ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ – ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Πρόγραμμα Μεταπτυχιακών Σπουδών
«Προηγμένα Συστήματα Πληροφορικής»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	**Διαχωρισμένη Τοπική Αναζήτηση για το Πρόβλημα Ομαδικού Προσανατολισμού με Χρονικά Παράθυρα** Partitioned Local Search for the Team Orienteering Problem with Time Window**s**
Ονοματεπώνυμο Φοιτητή	**Ευστάθιος Καψιώτης**
Πατρώνυμο	**Ηλίας**
Αριθμός Μητρώου	**ΜΠΣΠ17030**
Επιβλέπων	**Χαράλαμπος Κωνσταντόπουλος**, **Καθηγητής**

Ημερομηνία Παράδοσης **Δεκέμβριος 2022**

Τριμελής Εξεταστική Επιτροπή

Καθηγητής Χαράλαμπος Κωνσταντόπουλος	Επικ. Καθηγητής Ιωάννης Βενέτης	Επικ. Καθηγητής Ιωάννης Τασούλας

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον καθηγητή μου, κ. Χαράλαμπο Κωνσταντόπουλο, για όλη την καθοδήγηση που μου παρείχε κατά την εκπόνηση της μεταπτυχιακής μου διατριβής.

Επίσης θα ήθελα να ευχαριστήσω τους γονείς και τους φίλους μου για την υποστήριξη και την υπομονή τους όλα αυτά τα χρόνια.

Περίληψη

Η τρέχουσα εργασία μελετά εκτενώς το Πρόβλημα Ομαδικού Προσανατολισμού με Χρονικά Παράθυρα, το οποίο αποτελεί επέκταση του Προβλήματος Προσανατολισμού. Το Πρόβλημα Προσανατολισμού ανήκει στα NP-hard προβλήματα, κάτι που το κάνει αδύνατο να λυθεί σε πολυωνυμικό χρόνο για μεγάλα δεδομένα εισόδου. Για το λόγο αυτό, η χρήση ευρετικών και προσεγγιστικών αλγορίθμων καθίσταται αναγκαία για την εύρεση ικανοποιητικών λύσεων σε μικρό χρονικό διάστημα. Για το Πρόβλημα Προσανατολισμού έχουν ήδη υλοποιηθεί αρκετοί αλγόριθμοι, ένας από τους οποίους είναι και ο αλγόριθμος Επαναλαμβανόμενης Τοπικής Αναζήτησης (ILS). Σκοπός της παρούσας εργασίας είναι να μειώσει το χρόνο εκτέλεσης του ILS, διαχωρίζοντας το γράφημα του προβλήματος με ικανοποιητικό τρόπο, εφαρμόζοντας μια διαχωρισμένη Τοπική Αναζήτηση στα επιμέρους υπο-γραφήματα, και αντιμετωπίζοντας τα προβλήματα που προκύπτουν από αυτόν τον διαχωρισμό. Τα πειραματικά αποτελέσματα δείχνουν τη μείωση του χρόνου εκτέλεσης του αλγορίθμου ειδικά σε παραδείγματα με μεγάλα δεδομένα εισόδου, με αντίκτυπο όμως τη μείωση της βαθμολογίας των λύσεων. Παρ’ όλα αυτά, έχουν γίνει βήματα για τη διατήρηση των λύσεων σε ικανοποιητικό επίπεδο, ενώ υπάρχουν και περαιτέρω περιθώρια βελτίωσης.

Abstract

The current work extensively studies the Team Orienteering Problem with Time Windows, which is an extension of the Orienteering Problem. The Orienteering Problem belongs to NP-hard problems, which makes it impossible to solve in polynomial time for large input data. For this reason, the use of heuristic and approximate algorithms becomes necessary to find satisfactory solutions in a short period of time. Several algorithms have already been implemented for the Orienteering Problem, one of which is the Iterated Local Search (ILS) algorithm. The purpose of this paper is to reduce the execution time of ILS by partitioning the problem graph in a satisfactory way, applying a partitioned Local Search to the individual sub-graphs, and dealing with the problems arising from this partitioning. The experimental results show the reduction of the execution time of the algorithm especially in examples with large input data, but with the impact of the reduction of the score of the solutions. Nevertheless, steps have been taken to maintain the solutions at a satisfactory level, while there is also room for further improvement.

Περιεχόμενα

1. Εισαγωγή

Το Πρόβλημα Προσανατολισμού (Orienteering Problem, OP) αναφέρθηκε για πρώτη φορά από τον Tsiligirides (1984) [1] και οφείλει το όνομα του στο άθλημα «orienteering» που πραγματοποιείται συνήθως σε ορεινές ή δασικές περιοχές. Οι αθλητές , χρησιμοποιώντας μια πυξίδα και ένα χάρτη, πρέπει να επισκεφτούν όσο το δυνατόν περισσότερα σημεία ενδιαφέροντος χωρίς να παραβιάζεται ένα προκαθορισμένο χρονικό παράθυρο. Σε κάθε σημείο ενδιαφέροντος αντιστοιχεί μία τιμή κέρδους και στόχος των συμμετεχόντων είναι να μεγιστοποιήσουν την τιμή αυτή. Το OP συναντάται στην βιβλιογραφία επίσης ως το Πρόβλημα του Επιλεκτικού Περιοδεύοντος Πωλητή (Selective Traveling Salesman Problem, Laporte & Martelo, 1990) [2] και σαν το Πρόβλημα της Μέγιστης Συλλογής (Maximum Collection Problem, Kataoka & Morito, 1988 [3]).

Ένα από τα σημαντικότερα πεδία εφαρμογής του OP είναι ο τουρισμός. Ένα πρόβλημα που συναντούν συχνά οι τουρίστες είναι πως δεν μπορούν να αποφασίσουν ποια αξιοθέατα πρέπει να επισκεφθούν, έτσι ώστε να γίνει πιο ευχάριστη η περιήγηση τους στην πόλη, περιοριζόμενοι πάντα από το χρόνο που διαθέτουν.

Για το λόγο αυτό, έχουν κατασκευαστεί προσωποποιημένοι ηλεκτρονικοί τουριστικοί οδηγοί (PETs) οι οποίοι χρησιμοποιούνται για την εξαγωγή τουριστικών διαδρομών δίνοντας πάντα έμφαση στις προτιμήσεις του εκάστοτε χρήστη. Οι βασικές λειτουργίες των PETs είναι τρεις (Carcia et al. , 2010 [4]):

- η δημιουργία μιας λίστας από POIs που πιθανώς να ενδιαφέρουν το χρήστη καθώς έχουν προκύψει από τις δικές του προτιμήσεις (recommendation)
- η κατασκευή διαδρομών εφαρμόζοντας κάποιον αλγόριθμο και χρησιμοποιώντας POIs από τη πρώτη λειτουργία (route generation)
- η δυνατότητα προσαρμογής των διαδρομών από το χρήστη (customization)

Οι λειτουργίες αυτές έχουν πρόσφατα ενσωματωθεί και στην λειτουργικότητα πολλών εφαρμογών και ιστοσελίδων. Το πρόβλημα που σχετίζεται με τη δεύτερη λειτουργία των PETs έχει οριστεί ως Πρόβλημα Σχεδίασης Τουριστικών Διαδρομών [5]. Οι πληροφορίες εισόδου σε ένα πρόβλημα Σχεδίασης Τουριστικών Διαδρομών είναι οι εξής:

- ένα σύνολο σημείων ενδιαφέροντος (POIs)
- οι χρόνοι ταξιδιού μεταξύ των σημείων ενδιαφέροντος
- το κέρδος του κάθε σημείου ενδιαφέροντος που έχει υπολογιστεί με βάση τις προτιμήσεις του χρήστη
- ο αριθμός των διαδρομών που πρέπει να κατασκευαστούν
- η προβλεπόμενη διάρκεια επίσκεψης του χρήστη σε ένα σημείο ενδιαφέροντος
- ο χρόνος που σκοπεύει να διαθέτει ο χρήστης καθημερινά για την όλη διαδικασία επίσκεψης των σημείων ενδιαφέροντος

Γίνεται λοιπόν εμφανές πόσο αποδοτικά το OP και οι επεκτάσεις του μπορούν να μοντελοποιήσουν το TTDP και τις πιο πολύπλοκες παραλλαγές του. Στη παρούσα εργασία, το TTDP μοντελοποιείται μέσω του Προβλήματος Ομαδικού Προσανατολισμού με Χρονικά Παράθυρα (TOPTW) το οποίο επεκτείνει το Πρόβλημα Ομαδικού ΠροσανατολισμούTOP,

προσθέτοντας χρονικά παράθυρα λειτουργίας σε κάθε κόμβο, ενώ το TOP με τη σειρά του επεκτείνει το OP σε πολλαπλές διαδρομές.

Σκοπός επίσης της παρούσας εργασίας, είναι να βελτιώσει τον αλγόριθμο Επαναλαμβανόμενης Τοπικής Αναζήτησης των Vansteenwegen et al. (2009) [6] διαχωρίζοντας το εκάστοτε γράφημα σε υπό-γραφήματα, βελτιώνοντας έτσι την ταχύτητα του αλγορίθμου αλλά και ενισχύοντας τη διερεύνηση διαφορετικών λύσεων.

