Πανεπιστήμιο Πειραιώς – Τμήμα Πληροφορικής
Πρόγραμμα Μεταπτυχιακών Σπουδών
«Πληροφορική»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	**Αλγοριθμικές Τεχνικές για το Πρόβλημα του Ομαδικού Προσανατολισμού με Εφαρμογή στο Σχεδιασμό Τουριστικών Δρομολογίων.** **Algorithmic Techniques for the Team Orienteering Problem with Application to the Tourist Itineraries Design.**
Ονοματεπώνυμο Φοιτητή	**Γεωργούλης Αριστοτέλης**
Πατρώνυμο	**Αντώνιος**
Αριθμός Μητρώου	**ΜΠΠΛ15012**
Επιβλέπων	**Χαράλαμπος Κωνσταντόπουλος, Αναπληρωτής Καθηγητής**

Ημερομηνία Παράδοσης **Ιανουάριος 2020**

Τριμελής Εξεταστική Επιτροπή

(υπογραφή)	(υπογραφή)	(υπογραφή)
Χαράλαμπος Κωνσταντόπουλος Αναπληρωτής Καθηγητής	Άγγελος Πικράκης Επίκουρος Καθηγητής	Ιωάννης Τασούλας Επίκουρος Καθηγητής

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τους γονείς μου και τον αδερφό μου για την στήριξη τους όλο αυτό το διάστημα. Επίσης, θα ήθελα να ευχαριστήσω τον καθηγητή μου κ. Χαράλαμπο Κωνσταντόπουλο για την βοήθεια του και την υποστήριξη του στην εκπόνηση της παρούσας μεταπτυχιακής διατριβής.

Περιεχόμενα

Πίνακας Εικόνων

Περίληψη

Το πρόβλημα του Προσανατολισμού (Orienteering Problem - OP) και το πρόβλημα του Ομαδικού Προσανατολισμού (Team Orienteering Problem - TOP) αποτελούν μία άμεση επέκταση του προβλήματος του Πλανόδιου Πωλητή (Traveling Salesperson Problem). Στο πρόβλημα OP δίνεται ένα σύνολο κόμβων, όπου κάθε κόμβος έχει ένα όφελος και ζητείται η διαδρομή που θα συλλέξει το μεγαλύτερο όφελος, με το μήκος της διαδρομής να περιορίζεται από ένα χρονικό περιθώριο. Το πρόβλημα του TOP επεκτείνει το OP χρησιμοποιώντας τους ίδιους περιορισμούς και έχοντας παρόμοια μαθηματική μοντελοποίηση, προσπαθώντας όμως να βρει ένα πλήθος διαδρομών. Τα προβλήματα αυτά ανήκουν στην κατηγορία των προβλημάτων NP-Hard.

Στην παρούσα εργασία θα επικεντρωθούμε στα προβλήματα OP και TOP, καθώς και σε μία διαδεδομένη επέκταση τους, συγκεκριμένα, το πρόβλημα του Ομαδικού Προσανατολισμού με χρονικά παράθυρα (Team Orienteering Problem with Time Windows). Θα αναλύσουμε τα προβλήματα αυτά και θα παρουσιάσουμε την μαθηματική τους μοντελοποίηση. Επίσης, θα αναφερθούμε σε κάποιους αλγόριθμους που στόχευσαν στην επίλυση του προβλήματος όπως ο GRASP, ο ILS και ο GRASP-ELS και θα παρουσιάσουμε μια τροποποίηση του συγκεκριμένου αλγορίθμου για αυστηρότερη επιλογή τοποθεσιών και εισαγωγή εστιατορίου στις διαδρομές μας.

Ο συγκεκριμένος αλγόριθμος ουσιαστικά έχει άμεση εφαρμογή στα προβλήματα που αφορούν τις εφαρμογές σχεδιασμού τουριστικών διαδρομών.

Abstract

The Orienteering Problem (OP) and the Team Orienteering Problem (TOP) is an extension of the well-known Traveling Salesperson Problem. In OP a set of nodes is given, each associated with a profit and we are searching for a route with maximum collected profit and length limited by a time budget. TOP extends OP by using the same constraints and having similar mathematical formulation, while searching for multiple routes. Both of these problems belong to the category of NP-Hard problems.

In this thesis, we will concentrate on presenting the OP and TOP and a known extension of it, the Team Orienteering Problem with Time Windows. We will analyze these problems and present their mathematical formulation. Furthermore, we will refer to some algorithms that focus on solving these problems such as GRASP, ILS and GRASP-ELS and we will present a modification of the ILS algorithm with stricter constraints on selecting visits and an option to add a restaurant in each tour.

The proposed algorithm has an immediate application in the Tourist Trip Design problems.

Κεφάλαιο 1°

1.1 Εισαγωγή

Η Θεωρία Γραφημάτων (Graph Theory) αποτελεί σημαντικό τομέα της Επιστήμης της Πληροφορικής και συντέλεσε τα μέγιστα στην επίλυση σημαντικών αλγοριθμικών προβλημάτων. Ένα τέτοιο πρόβλημα που μας απασχολεί δεκαετίες τώρα, είναι το πρόβλημα του Περιοδεύοντος ή Πλανόδιου Πωλητή (Travel Salesman Problem ή TSP) [1] το οποίο αποτελεί επέκταση του κύκλου Hamilton.

Το πρόβλημα του TSP παρουσιάζει τεράστιο ενδιαφέρον. Έχοντας ένα πράκτορα και ένα μεγάλο αριθμό σημείων (vertices) στο χάρτη, σκοπός μας είναι ο πράκτορας να επισκεφθεί όλα τα σημεία με το μικρότερο δυνατόν μήκος διαδρομής. Παρά την υψηλή υπολογιστική πολυπλοκότητα (Computational Complexity) που παρουσιάζει, καθώς ανήκει στην κατηγορία των NP-Πληρών (NP-Complete) προβλημάτων, έχει παρουσιασθεί μεγάλος αριθμός Ευρετικών Αλγορίθμων (Heuristic Algorithms) και Μετά-Ευρετικών Αλγορίθμων (Meta-Heuristic Algorithm) που μπορούν να λύσουν το πρόβλημα σε εύλογο, πολυωνυμικό χρόνο παρουσιάζοντας μια απόκλιση από την βέλτιστη λύση αρκετά μικρή.

Μια παραλλαγή του προβλήματος TSP είναι το πρόβλημα του Προσανατολισμού (Orienteering Problem - OP). Η πρώτη αναφορά στο OP γίνεται από τον Tsiligirides, T. [2]. Το όνομα του αποδόθηκε από ένα άθλημα που είναι συνδυασμός τρεξίματος και προσανατολισμού σε δάσος. Ο τρόπος που παρουσιάστηκε το OP από τον Tsiligirides ήταν περιγράφοντας ένα παιχνίδι στην ύπαιθρο (outdoor). Οι παίκτες – πράκτορες έπρεπε να ξεκινήσουν από ένα σημείο – αφετηρία και να περάσουν από όσα περισσότερα σημεία μπορούσαν φτάνοντας στο τερματισμό – τελικό σημείο. Κάθε σημείο είχε ένα όφελος – πόντους (profits) που έπαιρνε ο παίκτης περνώντας από εκεί, ενώ οι παίκτες – πράκτορες είχαν ένα καθορισμένο χρονικό περιθώριο (budget). Σκοπός, λοιπόν, των παικτών ήταν να μεγιστοποιήσουν τους πόντους που θα μάζευαν, περνώντας από όσο περισσότερα σημεία μπορούσαν στο δοθέν χρονικό όριο. Πέραν της ονομασίας OP, το συγκεκριμένο πρόβλημα μπορεί να το συναντήσει κάποιος και με άλλες ονομασίες όπως το Πρόβλημα του Επιλεκτικού Περιοδεύοντος Πωλητή (Selective Traveling Salesman Problem) [3], το Πρόβλημα της Μέγιστης Συλλογής (Maximum Collection Problem) [4] και το Πρόβλημα του Ληστή της Τράπεζας (The Bank Robber Problem) [5].

Οι παραλλαγές του OP είναι αρκετές και το πρόβλημα αυτό έχει οδηγήσει σε αρκετές μελέτες και επεκτάσεις. Οι εργασίες που έγιναν σχετικά με το πρόβλημα του TSP από τους Feillet et al. [6] και το Χαμιλτονιανό και μη-Χαμιλτονιανό πρόβλημα που παρουσιάστηκε από τους Laporte and Rodriguez Martin [7], τοποθετούν το OP ανάμεσα σε άλλα προβλήματα δρομολόγησης (με και χωρίς οφέλη - profits). Και οι δύο αυτές εργασίες αναλύουν το OP, ενώ παρουσιάζουν διάφορες στρατηγικές και επεκτάσεις του προβλήματος.

Οι διαφορές στο OP έχουν να κάνουν συνήθως με το γράφημα. Κάποιες από τις πιο γνωστές παραλλαγές είναι το γράφημα να είναι κατευθυνόμενο (directed graph) ή μη-κατευθυνόμενο (undirected graph). Μπορεί, επίσης, το γράφημα να διαθέτει χρονικά παράθυρα στους κόμβους, οπότε η επίσκεψη των κόμβων θα πρέπει να γίνει στα χρονικά περιθώρια που καθορίζει ο κάθε κόμβος. Μια ακόμα παραλλαγή του γραφήματος είναι να υπάρχουν διαφορές στον αρχικό και τελικό κόμβο, δηλαδή ο αρχικός κόμβος να μην είναι ίδιος με τον τελικό κόμβο ή άλλη περίπτωση είναι ο αρχικός και ο τελικός κόμβος να συμπίπτουν. Τέλος, δυο ακόμα γνωστές παραλλαγές είναι να υπάρχει μείωση οφέλους για τον πράκτορα μετά την επίσκεψη σε ένα κόμβο, εφόσον ο πράκτορας επισκεφθεί ξανά τον κόμβο αυτό, και να υπάρχουν πολλαπλές διαδρομές. Στην πρόσφατη έρευνα του Vansteenwegen P. et al. [8] και του Gunawan A. et al. [9] παρουσιάζονται εκτενώς κάποιες από τις παραπάνω αναφερθείσες παραλλαγές και επεκτάσεις του OP.

Στην παρούσα εργασία θα παρουσιάσουμε το πρόβλημα του OP και κάποιες από τις σημαντικότερες παραλλαγές του όπως το Ομαδικό Πρόβλημα Προσανατολισμού (Team Orienteering Problem ή TOP) [10] και το Ομαδικό Πρόβλημα Προσανατολισμού με Χρονικά Περιθώρια στους κόμβους (Team Orienteering Problem with Time Windows ή TOPTW) [11]. Θα

αποτυπώσουμε τους μαθηματικούς τύπους του κάθε προβλήματος, τους περιορισμούς τους και θα υλοποιήσουμε ορισμένους από τους ευρετικούς και μετα-ευρετικούς αλγορίθμους που έχουν βρεθεί και έχουν καταφέρει να οδηγήσουν στην προσεγγιστική επίλυση του σε πολυωνυμικό χρόνο. Θα επιδείξουμε τα αποτελέσματα των αλγορίθμων και θα τα συγκρίνουμε, ενώ στο τέλος θα υλοποιήσουμε μια παραλλαγή ενός ήδη υπάρχοντος αλγορίθμου.

Οι παραπάνω επεκτάσεις που θα αναλυθούν περαιτέρω παρακάτω, όπως αναφέραμε, θα συζητηθούν μέσα στα πλαίσιο του προβλήματος Σχεδιασμού Τουριστικών Διαδρομών (Tourist Trip Design Problem) [12]. Έχοντας ένα αριθμό από σημεία ενδιαφέροντος (Points of Interest ή POIs) ενός τουρίστα, θα προσπαθήσουμε να βρούμε διαδρομές οι οποίες θα μεγιστοποιήσουν το όφελος του και οι διαδρομές αυτές θα πραγματοποιούνται εντός ενός συγκεκριμένου χρονικού διαστήματος για κάθε ημέρα διαμονής. Τις διαδρομές αυτές θα προσπαθήσουμε να τις επεκτείνουμε με την εισαγωγή ενός εστιατορίου – γεύματος, φροντίζοντας να διατηρήσουμε το όφελος του τουρίστα. Αυτή ουσιαστικά θα είναι και η παραλλαγή μας στον αλγόριθμο όπως θα παρουσιαστεί παρακάτω.

1.2 Δομή Εργασίας

Αρχικά, θα γίνει μια περίληψη της εργασίας στα Ελληνικά και στα Αγγλικά, στην οποία θα αναφερθούμε στα κυριότερα σημεία της μεταπτυχιακής διατριβής.

Στο 1ο Κεφάλαιο θα γίνει μια σύντομη εισαγωγή στο θέμα του OP και στις επεκτάσεις του. Θα παρουσιάσουμε κάποια ιστορικά στοιχεία και πως από το πρόβλημα του Περιοδεύοντος Πωλητή (TSP) οδηγηθήκαμε στο OP. Θα παρουσιαστεί ακόμα η ανασκόπηση της εργασίας.

Στο 2ο Κεφάλαιο θα ξεκινήσουμε να αναλύουμε καλύτερα το πρόβλημα του OP. Θα αναφερθούμε και θα αποτυπώσουμε τις μαθηματικές συναρτήσεις – πράξεις του προβλήματος. Τέλος, θα αναλύσουμε κάποιες άμεσες επεκτάσεις του OP σε μια προσπάθεια να βρεθούν αλγοριθμικές προσεγγίσεις που θα παράγουν βέλτιστες διαδρομές.

Στο 3ο Κεφάλαιο θα παρουσιαστεί η επέκταση του OP σε TOP, έχοντας σαν σκοπό την παραγωγή πολλαπλών διαδρομών. Θα αναφερθούμε στις προσεγγίσεις πάνω στο συγκεκριμένο θέμα και θα γίνει ανάλυση του αλγόριθμου GRASP [30]. Τέλος, θα γίνει αναφορά στην επέκταση του GRASP σε GRASP με Path Relinking [31, 32].

Στο 4ο Κεφάλαιο θα παρουσιάσουμε το TOPTW, που αποτελεί μια ακόμη επέκταση του προβλήματος TOP. Θα αναφερθούμε στον ορισμό του προβλήματος και θα γίνει αναφορά στην μαθηματική μοντελοποίηση του TOPTW. Θα συζητηθεί ο αλγόριθμος ILS [36] και τα βήματα που ακολουθεί για να βρεθεί μία προσεγγιστική λύση σε πολυωνυμικό χρόνο. Θα παρουσιαστεί ο CSCRatio και CSCRoutes [37] που κατατάσσουν τις πιθανές τοποθεσίες σε συστάδες. Ακόμα, θα παρουσιαστεί ένας αλγόριθμος παραπλήσιος του ILS, ο υβριδικός GRASP-ELS [38] και θα αναφερθούμε στον τρόπο με τον οποίο ο συγκεκριμένος αλγόριθμος προσπαθεί να λύσει το πρόβλημα.

Στο 5ο Κεφάλαιο θα παρουσιάσουμε τον αλγόριθμο μας που αποτελεί μια διαφοροποίηση του ILS. Η παραλλαγή μας θα εισάγει στις διαδρομές την παράμετρο του γεύματος σε εστιατόριο μια συγκεκριμένη ώρα που θα ορίζει ο χρήστης. Θα παρουσιάσουμε τα αποτελέσματα του ILS και του τροποποιημένου ILS, ενώ θα αναφερθούμε και στα συμπεράσματα τους.

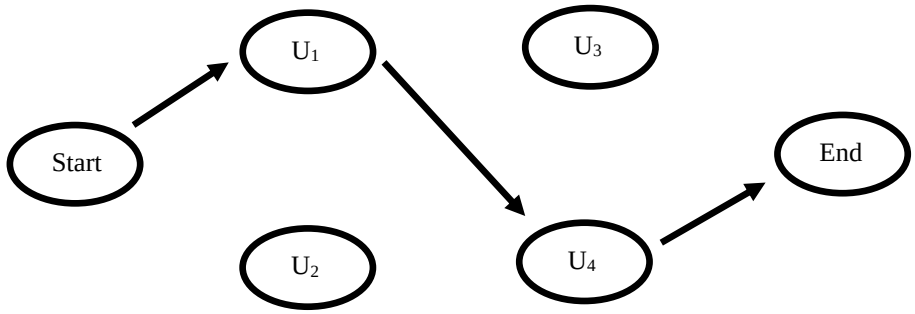
Τέλος, στο 6ο Κεφάλαιο θα εξάγουμε τα συμπεράσματα μας και θα παρουσιάσουμε την βιβλιογραφία της εργασίας.

Κεφάλαιο 2ο

2.1 Το πρόβλημα του Προσανατολισμού (OP)

Στο πρόβλημα του προσανατολισμού (Orienteering Problem ή OP), όπως παρουσιάσθηκε από τον Tsiligirides T. [2], έχουμε ένα σύνολο από *N* κορυφές (vertices) και κάθε κορυφή $i \in N$, διαθέτει ένα όφελος (score) S_i. Το σημείο της εκκίνησης (κορυφή 1) και το σημείο τερματισμού (κορυφή *N*) είναι καθορισμένα. Ο χρόνος που χρειαζόμαστε για να πάμε από την κορυφή *i* στην κορυφή *j* συμβολίζεται με t_{ij} και μας είναι ήδη γνωστά. Δεν μπορούμε να επισκεφθούμε όλες τις κορυφές καθώς έχουμε ένα περιορισμένο χρονικό περιθώριο το οποίο είναι άνω φραγμένο και μας είναι γνωστό. Το περιορισμένο χρονικό περιθώριο θα το συμβολίζουμε με T_{max}. Στόχος του OP είναι να βρει ένα μονοπάτι, που δεν θα ξεπεράσει το μέγιστο χρονικό περιθώριο T_{max}, το οποίο επισκέπτεται μερικές από τις κορυφές, με σκοπό να μεγιστοποιήσει το συνολικό συλλεγόμενο όφελος (score). Η κάθε κορυφή δεν μπορεί να προσπελαστεί παραπάνω από μια φορές.

Ένα στιγμιότυπο του OP, μπορεί να περιγραφεί με την βοήθεια ενός γράφου (κατευθυνόμενου ή μη-κατευθυνόμενου) *G=(V, E)* όπου το $V = \{v_1, v_2, v_3, \ldots, v_N\}$ είναι ένα σύνολο κορυφών (vertex set) και *E* είναι ένα σύνολο τόξων (arc set), τέτοιο ώστε ένα μη-αρνητικό όφελος S_i να συσχετίζεται με κάθε κορυφή $v_i \in V$, όπως φαίνεται και στην εικόνα 1. Επίσης, υπάρχει ένα χρόνος μεταφοράς t_{ij} από μία κορυφή σε μία άλλη συσχετισμένος με κάθε τόξο $a_{ij} \in A$. Υπάρχει, ακόμα, όπως αναφέραμε παραπάνω ένα χρονικό περιθώριο (T_{max}), που δεν πρέπει να το υπερβούμε προκειμένου να μεγιστοποιήσουμε το συνολικό όφελος από την διαδρομή. Το OP αποτελείται από ένα καθορισμένο Χαμιλτονιανό μονοπάτι G' ($\subset$ G) ενός υποσυνόλου *V* και μπορεί καθοριστεί σαν ένα μονοπάτι με διακριτές κορυφές, όμως, σε πολλές παραλλαγές του OP το αρχικό σημείο έναρξης συμπίπτει με το τελικό σημείο $v_1 = V_n$.



Εικόνα 1 – Μια διαδρομή (κατευθυνόμενος γράφος)

Χρησιμοποιώντας τις παρατηρήσεις που αναφέραμε παραπάνω, μπορούμε να τις χρησιμοποιήσουμε και να μοντελοποιήσουμε το OP σαν ένα ακέραιο γραμμικό πρόβλημα (integer linear problem).

Αρχικά, θα χρειαστούμε 3 σταθερές μεταβλητές για να μπορέσουμε να τυποποιήσουμε το πρόβλημα του προσανατολισμού.

