0	33,6	0	1,8	0	60	0,52	4,00
sko81	90998	0,06	6	0	39,9	0	2,4	0	60	0,50	4,00
sko90	115534	0,07	12	0	40,5	0	3,2	0	60	0,61	4,00
sko100a	152002	0,09	30	0	41,7	0	29,8	0	60	0,47	4,00
sko100b	153890	0,06	30	0	42,3	0	8,5	0	60	0,56	4,00
sko100c	147862	0,06	30	0	42,2	0	4,3	0	60	0,46	4,00
sko100d	149576	0,09	30	0	41,9	0	12,9	0	60	0,55	4,00
sko100e	149150	0,07	30	0	42,5	0	4,3	0	60	0,61	4,00
sko100f	149036	0,08	30	0	42	0	17,1	0	60	0,54	4,00
Average		0,09	16	0,04	21,3	0	44,5	0	31	0,65	3,47

Περνώντας στην ανάλυση της σύγκρισης, βλέπουμε ότι αν και οι υπόλοιποι λύτες έχουν χαμηλότερες τιμές σφαλμάτων, το πρόγραμμα που αναπτύχθηκε στην παρούσα διπλωματική χρειάζεται πολύ λίγο χρόνο. Ισχύει ότι έχει χειρότερα ποιοτικά αποτελέσματα, της τάξης του συν 0.6 τοις εκατό, αλλά επίσης έχει μέσο χρόνο τα 3.5 λεπτά. Οι υπόλοιποι λύτες βλέπουμε ότι χρειάζονται από 16 μέχρι και 45 λεπτά μέσο χρόνο για να φτάσουν στα δικά τους αποτελέσματα. Το μειονέκτημα της δικής μου προσέγγισης είναι ότι με μεγαλύτερο χρονικό όριο δεν παρατηρείται αρκετά μεγάλη βελτίωση έτσι ώστε να διαθέσουμε επιπλέον χρόνο. Όπως είπαμε, το ιδανικό χρονικό όριο είναι τα 4 λεπτά. Οπότε αυτό σημαίνει ότι αν θέλουμε μια αρκετά ποιοτική λύση σε πολύ μικρό χρονικό διάστημα, η χρήση της μεθόδου που αναπτύχθηκε κάνει και με το παραπάνω τη δουλειά της. Τέλος, ας μη ξεχνάμε ότι οι περισσότερες άλλες προσεγγίσεις βασίζονται σε παραλληλοποιημένες τεχνικές, οι οποίες όπως είναι φυσικό, παράγουν πιο ποιοτικά αποτελέσματα. Στο επόμενο και τελευταίο

κεφάλαιο θα παρουσιαστούν τα συμπεράσματα της παρούσας διπλωματικής, καθώς και ιδέες για μελλοντική επέκταση ή βελτίωση του προγράμματος.

ΚΕΦΑΛΑΙΟ 6

Συμπεράσματα και Μελλοντικές Βελτιώσεις

Σε αυτό το τελευταίο κεφάλαιο θα παρουσιαστούν τα συμπεράσματα της παρούσας διπλωματικής και έπειτα κάποιες προτάσεις για μελλοντική βελτίωση. Αρχικά είδαμε το θεωρητικό υπόβαθρο του προβλήματος της τετραγωνικής αντιστοίχισης και το πόσο υπολογιστικά δύσκολο παραμένει ακόμα και μετά από τόσο καιρό. Αφού παρουσιάστηκαν κάποιες τεχνικές επίλυσής του, είδαμε και τη μροφή των στιγμιοτύπων του από τη Διαδικτυακή βιβλιοθήκη QAPLIB. Συνεχίζοντας, μελετήσαμε τη μέθοδο με την οποία έγινε η προσπάθεια επίλυσης του προβλήματος, αυτή της αναζήτησης μεταβλητής γειτνίασης. Αναφέρθηκαν πρώτα από όλα οι ξεχωριστές φάσεις που απαρτίζουν τη συγκεκριμένη μέθοδο και έπειτα αναλύθηκαν οι διαφορετικές εκδοχές της.