Στο Κεφάλαιο 2 γίνεται ανασκόπηση της βιβλιογραφίας σχετικά με το πρόβλημα OP και μερικών επεκτάσεών του και αναφέρονται διάφορες αλγοριθμικές προσεγγίσεις που έχουν προταθεί. Στο Κεφάλαιο 3 παρουσιάζεται ο αλγόριθμος ILS (Vansteenwegen et al. 2009) [6] που επιλέχθηκε και υλοποιήθηκε για την επίλυση περιπτώσεων του προβλήματος TOPTW καθώς γίνεται και μια πιο λεπτομερής αναφορά στα επιμέρους συστατικά του ILS. Στο Κεφάλαιο 4, αναλύεται η διαδικασία διαχωρισμού του εκάστοτε προβλήματος και η διαχωρισμένη Τοπική Αναζήτηση που αναπτύχθηκε για τη βελτίωση του αλγορίθμου. Στο Κεφάλαιο 5 παρουσιάζονται τα πειραματικά αποτελέσματα του τροποποιημένου αλγορίθμου για διάφορες περιπτώσεις του TOPTW αλλά και για ένα πιο ρεαλιστικό παράδειγμα με φόντο την περιοχή της Αθήνας. Τέλος στο Κεφάλαιο 6 παρουσιάζονται μερικά συμπεράσματα που προέκυψαν από την τροποποίηση του ILS ενώ αναφέρονται και μερικές προσθήκες που θα μπορούσαν να βελτιώσουν περαιτέρω τις λύσεις.

2. Το Πρόβλημα Προσανατολισμού

Το πρόβλημα του προσανατολισμού μπορεί να αναπαρασταθεί ως εξής: Έστω το γράφημα G = (V,E) όπου σε κάθε ακμή του γραφήματος αντιστοιχεί μία τιμή κόστους και σε κάθε κόμβο μία τιμή κέρδους. Έχοντας καθορίσει έναν κόμβο ως αρχικό και έναν άλλον (ή τον ίδιο) ως τελικό, σκοπός είναι να βρεθεί μια διαδρομή από τον αρχικό προς τον τελικό κόμβο που να μεγιστοποιεί το κέρδος αλλά να μην ξεπερνάει το μέγιστο όριο χρόνου που έχει προκαθοριστεί. Επίσης το πρόβλημα του προσανατολισμού μπορεί να αναπαρασταθεί ως ένα μοντέλο ακέραιου προγραμματισμού (Vansteenwegen et al. 2011 [7]) χρησιμοποιώντας τις εξής μεταβλητές:

- N ο αριθμός των κόμβων (1, 2, ..., N) με αρχικό κόμβο s = 1 και τελικό t = N
- P_i η τιμή κέρδους της επίσκεψης στον κόμβο i
- C_{ij} το κόστος μετακίνησης από τον κόμβο i στον κόμβο j
- $x_{ij} = 1$ εάν η επίσκεψη στον κόμβο i ακολουθείται από την επίσκεψη στον κόμβο j, ειδάλλως $x_{ij} = 0$

Χρησιμοποιώντας λοιπόν τους παραπάνω συμβολισμούς προκύπτουν οι παρακάτω σχέσεις:

$$maximize \sum_{i=2}^{N-1} \sum_{j=2}^{N} S_i X_{ij} \quad (2.1)$$

$$\sum_{j=2}^{N} X_{1j} = \sum_{i=1}^{N-1} X_{iN} = 1 \quad (2.2)$$

$$\sum_{i=1}^{N-1} X_{ik} = \sum_{j=2}^{N} X_{kj} \leq 1 \; \forall k = 2, \cdots, (N-1) \quad (2.3)$$

$$\sum_{i=1}^{N-1} \sum_{j=2}^{N} t_{ij} X_{ij} \leq T_{max} \quad (2.4)$$

$$2 \leq u_i \leq N \; \forall i = 2, \cdots, N \quad (2.5)$$

$$u_i - u_j + 1 \leq (N-1)\big(1 - X_{ij}\big) \forall i = 2, \cdots, N \quad (2.6)$$

Η σχέση 2.1 αντιπροσωπεύει το στόχο που πρέπει να επιτευχθεί δηλαδή την μεγιστοποίηση της τιμής κέρδους. Η σχέση 2.2 διασφαλίζει πως η διαδρομή θα ξεκινάει από τον αρχικό κόμβο 1 και θα καταλήγει στον τελικό κόμβο N καθώς το πλήθος των ακμών που ανήκουν στη διαδρομή και ξεκινάνε από τον κόμβο 1 όπως και το πλήθος των ακμών που ανήκουν στη διαδρομή και καταλήγουν στον κόμβο N ισούται με 1. Η σχέση 3 διασφαλίζει τη συνεκτικότητα της διαδρομής καθώς και την μοναδικότητα της κάθε επίσκεψης καθώς δεν επιτρέπεται το πλήθος των ακμών που αρχίζουν από οποιοδήποτε κόμβο k να διαφέρει από το πλήθος τον ακμών που καταλήγουν στον κόμβο k ή να είναι μεγαλύτερο από 1. Η σχέση 4 περιορίζει το συνολικό χρόνο περιήγησης σε ένα καθορισμένο χρονικό όριο T_{max}. Οι σχέσεις 5 και 6 εμποδίζουν την ύπαρξη υπό-διαδρομών (Miller, Tucker, & Zermin 1960 [8]).

Στη βιβλιογραφία συναντώνται διαφορετικές εκδοχές του Προβλήματος Προσανατολισμού. Οι κυριότερες διαφοροποιήσεις είναι οι εξής:

- Το γράφημα μπορεί να είναι κατευθυνόμενο (directed OP) (Nagarajan and Ravi 2011 [9]) ή μη κατευθυνόμενο (Tsiligirides 1984 [1] , Bansal et al. 2004 [10])
- Έχει προκαθοριστεί ένας αρχικός κόμβος αλλά όχι ένας τελικός (rooted OP) (Arkin et al. 1998 [11], Chen and Har-Peled 2006 [12]). Το rooted OP αποτελεί ευκολότερο πρόβλημα από το κλασσικό OP
- Δεν έχει καθοριστεί ούτε αρχικός ούτε τελικός κόμβος (unrooted OP) (Gendreau et al. 1998 [13] [14])

Σε περίπτωση που το πλήθος κόμβων είναι μικρό, η βέλτιστη λύση είναι προσιτή σε λογικά πλαίσια χρόνου. Παρ' όλα αυτά επειδή σύμφωνα με τους Golden et al. (1987) [15], Laporte και Martello (1990) [2] το OP είναι NP-hard, είναι προφανής η ανάγκη εύρεσης προσεγγιστικών και ευρετικών αλγορίθμων για την επίλυση στιγμιότυπων με μεγάλο πλήθος κόμβων σε πολυωνυμικό χρόνο. Μερικοί αλγόριθμοι για την εύρεση της βέλτιστης λύσης βασίζονται σε τεχνικές branch-and-cut Genrau et al. (1998) [14], Fischetti et al. (1998) [16] και branch-and-bound Laporte and Martello (1990) [2], Ramesh et al. (1992) [17]. Πολλοί από τους προσεγγιστικούς αλγορίθμους που μπορεί να συναντήσει κανείς στη βιβλιογραφία είναι είτε δύσκολα υλοποιήσιμοι είτε απαιτούν παραπάνω χρόνο από το επιθυμητό.

Παρακάτω αναφέρονται μερικοί από τους ευρετικούς αλγόριθμους για την επίλυση του OP που μελετήθηκαν για την υλοποίηση της παρούσας εργασίας. Όπως έχει ήδη αναφερθεί το Πρόβλημα Προσανατολισμού αποτελεί το ευκολότερο μοντέλο του Προβλήματος Σχεδιασμού Τουριστικών Διαδρομών. Παρ΄ όλα αυτά συναντώνται και διαφορετικές εφαρμογές του OP.

Η πρώτη πρακτική εφαρμογή του OP αναφέρθηκε από τον Tsiligirides (1984) [1], όπου εξετάζεται η περίπτωση που ο πλανόδιος πωλητής δεν έχει αρκετό χρόνο για να επισκεφθεί όλες τις πόλεις. Γνωρίζοντας όμως το κέρδος που θα αποκομίσει σε κάθε πόλη, προσπαθεί να μεγιστοποιήσει το συνολικό κέρδος ενώ ταυτόχρονα να μην ξεπεράσει ένα καθορισμένο χρονικό όριο.

Μία άλλη εφαρμογή του OP είναι το Πρόβλημα Παράδοσης Καυσίμων (Fuel Delivery Problem, Golden et al. (1987) [15]), όπου ένα πλήθος φορτηγών πρέπει να εφοδιάζουν καθημερινά διάφορους πελάτες με καύσιμα. Κάθε πελάτης πρέπει να έχει στη διάθεση του συνεχώς παραπάνω από μία συγκεκριμένη ποσότητα καυσίμων. 'Ετσι λοιπόν, η ανάγκη καυσίμων μπορεί να θεωρηθεί ως το κέρδος στο Πρόβλημα Προσανατολισμού και ως στόχος η δημιουργία ενός πλάνου διαδρομών έτσι ώστε να εξυπηρετείται ένα υποσύνολο πελατών με τη μεγαλύτερη ανάγκη από καύσιμα. Παρακάτω, αναλύονται μερικές από τις βασικές επεκτάσεις του OP.

2.1 Το πρόβλημα Προσανατολισμού με Χρονικά Παράθυρα (OPTW)

Η διαφορά του OPTW με το OP είναι πως η επίσκεψη σε έναν κόμβο i μπορεί να πραγματοποιηθεί μόνο μέσα σε ένα προκαθορισμένο χρονικό παράθυρο του i. Τα χρονικά παράθυρα μπορεί να διαφέρουν μεταξύ των κόμβων.