1) $x_{ij} = 1$, εάν η κορυφή *i* ακολουθείται από μια επίσκεψη στην κορυφή *j* ενώ σε διαφορετική περίπτωση η συγκεκριμένη μεταβλητή θα είναι 0

2) $u_i =$ η θέση της κορυφής *i* στο μονοπάτι

3) $S_i > 0$ αφού το όφελος της κάθε κορυφής δεν μπορεί να είναι αρνητικό

4) $Max \sum_{i=2}^{N-1} \sum_{j=2}^{N} S_i X_{ij},$

5) $\sum_{j=2}^{N} x_{1j} = \sum_{i=1}^{N-1} x_{iN} = 1$

6) $\sum_{i=1}^{N-1} x_{ik} = \sum_{j=2}^{N} x_{kj} \leq 1; \quad \forall k = 2,3,\dots,N-1; \; i,j \neq k$

7) $\sum_{i=1}^{N-1} \sum_{j=2}^{N} t_{ij} x_{ij} \leq T_{max} \quad i \neq j$

8) $2 \leq u_i \leq N; \quad \forall i = 2,\dots,N$

9) $u_i - u_j + 1 \leq (N-1)\left(1 - x_{ij}\right); \quad \forall i,j = 2,\dots,N$

10) $x_{ij} \in \{0,1\}; \; \forall i,j = 1,\dots,N$

Οι παραπάνω μεταβλητές και συναρτήσεις χρησιμεύουν στους περιορισμούς που υπόκειται το OP. Η μεταβλητή (1) χρησιμοποιείται για να δείξει το γεγονός ότι σε ένα μονοπάτι (path), η κορυφή *i* ακολουθείται από την κορυφή *j*. Τότε, η μεταβλητή παίρνει την τιμή 1, ενώ σε διαφορετική περίπτωση η μεταβλητή θα πάρει την τιμή 0. Η μεταβλητή (2) αφορά την θέση της εκάστοτε κορυφής στο μονοπάτι αυτό και η μεταβλητή (3) χρησιμοποιείται λόγω του οφέλους της κορυφής, που πρέπει να είναι θετικός αριθμός. Στην συνέχεια ακολουθούν οι συναρτήσεις, με την αντικειμενική συνάρτηση (4) να εφαρμόζεται για να μεγιστοποιήσουμε το συνολικό όφελος που συλλέγουμε σε ένα μονοπάτι. Ο περιορισμός (5) μας εγγυάται ότι το μονοπάτι θα ξεκινήσει από την κορυφή 1 και θα τελειώσει στην κορυφή *N* και η συνάρτηση (6) ότι όλες οι κορυφές του μονοπατιού είναι συνδεδεμένες μεταξύ τους, καθώς επίσης, και ότι κάθε κορυφή την επισκεπτόμαστε το πολύ μία φορά. Ο περιορισμός (7) διασφαλίζει το γεγονός ότι το μονοπάτι θα ακολουθήσει το περιορισμένο χρονικό περιθώριο που έχει, $T_{max,}$ ενώ, τέλος, οι δύο τελευταίοι περιορισμοί (8) και (9) αποκλείουν την πιθανότητα δημιουργίας υπο-κύκλων, και ο περιορισμός (10) συνδέεται με τον περιορισμό (1) ώστε το x_{ij} να ανήκει στο σύνολο {0, 1}. Οι συγκεκριμένοι περιορισμοί μοντελοποιήθηκαν σύμφωνα με το μοντέλο Miller–Tucker–Zemlin (MTZ) που αρχικά εφαρμόστηκαν στο πρόβλημα του TSP [13].

Στο πρόβλημα του OP, υπάρχει μια σημαντική υπόθεση που δεν αναφέραμε παραπάνω. Για τον υπολογισμό της απόστασης των κορυφών χρησιμοποιείται σιωπηλά μια μετρική, αυτή της Ευκλείδειας απόστασης, που υποθέτει ότι το $t_{ij} = t_{ji}$. Αυτό εφαρμόζεται σε ένα μη-κατευθυνόμενο πλήρες γράφημα *G* (undirected complete graph), που όμως εύκολα μπορεί να

τροποποιηθεί και να βρει εφαρμογή σε έναν κατευθυνόμενο μη-πλήρες γράφημα *G* (directed un-completed graph).

Η απόδειξη της πολυπλοκότητας του OP δόθηκε το 1987 από τον Golden et al. [14] και αποδείχθηκε ότι το πρόβλημα του OP ανήκει στην κατηγορία των NP-Hard προβλημάτων. Επίσης, απέδειξαν ότι κανένας αλγόριθμος επί του παρόντος δεν μπορεί να λύσει το πρόβλημα αυτό σε πολυωνυμικό χρόνο, ούτε μπορεί να σχεδιαστεί αλγόριθμος που θα βρει σε εύλογο χρόνο βέλτιστη λύση. Οι αλγόριθμοι του OP απαιτούν αρκετή υπολογιστική ισχύ και για αυτό είναι απαραίτητη μια προσέγγιση ευριστική. Επιπροσθέτως, ο Gendreau et al. [15] εξήγησε τους λόγους που είναι δύσκολο για το πρόβλημα του OP να σχεδιαστεί καλός ευρετικός αλγόριθμος. Το όφελος της κορυφής και ο χρόνος προσέγγισης της κορυφής είναι ανεξάρτητα μεταξύ τους και συχνά αντιφατικά το ένα με το άλλο. Αυτό δημιουργεί τεράστια δυσκολία στο να βρεθεί αλγόριθμος που θα διαλέξει κορυφές που θα δημιουργήσουν βέλτιστη λύση. Ο Vansteenwegen [11] το 2008 έδειξε ότι οι πιο δύσκολες περιπτώσεις του προβλήματος προσανατολισμού για να λυθούν, είναι εκείνες που ο επιλεγμένος αριθμός των κορυφών είναι λίγο περισσότερος από τις μισές συνολικές κορυφές. Τέλος, καθορίζοντας ένα μονοπάτι μεταξύ των επιλεγμένων κορυφών κάνει το πρόβλημα πιο περίπλοκο όταν ο αριθμός των κορυφών αυξάνεται.

2.2 Επεκτάσεις του Προβλήματος Προσανατολισμού (OP)

Το πρόβλημα OP, που παρουσιάσθηκε από τον Tsiligirides T. [2], αποτέλεσε αφορμή για να υπάρξουν αρκετές παραλλαγές ή επεκτάσεις του.

Όπως, αναφέραμε και στην εισαγωγή δεν είναι λίγες οι διαφορετικές εκδοχές του OP που έχουν μελετηθεί στην βιβλιογραφία. Οι διαφοροποιήσεις του OP έχουν να κάνουν κυρίως με το εάν το γράφημα είναι κατευθυνόμενο (directed graph) [16] ή μη-κατευθυνόμενο (undirected graph) [17].

Ένα από τα προβλήματα που έχουν μελετηθεί είναι να δίνεται μόνο ένας αρχικός κόμβος, από τον οποίο θα ξεκινάει ο πράκτορας, χωρίς όμως να υπάρχει τελικός κόμβος, δηλαδή η διαδρομή μπορεί να τελειώσει σε οποιονδήποτε κόμβο του γραφήματος (rooted OP) [18] ή μπορεί να διαφοροποιηθεί με την απουσία αρχικού κόμβου, όπου δίνεται η ελευθερία στον πράκτορα να ξεκινήσει από όπου θέλει και να τελειώσει σε ένα οποιονδήποτε κόμβο επιθυμεί αυτός (unrooted OP) [15].

Μια παραλλαγή του αλγορίθμου TSP, που βασίζεται στα προβλήματα απόφασης και ανήκει στην κατηγορία των προβλημάτων NP-Complete, παρουσιάζεται από τους Garey and Johnson [19] και τον Papadimitriou [20]. Στο πρόβλημα του πλανόδιου πωλητή με την διαδικασία της απόφασης (dTSP), δίνεται ένα γράφημα G με κόστη στις ακμές και ένας στόχος C και προσπαθεί να δημιουργηθεί κύκλος ο οποίος θα περνάει από όλες τις κορυφές του γράφου με μήκος το πολύ C.

Παρά το γεγονός ότι οι ακριβείς αλγόριθμοι για την επίλυση του OP απαιτούν αρκετό χρόνο, έχουν προταθεί κάποιες τεχνικές branch-and-bound [21] και branch-and-cut [22] οι οποίες όμως αφορούν γραφήματα με περιορισμένο αριθμό κόμβων.

Για το OP, έχουν προταθεί όμως και αρκετοί ευρετικοί και μετα-ευρετικοί αλγόριθμοι. Ο Tsiligirides [2] ήταν αυτός που αρχικά πρότεινε έναν στοχαστικό (S-Algorithm) και έναν ντετερμινιστικό (D-Algorithm) αλγόριθμο. Ο στοχαστικός χρησιμοποιεί τεχνικές και μεθόδους Monte-Carlo, όπου κατασκευάζει ένα πλήθος διαδρομών και διαλέγει την διαδρομή με το καλύτερο όφελος. Η πιθανότητα της επιλογής βασίζεται στην Ευκλείδεια απόσταση. Στον ντετερμινιστικό αλγόριθμο, η περιοχή χωρίζεται σε κύκλους που περιορίζουν τις διαδρομές.

Άλλοι αλγόριθμοι που παρουσιάστηκαν ήταν του Golden et al. [23] που πρότεινε έναν ευρετικό αλγόριθμο στον οποίο λαμβάνεται υπόψη το κέντρο βάρους της διαδρομής. Οι κόμβοι με τον μεγαλύτερο λόγο του κέρδους προς την απόσταση τους από το κέντρο βάρους της διαδρομής εισάγονται στην θέση με το ελάχιστο κόστος εισαγωγής, ενώ οι Ramesh and Brown [24] εισήγαγαν ένα ευρετικό αλγόριθμο με 4-στάδια. Στον συγκεκριμένο αλγόριθμο τα δύο

πρώτα στάδια εισαγωγής χαλαρώνουν το χρονικό περιθώριο και βελτιώνουν την διαδρομή με την χρησιμοποίηση των 2-Opt και 3-Opt. Στο επόμενο στάδιο, διαγράφεται ένας κόμβος και εισάγεται ένα άλλος κόμβος με σκοπό την μείωση του μήκους της διαδρομής, ενώ, το τελευταίο στάδιο περιλαμβάνει την εισαγωγή περισσότερων διαθέσιμων κόμβων.

Ο Chao at al. (1996b) πρότεινε ένα αλγόριθμο με 5-στάδια με καλύτερα αποτελέσματα από τον προηγούμενο ενώ οι Wang et al. [25] πρότειναν έναν αλγόριθμο εφαρμόζοντας ένα συνεχές Hopfield νευρωνικό δίκτυο.

Οι πιο πρόσφατες λύσεις για το πρόβλημα OP, παρουσιάστηκαν από τον Schilde et al. (2009) [27] όπου παρουσίασαν μια προσεγγιστική πολυ-κριτηριακή παραλλαγή του OP. Οι τελευταίες παραλλαγές που παρουσιάστηκαν ωστόσο δεν έχουν προσφέρει σημαντικές βελτιώσεις για την επίλυση του OP.

Κεφάλαιο 3°

3.1 Το πρόβλημα του Ομαδικού Προσανατολισμού (TOP)

Άμεση επέκταση του προβλήματος OP είναι το πρόβλημα του Ομαδικού Προσανατολισμού (TOP) που παρουσιάστηκε από τον Chao et al. [28]. Το TOP παρουσιάστηκε αρχικά από τους Butt and Cavalier [29] ως το πρόβλημα μέγιστης συλλογής πολλαπλών διαδρομών (Multiple tour maximum collection problem MTMCP).

Στο TOP στόχος είναι να καθοριστούν πολλαπλά, P, μονοπάτια μέσα στο ίδιο γράφημα, όπου το κάθε μονοπάτι (p) περιορίζεται από ένα χρονικό περιθώριο, $T_{max,}$ με σκοπό να μεγιστοποιήσουμε το συνολικό όφελος που θα συλλέξουμε από τις επισκέψεις μας στους κόμβους των μονοπατιών. Κάθε ένα από αυτά τα μονοπάτια θα ξεκινάει και θα τελειώνει σε ένα προκαθορισμένο κόμβο ($N_{[1]}$ kai $N_{[n]}$).

Η μαθηματική μοντελοποίηση του TOP μοιάζει αρκετά με αυτή του OP.

11) x_{ijp} =1, εάν η επίσκεψη από τον κόμβο *i* στον κόμβο *j* είναι εφικτή στο μονοπάτι p, σε διαφορετική περίπτωση θα είναι 0

12) y_{ip} =1, εάν ο κόμβος *i* βρίσκεται στο μονοπάτι *p*

13) u_{ip} = η θέση του κόμβου *i* στο μονοπάτι *p*

14) $Max \ \sum_{p=1}^{P} \sum_{i=2}^{N-1} \ S_i y_{ip}$,

15) $\sum_{p=1}^{P} \sum_{j=2}^{N} \ x_{1jp} = \sum_{p=1}^{P} \sum_{i=1}^{N-1} \ x_{iNp} = P,$

16) $\sum_{p=1}^{P} y_{kp} \leq 1; \quad \forall k = 2, \ldots, N-1,$

17) $\sum_{i=1}^{N-1} x_{ikp} = \sum_{j=2}^{N} \ x_{kjp} = \ y_{kp}; \quad \forall k = 2, \ldots, N-1; \forall p = 1, \ldots, P$,

18) $\sum_{i=1}^{N-1} \sum_{j=2}^{N} t_{ij} \, x_{ijp} \leq T_{max} \quad \forall p = 1, \ldots, P$,

19) $2 \leq u_{ip} \leq N; \quad \forall i = 2,\ldots,N; \; \forall p = 1,\ldots,P$,

20) $u_{ip} - u_{jp} + 1 \leq (N-1)\left(1 - x_{ijp}\right); \quad \forall i,j = 2,\ldots,N; \; \forall p = 1,\ldots,P$,

21) $x_{ijp}, y_{ip} \in \{0,1\}; \quad \forall i,j = 1,\ldots,N; \; \forall p = 1,\ldots,P$

Το TOP στην ουσία μπορεί να μοντελοποιηθεί και σαν ένα πρόβλημα ακεραίων με μεταβλητές τις παραπάνω αναφερθείσες. Η μεταβλητή (11) παίρνει τιμή 1 εφόσον στο μονοπάτι *p*, η κορυφή *i* ακολουθείται από μια επίσκεψη στην κορυφή *j*, ενώ σε αντίθετη περίπτωση θα είναι 0. Οι δυο επόμενες μεταβλητές, (12) και (13) αφορούν το εάν η κορυφή *i* υπάρχει στο μονοπάτι *p* και την θέση της κορυφής στο συγκεκριμένο μονοπάτι.

Οι επόμενες συναρτήσεις χρησιμοποιούνται για να επιβληθούν κάποιοι περιορισμοί στις κορυφές που εισάγονται. Η συνάρτηση (14) έχει σαν στόχο να μεγιστοποιήσει το συλλεγόμενο όφελος από τις κορυφές, ενώ ο περιορισμός (15) εγγυάται ότι κάθε μονοπάτι θα ξεκινάει από την κορυφή 1 θα τελειώνει στην κορυφή *N*. Ο περιορισμός (16) εγγυάται ότι κάθε κορυφή θα έχει το πολύ μια επίσκεψη και ο περιορισμός (17) εγγυάται την συνδεσιμότητα του κάθε μονοπατιού. Ο περιορισμός (18) εξασφαλίζει το περιορισμένο χρονικό περιθώριο για κάθε μονοπάτι ενώ οι περιορισμοί (19) και (20) είναι απαραίτητοι για να διασφαλίσουν την μοναδικότητα των μονοπατιών και να αποφευχθούν τα υπο-μονοπάτια.

Ο πρώτος ευρετικός αλγόριθμος που παρουσιάστηκε ήταν από τον Chao at al. [28] και έμοιαζε αρκετά με τον ευρετικό αλγόριθμο των 5 βημάτων που παρουσίασε για το πρόβλημα του OP. Παρά το γεγονός ότι και εδώ παρουσιάστηκαν κάποιοι ακριβείς αλγόριθμοι για την επίλυση του προβλήματος TOP, δεν βρέθηκε κάποιος που να καταφέρνει να λύνει το πρόβλημα του TOP σε εύλογο χρονικό διάστημα, ενώ σημαντική προσπάθεια έγινε από τους Butt and Ryan (1999) [42] που χρησιμοποίησαν μια τεχνική παραγωγής στηλών (columns generation) και στόχευσαν στην επίλυση προβλημάτων 100 κόμβων.

Οι κύριοι ευρετικοί αλγόριθμοι που παρουσιάστηκαν από τους Archetti et al. (2007), Ke et al. (2008), Vansteenwegen et al. (2009 b,c) και Souffriau et al. (2009) είχαν όλοι κάτι κοινό, καθώς και οι τέσσερις ξεκινούσαν από μια αρχική λύση και προσπαθούσαν να παράγουν μια δεύτερη που θα αντικαταστούσε την πρώτη παραχθείσα, εφόσον όμως ήταν πιο επικερδής από αυτή. Επιπλέον, έδιναν ιδιαίτερη έμφαση στο να μειώσουν τον χρόνο που απαιτείται για να πάει ο ταξιδιώτης από τον ένα κόμβο στον επόμενο. Οι αλγόριθμοι αυτοί, επίσης χρησιμοποιούσαν και μια μέθοδο τοπικής αναζήτησης που περιελάβανε 5 βήματα για την μεγιστοποίηση του συνολικού οφέλους και δυο βήματα για την μείωση του χρόνου ταξιδιού από τον ένα κόμβο στον άλλο.

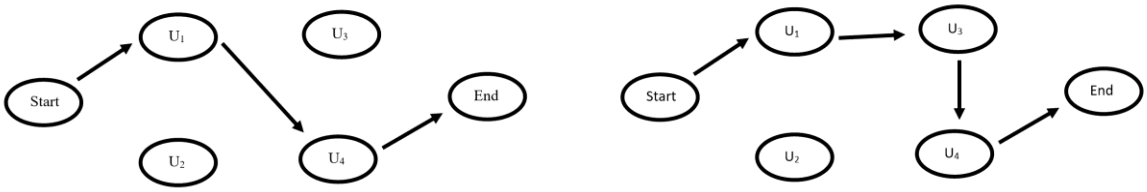
Τα βήματα μεγιστοποίησης είναι τα Insert, Two-Insert, Replace, Two-Replace, Change ενώ τα βήματα μείωσης του χρόνου ταξιδιού στους κόμβους είναι τα 2-Opt και Swap.

Στην συνέχεια, θα παρουσιάσουμε αναλυτικά τον αλγόριθμο GRASP και GRASP με Path Relinking.

3.2 Greedy Randomized Adaptive Search Procedure (GRASP)

Ένας αλγόριθμος που υλοποιήθηκε για την επίλυση του προβλήματος TOP ήταν ο GRASP. Ο Greedy Randomized Adaptive Search Procedure (GRASP) είναι ένας μετά-ευρετικός αλγόριθμος που παρουσιάστηκε αρχικά από τον Feo and Resende [30]. Γενικά ο GRASP

εκτελεί μια σειρά από επαναλήψεις, που κάθε μία ακολουθεί μια προπαρασκευαστική διαδικασία ακολουθούμενη από την τοπική αναζήτηση (εικόνα 2) για να βελτιώσει το αποτέλεσμα.



Εικόνα 2 – Διαδικασία τοπικής αναζήτησης

Η συμπεριφορά της προπαρασκευαστικής διαδικασίας εξαρτάται από μια και μόνο παράμετρο – μεταβλητή (greediness), η οποία καθορίζει την αναλογία μεταξύ της τυχαιότητας και της απληστίας. Οι επαναλήψεις είναι ανεξάρτητες μεταξύ τους και η καλύτερη λύση που θα βρεθεί θα επιστραφεί σαν αποτέλεσμα. Γενικότερα, ο GRASP θεωρείται ένας αλγόριθμος που έχει αρκετές εφαρμογές σε προβλήματα όπως για παράδειγμα το πρόβλημα δρομολόγησης οχημάτων (vehicle routing problem).