Εφόσον ήρθε το τέλος των θεωρητικών κεφαλαίων, περάσαμε στο σχεδιασμό και την υλοποίηση του προγράμματος. Ξεκινώντας, μπορούμε να πούμε ότι η προσπάθεια ανάπτυξης και υλοποίησης ενός αλγορίθμου για την επίλυση του προβλήματος τετραγωνικής αντιστοίχισης, στέφθηκε με επιτυχία. Αν και η αρχική σχεδίαση προέβλεπε τη χρήση μόνο της μεθόδου αναζήτησης μεταβλητής γειτνίασης, είδαμε πως εν τέλει η μέθοδος με τα καλύτερα αποτελέσματα ήταν μια υβριδική μέθοδος. Αυτή λοιπόν η υβριδική μέθοδος συνδυάζει την τοπική αναζήτηση με δύο γειτονιές και τη μέθοδο της πολυεναρκτήριας τοπικής αναζήτησης (Multi-start). Με αυτόν το συνδυασμό καταφέραμε να προσπεράσουμε το εμπόδιο των τοπικών βέλτιστων και κατά συνέπεια να βελτιώσουμε σημαντικά την ποιότητα των λύσεών μας.

Εκτός από τη βελτίωση στα αριθμητικά αποτελέσματα, η υβριδική αυτή μέθοδος είχε ως αποτέλεσμα την απλούστευση της γραφικής διεπιφάνειας του προγράμματος, καθώς ο χρήστης δε χρειάζεται να επιλέξει ανάμεσα σε διάφορες παραμέτρους. Έτσι η χρήση γίνεται ευκολότερα και αυτό προφανώς είναι ακόμα ένα πλεονέκτημα. Αφού παρουσιάστηκε η υλοποίηση του προγράμματος, είδαμε συνολικά τα αριθμητικά αποτελέσματα και καταλήξαμε στο πιο αποδοτικό ανώτατο χρονικό όριο για κάθε στιγμιότυπο, που είναι τα τέσσερα λεπτά. Παρουσιάστηκε επίσης και η σύγκριση του προγράμματος που αναπτύχθηκε με άλλους State of the art λύτες. Εκεί διαπιστώσαμε ότι το παρόν πρόγραμμα είχε λίγο χειρότερα ποιοτικά αποτελέσματα, αλλά χρειαζόταν ελάχιστο χρόνο για την επίτευξή τους. Επίσης είναι σημαντικό να σημειωθεί ότι οι υπόλοιποι λύτες ήταν καθαρά παράλληλοι

και όχι απλά βελτιστοποιημένοι όπως το παρόν πρόγραμμα.

Αυτό μας οδηγεί στην πιο εμφανής μελλοντική βελτίωση, που είναι η προσπάθεια της πλήρους παραλληλοποίησης του προγράμματος. Δυστυχώς όμως αυτό δεν μπορεί να γίνει με τη συγκεκριμένη γλώσσα και λογικά θα ήταν απαραίτητη η μεταγλώττιση του προγράμματος σε C++. Επίσης σε περίπτωση μεταγλώττισης, δε θα έχουμε τον αυτόματο βελτιστοποιητή του Visual Studio για τη C sharp, αλλά θα πρέπει να χρησιμοποιηθεί κάποιος άλλος αντίστοιχος. Πάντως για την ώρα, ο αυτόματος βελτιστοποιητής για τη C sharp μας έδωσε πολύ μεγάλη βελτίωση και παραλληλοποίησε αυτόματα όποια τμήματα του κώδικα ήταν δυνατόν.

Εν κατακλείδι, μπορούμε να πούμε με ασφάλεια ότι οι στόχοι της διπλωματικής επιτεύχθηκαν, αλλά πάντα υπάρχει περιθώριο βελτίωσης.

ΑΝΑΦΟΡΕΣ

Aksan, Y., T. Dokeroglu, and A. Cosar (2017). A stagnation-aware cooperative parallel breakout local search algorithm for the quadratic assignment problem. *Computers and Industrial Engineering 103*, 105–115.