Οι Duque et al.(2015) [18] πρότειναν έναν αλγόριθμο Pulse για την αντιμετώπιση του OPTW, ο οποίος αλγόριθμος χρησιμοποιείται γενικότερα για δύσκολα προβλήματα συντομότερης διαδρομής και χρησιμοποιήθηκε πρώτη φορά για το πρόβλημα Constrained Shortest Path από τους Lozano και Medaglia (2013) [19]. Ο αλγόριθμος πυροδοτώντας έναν παλμό από τον αρχικό κόμβο v_s και ωθώντας τον προς τον τελικό v_e, και αποθηκεύοντας το συσσωρευμένο κέρδος και το χρόνο, δημιουργεί μια τροχιά-λύση. Στη συνέχεια πραγματοποιώντας οπισθοδρόμηση και επαναλαμβάνοντας τη παραπάνω διαδικασία αναζητά καινούριες λύσεις. Επειδή ο αλγόριθμος επρόκειτο να εξετάσει ολόκληρο το πλήθος λύσεων, χρησιμοποιούνται 4 τεχνικές κλαδέματος για το περιορισμό του χώρου λύσεων, από τις οποίες οι 2 (Pruning by soft dominance, Pruning by detour) είναι προτεινόμενες για το συγκεκριμένο πρόβλημα από τους ίδιους τους συγγραφείς.

Οι Gunawan et al.(2015) [20] προτείνουν έναν αλγόριθμο Επαναλαμβανόμενης Τοπικής Αναζήτησης (ILS) παρόμοιο με τον ILS των Vansteenwegen et al. (2009) [6] που θα αναλυθεί περαιτέρω στο 3ο Κεφάλαιο. Ο αλγόριθμος των Gunawan et al.(2015) [20] στο βήμα Εισαγωγής εισάγουν τον κόμβο με τη μεγαλύτερη πιθανότητα εισαγωγής και όχι με το μεγαλύτερο ratio.Η πιθανότητα υπολογίζεται από τη σχέση prob{n,p} = ratio{n,p}/ X (i,j)∈F ratio{i,j} η οποία απεικονίζει πως η πιθανότητα εισαγωγής του κόμβου n στο σημείο p ισούται με το ratio του n στο σημείο αυτό δια το άθροισμα όλων των ratio όλων των κόμβων προς εισαγωγή σε όλες τις πιθανές θέσεις εισαγωγής. Επίσης ο ILS των Gunawan et al.(2015) [20] διαφέρει από τον ILS των Vansteenwegen et al.(2009) [6] σε μερικά σημεία στο βήμα Διαταραχής ενώ μάλιστα για τη παραγωγή λύσεων χρησιμοποιεί τεχνικές 2-opt, Insert, Swap, Replace. Τέλος το κριτήριο τερματισμού είναι διαφορετικό καθώς μετά από έναν αριθμό αποτυχημένων επαναλήψεων ο πρώτος αλγόριθμος δεν τερματίζεται αλλά επαναλαμβάνεται ξεκινώντας από τη βέλτιστη μέχρι εκείνη τη στιγμή λύση τερματίζοντας εν τέλει μετά από ένα προκαθορισμένο χρονικό διάστημα.

2.2 Το Χρονικά Εξαρτώμενο Πρόβλημα Προσανατολισμού (TDOP)

Οι Fomin and Lingas (2002) [21] ανέφεραν για πρώτη φορά το TDOP δηλώνοντας πως είναι NP-hard πρόβλημα επειδή και το OP είναι NP-hard. Το TDOP ανταποκρίνεται περισσότερο σε συνθήκες της πραγματικής ζωής καθώς σε πολλές περιπτώσεις ο χρόνος ταξιδιού από το σημείο A στο σημείο B δεν είναι σταθερός καθ' όλη τη διάρκεια της ημέρας όπως θεωρείται στο OP αλλά εξαρτάται από την ώρα αναχώρησης από το A. Η χρονική διάρκεια μιας ημέρας λοιπόν, μπορεί να χωριστεί σε περιόδους και στην κάθε ακμή, ανάλογα με την εκάστοτε χρονική περίοδο, μπορεί αντιστοιχεί ένας διαφορετικός χρόνος ταξιδιού. Για παράδειγμα ένας κεντρικός δρόμος συχνά έχει μεγαλύτερη συμφόρηση σε ώρες αιχμής ενώ ένας σχετικά πιο απόμερος δρόμος δεν επηρεάζεται, τουλάχιστον σημαντικά, κατά τη διάρκεια της ημέρας. Οι Verbeeck et al.(2014a) [22] περιέγραψαν ένα μοντέλο Μεικτού Ακέραιου Προγραμματισμού (MIP) με βάση το MIP για το OP (Vansteenwegen, Souffriau, & Van Oudheusden, (2011) [23]).

- $x_{i,j,t} = 1$ εάν ο χρήστης διατρέχει το τόξο i → j και η ώρα αναχώρησης από τον i είναι μέσα στο χρονικό διάστημα t, ειδάλλως $x_{i,j,t} = 0$
- $w_{i,j,t}$ η ώρα αναχώρησης στο χρονικό διάστημα t όταν διατρέχεται το i → j
- $\theta_{i,j,t}$ ο συντελεστής κλίσης του γραμμικού χρονικά εξαρτώμενου χρόνου ταξιδιού
- $\eta_{i,j,t}$ ο συντελεστής παρεμπόδισης του γραμμικού χρονικά εξαρτώμενου χρόνου ταξιδιού
- $\tau_{i,j,t}$ το κάτω όριο του χρονικού διαστήματος t για την ακμή i → j

- $T_{i,j}$ ο αριθμός των χρονικών διαστημάτων για την ακμή i → j
- S_i κέρδος του κόμβου i
- t_{max} ο μέγιστος συνολικός χρόνος ταξιδιού
- Ο αρχικός κόμβος είναι ο 1 και ο τελικός είναι ο N

$$maximize \sum_{i=2}^{N-1} \sum_{j=2}^{N} \sum_{t=1}^{T_{i,j}} S_i x_{i,j,t} \quad (2.7)$$

$$\sum_{j=2}^{N} x_{1,j,1} = \sum_{i=1}^{N-1} \sum_{t=1}^{T_{iN}} x_{i,N,t} = 1 \quad (2.8)$$

$$\sum_{i=1}^{N-1} \sum_{t=1}^{T_{i,h}} x_{i,h,t} = \sum_{j=2}^{N} \sum_{t=1}^{T_{h,j}} x_{h,j,t} \leq 1 \forall h = 2, \cdots, N-1 \quad (2.9)$$

$$\sum_{i=1}^{N-1} \sum_{t=1}^{T_{i,h}} \left[w_{i,h,t} + (\theta \cdot w_{i,h,t} + \eta_{i,h,t} \cdot x_{i,h,t}) \right] = \sum_{j=2}^{N} \sum_{t=1}^{T_{h,j}} w_{h,j,t} \; \forall h = 2, \cdots, N-1 \quad (2.10)$$

$$x_{i,j,t} \cdot \tau_{i,j,t} \leq w_{i,j,t} \leq x_{i,j,t} \cdot \tau_{i,j,t+1} \; i = 1, \cdots, N-1, j = 2, \cdots, N, \forall t \quad (2.11)$$

$$\sum_{i=1}^{N-1} \sum_{j=2}^{N} \sum_{t=1}^{T_{ij}} \left[\theta_{i,j,t} \cdot w_{i,j,t} + \eta_{i,j,t} \cdot x_{i,j,t} \right] \leq t_{max} \quad (2.12)$$

$$w_{1,i,1} = 0 \; \forall 1, \cdots, N \quad (2.13)$$

$$x_{i,j,t} \in (0,1); \; 0 \leq w_{i,j,t} \leq t_{max} \; \forall t, i, j = 1, \cdots, N \quad (2.14)$$

Η σχέση 2.7 αντιπροσωπεύει το στόχο μεγιστοποίησης του συνολικού κέρδους. Η σχέση 2.8 διασφαλίζει πως κάθε τροχιά, αρχίζει από τον κόμβο 1 και καταλήγει στον κόμβο N. Η σχέση 2.9 διασφαλίζει πως σε κάθε κόμβο θα πραγματοποιείται το πολύ μία επίσκεψη. Η σχέση 2.10 διαβεβαιώνει πως η ώρα αναχώρησης από έναν κόμβο j που έπεται από έναν κόμβο i ισούται με την ώρα αναχώρησης από τον i συν το χρόνο ταξιδιού της ακμής i → j. Για να ισχύει η σχέση αυτή, θεωρείται πως δεν υπάρχουν χρόνοι αναμονής. Η σχέσεις 2.11 και 2.12 κατηγοριοποιούν κάθε ώρα αναχώρησης σε χρονοθυρίδες (time slots) χρησιμοποιώντας τα αντίστοιχα της θ και η. Η σχέση 2.13 διασφαλίζει πως η τροχιά ξεκινάει στο πρώτο timeslot ενώ η 2.14 πως όλες οι ώρες αναχώρησης είναι μικρότερες ή ίσες του T_{max}. Μια χρονοθυρίδα δημιουργείται εάν κατά τη διάρκεια ενός ταξιδιού από έναν κόμβο i προς έναν κόμβο j αλλάξει ο χρόνος ταξιδιού του i → j. Στη περίπτωση αυτή, η χρονική στιγμή της αλλαγής αποθηκεύεται σαν το κάτω όριο της νέας χρονοθυρίδας $\tau_{i,j,t}$ μαζί με τους αντίστοιχους συντελεστές $\theta_{i,j,t}$ και $\eta_{i,j,t}$. Με βάση αυτούς τους συντελεστές, μπορεί να υπολογιστεί ο εκάστοτε χρόνος ταξιδιού

$$traveltime_{i,j,w_{i,j,t}} = \theta_{i,j,t} \cdot w_{i,j,t} + \eta_{i,j,t}$$

Επίσης με βάση τους χρόνους ταξιδιού και το σύνολο των χρονοθυρίδων ($\tau_{i,j,t}$) τα θ και η μπορούν να υπολογιστούν ως εξής:

$$\theta_{i,j,t} = \frac{traveltime_{i+1} - traveltime_t}{\tau_{i,j,t+1} - \tau_{i,j,t}}$$

$$\eta_{i,j,t} = traveltime_t - \theta_{i,j,t} \cdot \eta_{i,j,t}$$

Οι Verbeeck et al.(2014a) [22] επίσης πρότειναν για την επίλυση του TDOP έναν αλγόριθμο που συνδυάζει τον αλγόριθμο Ant Colony System (ACS) με τεχνικές εισαγωγής-τοπικής αναζήτησης και 2-opt. Ο ACS είναι ένας μεταευρετικός αλγόριθμος ο οποίος δημιουργεί διάφορες λύσεις και χρησιμοποιεί μία δομή, αποκαλούμενη ως «ίχνη φερομονών», στην οποία αποθηκεύει τα ίχνη της διαδικασίας, όπως τις καλύτερες ακμές της εκάστοτε βέλτιστης λύσης, έτσι ώστε σε κάθε επανάληψη να βελτιώνονται οι προκύπτουσες λύσεις. Η τεχνική της εισαγωγής-τοπικής αναζήτησης, λόγω της φύσης του προβλήματος, εξετάζει περιπτώσεις εισαγωγής κόμβων σε σημεία που θα μειώσουν τον συνολική χρονική ολίσθηση της τροχιάς. Πιο συγκεκριμένα σε περίπτωση που η ακμή A → C περιλαμβάνει κεντρικούς δρόμους με μεγάλη πιθανότητα συμφόρησης ενώ οι ακμές A → B και B → C περιλαμβάνουν πιο ερημικούς δρόμους με μικρότερη πιθανότητα συμφόρησης, ίσως σε ώρα αιχμής να ισχύει $TravelTime_{A\to B} + TravelTime_{B\to C} < TravelTime_{A\to C}$. Επίσης, η τεχνική 2-opt που εφαρμόζεται, δηλαδή η αντικατάσταση 2 ακμών, είναι τροποποιημένη, καθώς η κανονική, πιθανότατα θα οδηγούσε σε μια ανέφικτη τροχιά, λόγω της διαφοροποίησης των χρόνων ταξιδιού κατά τη διάρκεια της μέρας.

Οι Gunawan et al. (2014) [24] ανέπτυξαν το δικό τους μοντέλο Ακέραιου Γραμμικού Προγραμματισμού για το TDOP. Χρησιμοποιώντας τους παρακάτω συμβολισμούς:

- $X_{i,j,t} = 1$ εάν το ταξίδι από τον κόμβο i προς τον κόμβο j αρχίζει στη χρονική περίοδο t, ειδάλλως $X_{i,j,t} = 0$
- u_i η τιμή κέρδους του κόμβου i
- T_{max} ο συνολικός διαθέσιμος χρόνος
- $d_{i,j,t}$ ο χρόνος ταξιδιού της ακμής i → j που ξεκίνησε κατά τη χρονική περίοδο t

προκύπτουν οι σχέσεις

$$Maximize \sum_{i=1}^{n} \sum_{j=1, j\neq i}^{n} \sum_{t=1}^{T_{max}} u_i X_{i,j,t} \quad (2.15)$$

$$\sum_{i>1}^{n} \sum_{t=1}^{T_{max}} X_{i,1,t} = 0 \quad (2.16)$$

$$\sum_{j>1}^{n} \sum_{t=1}^{T_{max}} X_{1,j,t} = 1 \quad (2.17)$$

$$\sum_{j=1}^{n-1}\sum_{t=1}^{T_{max}} X_{n,j,t} = 0s \quad (2.18)$$

$$\sum_{i=1}^{n-1}\sum_{t=1}^{T_{max}} X_{i,n,t} = 1 \quad (2.19)$$

$$\sum_{i=1,i\neq e}^{n-1}\sum_{\mathrm{t}=1,T_{max}}^{n} X_{i,e,t} = \sum_{\mathrm{j}=2,\mathrm{j}\neq \mathrm{e}}^{n}\sum_{T=1}^{T_{max}} X_{e,j,t} \ \forall e = 2,3,\cdots,(n-1) \quad (2.20)$$

$$\sum_{j=2,j\neq i}^{n}\sum_{t=1}^{T_{max}} X_{i,j,t} \leq 1 \ \forall\ 2,3,\cdots,(n-1) \quad (2.21)$$

$$\sum_{e\neq i,j}\sum_{u=t+d_{i,j,t}}^{T_{max}} X_{j,e,u} \leq X_{i,j,t} \ \forall i,j = 1,\cdots,n-1, i \neq j, j \neq 1, t \leq T_{max} - d_{i,j,t} \quad (2.22)$$

$$X_{i,j,t} = 0 \ \forall i \neq j, t > \ T_{max} - d_{i,j,t} \quad (2.23)$$

Η σχέση 2.15 αντιπροσωπεύει το στόχο μεγιστοποίησης του κέρδους (ωφελιμότητας). Η σχέση 2.16 διασφαλίζει πως δεν θα υπάρχει τόξο που θα καταλήγει στον αρχικό κόμβο, ενώ η σχέση 2.17 πως ο κόμβος 1 είναι ο αρχικός κόμβος καθώς το πλήθος των τόξων που ξεκινούν από τον κόμβο 1 σε όλες τις χρονικές περιόδους t ισούται με 1. Οι σχέσεις 2.18 και 2.19, ορίζοντας πως το πλήθος των τόξων που αρχίζουν από τον κόμβο n σε όλες τις χρονικές περιόδους ισούται με 0 και πως το πλήθος των τόξων που καταλήγουν στον κόμβο n σε όλες τις χρονικές περιόδους ισούται με 1, διασφαλίζει πως ο n είναι ο τελικός κόμβος. Η σχέση 2.20, ορίζοντας πως το πλήθος των τόξων που καταλήγουν σε έναν κόμβο ισούται με το πλήθος των τόξων που αρχίζουν από αυτόν (είτε βρίσκεται στη τροχιά είτε όχι), διασφαλίζει την συνεκτικότητα της τροχιάς. Η σχέση 2.21 διασφαλίζει πως η επίσκεψη σε κάθε κόμβο πραγματοποιείται το πολύ μία φορά. Η σχέση 2.22 θέτει το περιορισμό πως εάν το ταξίδι i → j ξεκινάει κατά τη χρονική περίοδο t και ο j δεν είναι ο τελικός κόμβος, τότε το ταξίδι j → j + 1 πρέπει να αρχίζει μία χρονική περίοδο μετέπειτα από αυτήν της ώρα επίσκεψης στον j. Τέλος η σχέση 2.23 αποκλείει ταξίδια που καθυστερούν σημαντικά να ξεκινήσουν.

Επίσης, οι Gunawan et al. (2014) [24] πρότειναν έναν αλγόριθμο για την επίλυση του TDOP, βασισμένο σε περιβάλλον θεματικού πάρκου όπου οι επισκέπτες κάθε φορά χρειάζονται άμεσα μία λύση-διαδρομή. Ο αλγόριθμος τους αποτελείται από έναν Άπληστο Κατασκευαστικό ευρετικό αλγόριθμο, από τεχνικές Εισαγωγής, Αντικατάστασης, καθώς και από 2 τεχνικές ILS (Basic,Adaptive). Ο Άπληστος Κατασκευαστικός αλγόριθμος κατασκευάζει μία λύση εισάγοντας κόμβους που πληρούν ορισμένα κριτήρια. Οι τεχνικές Εισαγωγής, Αντικατάστασης εφαρμόζονται αφότου ενημερωθεί η νωρίτερη ώρα άφιξης (t_i^{arr}) και η αργότερη ώρα αναχώρησης (t_i^{dep}) κάθε κόμβου i μέσα στην εκάστοτε τροχιά. Επίσης εξετάζεται η υβριδοποίηση των 2 αυτών τεχνικών (Variable Neighborhood Descent). Τέλος η κύρια διαφορά του βασικού ILS από τον προσαρμόσιμο ILS είναι ο τρόπος με τον οποίο αντιμετωπίζεται η στασιμότητα των παραγόμενων λύσεων.

2.3 Το πρόβλημα Ομαδικού Προσανατολισμού (TOP)

Το πρόβλημα προσανατολισμού επεκτείνεται στο πρόβλημα Ομαδικού Προσανατολισμού (Team Orienteering Problem) στο οποίο στόχος πλέον αποτελεί η εύρεση πολλαπλών διαδρομών από τον αρχικό κόμβο προς τον τελικό επιδιώκοντας πάντα τη μεγιστοποίηση του κέρδους. Τονίζεται πως η ύπαρξη πολλαπλών διαδρομών δεν αναιρεί τους κανόνες πως η κάθε επίσκεψη σε ένα κόμβο πρέπει να είναι μοναδική και το πως κάθε διαδρομή θα πρέπει να μην υπερβαίνει ένα χρονικό όριο. Το μοντέλο των πολλαπλών διαδρομών μπορεί να εφαρμοσθεί εύκολα και στο πρόβλημα Σχεδιασμού Τουριστικών Διαδρομών (TTDP) καθώς κάθε διαδρομή μπορεί να αντιστοιχιστεί σε μία μέρα περιήγησης του τουρίστα. Στη βιβλιογραφία συναντώνται διάφορες παραλλαγές του TOP που μοντελοποιούν διαφορετικές εκδοχές του TTDP.