Ο ψευδοκώδικας του αλγορίθμου GRASP παρουσιάζεται παρακάτω και αφορά μια επανάληψη για το πρόβλημα του TOP. Κάθε επανάληψη ξεκινάει με μια άδεια λύση από *m* διαδρομές, που αρχικά το μόνο που περιλαμβάνουν είναι ο αρχικός και ο τελικός κόμβος. Προκειμένου να κατασκευαστεί μια αρχική λύση, κατασκευάζουμε αρχικά μια λίστα *C* με όλες τους υποψήφιους κόμβους που μπορούν να εισαχθούν. Μια κίνηση εισαγωγής ενός κόμβου IM_{ilj} θεωρείται η εισαγωγή ενός κόμβου *l* εκτός διαδρομής, στην υπάρχουσα διαδρομή μεταξύ των *i* και *j*. Το IM_{ilj} χαρακτηρίζεται από ένα όφελος S_l και μια αύξηση της διάρκειας της διαδρομής, που ισούται με $t_{ilj} = t_{il} + t_{lj} - t_{ij}$.

Για να θεωρηθεί μια κίνηση εφικτή, είναι απαραίτητο ο χρόνος που θα αυξήσει την διαδρομή αυτή η κίνηση να είναι μικρότερος ή ίσος από τον υπολειπόμενο διαθέσιμο χρόνο. Όλες οι πιθανές κινήσεις εισάγονται στην λίστα *C* και για κάθε μια από αυτές υπολογίζεται μια ευρετική τιμή $h_{ilj} = \frac{S_l}{t_{ilj}}$. Ανάλογα την μέγιστη και την μικρότερη τιμή των πιθανών κινήσεων της λίστας *C*, και με βάση την απληστία που ορίσαμε σαν παράμετρο, θα υπολογιστεί ένα όριο. Στην συνέχεια οι πιθανές λίστες θα φιλτραριστούν και θα κρατηθούν μόνο οι κινήσεις αυτές οι οποίες ξεπερνάνε το όριο αυτό (Restricted Candidate List - RCL). Μια τυχαία κίνηση από την λίστα αυτή διαλέγεται και προστίθεται στην αρχική λύση. Η διαδικασία αυτή επαναλαμβάνεται μέχρις ότου δεν υπάρχουν διαθέσιμες λύσεις.

Ψευδοκώδικας GRASP

διαδικασία GRASP(Επαναλήψεις, Seed)

Διάβασμα_διαθέσιμων_σημείων();

Για k = 1, . . . , Επαναλήψεις κάνε:

Solution ← Greedy_Randomized_Construction(Seed);

Solution ← Τοπική_Αναζήτηση(Solution);

Update_Solution(Solution, Best Solution);

Τέλος Για;

Επέστρεψε Καλύτερη_Λύση;

τέλος GRASP;

διαδικασία Greedy_Randomized_Construction(greediness)

Solution = 0;

C = Σύνολο κόμβων εφικτοί για εισαγωγή;

Όσο το C ≠ 0 κάνε:

Min = Ελάχιστη ευρετική τιμή του C;

Max = Μέγιστη ευρετική τιμή του C;

Threshold = Min + greediness x (Max - Min);

RCL = 0;

Για κάθε c που ανήκει στο C κάνε:

Αν ευρετική τιμή του c ≥ Threshold τότε:

Πρόσθεσε το c στην RCL

Τέλος Αν;

Τέλος Για;

Υποψήφιος κόμβος = Τυχαία επιλογή από την RCL;

Solution = Solution U {Υποψήφιος κόμβος};

C = Κόμβοι εφικτοί για εισαγωγή\{Υποψήφιος κόμβος};;

Τέλος Όσο;

Επέστρεψε Solution;

Τέλος Greedy_Randomized_Construction;

Στο τέλος κάθε επανάληψης η παραγόμενη λύση θα βελτιωθεί εφαρμόζοντας την διαδικασία της τοπικής αναζήτησης, η οποία αποτελείται από μια σειρά επαναλήψεων. Αρχικά, για να μειωθεί ο συνολικός χρόνος εφαρμόζεται η κίνηση 2-Opt και η κίνηση Swap, όπου εισάγεται η ανταλλαγή κόμβων μεταξύ των διαδρομών. Στην συνέχεια, η Replace προσπαθεί να αλλάξει τους κόμβους της παρούσας λύσης με κόμβους που βρίσκονται εκτός, και τέλος εφαρμόζεται πάλι η insert για την εισαγωγή των κόμβων.

Παρακάτω παρουσιάζεται ο αλγόριθμος της τοπικής αναζήτησης.

διαδικασία Τοπική_Αναζήτηση(Solution, Best Solution)

Όσο δεν έχει βρεθεί τοπικό μέγιστο κάνε:

2-opt()

Swap()

Replace()

Insert()

Τέλος Όσο;

Τέλος Τοπική_Αναζήτηση;

3.3 Greedy Randomized Adaptive Search Procedure με Path Relinking

Ένα σημαντικό μειονέκτημα που παρατηρήθηκε στον αλγόριθμο GRASP, τον οποίο παρουσιάσαμε παραπάνω, είχε να κάνει με την ανεξαρτησία των επαναλήψεων μεταξύ τους. Εκτός του να διατηρούμε την καλύτερη λύση που βρέθηκε, κάθε μια από τις επαναλήψεις ξεκινάει ξανά από μια εντελώς άδεια λύση, ανεξάρτητα αν στις προηγούμενες επαναλήψεις είχε βρεθεί κόμβος, ο οποίος θα οδηγούσε σε βελτίωση του χρόνου της διαδρομής.

Προκειμένου να βρεθεί μια λύση στο συγκεκριμένο πρόβλημα που αναφέρθηκε, μια επέκταση του βασικού αλγορίθμου GRASP παρουσιάστηκε από τους Souffriau et al. [45] για την αντιμετώπιση του προβλήματος TOP. Η επέκταση αυτή συνδυάζει τον αλγόριθμο GRASP με τον αλγόριθμο του Path Relinking. Η τεχνική path relinking παρουσιάστηκε από τον Glover [31] και εφαρμόστηκε σε συνδυασμό με τον GRASP από τους Laguna and Marti [32].

Παρακάτω παρουσιάζεται και ο ψευδοκώδικας του GRASP με Path Relinking. Στην ουσία η διαδικασία αρχικά δεν διαφέρει αρκετά από αυτή του αλγόριθμου GRASP. Μέχρι να εκτελεστεί ένας καθορισμένος αριθμός επαναλήψεων και εφόσον δεν έχει προκύψει καμία βελτίωση, ακολουθείται η ίδια διαδικασία κατασκευής διαδρομών και καλείται η τοπική αναζήτηση. Στην συνέχεια εισέρχεται η παραλλαγή της τεχνικής path relinking. Θα δοθεί σαν όρισμα η λύση που βρέθηκε από τον GRASP και μια λύση ‘οδηγός’, η οποία ανήκει σε μια ‘δεξαμενή’ λύσεων που ορίζεται ως ‘pool elite solutions’. Η τεχνική αυτή θα ψάξει το μονοπάτι μεταξύ των δύο αυτών λύσεων προκειμένου να βρεθεί μια καλύτερη λύση. Η λύση που θα βρεθεί, θα θεωρηθεί υποψήφια προς εισαγωγή στην ‘δεξαμενή’ των καλών λύσεων χρησιμοποιώντας ένα μέτρο ομοιότητας και εφόσον είναι καλύτερη θα εισαχθεί στη ‘δεξαμενή’ αυτή.

Ψευδοκώδικας GRASP με Path Relinking

διαδικασία GRASP-PR()

Όσο ο αριθμός των επαναλήψεων χωρίς βελτίωση είναι κάτω ενός ορίου κάνε:

Construct;

Local search;

Link to elites;

Update elite pool;

Τέλος όσο;

Επέστρεψε την καλύτερη λύση;

Τέλος GRASP-PR;

Στην συνέχεια θα περιγραφεί η διαδικασία της σύνδεσης των λύσεων με μεγαλύτερη ακρίβεια και θα ακολουθήσει ο ψευδοκώδικας του αλγορίθμου. Αυτή η διαδικασία παίρνει σαν όρισμα δυο λύσεις, την πρώτη που είναι η αρχική λύση (start solution) και η δεύτερη που είναι η λύση οδηγός (guiding solution), και ψάχνει το μονοπάτι μεταξύ των δύο αυτών λύσεων προκειμένου να βρεθεί μια καλύτερη λύση. Από την λύση οδηγό (guiding solution), όλες οι κοινές τοποθεσίες με την αρχική λύση αφαιρούνται, για να καταλήξουμε σε μια λίστα με τοποθεσίες που μπορούν να προστεθούν (set of features to add). Στην συνέχεια, οι τοποθεσίες από την λίστα που δημιουργήθηκε προηγουμένως (set of features to add), προστίθενται η μία μετά την άλλη στην αρχική λύση, προσπαθώντας να εισαχθούν σε αυτή.

Σε αντίθεση με τις κινήσεις εισαγωγής που αναφέραμε παραπάνω, μια τοποθεσία θεωρείται εφικτή προς εισαγωγή ακόμα και αν παραβιάζει το χρονικό περιθώριο (time budget) και ανήκει στο set of features to add. Όταν όλες οι *m* διαδρομές υπερβούν το χρονικό περιθώριο και δεν είναι πλέον εφικτές, δεν προστίθενται άλλες τοποθεσίες και ακολουθείται μια διαδικασία η οποία επαναφέρει τις τοποθεσίες στην αρχική τους κατάσταση, πριν την εισαγωγή, ώστε να γίνουν πάλι εφικτές προς εισαγωγή για κάθε διαδρομή.

Για κάθε κόμβο *l* που ανήκει σε μια μη επιτρεπτή διαδρομή και τοποθετείται μεταξύ των κόμβων *i* και *j*, υπολογίζεται το $\frac{1}{h_{ilj}} = \frac{t_{ilj}}{s_l}$. Ο κόμβος *l* με τη ψηλότερη τιμή αφαιρείται από την διαδρομή. Η αφαίρεση αυτή επαναλαμβάνεται μέχρι κάθε διαδρομή να είναι εφικτή ξανά.

Ψευδοκώδικας διαδικασίας elite

διαδικασία Elite()

Λίστα Intersection = κοινές τοποθεσίες και των δύο λύσεων;

Λίστα featuresToAdd = λύση οδηγός μείον την λίστα Intersection;

Όσο η λίστα featuresToAdd δεν είναι άδεια κάνε:

Insert τοποθεσίες, οι οποίες είναι μη επιτρεπόμενες;

Delete τοποθεσίες για επαναφορά των εφικτών τοποθεσιών;

Τέλος Όσο;

Τέλος elite;

Από την παραπάνω διαδικασία θα πάρουμε την καλύτερη λύση και θα την θεωρήσουμε υποψήφια για εισαγωγή στην δεξαμενή (pool) των elite λύσεων. Ταυτόχρονα, υπολογίζεται και ένα μέτρο ομοιότητας μεταξύ της καλύτερης λύσης και των άλλων μελών της δεξαμενής των λύσεων. Η ακόλουθη φόρμουλα υπολογίζει την ομοιότητα των λύσεων *X* και *Y*, όπου το n_x και το n_y είναι ο αριθμός των τοποθεσιών των λύσεων *X* και *Y* αντίστοιχα, ενώ $n_{x \cap y}$ είναι ο αριθμός των κοινών τοποθεσιών.

$$\frac{2n_{x \cap y}}{n_x + n_y}$$

Εάν η δεξαμενή των elite λύσεων δεν έφτασε το μέγιστο αριθμό τοποθεσιών και το μέτρο ομοιότητας μεταξύ των υποψήφιων λύσεων είναι κάτω από το όριο, τότε η λύση προστίθεται στην δεξαμενή elite. Εάν η υποψήφια λύση παρουσιάζει ομοιότητα, εισέρχεται στην δεξαμενή elite μόνο στην περίπτωση που η υποψήφια λύση είναι καλύτερη από την καλύτερη λύση της

δεξαμενής elite. Εάν η δεξαμενή elite είναι γεμάτη και η υποψήφια λύση είναι καλύτερη από την χειρότερη λύση της πισίνας elite, τότε η χειρότερη λύση αντικαθίσταται από την υποψήφια λύση.

Κεφάλαιο 4°

4.1 Το πρόβλημα του Προσανατολισμού με χρονικά Παράθυρα (OPTW)

Το πρόβλημα του προσανατολισμού με χρονικά παράθυρα (OPTW) και το πρόβλημα του ομαδικού προσανατολισμού με χρονικά παράθυρα (TOPTW), το οποίο θα παρουσιάσουμε στην επόμενη ενότητα, έλαβαν αρκετή προσοχή στη βιβλιογραφία (Boussier et al., 2007 [33], Righini and Salani, 2009 [34], Montemanni and Gambardella, 2009 [35], Vansteenwegen et al., 2009 [36)].

Ο κύριος λόγος για τον οποίο συνέβη αυτό, οφείλεται στο γεγονός ότι στα γραφήματα που οι κόμβοι έχουν χρονικά παράθυρα εξετάζονται και επιλύονται με διαφορετικό τρόπο από αυτόν που παρουσιάσαμε στις προηγούμενες ενότητες όπου οι κόμβοι δεν διαθέτουν χρονικά παράθυρα. Για παράδειγμα, ο 2-Opt καταφέρνει να παράγει επιθυμητά αποτέλεσμα για το πρόβλημα OP αλλά δεν μπορεί να εφαρμοστεί αποτελεσματικά στο πρόβλημα του OPTW και αυτό οφείλεται στο γεγονός ότι η αλλαγή της σειράς των κόμβων δεν είναι πλέον εφικτή λόγω των χρονικών παραθύρων.

Πιο αναλυτικά, στο πρόβλημα OPTW σε κάθε κόμβο αναθέτεται ένα χρονικό παράθυρο *[Oi, Ci]* και κάθε επίσκεψη στον συγκεκριμένο κόμβο μπορεί να γίνει μόνο εφόσον η υπηρεσία στο συγκεκριμένο κόμβο έχει ξεκινήσει και δεν έχει ολοκληρωθεί.

Συγκεκριμένα, το πρόβλημα OPTW μπορεί να μοντελοποιηθεί σαν ένα ακέραιο γραμμικό πρόβλημα με τις ακόλουθες μεταβλητές και συναρτήσεις:

22) X_{ij} =1

23) y_i = 1

24) s_i = έναρξη επίσκεψης του κόμβου i

25) M = μεγάλη σταθερά

26) $Max \sum_{i=2}^{N-1} \sum_{j=2}^{N} S_i X_{ij},$

27) $\sum_{j=2}^{N} x_{1j} = \sum_{i=1}^{N-1} x_{iN} = 1,$

28) $\sum_{i=1}^{N-1} x_{ik} = \sum_{j=2}^{N} x_{kj} \leq 1; \quad \forall k = 2, \ldots, N-1,$

29) $s_i + t_{ij} - s_j \leq M(1 - x_{ij}); \ \forall i, j = 1, \ldots, N,$

30) $\sum_{i=1}^{N-1} \sum_{j=2}^{N} t_{ij} x_{ij} \leq T_{max}$,

31) $O_i \leq s_i; \ \forall i = 1, \ldots, N,$

32) $s_i \leq C_i; \ \forall i = 1, \ldots, N,$

33) $x_{ij} \in \{0,1\};\ \forall i,j = 1,\ldots,N$

Οι τέσσερις πρώτες μεταβλητές είναι μεταβλητές απόφασης. Η μεταβλητή (22) παίρνει τιμή 1 όταν η επίσκεψη στον κόμβο *i* ακολουθείται από μια επίσκεψη στον κόμβο *j,* σε διαφορετική περίπτωση είναι 0. Η μεταβλητή (23) αφορά τον κόμβο *i* και παίρνει τιμή 1 εφόσον ο πράκτορας έχει περάσει από τον συγκεκριμένο κόμβο, διαφορετικά παίρνει 0, ενώ οι μεταβλητές (24) και (25) αφορούν την έναρξη της επίσκεψης στον συγκεκριμένο κόμβο και το *M* είναι μια επαρκώς μεγάλη σταθερά.

Όσον αφορά τις συναρτήσεις, η συνάρτηση (26) προβλέπει στο να μεγιστοποιηθεί το όφελος που αποκτάμε από τις επισκέψεις στους κόμβους. Ο περιορισμός (27) εγγυάται ότι το μονοπάτι θα ξεκινήσει από τον κόμβο 1 και θα τελειώσει στον *N* και ο περιορισμός (28) εγγυάται την συνδεσιμότητα, όπως και ότι κάθε κόμβος θα έχει το πολύ μια επίσκεψη. Ακόμα, ο περιορισμός (29) διαβεβαιώνει την τήρηση του μονοπατιού και η συνάρτηση (30) εξασφαλίζει την τήρηση του χρονικού περιθωρίου. Οι συναρτήσεις (31), (32) περιορίζουν την έναρξη της επίσκεψης εντός του χρονικού παραθύρου και, τέλος, η συνάρτηση (33) αφορά αν η επίσκεψη από τον κόμβο i στο κόμβο j θα πραγματοποιηθεί άρα θα πάρει και τιμή 1 ή όχι, οπότε θα πάρει τιμή 0.

Δεν αναφέρονται πολλές εφαρμογές του προβλήματος OPTW, αλλά οι περισσότερες εφαρμογές του προβλήματος OPTW απαιτούν χρονικά παράθυρα στους κόμβους και έχουν εφαρμογή σε πολλά προβλήματα του σύγχρονου κόσμου. Για παράδειγμα, το πρόβλημα του τουρίστα, που θέλει να επισκεφθεί ένα μουσείο ή ένα αξιοθέατο και θα πρέπει η επίσκεψη στα συγκεκριμένα μέρη να πραγματοποιηθεί εντός του ωραρίου που έχουν αντίστοιχα καθορίσει.

4.2 Το πρόβλημα του Ομαδικού Προσανατολισμού με χρονικά Περιθώρια (TOPTW)

Μία άμεση επέκταση του προβλήματος OP και μια από τις πιο διαδεδομένες επεκτάσεις του είναι το πρόβλημα του Ομαδικού Προσανατολισμού με χρονικά περιθώρια (TOPTW).

Μπορεί να μοντελοποιηθεί σαν ένα ακέραιο γραμμικό πρόβλημα με κάποιες μεταβλητές και συναρτήσεις απόφασης. Παρακάτω παρουσιάζουμε πιο αναλυτικά τις συγκεκριμένες μεταβλητές και συναρτήσεις.

34) $x_{ijp} = 1$

35) $y_{ip} = 1$

36) s_{ip} = έναρξη επίσκεψης στον κόμβο *i* στο μονοπάτι *p*

37) M = μια μεγάλη σταθερά

38) $Max\ \sum_{p=1}^{P} \sum_{i=2}^{N-1}\ S_i y_{ip}$,

39) $\sum_{p=1}^{P} \sum_{j=2}^{N}\ x_{1jp} = \sum_{p=1}^{P} \sum_{i=1}^{N-1}\ x_{iNp} = P,$

40) $\sum_{i=1}^{N-1} x_{ikp} = \sum_{j=2}^{N}\ x_{kjp} =\ y_{kp};\ \ \forall k = 2,\ldots,N-1; \forall p = 1,\ldots,P$,

41) $s_{ip} + t_{ij} - s_{jp} \leq M\left(1 - x_{ijp}\right);\ \ \forall i = 1,\ldots,N-1; \forall p = 1,\ldots,P$

$$42)\ \sum_{p=1}^{P} y_{kp} \leq 1\ \forall k = 2, \ldots, N-1,$$

$$43)\ \sum_{i=1}^{N-1} \sum_{j=2}^{N} t_{ij}\, x_{ijp} \leq T_{max}\ ,\ \forall p = 1, \ldots, P\ ,$$

$$44)\ O_i \leq s_{ip};\ \ \forall i = 1, \ldots, N-1; \forall p = 1, \ldots, P$$

$$45)\ s_{ip} \leq C_i;\ \ \forall i = 1, \ldots, N-1; \forall p = 1, \ldots, P$$

$$46)\ x_{ijp}\ , y_{ip}\ \in \{0,1\};\ \ \forall i, j = 1, \ldots, N; \forall p = 1, \ldots, P$$

Οι τέσσερις πρώτες μεταβλητές είναι μεταβλητές απόφασης. Η μεταβλητή (34) παίρνει τιμή 1, αν στο μονοπάτι *p* ο κόμβος *i* ακολουθείται από μια επίσκεψη στον κόμβο *j*, σε διαφορετική περίπτωση είναι 0. Η μεταβλητή (35) αφορά τον κόμβο *i* και παίρνει τιμή 1 εφόσον ο πράκτορας έχει περάσει από τον συγκεκριμένο κόμβο στο μονοπάτι *p*, διαφορετικά παίρνει 0. Η μεταβλητή (36) είναι η έναρξη της επίσκεψης στην κορυφή *i* στο μονοπάτι *p* και η μεταβλητή (37) είναι μια μεγάλη σταθερά που χρησιμοποιείται.