Altner, D., R. Ahuja, O. Ergunm, and J. Orlin (2011). *Very Large-Scale Neighborhood Search*. American Cancer Society.

Andreatta, A. and C. Ribeiro (2002, July). Heuristics for the phylogeny problem. *Journal of Heuristics 8*(4), 429–447.

Battiti, R. and G. Tecchiolli (1994). The reactive tabu search. *ORSA Journal on Computing 6*(2), 126–140.

Benlic, U. and J. K. Hao (2015). Memetic search for the quadratic assignment problem. *Expert Systems with Applications 42*, 584–595.

Bentley, J. (1992). Fast algorithms for geometric traveling salesman problems. *ORSA Journal on Computing 4*(4), 387–411.

Burkard, R. E., S. Karisch, and F. Rendl (1997). Qaplib a quadratic assignment problem library. *Journal of Global Optimization 1 10*, 391–403.

Burkard, R. E., P. Pardalos, D. Du, and R. Graham (2013). *Quadratic Assignment Problems*. New York: Springer. ISBN 978-1-4419-7996-4.

Croes, G. (1958). A method for solving traveling-salesman problems. *Operations Research 6*(6), 791–812.

Fischetti, M., M. Monaci, and D. Salvagnin (2012, 08). Three ideas for the quadratic assignment problem. *Operations Research 60*, 954–964.

Heider, C. (1972). A computationally simplified pair-exchange algorithm for the quadratic assignment problem.

Koopmans, T. C. and M. J. Beckmann (1957). Assignment problems and the location of economic activities. *Econometrica 25*, 53–76.

Levente, K. and G. Andras (2009). Efficient multi-start strategies for local search algorithms. pp. 705–720.

Merz, P. and B. Freisleben (2000, 12). Fitness landscape analysis and memetic algorithms for the quadratic assignment problem. *Evolutionary Computation, IEEE Transactions on 4*, 337 – 352.

Misevicius, A. (2011). An implementation of the iterated tabu search algorithm for the quadratic assignment problem. *Springer-Verlag 34*, 665–690.

Mladenovic, N. and P. Hansen (1997). Variable neighborhood search. *Computers and Operations Research 24*(11), 1097 – 1100.

Mladenovic, N. and P. Hansen (2001, 02). Variable neighborhood search: Principles and applications. *European Journal of Operational Research 130*, 449–467.

Mladenovic, N. and P. Hansen (2006). First vs. best improvement: An empirical study. *Discrete Applied Mathematics 154*(5), 802 – 817.

Mladenovic, N., P. Hansen, and J. Perez (2010, March). Variable neighbourhood search: methods and applications. *Annals of Operations Research 175*(1), 367–407.

Mladenovic, N., P. Hansen, J. Perez, and J. Brimberg (2010, 09). In *Variable Neighborhood Search*, Volume 191, pp. 61–86.

Mladenovic, N., P. Hansen, R. Todosijevic, and S. Hanafi (2016, 08). Variable neighborhood search: basics and variants. *EURO Journal on Computational Optimization 5*.

Mustafa, A. and S. Topaloglu (2017). A multi-start iterated local search algorithm for the generalized quadratic multiple knapsack problem. *Computers and Operations Research 83*, 54 – 65.

Sahni, S. and T. Gonzalez (1976). P-complete approximation problems. *J. ACM 23*(3), 555–565.

Silva, A., L. C. Coelho, and M. Darvish (2019). A parallel iterated tabu search applied to several quadratic assignment problems.

Stutzle, T. (1999, 05). Iterated local search for the quadratic assignment problem.

Tosun, U. (2015). On the performance of parallel hybrid algorithms for the solution of the quadratic assignment problem. *Engineering Applications of Artificial Intelligence 39*, 267–278.

Wilhelm, R. and L. Ward (1987). Solving quadratic assignment problems by simulated annealing. *IIE Transactions 19*(1), 107–119.