Το πρόβλημα Ομαδικού Προσανατολισμού μπορεί να αναπαρασταθεί ως πρόβλημα ακέραιου προγραμματισμού (Vansteenwegen et al. 2011 [7]) ως εξής:

- k ο αριθμός των διαφορετικών διαδρομών
- $x_{ijm} = 1$ εάν η επίσκεψη στον κόμβο i ακολουθείται από την επίσκεψη στον κόμβο j στη διαδρομή m ειδάλλως $x_{ijm} = 0$
- $y_{im} = 1$ εάν πραγματοποιείται επίσκεψη στον κόμβο i στη διαδρομή m, ειδάλλως $y_{im} = 0$
- u_{im} η θέση της επίσκεψης στον κόμβο i στη διαδρομή m

Χρησιμοποιώντας τους παραπάνω συμβολισμούς προκύπτουν οι παρακάτω σχέσεις:

$$maximize \sum_{m=1}^{k} \sum_{i=2}^{N-1} p_i y_{im} \tag{2.24}$$

$$\sum_{m=1}^{k} \sum_{j=2}^{N} x_{1jm} = \sum_{m=1}^{k} \sum_{i=1}^{N-1} x_{iNm} = k \tag{2.25}$$

$$\sum_{m=1}^{k} y_{rm} \leq 1 \; \forall r = 2, \cdots, N-1 \tag{2.26}$$

$$\sum_{i=1}^{N-1} x_{irm} = \sum_{j=2}^{N} x_{rjm} = y_{rm} \; \forall r = 2, \cdots, N \;\; m = 1, \cdots, k \tag{2.27}$$

$$\sum_{i=1}^{N-1} \sum_{j=2}^{N} c_{ij} x_{ijm} \leq B \;\; \forall m = 1, \cdots, k \tag{2.28}$$

$$2 \leq u_{im} \leq N \;\; \forall i = 1, \cdots, N \; m = 1, \cdots, k \tag{2.29}$$

$$u_{im} - u_{jm} + 1 \leq (N-1)\left(1 - x_{ijm}\right) \forall i, j = 2, \cdots, N \; m = 1, \cdots, k \tag{2.30}$$

$$x_{ijm}, y_{im} \in \{0,1\} \; \forall i, j = 1, \cdots, N \; m = 1, \cdots, k \tag{2.31}$$

Η σχέση 2.24 αντιπροσωπεύει το στόχο που πρέπει να επιτευχθεί δηλαδή τη μεγιστοποίηση του κέρδους, όπου pi το κέρδος profit του κόμβου i. Οι σχέσεις 2.25 και 2.26 διασφαλίζουν πως όλα τα μονοπάτια ξεκινούν από τον αρχικό κόμβο 1 και καταλήγουν στον τελικό κόμβο N καθώς οι ακμές που αρχίζουν από τον αρχικό κόμβο και οι ακμές που καταλήγουν στον τελικό κόμβο πρέπει να ισούνται με το πλήθος των διαδρομών (σχέση 2.25) ενώ κάθε κόμβος επιτρέπεται να εμφανίζεται το πολύ μία φορά μέσα σε μια διαδρομή (σχέση 2.26). Η σχέση 2.27 διασφαλίζει τη συνεκτικότητα του μονοπατιού καθώς σε κάθε κόμβο ενός μονοπατιού m, πρέπει καταλήγει αλλά και να αρχίζει από αυτόν μία ακμή. Η σχέση 2.28 περιορίζει το χρόνο κάθε μονοπατιού έτσι ώστε να μη ξεπερνάει το χρονικό όριο B. Οι σχέσεις 2.29 και 2.30 εμποδίζουν την ύπαρξη υπο-διαδρομών (Miller, Tucker, & Zermin 1960 [8]).

Οι Ke et al. (2008) [25] προσεγγίζουν το TOP με έναν Ant Colony Optimization αλγόριθμο, στον οποίο αναθέτουν σε κάθε «μυρμήγκι» να δημιουργήσει μία λύση, η οποία βελτιώνεται στη συνέχεια με Τοπική Αναζήτηση. Αποθηκεύοντας τα μονοπάτια (ίχνη φερομονών) από τα οποία προέκυψαν οι εκάστοτε λύσεις και ενημερώνοντας τα σε κάθε κύκλο, ο αλγόριθμος οδηγεί σε συνεχώς καλύτερες λύσεις.

Οι Souffriau et al. (2008) [26] προτείνουν έναν αλγόριθμο GRASP ενισχυμένο με έναν αλγόριθμο Επανασύνδεσης Μονοπατιών (Path Relinking) για την επίλυση του TOP. Ο αλγόριθμος τους, δημιουργεί σε κάθε επανάληψη μία λίστα CL από εφικτές εισαγωγές κόμβων στη λύση. Για κάθε εισαγωγή της λίστας CL υπολογίζεται μια ευρετική τιμή και προκύπτει ένα κατώφλι, με βάση το οποίο ορισμένες εισαγωγές κόμβων από την λίστα CL αποθηκεύονται σε μία δεύτερη λίστα RCL από την οποία στη συνέχεια επιλέγεται τυχαία μία εισαγωγή. Στο τέλος κάθε επανάληψης ακολουθεί μία τοπική αναζήτηση σε 4 γειτονικές λύσεις με σκοπό τη βελτίωση τους. Οι τεχνικές της τοπικής αναζήτησης είναι οι 2-opt,swap,replace,insert και εφαρμόζονται διαδοχικά σε κάθε λύση. Παρ΄ όλα αυτά επειδή το βασικό μειονέκτημα του GRASP είναι πως δεν αποθηκεύει τις προηγούμενες βέλτιστες λύσεις, οι Souffriau et al. (2008) [26] προτείνουν την προσθήκη του αλγορίθμου Επανασύνδεσης Μονοπατιών κατά τον οποίο, κρατείται ένα σύνολο (pool) από βέλτιστες λύσεις. Κάθε βέλτιστη λύση που προκύπτει από τον GRASP συγκρίνεται με μία λύση «οδηγό» από το pool και τροποποιείται. Η τροποποιημένη λύση συγκρίνεται με αυτές του pool και εάν ο δείκτης ομοιότητας της με αυτές του pool ξεπερνάει ένα κατώφλι τότε προστίθεται και αυτή στο pool.

Οι Bouly et al. (2010) [27] ανέπτυξαν έναν υβριδικό αλγόριθμο που ονόμασαν memetic, ο οποίος αποτελείται από μία διεργασία βέλτιστου διαχωρισμού (Optimal split) και από τεχνικές τοπικής αναζήτησης. Η αρχικοποίηση του αλγορίθμου γίνεται με την εφαρμογή ενός άλλους ευρετικού αλγόριθμου που ανέπτυξαν, τον Iterative Destruction/Construction Heuristic (IDCH). Επίσης ο αλγόριθμος τους αποτελείται από λειτουργίες κωδικοποίησης της λύσης (χρωμόσωμα) σε μια γιγάντια διαδρομή και εκτίμησης της διαδρομής αυτής (optimal split ή quick split). Έπειτα ως λειτουργία μετάλλαξης εφαρμόζεται μια τοπική αναζήτηση με λειτουργίες:

- Shift: Εξετάζεται η εξαγωγή ενός κόμβου από τη γιγάντια διαδρομή και η εισαγωγή του σε ένα άλλο σημείο.
- Swap: Εξετάζονται όλες ανταλλαγές μεταξύ όλων των ζευγαριών των πελατών της γιγάντιας διαδρομής.

Ο Lin (2013) [28] πρότεινε έναν αλγόριθμο Multi-start Simulated Annealing (MSA) ο οποίος συνδυάζει τον αλγόριθμο Simulated Annealing με τη στρατηγική multi-start hill climbing για να διαφύγει από τοπικά βέλτιστα. Αρχικά κατασκευάζονται P_{size} αρχικές λύσεις σ_κ, όπου το P_{size} αντιπροσωπεύει το πλήθος των σημείων έναρξης του MSA. Για κάθε αρχική λύση παράγεται μια γειτονιά λύσεων από την οποία επιλέγεται κάποια νέα λύση σ_κ'. Εάν η σ_κ' υπερτερεί στην τιμή κέρδους από την σ_κ, τότε η πρώτη αντικαθιστά τη δεύτερη. Εάν όμως μειονεκτεί, τότε της ανατίθεται μία πιθανότητα αντικατάστασης της σ_κ που είναι αντιστρόφως ανάλογη με τη διαφορά των 2 λύσεων καθώς όσο καλύτερη είναι η σ_κ από την σ_κ', τόσο λιγότερο πιθανή είναι η αντικατάσταση της από τη τελευταία. Μετά από ένα πλήθος επαναλήψεων, εφαρμόζονται τεχνικές εισαγωγής (insert) και ανταλλαγής (swap) στην καλύτερη, μέχρι εκείνη τη στιγμή, λύση για την περαιτέρω βελτιστοποίηση της. Οι τεχνικές αυτές εφαρμόζονται και για τη παραγωγή γειτονικών λύσεων. Μετά από αρκετές επαναλήψεις, εάν η πιθανότητα αντικατάστασης της καλύτερης λύσης από κάποια λιγότερο καλή, είναι χαμηλότερη από ένα κατώφλι, τότε ο αλγόριθμος τερματίζεται.

Οι Ferreira et al. (2013) [29] προτείνουν ένα γενετικό αλγόριθμο που αποτελείται από τρία βασικά συστατικά: το χρωμόσωμα-λύση που αντιπροσωπεύει τα οχήματα και τις αντίστοιχες διαδρομές τους, την εξελικτική διαδικασία που είναι υπεύθυνη για τις διεργασίες διασταύρωσης (crossover) και μετάλλαξης (mutation), και τον έλεγχο της εγκυρότητας των χρωμοσωμάτων που προκύπτουν από την εξελικτική διαδικασία.