Για τις συναρτήσεις του προβλήματος TOPTW, ακολουθούμε λίγο πολύ την ίδια λογική με το OPTW. Η συνάρτηση (38) μεγιστοποιεί το συλλεγόμενο όφελος. Ο περιορισμός (39) εγγυάται ότι όλα τα μονοπάτια θα ξεκινήσουν από τον κόμβο 1 και θα τελειώσουν στον κόμβο *N*. Οι περιορισμοί (40) και (41) καθορίζουν την συνδεσιμότητα του κάθε μονοπατιού, ενώ ο περιορισμός (42) εγγυάται ότι κάθε κορυφή – κόμβος θα έχει το πολύ μια επίσκεψη. Ο περιορισμός (43) περιορίζει το διαθέσιμο χρονικό περιθώριο ενώ τέλος οι συναρτήσεις (44), (45) και (46) περιορίζουν την έναρξη της επίσκεψης στο συγκεκριμένο χρονικό παράθυρο που έχει ο κόμβος.

4.3 Ο αλγόριθμος Iterated Local Search (ILS)

Ένας από τους πιο διαδεδομένους αλγορίθμους που παρουσιάστηκαν για την επίλυση του προβλήματος του TOPTW είναι ο επαναλαμβανόμενος αλγόριθμος τοπικής αναζήτησης ILS.

Ο ILS παρουσιάστηκε από τον Vansteenwegen et al. [36] το 2009 και αποτελεί ένα σημαντικό σημείο αναφοράς στο πρόβλημα επίλυσης του TOPTW, καθώς καταφέρνει να βρει μια αρκετά καλή λύση, σε πολυωνυμικό χρόνο.

Η συγκεκριμένη εργασία βασίστηκε στην ιδέα μιας ηλεκτρονικής εφαρμογής, η οποία θα βοηθήσει τους τουρίστες στον καλύτερο σχεδιασμό του ταξιδιού τους. Στην ουσία, το πρόβλημα του σχεδιασμού μπορεί να λυθεί σε πραγματικό χρόνο, εάν μοντελοποιηθεί σαν ένα πρόβλημα TOPTW. Στο πρόβλημα TOPTW, δίνεται ένα σύνολο από τοποθεσίες, που κάθε μια έχει ένα όφελος και ένα χρονικό παράθυρο εντός του οποίου μπορεί να επισκεφθεί ο τουρίστας την τοποθεσία. Σκοπός είναι να μεγιστοποιήσει το όφελος με την επίσκεψη του σε όσο το δυνατόν πιο πολλές τοποθεσίες με υψηλή βαθμολογία.

Το TOPTW αποτελεί μια μοντελοποίηση του προβλήματος σχεδιασμού τουριστικών διαδρομών (Tourist Trip Design Problem - TTDP). Στο πρόβλημα αυτό ο τουρίστας – χρήστης προσπαθεί να σχεδιάσει την διαδρομή της κάθε ημέρας που έχει διαθέσιμη, με σκοπό να επισκεφθεί τις περισσότερες τοποθεσίες. Μια διαδρομή αντιστοιχεί σε μια μέρα ενώ οι πιθανές τοποθεσίες (POI) έχουν ένα όφελος. Κάθε διαδρομή ξεκινάει και τελειώνει στο ίδιο σημείο.

Ο ILS συνδυάζει δυο τεχνικές για να βρεθεί η βέλτιστη λύση, ένα βήμα εισαγωγής (insertion step) και ένα βήμα διαταραχής (shake step). Το βήμα εισαγωγής προσπαθεί να εισάγει

τοποθεσίες ενώ το βήμα διαταραχής προσπαθεί να απεγκλωβίσει τον αλγόριθμο από τοπικά μέγιστα. Γενικότερα, ο αλγόριθμος ξεκινά με μια αρχική λύση και προσπαθεί να εισάγει τοποθεσίες ώστε να βρεθεί μια τοπικά βέλτιστη λύση. Αυτή η βέλτιστη λύση θα διαταραχθεί, αφαιρώντας κάποιες τοποθεσίες και ακολουθώντας ξανά την διαδικασία της εισαγωγής μέχρι να εκτελεστεί ένας συγκεκριμένος αριθμός επαναλήψεων. Αναλυτικότερη, παρουσίαση του αλγορίθμου ILS ακολουθεί στις επόμενες ενότητες.

4.3.1 Βήμα Εισαγωγής (Insertion Step)

Το βήμα εισαγωγής προσπαθεί να προσθέσει μια καινούργια επίσκεψη μέσα στη διαδρομή. Πριν μία υποψήφια τοποθεσία θεωρηθεί ότι μπορεί να εισαχθεί στη διαδρομή, πρέπει πρώτα να πιστοποιηθεί ότι όλες οι επισκέψεις μετά την εισαγόμενη τοποθεσία θα εξακολουθήσουν να ικανοποιούν τα χρονικά τους παράθυρα.

Προκειμένου να αναπτυχθεί ένα γρήγορος ευρετικός αλγόριθμος, μια σύντομη αξιολόγηση των πιθανών κινήσεων είναι απαραίτητη. Για να ελεγχθούν όλες οι άλλες επισκέψεις απαιτείται αρκετός χρόνος. Για να αποφευχθεί αυτό καταγράφουμε τον χρόνο αναμονής (Wait) και το *MaxShift* για κάθε ήδη τοποθετημένη τοποθεσία στη διαδρομή. Σαν χρόνο αναμονής ορίζουμε τον χρόνο που πρέπει να περιμένει ο τουρίστας που έφθασε εφόσον ο χρόνος άφιξης (a_i) έγινε πριν την έναρξη του χρονικού παραθύρου. Η επίσκεψη στο κόμβο μπορεί να ξεκινήσει μόνο εντός του χρονικού παραθύρου. Εάν η άφιξη στο σημείο γίνει μετά την έναρξη της υπηρεσίας, ο χρόνος αναμονής είναι 0.

$$Wait_i = max[0, O_i - a_i]$$

Σαν *MaxShift* καθορίζεται ο μέγιστος χρόνος που η ολοκλήρωση της επίσκεψης μπορεί να καθυστερήσει, ώστε να μην κάνει τις επόμενες τοποθεσίες μετά από αυτήν ανέφικτες και τις βγάλει εκτός του χρονικού τους παραθύρου. Το *MaxShift* της τοποθεσίας *i* ισούται με την μικρότερη τιμή είτε του αθροίσματος του χρόνου αναμονής και του *MaxShift* της επόμενης τοποθεσίας *i + 1* ή την διάρκεια της επίσκεψης στον κόμβο αυτό.

$$MaxShift_i = min[C_i - s_i, Wait_{i+1} + MaxShift_{i+1}]$$

Με την χρήση της μεταβλητής $MaxShift_i$ μπορούμε να εκτιμήσουμε την εισαγωγή μιας υποψήφιας τοποθεσίας σε σταθερό χρόνο αντί για γραμμικό.

Η συνολική χρονική επιβάρυνση (*Shift*) που προκαλεί η εισαγωγή της υποψήφιας τοποθεσίας στην διαδρομή ανάμεσα στις τοποθεσίες *i* και *k*, υπολογίζεται από την εξής συνάρτηση:

$$Shift_j = c_{ij} + Wait_j + T_j + c_{jk} - c_{ik} \leq Wait_k + MaxShift_k$$

Φυσικά η έναρξη της επίσκεψης της υποψήφιας τοποθεσίας θα πρέπει να ταιριάζει στο χρονικό παράθυρο της τοποθεσίας αυτής.

Για τον υπολογισμός του χρόνου άφιξης (Arrival) αθροίζουμε την ώρα που έφυγε από την προηγούμενη τοποθεσία και τον χρόνο που χρειαζόμαστε για να πάμε από την προηγούμενη τοποθεσία στην παρούσα. Για τον υπολογισμό του χρόνου χρησιμοποιούμε την Ευκλείδεια απόσταση.

Ο χρόνος που ξεκινάει η επίσκεψη μας (Start) παίρνει την μεγαλύτερη τιμή μεταξύ της ώρας άφιξης και της ώρας που έχει ορίσει η τοποθεσία σαν ώρα έναρξης (O_i).

Για κάθε επίσκεψη καθορίζουμε το χαμηλότερο πιθανό Shift, δηλαδή την καλύτερη δυνατή εισαγωγή. Μετά, προκειμένου να αποφασίσουμε αν η επίσκεψη αυτή θα εισαχθεί στην διαδρομή, υπολογίζουμε το ratio για κάθε μία πιθανή επίσκεψη.

$$Ratio_i = \frac{(S_i)^2}{Shift_i}$$

Και επιλέγουμε την επίσκεψη με το υψηλότερο Ratio. Λόγω των χρονικών παραθύρων, αποφασίζουμε να κάνουμε το *Shift* λιγότερο σημαντικό στον υπολογισμό από ότι το όφελος της κάθε τοποθεσίας και για αυτό υψώνουμε το όφελος στο τετράγωνο.

Παρακάτω παρουσιάζουμε τον ψευδο-κώδικα του βήματος εισαγωγής.

διαδικασία Insert()

Για κάθε κόμβο που δεν ανήκει στη λύση

Καθόρισε την καλύτερη πιθανή θέση εισαγωγής με το μικρότερο Shift;

Υπολόγισε το ratio;

Τέλος Για;

Εισήγαγε τον κόμβο με το μεγαλύτερο ratio (j);

Επίσκεψη j: υπολόγισε τα Arrive, Start και Wait;

Για κάθε επίσκεψη μετά τον j (μέχρι το Shift == 0):

Ενημέρωσε τα Arrive, Start, Wait, MaxShift και Shift:

Τέλος Για;

Επίσκεψη j: ενημέρωσε το MaxShift;

Για κάθε κόμβο πριν τον j:

Ενημέρωσε το MaxShift;

Τέλος Για;

Τέλος διαδικασίας;

Μετά την εισαγωγή όλες οι άλλες επισκέψεις πρέπει να ενημερωθούν. Οι επισκέψεις μετά την τοποθεσία που εισήχθη, θα πρέπει να ενημερώσουν τον χρόνο αναμονής (*Wait*), τον χρόνο άφιξης (*a*), την έναρξη της υπηρεσίας (*s*) και το *MaxShift*. Κάθε φορά που η υποψήφια επίσκεψη έχει ένα χρόνο αναμονής, το shift των επόμενων επισκέψεων θα πρέπει να μειωθεί από τον χρόνο αυτό.

Η επόμενη τοποθεσία μετά το σημείο εισαγωγής ενημερώνεται από τις παρακάτω συναρτήσεις:

$$Wait_k = max[0, Wait_k - Shift_j];$$

$$a_k = a_k + Shift_j;$$

$$Shift_k = max[0, Shift_j - Wait_k];$$

$$s_k = s_k + Shift_k$$

$$MaxShift_k = MaxShift_k - Shift_k$$

Γενικότερα, οι επόμενες συναρτήσεις χρησιμοποιούνται για την ενημέρωση όλων των επισκέψεων μετά την εισαγόμενη τοποθεσία (*j=k+1,k+2,…*):

$$Wait_j = max[0, Wait_j - Shift_{j-1}];$$

$$a_j = a_j + Shift_{j-1};$$

$$Shift_j = max[0, Shift_{j-1} - Wait_j];$$

$$s_j = s_j + Shift_j;$$

$$MaxShift_j = MaxShift_j - Shift_j;$$

Οι επισκέψεις πριν την τοποθεσία που προστέθηκε, θα ενημερώσουν την τιμή της μεταβλητής *MaxShift* με βάση την ακόλουθη συνάρτηση (*j=1,2,…,i-1*):

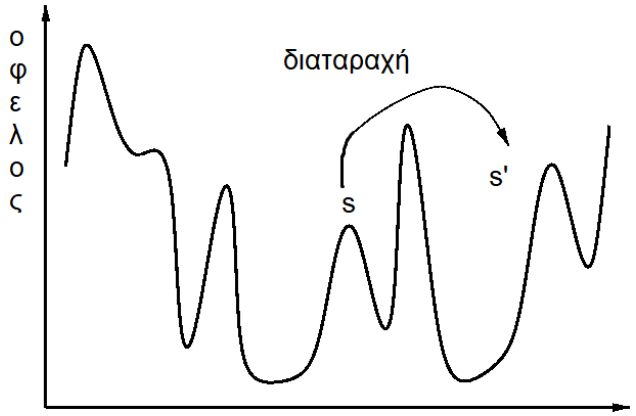
$$MaxShift_j = min[C_j - s_j, Wait_{j+1} + MaxShift_{j+1}];$$

4.3.2 Βήμα Διαταραχής (Shake Step)

Το βήμα διαταραχής χρησιμοποιείται για να δραπετεύσει ο αλγόριθμος από το τοπικά βέλτιστο. Κατά την διάρκεια αυτού του βήματος, μία ή περισσότερες τοποθεσίες θα διαγραφούν από κάθε διαδρομή. Κάθε βήμα διαταραχής χρησιμοποιεί δυο σταθερές, η πρώτη σταθερά υποδεικνύει το πόσες συνεχόμενες τοποθεσίες/επισκέψεις θα διαγραφούν από μια διαδρομή (R_d) ενώ η δεύτερη σταθερά υποδεικνύει την τοποθεσία της διαδρομής από όπου θα ξεκινήσει η διαδικασία της διαγραφής (S_d). Εάν κατά την διάρκεια της διαγραφής φτάσουμε στο τελικό σημείο, τότε συνεχίζουμε μετά την αρχική τοποθεσία. Λόγω των διαφορετικών μεγεθών των διαδρομών, η τιμή του S_d θα γίνει διαφορετική για κάθε διαδρομή, κατά την διάρκεια εκτέλεσης του αλγορίθμου. Διαφορετικά S_d σε διαφορετικές διαδρομές αυξάνουν την πιθανότητα η διαδρομή να δραπετεύσει από το τοπικό βέλτιστο, όπως φαίνεται στην εικόνα 3.

Μετά την διαγραφή, όλες οι τοποθεσίες που ακολουθούν τις διαγραφόμενες τοποθεσίες θα μετατοπιστούν στην αρχή της διαδρομής, προκειμένου να αποφευχθεί η αναμονή. Εάν μια τοποθεσία δεν μπορεί να μετατοπιστεί λόγω του χρονικού παραθύρου τότε αυτή και όλες οι άλλες διαδρομές παραμένουν αμετάβλητες. Οι μετατοπισμένες τοποθεσίες θα πρέπει να

ενημερωθούν με βάση τις συναρτήσεις που παρουσιάσαμε στο βήμα εισαγωγής. Για τοποθεσίες που υπήρχαν πριν τις διαγραμμένες τοποθεσίες, ενημερώνεται μόνο το *MaxShift*.



Εικόνα 3 – Βήμα διαταραχής για να φύγουμε από το τοπικό μέγιστο

Παρακάτω παρουσιάζουμε τον ψευδοκώδικα του βήματος διαταραχής.

διαδικασία Shake()

Για κάθε tour:

Διέγραψε το σύνολο των επισκέψεων από το i μέχρι το j;

Υπολόγισε το Shift;

Για κάθε επίσκεψη μετά το j (μέχρι το Shift ==0):

Μετατόπισε τις επισκέψεις προς την αρχή της διαδρομής;

Ενημέρωσε Arrive, Start, Wait, MaxShift, Shift;

Τέλος Για;

Για κάθε επίσκεψη πριν το i:

Ενημέρωσε το MaxShift;

Τέλος Για;

Τέλος Για;

Τέλος διαδικασίας;

4.3.3 Ευρετικός αλγόριθμος

Ο ευρετικός αλγόριθμος ξεκινάει με ένα σύνολο από άδειες διαδρομές και αρχικοποιεί τις παραμέτρους του βήματος διαταραχής σε 1. Ο ευρετικός αλγόριθμος τρέχει συνεχόμενα μέχρι, μετά από ένα καθορισμένο αριθμό επαναλήψεων, να μην μπορούν να βρεθούν βελτιωμένες λύσεις.

Αρχικά, το βήμα της εισαγωγής εφαρμόζεται μέχρι να βρεθεί ένα τοπικό βέλτιστο. Εάν αυτή η λύση είναι καλύτερη από την τωρινή λύση, καταγράφεται και το *R* γίνεται πάλι ένα, για το επόμενο βήμα διαταραχής. Μετά, εφαρμόζεται το βήμα διαταραχής, το *S* αυξάνεται όσο είναι η τιμή του *R* και το *R* αυξάνεται κατά ένα για το επόμενο βήμα διαταραχής. Εάν το *S* είναι ίσο ή μεγαλύτερο από το μέγεθος της μικρότερης διαδρομής, τότε το μέγεθος της μικρότερης διαδρομής θα αφαιρεθεί από το *S*. Εάν το *R* ισούται με *n/(3*m),* επανέρχεται σε ένα.

Χρησιμοποιώντας τις παραμέτρους που περιγράψαμε παραπάνω, τότε είναι πιθανό κάθε τοποθεσία της διαδρομής να διαγραφεί τουλάχιστον μια φορά. Αυτό αποδεικνύεται μια εξαιρετική τεχνική σε αρκετά γνωστά προβλήματα όταν χρησιμοποιούνται απλές ευρετικές βελτιώσεις όπως αναφέρθηκε από τον Gendreau et al. [23]. Ολόκληρος ο χώρος λύσης εξερευνείται καλύτερα και αποφάσεις που πάρθηκαν και ήταν λανθασμένες βελτιώνονται. Αυτή η ικανότητα ενισχύεται από το γεγονός ότι ο ευρετικός αλγόριθμος συνεχίζει την αναζήτηση από την τωρινή λύση και δεν επιστρέφει στην καλύτερη λύση που έχει βρεθεί μέχρι τότε.

Ο μέγιστος αριθμός τοποθεσιών προς διαγραφή *(n/(3∗m))* και ο μέγιστος αριθμός επαναλήψεων χωρίς να έχει παρατηρηθεί κάποια βελτίωση (150) είναι οι μόνες παράμετροι που είναι προκαθορισμένες στον ευρετικό αλγόριθμο. Για τον μέγιστο αριθμό τοποθεσιών προς διαγραφή, ένα ποσοστό *n/m* χρησιμοποιείται. Αλλάζοντας αυτό το ποσοστό δεν παρατηρείται κάποια βελτίωση στον χρόνο υπολογισμού ή στα αποτελέσματα, ενώ τέλος παρατηρήθηκε ότι μια αύξηση στον αριθμό των επαναλήψεων δεν επιφέρει σημαντική βελτίωση στα παραγόμενα αποτελέσματα ενώ ταυτόχρονα μεγαλώνει αισθητά ο χρόνος υπολογισμού.

Παρακάτω παρουσιάζουμε τον ψευδοκώδικα του βήματος διαταραχής.

διαδικασία Heuristic()

S = 1;

R = 1;

numberOfTimesNotImproved = 0;

BestFoundSolution = null;

Όσο numberOfTimesNotImproved < 150 Κάνε:

Όσο η λύση δεν είναι τοπικά βέλτιστη Κάνε:

Insertion Step:

Τέλος Όσο;

Αν η Λύση που βρέθηκε είναι καλύτερη από τη BestFoundSolution Τότε:

BestFoundSolution = Λύση;

R = 1;

numberOfTimesNotImproved = 0;

Αλλιώς

numberOfTimesNotImproved = numberOfTimesNotImproved + 1;

Τέλος Αν;

Shake(R,S);

S = S+R;

R = R+1;

Αν S > μήκος της μικρότερης διαδρομής τότε:

S = S - μήκος μικρότερης διαδρομής;

Τέλος Αν;

*Αν R == n/3*m τότε:*

R = 1;

Τέλος Αν;

Τέλος Όσο;

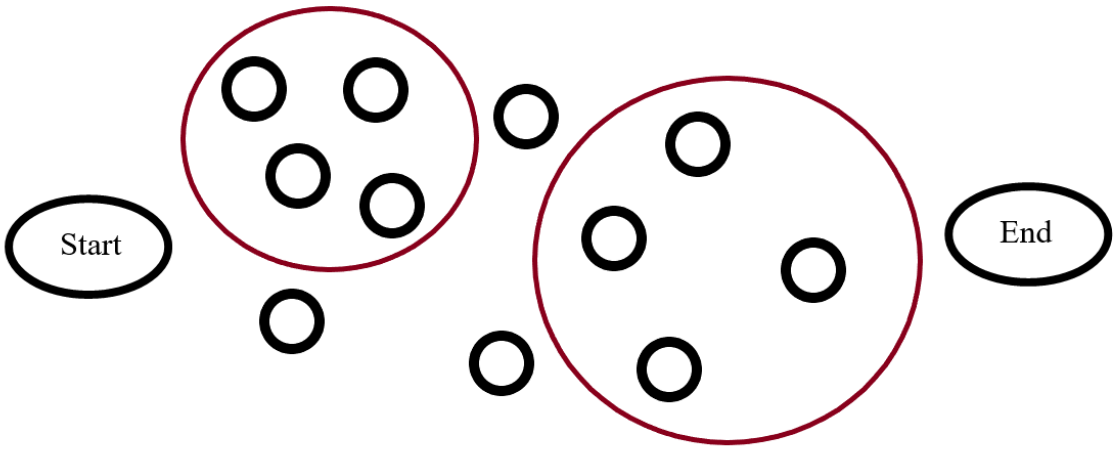
Επέστρεψε την BestFoundSolution;

Τέλος διαδικασίας;

4.4 Αλγόριθμοι με συστάδες (Cluster-Based Heuristics)

Μία διαφοροποίηση του ILS, που αναφέραμε παραπάνω, αφορά τους αλγορίθμους που εισάγουν τις πιθανές τοποθεσίες σε συστάδες (clusters), για να αντιμετωπίσουν το πρόβλημα των επισκέψεων σε τοποθεσίες που είναι απομακρυσμένες σε σχέση με τις υπόλοιπες τοποθεσίες της διαδρομής. Δυο τέτοιοι αλγόριθμοι είναι οι Cluster Search Cluster Ratio (CSCRatio) και Cluster Search Cluster Routes (CSCRoutes), που παρουσιάστηκε από τον Gavalas et. al. [37].

Οι δύο αυτοί αλγόριθμοι χωρίζουν τις πιθανές τοποθεσίες (POIs) σε ομάδες σύμφωνα με ένα κριτήριο εγγύτητας. Οι τοποθεσίες που ανήκουν σε μια συστάδα είναι σε κοντινή απόσταση η μία με την άλλη (Εικόνα 4). Έχοντας δηλαδή επισκεφθεί ένας τουρίστας μια τοποθεσία με υψηλή βαθμολογία από μία ομάδα, θα έχει ως αποτέλεσμα οι αλγόριθμοι αυτοί να ενθαρρύνουν την επίσκεψη τοποθεσιών που ανήκουν στην ίδια συστάδα – ομάδα.



Εικόνα 4 – Συστάδες από POI

Για να γίνει ο χωρισμός των τοποθεσιών σε συστάδες και οι δύο αλγόριθμοι χρησιμοποιούν των αλγόριθμο k-means. Ο αλγόριθμος αυτός δέχεται ως είσοδο τον αριθμό των συστάδων που επιθυμούμε να δημιουργήσουμε. Μόλις δημιουργηθούν οι συστάδες, η φάση αρχικοποίησης αρχίζει (RouteInitPhase). Κατά την διάρκεια αυτής της φάσης μια τοποθεσία (POI) εισέρχεται σε κάθε μία από τις *k* αρχικές διαδρομές. Κάθε ένα από *k* εισηγμένα POI ανήκει σε διαφορετική συστάδα. Για να αποφασίσουμε ποια POI θα εισαχθούν αρχικά, έχουν αναπτυχθεί διάφορες τεχνικές. Εμείς εισάγουμε τα POI με τους μεγαλύτερους λόγους οφέλους προς χρονική επιβάρυνση και στην συνέχεια τρέχει ο CSCRatio και CSCroutes.

Παρακάτω παρουσιάζουμε τον ψευδοκώδικα CSCRatio

διαδικασία CSCratio()

Τρέχουμε των k-means με k = αριθμόςΣυστάδων;

Κατασκευάζουμε την λίστα με το σύνολο των συστάδων;

It1 = maxIterations/4;

*It2 = 2*maxIterations/4;*

*It3 = 3*maxIterations/4;*

Όσο η λίστα με τις συστάδες δεν είναι άδεια κάνε:

Διέγραψε τα POI που περιλαμβάνονται στη παρούσα λύση;

theClusterSetIdToInsert = τελευταία τοποθεσία του listOfClusterSet;

RouteInitPhase(theClusterSetIdToInsert);

startNumber = 1;

removeNumber = 1;

notImproved = 0;

Όσο notImproved < maxIterations κάνε:

Εάν notImproved < it2 κάνε:

Εάν notImproved < it1 τότε clusterParameter = 1.3:

Διαφορετικά clusterParameter = 1.2;

Τέλος εάν;

Διαφορετικά:

Εάν notImproved < it3 τότε clusterParameter = 1.1:

Διαφορετικά clusterParameter = 1.0;

Τέλος εάν;

Τέλος εάν;

Όσο δεν είναι τοπικό βέλτιστο κάνε:

CSCRatio Insert(clusterParameter);

Τέλος όσο;

Εάν όφελος παρούσας λύσης > όφελος καλύτερης λύσης τότε:

bestSolution = currentSolution;

removeNumber = 1;

notImproved = 0;

Διαφορετικά άυξησε το notImproved κατά 1:

Τέλος εάν;

Εάν removeNumber > μέγεθος παρούσα λύσης / 2 τότε:

removeNumber = 1;

Τέλος εάν;

Διαταραχή(removeNumber,startNumber);

Αύξησε το startNumber κατά removeNumber;

Αύξησε το removeNumber κατά:

Εάν startNumber ≥ μικρότερο μέγεθος παρούσας λύσης τότε:

Μείωσε startNumber κατά μικρότερο μέγεθος παρούσας λύσης;

Τέλος εάν;

Τέλος όσο;

Τελος όσο;

Επιστροφή καλύτερης λύσης;

Τέλος διαδικασίας;

Όπως, παρατηρούμε ο CSCRatio μοιάζει αρκετά με τον ILS. Η κύρια διαφοροποίηση του έγκειται στην παράμετρο clusterParameter (clusterParameter ≥ 1), η οποία ευνοεί την εισαγωγή κοντινών τοποθεσιών που ανήκουν στην ίδια συστάδα. Η μεταβλητή clusterParameter στην ουσία αποτελεί την έμφαση που θα δοθεί από τον αλγόριθμο στην εισαγωγή τοποθεσιών από μια συστάδα. Όσο μεγαλύτερη είναι η τιμή, τόσο μεγαλύτερη είναι η πιθανότητα η επίσκεψη μιας τοποθεσίας να συνοδευτεί από επίσκεψη σε τοποθεσία που θα ανήκει στην ίδια συστάδα και άρα σε κοντινή απόσταση. Αρχικά χρησιμοποιείται η τιμή 1.3 για να αρχικοποιηθεί και όσο προχωρούν οι επαναλήψεις ελαττώνεται κατά 0.1. Στο τέλος το βήμα εισαγωγής CSCRatio Insert έχει γίνει ίδιο με το ILS Insert.

4.5 Ο αλγόριθμος GRASP-ELS

Μία εξέλιξη του αλγορίθμου Greedy Randomized Adaptive Search Procedure (GRASP) που παρουσιάσαμε στην ενότητα 3.2, είναι ο συνδυασμός του αλγορίθμου εκείνου με την εξελεγκτική μέθοδο τοπικής αναζήτησης (ELS). Ο αλγόριθμος GRASP – ELS παρουσιάστηκε από τον Labadie et. al. (2011) [38] και πρόκειται για ένα ευρετικό αλγόριθμο για την επίλυση του προβλήματος TOPTW. Συγκριτικά με τον ILS, τον οποίο επίσης παρουσιάσαμε παραπάνω, έχει καλύτερα αποτελέσματα ωστόσο ο υπολογιστικός χρόνος για την παραγωγή των αποτελεσμάτων είναι αρκετά μεγάλος.

Ο συγκεκριμένος αλγόριθμος αρχικά προτείνει πέντε απλούς κατασκευαστικούς ευρετικούς αλγόριθμους για να παράγουμε αρχικές λύσεις. Οι πρώτες τρεις μέθοδοι προσπαθούν να εισάγουν μία τοποθεσία στη λύση σε κάθε επανάληψη. Διαφέρουν ως προς τον τρόπο που διαλέγουν την καλύτερη κορυφή προς εισαγωγή. Οι άλλες δύο μέθοδοι είναι παραλλαγές των

sweep αλγορίθμων. Πρώτα δημιουργούν συστάδες από τοποθεσίες και μετά κατασκευάζουν μια διαδρομή από κάθε συστάδα.

Στην συνέχεια ο GRASP – ELS ακολουθεί περίπου την ίδια λογική με τον ILS, όπου προσπαθεί σε κάποιες επαναλήψεις να εισάγει τοποθεσίες, φροντίζοντας πάντα να τηρούνται οι όποιοι περιορισμοί υπάρχουν στη διαδρομή καθώς και οι περιορισμοί των υποψήφιων τοποθεσιών, ενώ ακολουθεί το βήμα της τοπικής αναζήτησης (Variable Neighborhood Descent).

Στη συνέχεια παρουσιάζουμε τους πέντε κατασκευαστικούς αλγόριθμους.

4.5.1 Insertion Heuristics

Οι ευρετικοί αλγόριθμοι εισαγωγής (Insertion Heuristics) κατασκευάζουν μια εφικτή λύση εισάγοντας τοποθεσίες. Στα κλασσικά προβλήματα δρομολόγησης το κριτήριο για την εισαγωγή μιας τοποθεσίας έχει σαν παράμετρο την αύξηση του συνολικού κόστους εάν πραγματοποιηθεί η εισαγωγή αυτή. Στο πρόβλημα TOPTW, ωστόσο, υπάρχουν δύο παράμετροι οι οποίοι λαμβάνονται υπόψιν, το συνολικό όφελος και το συνολικό μήκος της διαδρομής. Για την εισαγωγή μιας τοποθεσίας θα πρέπει να υπάρξει μια καλή αναλογία αυτών των δύο. Η διαφοροποίηση στο συγκεκριμένο αλγόριθμο έχει να κάνει ότι εδώ εξετάζεται και ο αριθμός των γειτονικών τοποθεσιών.

Πιο αναλυτικά, ας θεωρήσουμε ότι επιθυμούμε να εισάγουμε μια κορυφή v_j μετά την κορυφή v_i σε μια διαδρομή. Η μεταβλητή $shift_j$ συμβολίζει την χρονική επιβάρυνση της διαδρομής από την εισαγωγή της τοποθεσίας v_j μετά τον κόμβο v_i. Ως A_i συμβολίζουμε το σύνολο των τοποθεσιών που είναι προσβάσιμες από το v_i και για τις οποίες ισχύει ότι $z_i + t_{i,j} <= l_j$, όπου το z_i είναι η έναρξη της επίσκεψης στην τοποθεσία v_i, $t_{i,j}$ είναι η χρονική διάρκεια της μεταφοράς από το v_i στο v_j και l_j είναι το κλείσιμο της τοποθεσίας . Με q_i συμβολίζουμε την πληθικότητα (cardinality) των τοποθεσιών που ανήκουν στο σύνολο A_i και p_j το όφελος της κορυφής v_j.

Το κριτήριο, λοιπόν, για την εισαγωγή μιας τοποθεσίας καθορίζεται από την ακόλουθη ισότητα:

$$\hat{P}_j = q_j \left(\frac{p_j}{Shift_j} \right)$$

Η τοποθεσία με τη μεγαλύτερη τιμή $\hat{P}_j$ είναι και αυτή που εισάγεται. Αυτό το κριτήριο λαμβάνει υπόψιν τόσο το εισερχόμενο όφελος όσο και την αύξηση του χρόνου, που θα δημιουργήσει η εισαγωγή. Το q_j δίνει προτεραιότητα σε κόμβους με πολλούς γειτονικούς κόμβους που η πρόσβαση τους μπορεί να πραγματοποιηθεί εντός του αντίστοιχου χρονικού παραθύρου. Αυτό αυτόματα αυξάνει και την πιθανότητα να γίνει εισαγωγή στην επόμενη επανάληψη. Στις πρώτες επαναλήψεις το q_i επικρατεί του οφέλους. Όταν όμως οι διαδρομές αρχίζουν και έχουν πλέον περισσότερες τοποθεσίες τότε το όφελος είναι αυτό που παίζει τον σημαντικότερο ρόλο.

Ο λόγος που επιλέχθηκε το συγκεκριμένο κριτήριο έναντι του κριτηρίου του ILS είναι το γεγονός ότι σε κάποια στιγμιότυπα παράγει καλύτερα αποτελέσματα.

Heuristic Insert (I)

Αυτή η μέθοδος κατασκευάζει εφικτές λύσεις με το να εισάγει στην τελευταία θέση της διαδρομής τοποθεσίες οι οποίες δεν έχουν εισαχθεί. Ξεκινώντας από την διαδρομή $r = 1$, ψάχνει να εισάγει την τοποθεσία με την καλύτερη εισαγωγή, ανάμεσα σε όλες τις τοποθεσίες που δεν έχουν εισαχθεί, μέχρι να μην υπάρχει εφικτή εισαγωγή στην υπάρχουσα διαδρομή. Τότε το r

αυξάνεται κατά ένα μέχρι να γίνει ίσο με το πλήθος των διαδρομών. Η καλύτερη τοποθεσία επιλέγεται με βάση το κριτήριο που αναφέραμε παραπάνω. Η πολυπλοκότητα αυτής της ευριστικής τεχνικής είναι $O(n^2)$ αφού ψάχνουμε μόνο την τελευταία θέση στην υπάρχουσα διαδρομή όταν προσθέτουμε μια νέα κορυφή.

Heuristic Insert (II)

Αυτή η μέθοδος δημιουργεί διαδρομές που επισκέπτονται τοποθεσίες με καλύτερο όφελος. Στο πρώτο βήμα όλες οι τοποθεσίες ταξινομούνται κατά αύξουσα σειρά με βάση το όφελος τους. Τότε m διαδρομές αρχικοποιούνται με τις *m* πιο ωφέλιμες τοποθεσίες. Στη συνέχεια, κάθε διαδρομή συμπληρώνεται με περισσότερες τοποθεσίες. Για κάθε τοποθεσία που δεν έχουμε επισκεφθεί, ψάχνουμε να βρούμε την καλύτερη θέση ανάμεσα σε όλες τις διαδρομές και σε όλες τις πιθανές θέσεις. Σαν κριτήριο εισαγωγής έχουμε και εδώ το κριτήριο που αναφέραμε παραπάνω. Η διαδικασία σταματάει όταν δεν υπάρχει πλέον εφικτή εισαγωγή.

Heuristic Insert (III)

Αυτή η μέθοδος χρησιμοποιεί ξανά μια λίστα από τοποθεσίες οι οποίες έχουν ταξινομηθεί σε αύξουσα σειρά με βάση το όφελος. Σε κάθε επανάληψη, η τοποθεσία v_i από την αρχή της λίστας θα εισαχθεί στην διαδρομή που θα έχει το μικρότερο *Shift*. Εάν για την υπάρχουσα τοποθεσία δεν βρεθεί κάποια εφικτή τοποθέτηση, τότε ο αλγόριθμος προχωράει στην επόμενη τοποθεσία της λίστας. Η συγκεκριμένη διαδικασία ακολουθείται μέχρι όλες οι τοποθεσίες να έχουν αξιολογηθεί.

Η πολυπλοκότητα χρόνου τόσο του Heuristic Insert (II) και του Heuristic Insert (III) είναι $O(n^3)$.

4.5.2 Sweep Heuristics

Ο Sweep αλγόριθμος είναι ευρετικός αλγόριθμος που σχεδιάστηκε για το πρόβλημα δρομολόγησης οχημάτων (Vehicle Routing Problem - VRP). Αρχικά, σχηματίζει συστάδες με τοποθεσίες που βρίσκονται εντός μιας ακτίνας, την οποία περιστρέφουμε γύρω από την αρχική μας τοποθεσία. Στην συνέχεια κάθε συστάδα δημιουργεί μια διαδρομή. Στην περίπτωση του VRP, η περιστροφή της ακτίνας θα σταματήσει όταν ο αριθμός των τοποθεσιών της κάθε συστάδας φτάσει τη χωρητικότητα του οχήματος.

Για το πρόβλημα του TOPTW, οι τοποθεσίες ταξινομούνται σε μη-αύξουσα σειρά ανάλογα με την πολική ακτίνα τους. Η ταξινομημένη λίστα θα κοπεί σε *m* ίσα διαστήματα l_r $r = 1, 2, \ldots, m$. Κάθε διάστημα περιλαμβάνει τουλάχιστον n/m τοποθεσίες που μπορούν να επιλεχθούν για να σχηματίσουν μια διαδρομή.

Heuristic Sweep (I)

Αρχικά, κάθε διαδρομή αρχικοποιείται με τις πιο ωφέλιμες τοποθεσίες από το διάστημα l_r. Μετά ο αλγόριθμος θα εξερευνήσει τις υπόλοιπες τοποθεσίες του l_r και θα ψάξει να βρει την καλύτερη εφικτή τοποθεσία προς εισαγωγή. Για κάθε τοποθεσία που δεν έχουμε επισκεφθεί v_j που ανήκει στο l_r και *i* μια τοποθεσία της διαδρομής, ο λόγος p_{ij} υπολογίζεται και κάθε τοποθεσία με μέγιστο p_{ij} επιλέγεται. Εάν δεν υπάρχει εφικτή τοποθεσία προς εισαγωγή στο παρόν διάστημα, η διαδικασία ξεκινάει να χτίζει την επόμενη διαδρομή μέχρι να έχουν παραχθεί *m* διαδρομές.

Heuristic Sweep (II)

Η δεύτερη παραλλαγή του sweep heuristic προσπαθεί να εισάγει τοποθεσίες που έχουν το μεγαλύτερο όφελος μέσα στο παρούσα διαδρομή. Κάθε l_r είναι ταξινομημένο σε μη αύξουσα σειρά ανάλογα με το όφελος. Ξεκινώντας από την κορυφή της λίστας, η διαδικασία αξιολογεί τις καλύτερες και ταυτόχρονα εφικτές τοποθεσίες προς εισαγωγή, ανάλογα με την αύξηση του $Shift_j$. Αυτή με το μικρότερο κόστος θα επιλεχθεί. Εάν είναι εφικτή η εισαγωγή, τότε πραγματοποιείται, διαφορετικά εξετάζεται η επόμενη τοποθεσία της λίστας.