- crossover: ανταλλαγή τυχαίων διαδρομών μεταξύ δύο χρωμοσωμάτων, δημιουργώντας έτσι δύο νέα χρωμοσώματα
- mutation: αφαίρεση ενός ή περισσοτέρων τυχαίων πελατών από μια τυχαία διαδρομή από ένα χρωμόσωμα

2.4 Το πρόβλημα του Ομαδικού Προσανατολισμού με Χρονικά Παράθυρα

Το TOPTW επεκτείνει το TOP καθώς προσθέτει ένα χρονικό παράθυρο σε κάθε κόμβο του γραφήματος. Τα χρονικά παράθυρα δεν είναι απαραίτητα ίδια μεταξύ τους. Η επίσκεψη σε ένα κόμβο του γραφήματος πρέπει να ξεκινήσει πριν το τέλος του χρονικού παραθύρου που διαθέτει ο κόμβος αυτός. Εάν ο χρήστης καταφθάσει σε έναν κόμβο πριν την αρχή του χρονικού παραθύρου τότε μπορεί να παραμείνει σε αυτόν αναμένοντας την έναρξη της λειτουργίας του κόμβου.

Το TOPTW μπορεί να αναπαρασταθεί ως πρόβλημα Ακέραιου Προγραμματισμού ως εξής:

- $x_{ijd} = 1$ εάν στη διαδρομή d η επίσκεψη στον κόμβο i προηγείται από την επίσκεψη στον j, ειδάλλως $x_{ijd} = 0$
- $y_{id} = 1$ εάν πραγματοποιείται επίσκεψη στον κόμβο i στη διαδρομή d, ειδάλλως $y_{id} = 0$
- s_{id} η χρονική στιγμή έναρξης της επίσκεψης στον κόμβο i στη διαδρομή d
- μια σταθερά M

Χρησιμοποιώντας τους παραπάνω συμβολισμούς προκύπτουν οι σχέσεις:

$$maximize \sum_{d=1}^{m}\sum_{i=2}^{n-1} S_i y_{id} \quad (2.32)$$

$$\sum_{d=1}^{m}\sum_{j=2}^{n-1} x_{1jd} = \sum_{d=1}^{m}\sum_{i=2}^{n-1} x_{ind} = m \quad (2.33)$$

$$\sum_{i=1}^{n-1} x_{ikd} = \sum_{j=2}^{n} x_{kjd} = y_{kd} \ \forall k = 2,\cdots,n-1 \ d = 1,\cdots,m \quad (2.34)$$

$$s_{id} + T_i + c_{ij} - s_{jd} \leq M\left(1 - x_{ijd}\right) \forall i,j = 1,\cdots,n \ d = 1,\cdots,m \quad (2.35)$$

$$\sum_{d=1}^{m} y_{kd} \leq 1 \ \forall k = 2,\cdots,n-1 \quad (2.36)$$

$$\sum_{i=1}^{n-1}(T_i y_{id} + \sum_{y=2}^{n} c_{ij} x_{ijd}) \leq T_{max} \ \forall d = 1,\cdots,m \quad (2.37)$$

$$O_i \leq s_{id} \ \forall i = 1,\cdots,n \ d = 1,\cdots,m \quad (2.38)$$

$$s_{id} \leq C_i \ \forall i = 1,\cdots,n \ d = 1,\cdots,m \quad (2.39)$$

$$x_{ijd}, y_{id} \in \{0,1\} \ \forall i,j = 1,\cdots,n \ d = 1,\cdots,m \quad (2.40)$$

Η σχέση 2.32 αντιπροσωπεύει το στόχο του προβλήματος δηλαδή τη μεγιστοποίηση του κέρδους. Η σχέση 2.33 διαβεβαιώνει πως όλες οι τροχιές ξεκινούν από τον αρχικό κόμβο 1 και καταλήγουν στον τελικό κόμβο n καθώς οι ακμές που αρχίζουν από τον αρχικό κόμβο και οι ακμές που καταλήγουν στον τελικό κόμβο πρέπει να ισούνται με το πλήθος των διαδρομών. Η σχέση 2.34 διασφαλίζει τη συνεκτικότητα της κάθε τροχιάς καθώς το πλήθος των τόξων που καταλήγουν σε έναν κόμβο k πρέπει να ισούται με το πλήθος των τόξων που φεύγουν από αυτόν (εκτός και εάν πρόκειται για τον αρχικό η τον τελικό κόμβο). Η σχέση 2.35 διατηρεί το χρονοδιάγραμμα της κάθε τροχιάς. Η σχέση 2.36 διασφαλίζει πως η επίσκεψη στον κάθε κόμβο θα πραγματοποιηθεί το πολύ μία φορά και η σχέση 2.37 περιορίζει τη κάθε τροχιά στο όριο χρόνου T_{max}. Οι σχέσεις 2.38 και 2.39 περιορίζουν την έναρξη της επίσκεψης στο χρονικό παράθυρο του εκάστοτε κόμβου.

Οι Montemanni & Gambardella (2009) [30] προτείνουν έναν Ant Colony System αλγόριθμο ο οποίος αποτελείται από δύο φάσεις:

- Φάση Κατασκευής κατά την οποία ανατίθεται σειριακά σε κάθε μυρμήγκι να κατασκευάσει μία τροχιά προσθέτοντας έναν κόμβο i μετά τον κόμβο j λαμβάνοντας υπόψιν τα εξής:
    - Το ίχνος φερομόνης τ_{ij} (pheromone trail) η οποία διαθέτει τη πληροφορία σχετικά με το πόσο «καλή» υπήρξε στο παρελθόν η ακμή i – j.
    - Ο βαθμός επιθυμίας n_{ij} (desirability) με βάση τον οποίο οι επιθυμητοί κόμβοι προς επίσκεψη μετά τον i είναι αυτοί με υψηλή τιμή κέρδους, που δε βρίσκονται

μακριά από τον i και που τα χρονικά παράθυρα τους χρησιμοποιούνται με κατάλληλο τρόπο.

- Τοπική Αναζήτηση κατά την οποία οι τροχιές που προέκυψαν από την προηγούμενη φάση υπόκεινται σε μία διεργασία που βασίζεται στην ανταλλαγή υποδιαδρομών που μπορεί να βρίσκονται σε διαφορετικές τροχιές ή και στην ίδια, με σκοπό την εύρεση του τοπικού βέλτιστου τους.

Επίσης να προστεθεί πως στόχος των Montemanni & Gambardella (2009) [30] είναι μία ιεραρχική γενίκευση του TOPTW (HTOPTW) κατά την οποία κατασκευάζονται περισσότερες από k επιθυμητές τροχιές χρησιμοποιώντας τα αποδοτικά τμήματα των περισσευούμενων τροχιών για τη διαδικασία της Τοπικής Αναζήτησης που αναφέρθηκε προηγουμένως. Επίσης για την απλούστερη απεικόνιση του HTOPTW σε κλασσικό Πρόβλημα Περιοδεύοντος Πωλητή οι Montemanni & Gambardella (2009) [30] προτείνουν την προσάρτηση των διαδρομών σε μία ενιαία θεωρώντας τον αρχικό και τελικό κόμβο ίδιους.

Οι Vansteenwegen et al. (2009) [6] προτείνουν έναν αλγόριθμο Επαναλαμβανόμενης Τοπικής Αναζήτησης ο οποίος αποτελείται ουσιαστικά από 2 βήματα:

- Τοπική Αναζήτηση: Για κάθε κόμβο i που δεν έχει συμπεριληφθεί στις τροχιές, υπολογίζεται η καλύτερη θέση εισαγωγής δηλαδή η θέση που δίνει το μικρότερο shift ($minShift_i$). Στη συνέχεια, υπολογίζεται η τιμή $ratio_i = profit_i^2 / minShift_i$ για κάθε κόμβο i. Ο κόμβος με τη μεγαλύτερη τιμή της μεταβλητής ratio επιλέγεται για εισαγωγή στην καλύτερη θέση εισαγωγής που του έχει ανατεθεί προηγουμένως.
- Διαταραχή: Αφαιρείται ένα πλήθος συνεχόμενων κόμβων από κάθε τροχιά για να ξεφύγει η λύση από ένα πιθανόν τοπικό βέλτιστο.

Τα 2 παραπάνω βήματα επαναλαμβάνονται διαδοχικά έως ότου η λύση που προέκυψε να μην έχει βελτιωθεί για ένα προκαθορισμένο πλήθος επαναλήψεων. Ο συγκεκριμένος αλγόριθμος υλοποιήθηκε για τους σκοπούς της παρούσας εργασίας και εξετάζεται πιο αναλυτικά στο Κεφάλαιο 3.

Οι Lin & Yu (2012) [31] προτείνουν έναν ευρετικό αλγόριθμο βασισμένο στον Simulated Annealing αλγόριθμο (SA). Αρχικά κατασκευάζεται μία τυχαία λύση X η οποία αποτελείται από μία τυχαία ακολουθία όλων των κόμβων. Σε κάθε επανάληψη επιλέγεται μία λύση Y από τη γειτονιά λύσεων της X που έχει παραχθεί με τεχνικές swap, insertion και inversion. Εάν η τιμή κέρδους της Y είναι μεγαλύτερη από αυτήν της X τότε η X αντικαθίσταται από την Y. Σε αντίθετη περίπτωση ανατίθεται μία πιθανότητα αντικατάστασης της X από την Y η οποία είναι αντιστρόφως ανάλογη με τη διαφορά των κερδών των δύο λύσεων. Ο λόγος αποδοχής μιας χειρότερης λύσης είναι η αποφυγή ενός πιθανού τοπικού βέλτιστου. Μετά το πέρας ενός προκαθορισμένου πλήθος επαναλήψεων εφαρμόζεται και μία τοπική αναζήτηση στη βέλτιστη μέχρι εκείνη τι στιγμή λύση, κατά την οποία αρχικά εξετάζονται όλες οι πιθανές κινήσεις αντικατάστασης (swap) και επιλέγεται η καλύτερη λύση και στη συνέχεια εξετάζονται όλες οι πιθανές κινήσεις εισαγωγής (insertion) επιλέγοντας πάλι τη καλύτερη λύση.