Κάθε διάστημα περιλαμβάνει $N = \left\lceil \frac{n}{m} \right\rceil$ κορυφές, ενώ και οι δύο αλγόριθμοι sweep αξιολογούν την πιθανή εισαγωγή σε $O(n^3)$.

4.5.3 Μετα-Ευρετικός GRASP-ELS

Οι προτεινόμενοι μέθοδοι, που αφορούν την αρχική φάση χρησιμοποιούνται από τον GRASP για να παραχθούν λύσεις που θα χρησιμοποιήσει αργότερα ο ELS. Όπως αναφέραμε και στο δεύτερο κεφάλαιο, ο GRASP είναι απλά ένας ευρετικός αλγόριθμος που παράγει ανεξάρτητες τυχαίες λύσεις.

Ο ELS αρχικά παρουσιάστηκε από τους Merz and Wold (2007) [39] για τα προβλήματα τηλεπικοινωνίας. Η μέθοδος αυτή αποτελεί επέκταση του κλασσικού ILS. Ο ILS ξεκινάει με μία λύση *s* που αποκτήθηκε από τον ευρετικό αλγόριθμο και παράγει αρκετές λύσεις παιδιά (child solutions) εφαρμόζοντας το βήμα της διαταραχής. Στην συνέχεια αυτές οι λύσεις θα βελτιωθούν με το βήμα της τοπικής αναζήτησης. Ο ELS επιπλέον, παράγει αρκετά αντίγραφα των λύσεων αυτών και εφαρμόζει τον ILS σε κάθε ένα αντίγραφο.

Η διαδικασία της διαταραχής είναι απαραίτητη. Εάν η διαταραχή είναι πολύ ισχυρή, τότε η μέθοδος θα συμπεριφερθεί σαν τυχαίος ευρετικός αλγόριθμος, ενώ αν είναι αρκετά αδύναμος η λύση γρήγορα θα παγιδευτεί σε τοπικό βέλτιστο. Ο έλεγχος της διαταραχής γίνεται με την παράμετρο *p* που κυμαίνεται μεταξύ p_{min} και p_{max}.

Ο GRASP – ELS παρουσιάστηκε από τον Prins (2009) [40] για το πρόβλημα του VRP και περιγράφεται ακολούθως.

διαδικασία GraspEls()

Εισαγωγή: *Ένα στιγμιότυπο του TOPTW*

Αποτέλεσμα: *Εφικτές λύσεις του TOPTW*

Παράμετροι:

ns: μέγιστος αριθμός επαναλήψεων GRASP

ni : μέγιστος αριθμός επαναλήψεων ELS

nc: μέγιστος αριθμός παραγόμενων child λύσεων

p: παράμετρος διαταραχής

k_{max} *: μέγιστο μέγεθος από γειτονιές στην τοπική αναζήτηση*

f(s) = 0;*

Για i από 1 έως ns κάνε:

Αν i=1 τότε :

š = αρχική λύση (η καλύτερη που προκύπτει από τους 5 κατασκευαστικούς);

Αλλιώς :

š = τυχαία αρχική λύση;

Τέλος Αν;

s = Variable Neighborhood Descent (š, k_{max});

Αν f(s)>f(s) τότε:*

s = s*

Τέλος Αν;

P = p_{min};

Για j από 1 έως ni κάνε:

f(s') = 0;

Για c από 1 έως nc κάνε:

s'' = Perturbation(s, p);

ŝ = Variable Neighborhood Descent (s'' , k_{max});

Αν f(ŝ)>f(s') τότε :

s' = ŝ;

Τέλος Αν;

Τέλος Για;

Αν f(s') > f(s) Τότε:

s = s';

p = p_{min};

Αλλιώς:

P = min{ p_{max} , p+1};

Τέλος Αν;

Τέλος Για;

Αν f(s)>f(s) τότε:*

s = s;*

Τέλος Αν;

Τέλος Για;

Επέστρεψε τη λύση s;*

Τέλος διαδικασίας;

Ο παραπάνω αλγόριθμος παρουσιάζει την δομή του GRASP-ELS. Ο GRASP-ELS εξαρτάται από τρεις παραμέτρους: τον αριθμό των πανομοιότυπων λύσεων παιδιά (child) *nc* που παράγονται σε κάθε επανάληψη του ELS, τον μέγιστο αριθμό επαναλήψεων του ELS *ni* και τον αριθμό των λύσεων *ns* που παράγει ο GRASP.

Αρχικά, σε κάθε επανάληψη ο GRASP παράγει μια αρχική λύση *š*, στην οποία λύση θα εφαρμοστεί η διαδικασία της τοπικής αναζήτησης. Η λύση που θα προκύψει (*s*) εισάγεται στον ELS, ο οποίος δημιουργεί διάφορες παρόμοιες λύσεις παιδιά (child) για κάθε αντίγραφο της s

και στην συνέχεια εκτελείται στις λύσεις αυτές η διαδικασία της διαταραχής και της τοπικής αναζήτησης. Η καλύτερη λύση παιδί (child) s' που προέκυψε, αποθηκεύεται και αν είναι καλύτερη από την s την αντικαθιστά. Σε διαφορετική περίπτωση ο ELS θα εκτελέσει την επόμενη επανάληψη έχοντας ως αρχική λύση ξανά την s. Ο αλγόριθμος σταματάει όταν ο μέγιστος αριθμός επαναλήψεων *ns* έχει ολοκληρωθεί.

4.5.4 Διαδικασία Variable Neighborhood Descent

Η διαδικασία Variable Neighborhood Descent είναι μια διαδικασία τοπικής αναζήτησης, αφού κατά την διάρκεια της αναζήτησης εναλλάσσεται μεταξύ δύο γειτονιών με ντετερμινιστικό τρόπο. Η πρώτη γειτονιά περιλαμβάνει κλασσικές κινήσεις που περιλαμβάνουν μία ή δύο διακριτές διαδρομές. Οι κινήσεις αυτές είναι :

1. 2-opt - διαγράφει και αντικαθιστά δυο τοποθεσίες σε μια διαδρομή και τις αναδιοργανώνει
2. Or-opt - μετακινεί μια τοποθεσία σε άλλη θέση
3. 2-opt* - εναλλάσσει δυο υπο-μονοπάτια μεταξύ δυο διαδρομών
4. Exchange - εναλλάσσει δυο τοποθεσίες

Αυτές οι κινήσεις στοχεύουν στο να μειώσουν το μήκος των διαδρομών. Όταν η διαδρομή έχει μήκος 1, τότε εκτελούνται μόνο οι κινήσεις 2-Opt, Or-Opt και Exchange ενώ από 2 και πάνω εκτελούνται όλες οι κινήσεις. Στην γειτονιά αυτή ελέγχονται όλες οι εφικτές κινήσεις και η πρώτη που βελτιώνει την διαδρομή κρατείται. Ο έλεγχος για την εγκυρότητα της γειτονιάς και το κατά πόσο είναι εφικτή η εισαγωγή της στην διαδρομή γίνεται σε *O(1)* όπως έδειξε ο Kindervater and Savelsbergh (1997) [41].

Το δεύτερο σετ από γειτονιές εισάγει τοποθεσίες που δεν έχουν δεχθεί επίσκεψη μέχρι τώρα (unvisited) στην παρούσα διαδρομή. Μια γειτονιά $N_k(S)$, $k = 0, \ldots, k_{max}$ ορίζεται σαν ο αριθμός των συνεχόμενων τοποθεσιών k που θα διαγραφούν από μια διαδρομή και θα αντικατασταθούν από μια σειρά τοποθεσιών εκτός διαδρομής. Η αναζήτηση ξεκινά με $k = 0$, δηλαδή δεν διαγράφεται καμιά τοποθεσία, και ελέγχει όλες τις διαδρομές για να βρει μια κίνηση που θα μεγιστοποιήσει την διαφορά του συνολικού οφέλους των εισηγμένων τοποθεσιών και του συνολικού οφέλους των διαγραμμένων τοποθεσιών. Μόνο αυτές με θετική διαφορά θεωρούνται υποψήφιες προς εισαγωγή.

Η καλύτερη κίνηση για τη συγκεκριμένη τιμή της παραμέτρου k, που βρέθηκε ανάμεσα σε όλες τις διαδρομές, εκτελείται και η αναζήτηση συνεχίζεται στην παρούσα γειτονιά, μέχρι να μην βρεθεί άλλη κίνηση. Αν δεν σημειωθεί βελτίωση τότε η παράμετρος k αυξάνεται κατά 1 και αρχίζει ξανά η διαδικασία. Η διαδικασία σταματάει όταν το k γίνει ίσο με το k_{max} ή όταν δεν βρεθεί κάποια βελτιωμένη λύση. Ωστόσο, η διαδικασία δεν εξερευνά όλες τις γειτονιές καθώς αυτό θα απαιτούσε αρκετή υπολογιστική ισχύ.

Η διαδικασία ελέγχου των τοποθεσιών προς εισαγωγή στη διαδρομή ανήκει στην κατηγορία των προβλημάτων που είναι NP-Hard. Ο έλεγχος αυτός μπορεί να μοντελοποιηθεί ακολούθως: δοθέντος του χρονικού περιθωρίου T_{ij} μεταξύ των τοποθεσιών v_i και v_j, και ενός συνόλου από μη-επισκέψιμες τοποθεσίες $U \subset V$, βρίσκουμε μια ακολουθία τοποθεσιών $v_{l1}, v_{l2}, \ldots, v_{lq}$ από το U που να ταιριάζουν στο συγκεκριμένο χρονικό περιθώριο και να μεγιστοποιούν το όφελος της διαδρομής. Επιπλέον, φροντίζουμε το χρονικό παράθυρο κάθε τοποθεσίας να τηρείται.

Παρατηρούμε, λοιπόν, ότι το συγκεκριμένο πρόβλημα μπορεί να αναχθεί στο πρόβλημα του προσανατολισμού. Οι κόμβοι v_i και v_j μπορούν να θεωρηθούν σαν ο αρχικός και ο τελικός κόμβος, ενώ το T_{ij} είναι ο διαθέσιμος χρόνος. Παρά την υψηλή υπολογιστική πολυπλοκότητα,

ακριβής υπολογισμός της ακολουθίας μπορεί να είναι εφικτός για μέτριες τιμές του *k*, από την στιγμή που ο αριθμός των εφικτών ακολουθιών μειώνεται λόγω των χρονικών περιθωρίων.

4.5.5 Διαδικασία διαταραχής (perturbation)

Το βήμα διαταραχής μπορεί να γενικευτεί σαν τυχαίες μεταλλάξεις της παρούσας αρχικής λύσης. Η διαδικασία αυτή διασφαλίζει την διαφορετικότητα της λύσης αλλά μπορεί επίσης και να οδηγήσει στην επιδείνωση της ποιότητας της. Επιπλέον, απαιτείται πολύς υπολογιστικός χρόνος για να επιτευχθεί η επαναφορά της λύσης σε περίπτωση επιδείνωσης.

Η διαδικασία της διαταραχής επιλέγει τυχαία μια ακολουθία από *p* συνεχόμενες τοποθεσίες και τις διαγράφει από κάθε διαδρομή. Στην συνέχεια όλες οι τοποθεσίες που δεν έχουν δεχθεί ακόμα επίσκεψη ελέγχονται, για να σχηματιστούν ακολουθίες που μπορούν να εισαχθούν στα κενά διαστήματα της κάθε διαδρομής. Η ακολουθία με το μεγαλύτερο όφελος εισέρχεται στη διαδρομή και η συγκεκριμένη διαδρομή δεν εξετάζεται παραπάνω. Η διαδικασία συνεχίζεται μέχρι όλες οι διαδρομές να εισάγουν μια ακολουθία ή μέχρι να μην είναι εφικτή κάποια εισαγωγή σε μία διαδρομή. Εάν η ακολουθία που διαλέξαμε τυχαία παραπάνω δεν μπορεί να εισαχθεί σε καμία διαδρομή, θα διαγραφεί και δεν θα εξεταστεί περαιτέρω. Αυτό συμβαίνει συνήθως όταν το *p* είναι πολύ μικρό.

Κεφάλαιο 5°

5.1 Ο τροποποιημένος αλγόριθμος ILS

Στο συγκεκριμένο κεφάλαιο θα περιγράψουμε τις αλλαγές που πραγματοποιήσαμε στον αλγόριθμο του ILS.

Η πρώτη αλλαγή αφορά τις επισκέψεις στις τοποθεσίες. Ενώ στον κανονικό αλγόριθμο η επίσκεψη της τοποθεσίας μπορεί να πραγματοποιηθεί και μετά το τέλος του χρονικού παραθύρου, στον τροποποιημένο μας αλγόριθμο η επίσκεψη γίνεται αυστηρά στα πλαίσια που ορίζει το χρονικό παράθυρο.

Για να το πετύχουμε αυτό έχουμε εισάγει τον παρακάτω έλεγχο :

$$s_j + T_j < C_j$$

Ο περιορισμός αυτός φροντίζει ώστε η έναρξη της επίσκεψης μας στην τοποθεσία και ο χρόνος διαμονής μας εκεί, δεν θα επεκταθεί πέραν του τέλους του χρονικού παραθύρου της τοποθεσίας.

Η δεύτερη αλλαγή αφορά την εισαγωγή εστιατορίων στην διαδρομή μας σε μια προκαθορισμένη ώρα που δίνεται ως είσοδος. Για να πετύχουμε την εισαγωγή εστιατορίου στις διαδρομές μας θα κάνουμε χρήση των μεταβλητών *Shift* και *MaxShift*, ώστε να παραμείνει ο αλγόριθμος μας γρήγορος. Έστω λοιπόν ότι θέλουμε να επιλέξουμε να γευματίσουμε στις διαδρομές μας κάποια στιγμή εντός του χρονικού παραθύρου π.χ. 15:00 – 16:00. Ο αλγόριθμος μας θα φροντίσει να εισάγει σε κάθε διαδρομή μία επίσκεψη σε ένα εστιατόριο την ώρα που θα του πούμε ανάλογα την τοποθεσία μας. Αφού λοιπόν εκτελεστεί ο αλγόριθμος μας και προσδιορίσει τις επισκέψεις στις διάφορες τοποθεσίες, τότε θα εκτελέσουμε το βήμα εισαγωγής εστιατορίου.

Αρχικά, θα προσπαθήσουμε να εισάγουμε ένα εστιατόριο ανάμεσα στην ώρα που επιθυμούμε χωρίς να διαγράψουμε κάποια τοποθεσία. Εάν η διαδικασία δεν καταφέρει να εισάγει κάποιο εστιατόριο, τότε θα εκτελέσουμε το βήμα διαταραχής (perturbation step),

αφαιρώντας μία τοποθεσία στο συγκεκριμένο χρονικό όριο που επιθυμούμε να εισάγουμε το εστιατόριο και η τοποθεσία αυτή θα είναι η τοποθεσία με το μικρότερο όφελος για τον ταξιδιώτη. Στην συνέχεια θα εκτελεστεί πάλι το βήμα εισαγωγής κορυφής για το εστιατόριο.

Παρόμοια λύση για τοποθεσίες γεύματος έχει παρουσιασθεί και στην εργασία του D. Gavalas et al. [44]. Στον αλγόριθμο που παρουσιάζεται στην εργασία εκείνη, οι τοποθεσίες ομαδοποιούνται σε συστάδες και γίνεται χρήση των μέσων μαζικής μεταφοράς για την μετακίνηση του τουρίστα στις διαδρομές. Έτσι, η τοποθεσία που θα επισκεφθεί ο τουρίστας και συγκεκριμένα ο χρόνος έναρξης της επίσκεψης, προσαρμόζεται ανάλογα με τον χρόνο που απαιτείται από τα μέσα μαζικής μεταφοράς για την μετακίνηση στην εκάστοτε τοποθεσία. Στην παρούσα εργασία, η τροποποίηση που προτείνεται ‘εγκλωβίζει’ τον χρήστη σε αυστηρά κριτήρια επίσκεψης. Ο τουρίστας δεν παρεκκλίνει από το πρόγραμμα του και οι ώρες επίσκεψης στις τοποθεσίες και τα εστιατόρια πρέπει να είναι εντός του χρονικού παραθύρου και τηρώντας τον χρόνο επίσκεψης που έχουμε ορίσει.

Παρακάτω, παρουσιάζουμε το κύριο κομμάτι του ψευδοκώδικα για τον τροποποιημένο αλγόριθμο.

Διαδικασία restaurantsILS();

Λύση με εστιατόρια = κενή;

λύση = ILS(τοποθεσίες);

Ώρα γεύματος = αρχή και τέλος ώρας που θέλουμε να γευματίσουμε;

εστιατόρια = λίστα με τα διαθέσιμα εστιατόρια;

Τοποθεσίες εντός του χρόνου γεύματος = λίστα με τις τοποθεσίες που έχουμε σχεδιάσει να επισκεφθούμε στο χρονικό παράθυρο που θέλουμε να γευματίσουμε;

Εισαγωγή εστιατορίου = λάθος;

Όσο η εισαγωγή του εστιατορίου είναι λάθος κάνε:

Εάν μπορούμε να εισάγουμε εστιατόριο τότε:

Λύση με εστιατόρια = λύση συμπεριλαμβανομένης της τοποθεσίας γεύματος;

Εισαγωγή εστιατορίου = αληθής;

Αλλιώς:

Βήμα Διαταραχής(Τοποθεσίες εντός του χρόνου γεύματος); // Θα αφαιρέσουμε από την διαδρομή την τοποθεσία με το μικρότερο όφελος

Τέλος αν;

Τέλος όσο;

Επίστρεψε την λύση με τα εστιατόρια;

Τέλος διαδικασίας;

5.2 Αποτελέσματα τροποποιημένου αλγορίθμου ILS

Ο τροποποιημένος αλγόριθμος ILS με αυστηρή τήρηση των χρονικών παραθύρων και ο τροποποιημένος αλγόριθμος ILS για την εισαγωγή εστιατορίων, εκτελέστηκαν στα στιγμιότυπα που χρησιμοποιούνται και για τον ILS (Solomon datasets) [43]. Κάθε σύνολο δεδομένων έχει 100 πιθανές τοποθεσίες για επίσκεψη και κάθε τοποθεσία έχει ένα συγκεκριμένο χρονικό παράθυρο. Πρέπει να αναφέρουμε ότι τα συγκεκριμένα σύνολα δεδομένων είναι ευρέως χρησιμοποιούμενα για το πρόβλημα του TOPTW.

Οι υπολογισμοί έγιναν σε ένα προσωπικό υπολογιστή με Intel i7-8550U @ 1.80GHz και 8GB RAM. Εξετάζονται 4 σύνολα διαδρομών *(m=1,2,3,4)* για να υπάρχει άμεση συσχέτιση με τον ILS, ενώ για τον τροποποιημένο αλγόριθμο ILS, όπου εισάγουμε μια τοποθεσία γεύματος, επιλέγουμε να χωρίσουμε τα σύνολα σε 2 κατηγορίες, μία κατηγορία που θα αφορά τις τοποθεσίες προς επίσκεψη και μια κατηγορία που θα αφορά τα εστιατόρια. Επίσης, για κάθε τιμή της παραμέτρου m έχουμε διαλέξει διαφορετικό χρόνο εισαγωγής του εστιατορίου στην διαδρομή μας.

Πιο συγκεκριμένα, οι πίνακες 1-4 παρουσιάζουν τα αποτελέσματα του πρωτότυπου ILS και του τροποποιημένου ILS με αυστηρότερα κριτήρια επίσκεψης, ενώ, οι πίνακες 5-8 παρουσιάζουν τα αποτελέσματα του τροποποιημένου ILS με τα αυστηρότερα κριτήρια επίσκεψης και του τροποποιημένου ILS με την δυνατότητα γεύματος.