Οι Labadie et al. (2012) [32] προτείνουν έναν αλγόριθμο Αναζήτησης Μεταβλητής Γειτονιάς (Variable Neighborhood Search) που βασίζεται στη διάσπαρτη (granular) εξερεύνηση των γειτονιών. Αρχικά, εφαρμόζεται ένας ευρετικός αλγόριθμος εισαγωγής για την κατασκευή μιας

αρχικής εφικτής λύσης, ο οποίος αρχικά δημιουργεί m διαδρομές, τους οποίους αρχικοποιεί με τους m πιο επικερδείς κόμβους και συνεχίζει εισάγοντας κόμβους εξετάζοντας ένα συγκεκριμένο κριτήριο. Στη συνέχεια εφαρμόζεται ο Granular VNS αλγόριθμος ο οποίος εκτός από αυτά που πράττει ο απλός VNS, δηλαδή την αντικατάσταση μιας ακολουθίας από τη λύση με μία ακολουθία unscheduled κόμβων και η εφαρμογή της τοπικής αναζήτησης στη προκύπτουσα λύση, μειώνει το πλήθος των αναλυόμενων γειτονιών αποκλείοντας μη αποδοτικές ακμές κατά τη διάρκεια της τοπικής αναζήτησης. Η διαδικασία της τοπικής αναζήτησης εξετάζει 2 γειτονιές. Η πρώτη γειτονιά εξετάζεται με σκοπό τη μείωση του συνολικού χρόνου ταξιδιού και με τεχνικές:

- 2-opt: αντικαθιστά 2 ακμές σε μία διαδρομή και αναδιατάσσει τους κόμβους
- Or-opt: επανατοποθετεί έναν κόμβο
- 2-opt*: ανταλλάσσει 2 υπο-τροχιές μεταξύ 2 τροχιών
- Swap: ανταλλάσσει 2 κόμβους

Η δεύτερη γειτονιά εξερευνάται με την αντικατάσταση q συνεχόμενων κόμβων από μία ακολουθία κόμβων που δεν έχουν εισαχθεί στη λύση με σκοπό την αύξηση του συνολικού κέρδους. Η αναζήτηση εύρεσης της ακολουθίας προς εισαγωγή υλοποιείται με δυναμικό προγραμματισμό.

Οι Gavalas et al. (2013) [33] πρότειναν 2 αλγορίθμους για το πρόβλημα Σχεδιασμού Τουριστικών Διαδρομών (TTDP), τον CSCRatio και τον CSCRoutes. Πριν την εφαρμογή των 2 αλγορίθμων, απαιτείται μία προεργασία στα δεδομένα. Αρχικά λοιπόν, οι κόμβοι ομαδοποιούνται σε συστάδες με την εφαρμογή του κ-μέσων (k-means) αλγόριθμο. Στη συνέχεια ακολουθεί η φάση RouteInitPhase κατά την οποία προστίθεται στις τροχιές ένας κόμβος από κάθε συστάδα ενός συνόλου m συστάδων, ο οποίος διαθέτει την υψηλότερη τιμή ratio στη συστάδα του ($ratio_i = profit_i / insertionCost_i$). Η συνέχεια διαφέρει ανάλογα με τον αλγόριθμο που εκτελείται.

- CSCRoutes: Ο συγκεκριμένος αλγόριθμος είναι σχεδιασμένος ώστε να είναι γρήγορος. Κύριο του χαρακτηριστικό είναι πως δεν επιτρέπει την επιστροφή του τουρίστα σε μία συστάδα αφότου αποχωρήσει από αυτήν. Ο κανόνας αυτός φυσικά πρέπει να λαμβάνεται υπόψιν κατά την εισαγωγή των κόμβων στις τροχιές.
- CSCRatio: Ο συγκεκριμένος αλγόριθμος είναι σχεδιασμένος ώστε να ευνοεί την εισαγωγή των κόμβων πριν ή μετά από κόμβους της ίδιας συστάδας. Αυτό επιτυγχάνεται με τη βοήθεια της παραμέτρου clusterParameter και της μεταβλητής $shiftCluster_i^j = shift_i^j / clusterParameter$. Στη περίπτωση λοιπόν που εξετάζεται η εισαγωγή ενός κόμβου πριν η μετά από έναν κόμβο της ίδιας συστάδας, χρησιμοποιείται η shiftCluster και όχι η shift που για $clusterParameter = 1.3$ (στην αρχή του αλγορίθμου), ισχύει ότι $shiftCluster_i^j < shift_i^j$. Η clusterParameter μειώνεται στη πορεία του αλγορίθμου μέχρι να φτάσει το 1 όπου πλέον θα ισχύει $shiftCluster_i^j = shift_i^j$.

Οι Hu & Lim (2014) [34] προτείνουν για την επίλυση του TOPTW έναν ευρετικό αλγόριθμο, τον Iterative three-Component Heuristic (I3CH), ο οποίος αποτελείται από τις εξής διαδικασίες:

- Local search: Κατά τη τοπική αναζήτηση δημιουργείται ένα πλήθος γειτονικών λύσεων, των οποίων οι τροχιές προστίθενται σε ένα σύνολο τροχιών pool και η βέλτιστη μέχρι

στιγμής λύση αντικαθίσταται από την καλύτερη γειτονική εάν φυσικά η δεύτερη είναι πιο επικερδής από την πρώτη.

- Simulated annealing: Κατά τη διαδικασία της προσομοιωμένης ανόπτησης, παράγεται μονάχα μία γειτονική λύση, η οποία γίνεται αποδεκτή με βάση μία υπολογιζόμενη πιθανότητα. Οι τροχιές των προκύπτουσων γειτονικών λύσεων προστίθενται και πάλι στο σύνολο τροχιών pool.
- Route recombination: Κατά τον ανασυνδυασμό των τροχιών επιχειρείται η εύρεση ενός κατάλληλου συνδυασμού των τροχιών που βρίσκονται στο pool έτσι ώστε να κατασκευαστεί μια βέλτιστη λύση.

Για την κατασκευή γειτονικών λύσεων χρησιμοποιείται μια λειτουργία αναζήτησης γειτονικών λύσεων (eliminator) ή οποία αρχικά αφαιρεί ένα πλήθος κόμβων από τις τροχιές μιας λύσης Α και προσπαθεί να εισάγει κόμβους που δεν είχαν συμπεριληφθεί προηγουμένως σε αυτές, κατασκευάζοντας έτσι μία λύση Β. Στη συνέχεια η Β υπόκειται σε μία επιπλέον επεξεργαστική διαδικασία, η οποία αποτελείται από επτά λειτουργίες τύπου επανατοποθέτησης(relocate), ανταλλαγής(exchange) και 2-opt.

Οι José Ruiz-Meza et al. (2021) [35] μελέτησαν μια πιο εξειδικευμένη περίπτωση του TOPTW, το Multi-Constraints Multi-Modal Team Orienteering Problem with Time Windows for Groups with Heterogeneous Preferences (MCMMTOPTWGHP), στο οποίο μια ομάδα ατόμων, με διαφορετικές προτιμήσεις και περιορισμούς, θέλει να επισκεφθεί ένα πλήθος σημείων/αξιοθέατων με χρονικά παράθυρα. Για την επίλυση του MCMMTOPTWGHP P οι συγγραφείς προτείνουν έναν GRASP αλγόριθμο, που κατασκευάζει μια αρχική λύση στη φάση κατασκευής και στη συνέχεια τη βελτιώνει στη φάση της τοπικής αναζήτησης. Στη φάση της κατασκευής, ο αλγόριθμος κατασκευάζει μια λίστα RCL με τους καλύτερους κόμβους προς εισαγωγή από τους οποίους επιλέγει με τυχαίο τρόπο ποιος θα εισαχθεί στις διαδρομές ενισχύοντας έτσι τη διαφοροποίηση των λύσεων. Η τοπική αναζήτηση αποτελείται από 3 λειτουργίες:

- Εισαγωγή κόμβων που δεν έχουν μπει ακόμη στη λύση και ικανοποιούν κάποιες συνθήκες (insert)
- Αντικατάσταση ενός κόμβου της λύσης με έναν πιο επικερδή κόμβο που δεν έχει ακόμη εισαχθεί (replace)
- Ανταλλαγή δύο κόμβων μεταξύ δύο διαδρομών (swap)

2.5 Το Πρόβλημα Χρονικά Εξαρτώμενου Ομαδικού Προσανατολισμού με Χρονικά Παράθυρα (TDTOPTW)

Το TDTOPTW επεκτείνει το TOPTW και θεωρείται η καταλληλότερη επέκταση του OP, σε σύγκριση με τις προηγούμενες, για να μοντελοποιήσει το TTDP καθώς συνδυάζει τις χρονικές εξαρτήσεις των ακμών με τα χρονικά παράθυρα των κόμβων.

Οι Gavalas et al.(2015) [36] χρησιμοποίησαν το TDOPTW για να μοντελοποιήσουν το TTDP και ανέπτυξαν τρεις ευρετικούς αλγορίθμους, τον TDCSCRoutes, τον SlackCSCRoutes και τον AvgCSCRoutes.