Πίνακας 1 – Σύγκριση πρωτότυπου ILS και τροποποιημένου ILS με αυστηρή τήρηση των χρονικών παραθύρων για m = 1

	ΠΡΩΤΟΤΥΠΟΣ ILS			ILS ΜΕ ΑΥΣΤΗΡΑ ΚΡΙΤΗΡΙΑ ΕΠΙΣΚΕΨΗΣ		
Στιγμιότυπο Εισόδου	Συνολικό Όφελος	Επισκέψεις	Υπολογιστικός Χρόνος	Συνολικό Όφελος	Επισκέψεις	Υπολογιστικός Χρόνος
c101	320	10	0.04000 seconds	300	10	0.06700 seconds
c102	360	11	0.03000 seconds	320	11	0.05000 seconds
c103	390	10	0.05000 seconds	380	11	0.03200 seconds
c104	400	10	0.03000 seconds	390	11	0.02400 seconds
c105	340	10	0.03100 seconds	310	9	0.03400 seconds
c106	340	10	0.03000 seconds	310	10	0.02800 seconds
c107	360	11	0.03200 seconds	320	10	0.03300 seconds
c108	370	11	0.03000 seconds	320	10	0.02500 seconds
c109	380	11	0.03000 seconds	340	11	0.02900 seconds
r101	182	7	0.01000 seconds	186	8	0.06900 seconds
r102	286	11	0.02000 seconds	247	10	0.03100 seconds
r103	286	10	0.02000 seconds	252	10	0.03600 seconds

r104	297	11	0.02000 seconds	258	11	0.03200 seconds
r105	247	11	0.01000 seconds	215	9	0.03900 seconds
r106	293	11	0.02000 seconds	258	10	0.03700 seconds
r107	288	10	0.02000 seconds	243	11	0.03700 seconds
r108	297	11	0.02000 seconds	233	11	0.03400 seconds
r109	276	11	0.00200 seconds	264	11	0.05300 seconds
r110	281	11	0.03000 seconds	247	11	0.05500 seconds
r111	295	11	0.02000 seconds	276	11	0.05200 seconds
r112	295	11	0.02000 seconds	274	11	0.02900 seconds

	ΠΡΩΤΟΤΥΠΟΣ ILS			ILS ΜΕ ΑΥΣΤΗΡΑ ΚΡΙΤΗΡΙΑ ΕΠΙΣΚΕΨΗΣ		
Στιγμιότυπο Εισόδου	Συνολικό Όφελος	Επισκέψεις	Υπολογιστικός Χρόνος	Συνολικό Όφελος	Επισκέψεις	Υπολογιστικός Χρόνος
rc101	219	9	0.02000 seconds	203	8	0.01300 seconds
rc102	259	9	0.02000 seconds	245	10	0.00700 seconds
rc103	265	11	0.03000 seconds	245	10	0.01000 seconds
rc104	297	11	0.00300 seconds	240	10	0.00700 seconds
rc105	221	11	0.02000 seconds	162	7	0.00700 seconds
rc106	239	11	0.02000 seconds	200	8	0.01000 seconds
rc107	274	11	0.02000 seconds	240	10	0.01100 seconds
rc108	288	11	0.02000 seconds	240	10	0.01300 seconds

Πίνακας 2 – Σύγκριση πρωτότυπου ILS και τροποποιημένου ILS με αυστηρή τήρηση των χρονικών παραθύρων για m = 2

	ΠΡΩΤΟΤΥΠΟΣ ILS			ILS ΜΕ ΑΥΣΤΗΡΑ ΚΡΙΤΗΡΙΑ ΕΠΙΣΚΕΨΗΣ		
Στιγμιότυπο Εισόδου	Συνολικό Όφελος	Επισκέψεις	Υπολογιστικός Χρόνος	Συνολικό Όφελος	Επισκέψεις	Υπολογιστικός Χρόνος
c101	590	21	0.14000 seconds	540	18	0.10300 seconds
c102	650	22	0.09000 seconds	610	21	0.19100 seconds
c103	700	22	0.12000 seconds	660	22	0.07400 seconds
c104	750	22	0.15000 seconds	700	22	0.10800 seconds
c105	640	21	0.08000 seconds	540	18	0.04800 seconds
c106	320	20	0.08000 seconds	550	18	0.04300 seconds
c107	670	22	0.14000 seconds	570	18	0.04600 seconds
c108	670	22	0.08000 seconds	570	19	0.04600 seconds
c109	710	22	0.09000 seconds	640	20	0.05600 seconds
r101	330	13	0.04000 seconds	307	13	0.03100 seconds
r102	508	21	0.09000 seconds	455	18	0.06400 seconds
r103	513	20	0.09000 seconds	450	19	0.05700 seconds
r104	539	22	0.15000 seconds	483	21	0.04800 seconds
r105	430	18	0.08000 seconds	369	16	0.05800 seconds
r106	529	21	0.09000 seconds	438	19	0.05600 seconds
r107	529	21	0.10000 seconds	483	20	0.09300 seconds
r108	549	24	0.14000 seconds	486	21	0.11600 seconds

r109	498	22	0.05000 seconds	412	18	0.04200 seconds
r110	515	22	0.10000 seconds	435	19	0.06200 seconds
r111	535	23	0.06000 seconds	471	20	0.05000 seconds
r112	515	21	0.05000 seconds	450	20	0.04000 seconds

	ΠΡΩΤΟΤΥΠΟΣ ILS			ILS ΜΕ ΑΥΣΤΗΡΑ ΚΡΙΤΗΡΙΑ ΕΠΙΣΚΕΨΗΣ		
Στιγμιότυπο Εισόδου	Συνολικό Όφελος	Επισκέψεις	Υπολογιστικός Χρόνος	Συνολικό Όφελος	Επισκέψεις	Υπολογιστικός Χρόνος
rc101	427	19	0.06000 seconds	293	11	0.04100 seconds
rc102	494	20	0.08000 seconds	326	14	0.01900 seconds
rc103	519	20	0.11000 seconds	394	16	0.03100 seconds
rc104	565	22	0.07000 seconds	459	18	0.02300 seconds
rc105	459	22	0.08000 seconds	289	12	0.02200 seconds
rc106	458	20	0.06000 seconds	377	15	0.05000 seconds
rc107	515	21	0.05000 seconds	436	18	0.02700 seconds
rc108	546	23	0.06000 seconds	464	19	0.03100 seconds

Πίνακας 3 – Σύγκριση πρωτότυπου ILS και τροποποιημένου ILS με αυστηρή τήρηση των χρονικών παραθύρων για m = 3

	ΠΡΩΤΟΤΥΠΟΣ ILS			ILS ΜΕ ΑΥΣΤΗΡΑ ΚΡΙΤΗΡΙΑ ΕΠΙΣΚΕΨΗΣ		
Στιγμιότυπο Εισόδου	Συνολικό Όφελος	Επισκέψεις	Υπολογιστικός Χρόνος	Συνολικό Όφελος	Επισκέψεις	Υπολογιστικός Χρόνος
c101	790	29	0.11000 seconds	730	25	0.09200 seconds
c102	890	32	0.21000 seconds	820	30	0.09700 seconds
c103	960	33	0.22000 seconds	920	32	0.07500 seconds
c104	1010	34	0.13000 seconds	950	33	0.07200 seconds

c105	840	30	0.10000 seconds	780	26	0.06200 seconds
c106	840	30	0.11000 seconds	780	26	0.05800 seconds
c107	900	33	0.15000 seconds	760	25	0.13600 seconds
c108	900	33	0.12000 seconds	820	27	0.08400 seconds
c109	950	33	0.20000 seconds	860	28	0.10000 seconds

r101	481	21	0.08000 seconds	415	18	0.04900 seconds
r102	685	31	0.10000 seconds	606	25	0.08000 seconds
r103	720	31	0.20000 seconds	630	27	0.06400 seconds
r104	765	34	0.15000 seconds	662	29	0.06000 seconds
r105	609	27	0.23000 seconds	519	22	0.05500 seconds
r106	719	32	0.21000 seconds	604	26	0.06700 seconds
r107	747	33	0.11000 seconds	611	26	0.09800 seconds
r108	790	36	0.31000 seconds	691	31	0.09500 seconds
r109	699	31	0.18000 seconds	580	24	0.09900 seconds
r110	711	32	0.14000 seconds	618	26	0.06500 seconds
r111	764	34	0.18000 seconds	654	28	0.06000 seconds
r112	758	34	0.11000 seconds	665	29	0.07000 seconds

	ΠΡΩΤΟΤΥΠΟΣ ILS			ILS ΜΕ ΑΥΣΤΗΡΑ ΚΡΙΤΗΡΙΑ ΕΠΙΣΚΕΨΗΣ		
Στιγμιότυπο Εισόδου	Συνολικό Όφελος	Επισκέψεις	Υπολογιστικός Χρόνος	Συνολικό Όφελος	Επισκέψεις	Υπολογιστικός Χρόνος
rc101	604	29	0.14000	448	17	0.03400

			seconds			seconds
rc102	698	30	0.13000 seconds	486	20	0.03600 seconds
rc103	747	30	0.11000 seconds	588	22	0.04700 seconds
rc104	822	33	0.13000 seconds	690	27	0.04500 seconds
rc105	654	28	0.08000 seconds	422	17	0.04200 seconds
rc106	678	31	0.10000 seconds	546	21	0.04500 seconds
rc107	745	31	0.09000 seconds	602	24	0.03700 seconds
rc108	757	29	0.11000 seconds	648	26	0.04700 seconds

Πίνακας 4 – Σύγκριση πρωτότυπου ILS και τροποποιημένου ILS με αυστηρή τήρηση των χρονικών παραθύρων για m = 4

	ΠΡΩΤΟΤΥΠΟΣ ILS			ILS ΜΕ ΑΥΣΤΗΡΑ ΚΡΙΤΗΡΙΑ ΕΠΙΣΚΕΨΗΣ		
Στιγμιότυπο Εισόδου	Συνολικό Όφελος	Επισκέψεις	Υπολογιστικός Χρόνος	Συνολικό Όφελος	Επισκέψεις	Υπολογιστικός Χρόνος
c101	1000	39	0.38000 seconds	910	31	0.07100 seconds
c102	1090	43	0.18000 seconds	980	35	0.13300 seconds
c103	1150	44	0.25000 seconds	1080	39	0.06500 seconds
c104	1220	45	0.30000 seconds	1160	42	0.05700 seconds
c105	1030	40	0.18000 seconds	950	33	0.08000 seconds
c106	1040	40	0.21000 seconds	940	32	0.08600 seconds
c107	1100	43	0.20000 seconds	950	33	0.09800 seconds
108	1100	44	0.36000 seconds	970	34	0.10000 seconds
c109	1180	45	0.25000 seconds	1030	36	0.09800 seconds

r101	601	28	0.14000 seconds	507	22	0.08500 seconds
r102	807	39	0.17000 seconds	687	29	0.09100 seconds
r103	878	42	0.22000 seconds	782	33	0.10600 seconds
r104	941	45	0.38000 seconds	808	36	0.05600 seconds
r105	735	35	0.29000 seconds	634	27	0.08900 seconds
r106	870	41	0.35000 seconds	725	31	0.07700 seconds
r107	927	44	0.33000 seconds	765	34	0.05900 seconds
r108	982	47	0.32000 seconds	865	38	0.06100 seconds
r109	866	40	0.21000 seconds	763	33	0.09200 seconds
r110	870	42	0.20000 seconds	768	34	0.13200 seconds
r111	935	45	0.20000 seconds	805	35	0.08500 seconds
r112	939	44	0.31000 seconds	830	37	0.07300 seconds

	ΠΡΩΤΟΤΥΠΟΣ ILS			ILS ΜΕ ΑΥΣΤΗΡΑ ΚΡΙΤΗΡΙΑ ΕΠΙΣΚΕΨΗΣ		
Στιγμιότυπο Εισόδου	Συνολικό Όφελος	Επισκέψεις	Υπολογιστικός Χρόνος	Συνολικό Όφελος	Επισκέψεις	Υπολογιστικός Χρόνος
rc101	794	37	0.19000 seconds	598	23	0.04100 seconds
rc102	881	42	0.23000 seconds	656	27	0.04700 seconds
rc103	947	42	0.20000 seconds	777	30	0.05800 seconds
rc104	1019	43	0.17000 seconds	833	33	0.06300 seconds
rc105	841	37	0.15000 seconds	601	23	0.06900 seconds
rc106	874	37	0.25000 seconds	728	28	0.05900 seconds
rc107	951	42	0.19000 seconds	773	30	0.05000 seconds

rc108	998	43	0.20000 seconds	837	33	0.03800 seconds

Πίνακας 5 – Σύγκριση τροποποιημένου ILS με αυστηρή τήρηση των χρονικών παραθύρων και τροποποιημένου ILS με την δυνατότητα γεύματος για m = 1

	ILS ΜΕ ΑΥΣΤΗΡΑ ΚΡΙΤΗΡΙΑ ΕΠΙΣΚΕΨΗΣ			ΤΡΟΠΟΠΟΙΗΜΕΝΟΣ ILS ΜΕ ΕΠΙΛΟΓΗ ΕΣΤΙΑΤΟΡΙΟΥ		
Στιγμιότυπο Εισόδου	Συνολικό Όφελος	Επισκέψεις	Υπολογιστικός Χρόνος	Συνολικό Όφελος	Επισκέψεις	Υπολογιστικός Χρόνος
c101	300	10	0.06700 seconds	240	8	0.07100 seconds
c102	320	11	0.05000 seconds	240	9	0.03700 seconds
c103	380	11	0.03200 seconds	290	9	0.03100 seconds
c104	390	11	0.02400 seconds	340	10	0.02500 seconds
c105	310	9	0.03400 seconds	240	8	0.03500 seconds
c106	310	10	0.02800 seconds	230	8	0.03500 seconds
c107	320	10	0.03300 seconds	240	9	0.03500 seconds
c108	320	10	0.02500 seconds	250	9	0.02300 seconds
c109	340	11	0.02900 seconds	290	10	0.03000 seconds
r101	186	8	0.06900 seconds	157	7	0.01900 seconds
r102	247	10	0.03100 seconds	218	9	0.02200 seconds
r103	252	10	0.03600 seconds	260	10	0.01600 seconds
r104	268	11	0.03200 seconds	274	11	0.01700 seconds
r105	215	9	0.03900 seconds	186	7	0.01900 seconds
r106	258	10	0.03700 seconds	214	9	0.01600 seconds

r107	243	11	0.03700 seconds	260	10	0.01600 seconds
r108	233	11	0.03400 seconds	274	11	0.01200 seconds
r109	264	11	0.05300 seconds	236	10	0.01000 seconds
r110	247	11	0.05500 seconds	224	9	0.01400 seconds
r111	276	11	0.05200 seconds	231	9	0.01600 seconds
r112	274	11	0.02900 seconds	263	11	0.01500 seconds

	ILS ΜΕ ΑΥΣΤΗΡΑ ΚΡΙΤΗΡΙΑ ΕΠΙΣΚΕΨΗΣ			ΤΡΟΠΟΠΟΙΗΜΕΝΟΣ ILS ΜΕ ΕΠΙΛΟΓΗ ΕΣΤΙΑΤΟΡΙΟΥ		
Στιγμιότυπο Εισόδου	Συνολικό Όφελος	Επισκέψεις	Υπολογιστικός Χρόνος	Συνολικό Όφελος	Επισκέψεις	Υπολογιστικός Χρόνος
rc101	203	8	0.01300 seconds	133	5	0.06200 seconds
rc102	245	10	0.00700 seconds	196	7	0.02200 seconds
rc103	245	10	0.01000 seconds	196	7	0.02500 seconds
rc104	240	10	0.00700 seconds	213	9	0.02200 seconds
rc105	162	7	0.00700 seconds	84	4	0.01700 seconds
rc106	200	8	0.01000 seconds	150	6	0.01700 seconds
rc107	240	10	0.01100 seconds	170	7	0.02600 seconds
rc108	240	10	0.01300 seconds	210	9	0.02600 seconds

Πίνακας 6 – Σύγκριση τροποποιημένου ILS με αυστηρή τήρηση των χρονικών παραθύρων και τροποποιημένου ILS με την δυνατότητα γεύματος για m = 2

	ILS ΜΕ ΑΥΣΤΗΡΑ ΚΡΙΤΗΡΙΑ ΕΠΙΣΚΕΨΗΣ			ΤΡΟΠΟΠΟΙΗΜΕΝΟΣ ILS ΜΕ ΕΠΙΛΟΓΗ ΕΣΤΙΑΤΟΡΙΟΥ		
Στιγμιότυπο Εισόδου	Συνολικό Όφελος	Επισκέψεις	Υπολογιστικός Χρόνος	Συνολικό Όφελος	Επισκέψεις	Υπολογιστικός Χρόνος
c101	540	18	0.10300 seconds	420	15	0.07500 seconds
c102	610	21	0.19100 seconds	430	17	0.07600 seconds
c103	660	22	0.07400 seconds	590	19	0.09200 seconds
c104	700	22	0.10800 seconds	570	20	0.04600 seconds
c105	540	18	0.04800 seconds	430	15	0.04500 seconds
c106	550	18	0.04300 seconds	410	15	0.03300 seconds
c107	570	18	0.04600 seconds	430	15	0.02400 seconds
c108	570	19	0.04600 seconds	480	17	0.03300 seconds
c109	640	20	0.05600 seconds	600	20	0.03800 seconds
r101	307	13	0.03100 seconds	265	12	0.03000 seconds
r102	455	18	0.06400 seconds	393	15	0.03400 seconds
r103	450	19	0.05700 seconds	430	17	0.02800 seconds
r104	483	21	0.04800 seconds	452	19	0.04400 seconds
r105	369	16	0.05800 seconds	329	14	0.03900 seconds
r106	438	19	0.05600 seconds	325	14	0.03000 seconds
r107	483	20	0.09300 seconds	425	18	0.03300 seconds
r108	486	21	0.11600 seconds	462	19	0.03200 seconds
r109	412	18	0.04200 seconds	414	17	0.02900 seconds
r110	435	19	0.06200 seconds	454	19	0.02800 seconds
r111	471	20	0.05000 seconds	407	17	0.03200 seconds
r112	450	20	0.04000 seconds	454	19	0.03700 seconds

	ILS ΜΕ ΑΥΣΤΗΡΑ ΚΡΙΤΗΡΙΑ ΕΠΙΣΚΕΨΗΣ			ΤΡΟΠΟΠΟΙΗΜΕΝΟΣ ILS ΜΕ ΕΠΙΛΟΓΗ ΕΣΤΙΑΤΟΡΙΟΥ		
Στιγμιότυπο Εισόδου	Συνολικό Όφελος	Επισκέψεις	Υπολογιστικός Χρόνος	Συνολικό Όφελος	Επισκέψεις	Υπολογιστικός Χρόνος
rc101	293	11	0.04100 seconds	209	8	0.05700 seconds
rc102	326	14	0.01900 seconds	275	11	0.05800 seconds
rc103	394	16	0.03100 seconds	256	11	0.05100 seconds
rc104	459	18	0.02300 seconds	425	17	0.04000 seconds
rc105	289	12	0.02200 seconds	199	8	0.05200 seconds
rc106	377	15	0.05000 seconds	285	11	0.03400 seconds
rc107	436	18	0.02700 seconds	282	11	0.03800 seconds
rc108	464	19	0.03100 seconds	337	14	0.03700 seconds

Πίνακας 7 – Σύγκριση τροποποιημένου ILS με αυστηρή τήρηση των χρονικών παραθύρων και τροποποιημένου ILS με την δυνατότητα γεύματος για m = 3

	ILS ΜΕ ΑΥΣΤΗΡΑ ΚΡΙΤΗΡΙΑ ΕΠΙΣΚΕΨΗΣ			ΤΡΟΠΟΠΟΙΗΜΕΝΟΣ ILS ΜΕ ΕΠΙΛΟΓΗ ΕΣΤΙΑΤΟΡΙΟΥ		
Στιγμιότυπο Εισόδου	Συνολικό Όφελος	Επισκέψεις	Υπολογιστικός Χρόνος	Συνολικό Όφελος	Επισκέψεις	Υπολογιστικός Χρόνος
c101	730	25	0.09200 seconds	560	20	0.05800 seconds
c102	820	30	0.09700 seconds	660	24	0.06900 seconds
c103	920	32	0.07500 seconds	750	27	0.06500 seconds
c104	950	33	0.07200 seconds	790	29	0.05700 seconds
c105	780	26	0.06200 seconds	560	20	0.04900 seconds
c106	780	26	0.05800 seconds	550	21	0.05900 seconds
c107	760	25	0.13600 seconds	610	22	0.05300 seconds
c108	820	27	0.08400 seconds	700	24	0.07600 seconds

c109	860	28	0.10000 seconds	680	24	0.06700 seconds

r101	415	18	0.04900 seconds	334	15	0.05200 seconds
r102	606	25	0.08000 seconds	508	20	0.04300 seconds
r103	630	27	0.06400 seconds	520	22	0.04100 seconds
r104	662	29	0.06000 seconds	616	26	0.03000 seconds
r105	519	22	0.05500 seconds	445	20	0.06400 seconds
r106	604	26	0.06700 seconds	479	21	0.08900 seconds
r107	611	26	0.09800 seconds	624	27	0.06400 seconds
r108	691	31	0.09500 seconds	673	29	0.04800 seconds
r109	580	24	0.09900 seconds	572	24	0.04200 seconds
r110	618	26	0.06500 seconds	634	26	0.03900 seconds
r111	654	28	0.06000 seconds	551	24	0.03400 seconds
r112	665	29	0.07000 seconds	579	24	0.03300 seconds

	ILS ΜΕ ΑΥΣΤΗΡΑ ΚΡΙΤΗΡΙΑ ΕΠΙΣΚΕΨΗΣ			ΤΡΟΠΟΠΟΙΗΜΕΝΟΣ ILS ΜΕ ΕΠΙΛΟΓΗ ΕΣΤΙΑΤΟΡΙΟΥ		
Στιγμιότυπο Εισόδου	Συνολικό Όφελος	Επισκέψεις	Υπολογιστικός Χρόνος	Συνολικό Όφελος	Επισκέψεις	Υπολογιστικός Χρόνος
rc101	448	17	0.03400 seconds	318	13	0.05800 seconds
rc102	486	20	0.03600 seconds	426	17	0.04500 seconds
rc103	588	22	0.04700 seconds	422	16	0.03800 seconds
rc104	690	27	0.04500 seconds	560	22	0.03100 seconds
rc105	422	17	0.04200 seconds	371	14	0.04500 seconds
rc106	546	21	0.04500 seconds	401	15	0.03900 seconds

rc107	602	24	0.03700 seconds	408	15	0.03700 seconds
rc108	648	26	0.04700 seconds	498	20	0.02700 seconds

Πίνακας 8 – Σύγκριση τροποποιημένου ILS με αυστηρή τήρηση των χρονικών παραθύρων και τροποποιημένου ILS με την δυνατότητα γεύματος για m = 4

	ILS ΜΕ ΑΥΣΤΗΡΑ ΚΡΙΤΗΡΙΑ ΕΠΙΣΚΕΨΗΣ			ΤΡΟΠΟΠΟΙΗΜΕΝΟΣ ILS ΜΕ ΕΠΙΛΟΓΗ ΕΣΤΙΑΤΟΡΙΟΥ		
Στιγμιότυπο Εισόδου	Συνολικό Όφελος	Επισκέψεις	Υπολογιστικός Χρόνος	Συνολικό Όφελος	Επισκέψεις	Υπολογιστικός Χρόνος
c101	910	31	0.07100 seconds	660	25	0.05900 seconds
c102	980	35	0.13300 seconds	1020	38	0.08600 seconds
c103	1080	39	0.06500 seconds	1020	36	0.09500 seconds
c104	1160	42	0.05700 seconds	970	37	0.04400 seconds
c105	950	33	0.08000 seconds	700	26	0.06900 seconds
c106	940	32	0.08600 seconds	690	26	0.07300 seconds
c107	950	33	0.09800 seconds	720	27	0.05000 seconds
108	970	34	0.10000 seconds	840	31	0.05500 seconds
c109	1030	36	0.09800 seconds	920	34	0.10700 seconds

r101	507	22	0.08500 seconds	382	17	0.06600 seconds
r102	687	29	0.09100 seconds	587	24	0.09200 seconds
r103	782	33	0.10600 seconds	725	30	0.08700 seconds
r104	808	36	0.05600 seconds	750	32	0.07300 seconds
r105	634	27	0.08900 seconds	568	25	0.04900 seconds

r106	725	31	0.07700 seconds	633	28	0.04700 seconds
r107	765	34	0.05900 seconds	754	33	0.04400 seconds
r108	865	38	0.06100 seconds	752	33	0.07100 seconds
r109	763	33	0.09200 seconds	705	30	0.05500 seconds
r110	768	34	0.13200 seconds	689	30	0.07000 seconds
r111	805	35	0.08500 seconds	706	31	0.05900 seconds
r112	830	37	0.07300 seconds	690	30	0.06200 seconds

	ILS ΜΕ ΑΥΣΤΗΡΑ ΚΡΙΤΗΡΙΑ ΕΠΙΣΚΕΨΗΣ			ΤΡΟΠΟΠΟΙΗΜΕΝΟΣ ILS ΜΕ ΕΠΙΛΟΓΗ ΕΣΤΙΑΤΟΡΙΟΥ		
Στιγμιότυπο Εισόδου	Συνολικό Όφελος	Επισκέψεις	Υπολογιστικός Χρόνος	Συνολικό Όφελος	Επισκέψεις	Υπολογιστικός Χρόνος
rc101	598	23	0.04100 seconds	349	14	0.03200 seconds
rc102	656	27	0.04700 seconds	514	20	0.03100 seconds
rc103	777	30	0.05800 seconds	518	19	0.03900 seconds
rc104	833	33	0.06300 seconds	667	27	0.04900 seconds
rc105	601	23	0.06900 seconds	457	18	0.07100 seconds
rc106	728	28	0.05900 seconds	483	18	0.03100 seconds
rc107	773	30	0.05000 seconds	579	24	0.05200 seconds
rc108	837	33	0.03800 seconds	629	25	0.04600 seconds

Από τα αποτελέσματα των πινάκων 1-4 παρατηρούμε ότι ο τροποποιημένος αλγόριθμος ILS με τα αυστηρότερα κριτήρια επίσκεψης στις τοποθεσίες, παρουσιάζει μια μείωση στο όφελος και στο πλήθος των επισκέψιμων τοποθεσιών. Αυτό ήταν κάτι που αναμέναμε, καθώς η αυστηρή τήρηση των χρονικών παραθύρων καθιστά δυσκολότερη την επίσκεψη μιας τοποθεσίας. Η μείωση αυτή παρουσιάζει μια απόκλιση 1-10% τόσο για το όφελος όσο και για τις τοποθεσίες, ποσοστό αρκετά μικρό σε σύγκριση με τον πρωτότυπο ILS.

Αρκετά μικρή μείωση οφέλους και τοποθεσιών, παρατηρούμε επίσης και στον τροποποιημένο μας αλγόριθμο ILS, όπου υπάρχει η δυνατότητα γεύματος. Τα αποτελέσματα που παράγει ο αλγόριθμος μας είναι αρκετά καλά, δεδομένου ότι έχουμε χωρίσει τις τοποθεσίες σε 2 κατηγορίες (τοποθεσίες επίσκεψης και τοποθεσίες γεύματος) και έχουμε μειώσει αισθητά το πλήθος των διαθέσιμων τοποθεσιών. Τα αυστηρά κριτήρια τήρησης των χρονικών παραθύρων έχουν εφαρμοστεί και στις 2 προτεινόμενες παραλλαγές (ILS με αυστηρά κριτήρια επίσκεψης και ILS με την δυνατότητα γεύματος) και η απόκλιση του οφέλους στους πίνακες 5-8 κυμαίνεται μεταξύ 5-20%.

Συνοψίζοντας, τα αποτελέσματα των πινάκων 1-8, με την προσθήκη των αυστηρών κριτηρίων που εφαρμόστηκαν και στις 2 περιπτώσεις των τροποποιημένων αλγορίθμων, είναι ικανοποιητικά για μικρό *m* (αριθμός διαδρομών) λόγω του πλήθους των τοποθεσιών. Όσο αυξάνεται η τιμή της παραμέτρου *m,* αυξάνεται και η απόκλιση του συνολικού οφέλους και μειώνεται η απόδοση των αλγορίθμων μας.

Κεφάλαιο 6°

6.1 Συμπεράσματα

Στην παρούσα εργασία παρουσιάσαμε το πρόβλημα του προσανατολισμού (OP), του ομαδικού προσανατολισμού (TOP) και του ομαδικού προσανατολισμού με Χρονικά Παράθυρα (TOPTW). Έγινε μια επισκόπηση της περιοχής και παρουσιάσαμε αλγόριθμους που προτάθηκαν για την επίλυση τους (GRASP, ILS, GRASP-ELS).

Στηριζόμενοι στον αλγόριθμο ILS για το TOPTW, υλοποιήσαμε κάποιες τροποποιήσεις και εισηγάγαμε δύο περιορισμούς, κάνοντας τον αλγόριθμο αυστηρότερο στα κριτήρια επίσκεψης μιας τοποθεσίας.

Η πρώτη τροποποίηση αφορούσε την αυστηρότερη τήρηση των χρονικών παραθύρων που έχει η τοποθεσία, εμποδίζοντας τον επισκέπτη να παραμένει σε αυτή μετά το πέρας του χρονικού της παραθύρου. Θα πρέπει δηλαδή η επίσκεψη να γίνει μέσα στα πλαίσιο του χρονικού ορίου της τοποθεσίας χωρίς να το υπερβεί, αλλιώς θα θεωρηθεί ανέφικτη και δεν θα μπορεί να εισαχθεί στην διαδρομή.

Η δεύτερη και μεγαλύτερη τροποποίηση αφορούσε την εισαγωγή τοποθεσιών γεύματος στον αλγόριθμο μας. Ταυτόχρονα διατηρούμε τα αυστηρά κριτήρια επίσκεψης στις τοποθεσίες, όπως και στα εστιατόρια. Και οι δύο αλλαγές στον αλγόριθμο έχουν σαν σκοπό να εφαρμόσουν κάποια ρεαλιστικά κριτήρια στον σχεδιασμό τουριστικών διαδρομών.

Όσον αφορά, την απόδοση του αλγορίθμου μας, τα αποτελέσματα είναι αρκετά κοντά σε αυτά του πρωτότυπου ILS, παρά τους αυστηρούς περιορισμούς που προστέθηκαν.

Τέλος, ο αλγόριθμος θα μπορούσε να τροποποιηθεί περαιτέρω, ώστε να υλοποιηθούν και άλλα κριτήρια που έχουν πρακτική εφαρμογή στον σχεδιασμό των τουριστικών διαδρομών, όπως η δυνατότητα ο επισκέπτης να μπορεί να βαθμολογήσει μια επίσκεψη που πραγματοποίησε σε μια τοποθεσία και ο αλγόριθμος να ξανατρέχει ανάλογα με τα κριτήρια του εκάστοτε τουρίστα αναπροσαρμόζοντας τις προτεινόμενες διαδρομές με βάση τα κριτήρια αρεσκείας του. Μια ακόμα ενδιαφέρουσα επέκταση είναι η δυνατότητα εισαγωγής ρεαλιστικών χρόνων μεταφοράς εκτιμώντας την κίνηση των δρόμων. Αντί λοιπόν, για την χρήση της ευκλείδειας απόστασης, ο αλγόριθμος θα ενημερώνεται με τα δεδομένα της κίνησης στον δρόμο και θα φροντίζει ώστε ο επισκέπτης να τηρεί το αρχικό του πρόγραμμα που σχεδιάστηκε, προτείνοντας δρόμους με μικρότερη κυκλοφορία.

Βιβλιογραφία

[1] Applegate, D. L.; Bixby, R. M.; Chvátal, V.; Cook, W. J., The Traveling Salesman Problem, ISBN 978-0-691-12993-8, 2006.

[2] T. Tsiligirides. Heuristic methods applied to orienteering. The Journal of the Operational Research Society, 35(9): pp. 797–809, 1984.

[3] G. Laporte and S. Martello. The selective travelling salesman problem. Discrete Applied Mathematics, 26(2-3): pp. 193 – 207, 1990.

[4] S. Kataoka and S. Morito. An algorithm for single constraint maximum collection problem. Journal of the Operations Research Society of Japan, 31(4): pp. 515–530, 1988.

[5] Arkin, E., Mitchell, J., Narasimhan, G., Resource-constrained geometric network optimisation. In: Proc. 14th ACM Symposium on Computational Geometry, June, pp. 307–316, 1998

[6] Feillet, D., Dejax, P., Gendreau, M., Travelling salesman problems with profits. Transportation Science 39,pp. 188–205, 2005a.

[7] Laporte, G., Rodriguez Martin, I., Locating a cycle in a transportation or a telecommunications network. Networks,pp. 92–108, 2007.

[8] Pieter Vansteenwegen, Wouter Souffriau, Dirk Van Oudheusden, The orienteering problem: A survey, European Journal of Operational Research, ISSN 0377-2217, Volume 209, Issue 1, pp. 1-10, 2011.

[9] A. Gunawan, Hoong Chuin Lau, P. Vansteenwegen, Orienteering Problem: A survey of recent variants, solution approaches and applications, European Journal of Operational Research, ISSN 0377-2217, Volume 255, Issue 2, pp. 315-332, 2016.

[10] I-M. Chao, B. L. Golden, and E. A. Wasil. The team orienteering problem. European Journal of Operational Research, 88(3):pp. 464 – 474, 1996.

[11] P. Vansteenwegen. Planning in Tourism and Public Transportation – Attraction Selection by Means of a Personalised Electronic Tourist Guide and Train Transfer Scheduling. PhD thesis, Katholieke Universiteit Leuven, 2008.

[12] P. Vansteenwegen and D. Van Oudheusden. The mobile tourist guide: An OR opportunity. Operational Research Insight, 20(3): pp. 21–27, 2007.

[13] Miller, C., Tucker, A., Zemlin, R., 1960. Integer programming formulations and travelling salesman problems. Journal of the ACM 7, 326–329, 1960.

[14] Golden, B., Levy, L., Vohra, R., 1987. The orienteering problem. Naval Research Logistics 34, 307–318, 1987.

[15] Gendreau, M., Laporte, G., Semet, F., 1998a. A tabu search heuristic for the undirected selective travelling salesman problem. European Journal of Operational Research 106, 539–545, 1998.

[16] N. Bansal, A. Blum, S. Chawla, and A. Meyerson. Approximation algorithms for deadline-tsp and vehicle routing with time-windows. In Proceedings of the 36th Annual ACM Symposium on Theory of Computing, STOC '04, pages 166–174, 2004.

[17] V. Nagarajan and R. Ravi. The directed orienteering problem. Algorithmica, 1017–1030, 2011.

[18] K. Chen and S. Har-Peled. The orienteering problem in the plane revisited. In Proceedings of the 22nd Annual Symposium on Computational Geometry, SCG '06, pages 247–254, 2006.

[19] M.R. Garey and D.S. Johnson. Computers and Intractability: A Guide to the Theory of NP-Completeness . W.H. Freeman and Company, San Francisco, 1979

[20] Papadimitriou, C. H. Computational Complexity. Reading, MA: Addison-Wesley. 1994

[21] Laporte G, Martello S. The selective travelling salesman problem. Discrete Applied Mathematics ;26:193–207. 1990

[22] Fischetti et al. Solving the Orienteering Problem through Branch-and-Cut. Informs Journal on Computing 10(2):133-148. May 1998

[23] Golden, B., Levy, L., Vohra, R., 1987. The orienteering problem. Naval Research Logistics 34, 307–318

[24] Ramesh, R., Brown, K., 1991. An efficient four-phase heuristic for the generalized orienteering problem. Computers and Operations Research 18, 151–165

[25] Wang, Q., Sun, X., Golden, B., 1996. Using artificial neural networks to solve generalized orienteering problems. In: Dagli, C., Akay, M., Chen, C., Fernández, B., Ghosh, J. (Eds.), , Intelligent Engineering Systems Through Artificial Neural Networks, vol. 6. ASME Press, New York, pp. 063–1068.

[26] Gendreau, M., Laporte, G., Semet, F., 1998a. A tabu search heuristic for the undirected selective travelling salesman problem. European Journal of Operational Research 106, 539 – 545

[27] Schilde, M., Doerner, K., Hartl, R., Kiechle, G., 2009. Metaheuristics for the biobjective orienteering problem. Swarm Intelligence 3, 179–201

[28] Chao, I., Golden, B., Wasil, E. Theory and methodology – the team orienteering problem. European Journal of Operational Research 88, 464–474. 1996a

[29] Butt, S., Cavalier, T. A heuristic for the multiple tour maximum collection problem. Computers and Operations Research 21, 101–111. 1994

[30] T.A. Feo and M.G.C. Resende. A probabilistic heuristic for a computationally difficult set cover problem. Operations Research Letters , 8:67-71, 1989

[31] F. Glover. Interfaces in Computer Science and Operations Research, chapter Tabu search and adaptive memory programming, pages 1–75. Kluwer, 1996

[32] M. Laguna and R. Marti. Grasp and path relinking for 2-layer straight line crossing minimization. INFORMS Journal on Computing, 11:44–52, 1999

[33] Boussier, S., Feillet, D., Gendreau, M., 2007. An exact algorithm for the team orienteering problem. 4OR 5, 211–230.

[34] Righini, G., Salani, M., 2009. Decremental state space relaxation strategies and initialization heuristics for solving the orienteering problem with time windows with dynamic programming. Computers & Operations Research 4, 1191–1203

[35] Montemanni, R., Gambardella, L., 2009. Ant colony system for team orienteering problems with time windows. Foundations of computing and Decision Sciences 34 (4), 287–306.

[36] Vansteenwegen, P., Souffriau, W., Vanden Berghe, G., Van Oudheusden, D., 2009d. Iterated local search for the team orienteering problem with time windows. Computers and Operations Research 36 (12), 3281–3290.

[37] Gavalas, D. , Konstantopoulos, C. , Mastakas, K. , Pantziou, G. , & Tasoulas, Y. (2013). Cluster-based heuristics for the team orienteering problem with time windows. In V. Bonifaci, C. Demetrescu, & A. Marchetti-Spaccamela (Eds.), Experimental algorihtms. Lecture Notes in Computer Science: 7933 (pp. 390–401). Springer

[38] Labadie, N., Melechovsky, J., Wolfler Calvo , R.: Hybridized evolutionary local search algorithm for the team orienteering problem with time windows. J. Heuristics 17,729-753 (2011)

[39] Merz, P., Wolf, S.: Evolutionary local search for the super-peer selection problem and the p-hub median problem. In: Bartz-Beielstein, T., et al. (ed.) Lecture notes in computer science, vol. 4771, pp. 1–15 Springer, Berlin (2007)

[40] Prins, C.: A grasp x evolutionary local search hybrid for the vehicle routing problem. In: Pereira, F., Tavares, J. (eds.) Bio-inspired algorithms for the vehicle routing problem, vol. 16, pp. 35–53. Springer, Berlin (2009)

[41] Kindervater, G.A.P., Savelsbergh, M.W.P.: Vehicle routing: handling edge exchanges. In: Aarts, E., Lenstra,J.(eds.) Local Search in Combinatorial Optimization, pp.311 336.Wiley, NewYork (1997)

[42] Butt, S., Ryan, D., 1999. An optimal solution procedure for the multiple tour maximum collection problem using column generation. Computers and Operations Research 26, 427–441

[43] Solomon M. Algorithms for the vehicle routing and scheduling problem with time window constraints. Operations Research 1987;35:254–65.

[44] Damianos Gavalas, Charalampos Konstantopoulos, Konstantinos Mastakas, Grammati E. Pantziou, Nikolaos Vathis: Heuristics for the time dependent team orienteering problem: Application to tourist route

[45] Souffriau, W., Vansteenwegen, P., VandenBerghe, G. , VanOudheusden, G. .A path relinking approach for the team orienteering problem. Computers and Operations Research, doi: 10.1016/j.cor.2009.05.002.