- Time Dependent CSCRoutes (TDCSCRoutes):
  Ο TDCSCRoutes διαφέρει στο βήμα εισαγωγής από τον CSCRoutes που περιεγράφηκε στην υποενότητα του TOPTW καθώς τροποποιείται έτσι ώστε να μπορεί να χειριστεί χρονικά εξαρτημένα κόστη ακμών. Οπότε το προπαρασκευαστικό στάδιο ομαδοποίησης των κόμβων με τη χρήση του global k-means αλγορίθμου και ο κανόνας που δεν επιτρέπει την επιστροφή μιας διαδρομής σε μία συστάδα, παραμένουν ίδια. Υπενθυμίζεται πως λόγω του κανόνα αυτού, ένας κόμβος I δεν επιτρέπεται να εισαχθεί σε ορισμένα σημεία των διαδρομών ανάλογα με τη συστάδα που ανήκει. Η παράμετρος weight που υπολογίζεται κατά τη διαδικασία εισαγωγής του TDCSCRoutes, δίνει μεγαλύτερη έμφαση στο κέρδος της εισαγωγής (profit) παρά στο χρόνο κατανάλωσης αυτής (shift). Επίσης, αρχικά ευνοεί τις εισαγωγές κόμβων που δημιουργούν παρατεταμένες κενές χρονικές περιόδους, ενώ προς το τέλος ευνοεί εισαγωγές κόμβων που εκμεταλλεύονται στο έπακρο τα αναξιοποίητα χρονικά αποθέματα.
- Time Dependent Slack CSCRoutes (SlackCSCRoutes):
  Ο SlackCSCRoutes διαφοροποιείται από τον TDCSCRoutes στη βήμα εισαγωγής καθώς περιλαμβάνει ένα πιο καθολικό κριτήριο που λαμβάνει υπόψιν όλους τους κόμβους της διαδρομής κατά την εισαγωγή ενός κόμβου, αντίθετα με το κριτήριο του TDCSCRoutes που δίνει βάση κυρίως στο κέρδος της εισαγωγής ενός κόμβου σε μια διαδρομή και στη χρονική επιβάρυνση που θα επιφέρει σε αυτήν ως ένα σημείο. Η παράμετρος που προστίθεται στον SlackCSCRoutes είναι η slack για την οποία ισχύει $slack_i = maxStart_i - arrive_i$, όπου ο όρος $maxStart_i$ συμβολίζει το μέγιστο επιτρεπτό χρόνο έναρξης της επίσκεψης στον κόμβο i έτσι ώστε να μην επηρεάζει την εγκυρότητα της διαδρομής. Όσο μικρότερο είναι το $slack_i$, τόσο λιγότερο πιθανό είναι να είναι εφικτή η εισαγωγή ενός κόμβου πριν από αυτόν. Η εισαγωγή ενός καινούριου κόμβου σε μια διαδρομή έχει ως αποτέλεσμα τη μείωση της μεταβλητής slack σε ένα υποσύνολο κόμβων που ανήκουν σε αυτήν. Κατά την εξέταση μιας πιθανής θέσης εισαγωγής ενός κόμβου i σε μία διαδρομή, υπολογίζεται το μέσο slack των κόμβων στη διαδρομή (avgSlack). Η θέση που επιτυγχάνει το μέγιστο avgSlack είναι και αυτή που επιλέγεται. Ο κόμβος που θα εισαχθεί στο εκάστοτε βήμα είναι αυτός με τη μεγαλύτερη τιμή της μεταβλητής $slackWeight_i = p_i^2 \cdot avgSlack_i$.
- Average Travel Times (AvgCSCRoutes):
  Ο AvgCSCRoutes βασίζεται στη προσέγγιση των Garcia et al. (2013), καθώς ανάγει το TDTOPTW σε TOPTW προσπαθώντας να επιλύσει το δεύτερο και να εφαρμόσει τη λύση στο πρώτο. Πιο αναλυτικά, υπολογίζεται αρχικά το μέσο κόστος διάτρεξης της κάθε ακμής. Έπειτα εφαρμόζεται ο CSCRoutes για το στιγμιότυπο του προβλήματος με τα μέσα κόστη που υπολογίστηκαν προηγουμένως. Στη συνέχεια, επαναφέρει τα κόστη ακμών στις αρχικές τους τιμές, καθιστώντας έτσι αρκετές τροχιές από την παραγόμενη λύση του προηγούμενου βήματος ανέφικτες, είτε επειδή κάποια διαδρομή έχει πλέον ένα χρονικό κόστος μεγαλύτερο από το επιτρεπτό, είτε γιατί η επίσκεψη σε κάποιο κόμβο πραγματοποιείται εκτός του χρονικού παραθύρου του. Στη περίπτωση, λοιπόν, που όντως προκύψει μια μη έγκυρη διαδρομή, ο πρώτος χρονικά κόμβος που έχει χρόνο άφιξης μεγαλύτερο από τον αντίστοιχο χρόνο άφιξης που είχε μετά τον CSCRoutes, αφαιρείται από τη διαδρομή του. Εάν ο κόμβος αυτός είναι τερματικός, τότε αφαιρείται ο προηγούμενός του. Στο τέλος οι κόμβοι που βρίσκονται εκτός τροχιών ταξινομούνται σε

φθίνουσα σειρά με βάση το κέρδος τους και εισάγονται σειριακά στις αντίστοιχες βέλτιστες θέσεις τους, εάν φυσικά διατηρούν τις διαδρομές έγκυρες.

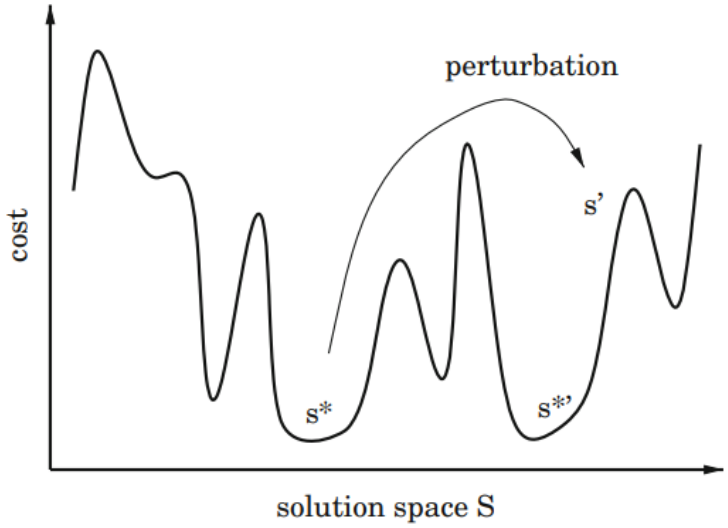
Οι Khodadadian et al. (2022) [37] μελέτησαν το πρόβλημα TDOPTW-STP όπου τα κέρδη των κόμβων είναι εξαρτώμενα από το χρόνο επίσκεψης/εξυπηρέτησης. Ο προτεινόμενος αλγόριθμος VNS τους αποτελείται από δύο κύριες φάσεις: αρχικοποίηση και βελτίωση. Η αρχικοποίηση με τη σειρά της περιλαμβάνει 3 βήματα: προεπεξεργασία, παραγωγή αρχικής λύσης και έναν αλγόριθμο VND. Κατά την προεπεξεργασία, για κάθε κόμβο i, ορίζεται ένα σύνολο κόμβων j ως λίστα πλησιέστερων γειτόνων με βάση πάντα την εφικτότητά τους. Έπειτα, παράγεται μια αρχική λύση S0 με βάση τις σχέσεις γειτνίασης από την προεπεξεργασία. Στη συνέχεια ο αλγόριθμος VND διερευνά γειτονικές λύσεις ψάχνοντας τη βέλτιστη. Στη φάση της βελτίωσης, ο αλγόριθμος εφαρμόζει αρχικά μια διαταραχή στην λύση που προέκυψε από την φάση της αρχικοποίησης και στη συνέχεια εφαρμόζει πάλι έναν αλγόριθμο VND με διάφορες κινήσεις τοπικής αναζήτησης για να ενισχύσει την λύση.

3. Αλγόριθμος επίλυσης του TOPTW

Για την παρούσα εργασία, χρησιμοποιήθηκε ο μεταευρετικός αλγόριθμος Επαναλαμβανόμενης Τοπικής Αναζήτησης (ILS) και συγκεκριμένα ο ILS των Vansteenwegen et al. (2009) [6]. Οι μεταευρετικοί αλγόριθμοι είναι αλγόριθμοι γενικού σκοπού , σχεδιασμένοι έτσι ώστε να βρίσκουν ικανοποιητικές λύσεις σε προβλήματα βελτιστοποίησης, σε σύντομο χρονικό διάστημα, χρησιμοποιώντας ευρετικές τεχνικές.

3.1 Μεταευρετικός αλγόριθμος Επαναλαμβανόμενης Τοπικής Αναζήτησης

Αρχικά η Επαναλαμβανόμενη Τοπική Αναζήτηση κατασκευάζει μια αρχική λύση S0, η οποία βελτιώνεται μέχρι να συναντηθεί ένα τοπικό βέλτιστο. Έπειτα, λαμβάνοντας υπόψιν την ιστορία της διαδικασίας, προκαλείται μια διαταραχή στην S0 και παράγεται εκ νέου μία λύση S1 ξεφεύγοντας έτσι από το τοπικό βέλτιστο. Αυτή τη φορά πραγματοποιείται τοπική αναζήτηση για την S1 μέχρι να προκύψει μία νέα τοπικά βέλτιστη λύση S2 η οποία συγκρίνεται με τη S0 και στη περίπτωση που υπερτερεί , η νέα τοπική βέλτιστη λύση γίνεται αποδεκτή ως η μέχρι στιγμής βέλτιστη. Η διαδικασία επαναλαμβάνεται και διακόπτεται εφόσον ικανοποιηθεί μια συνθήκη όπως το να φτάσει το μέγιστο επιτρεπτό πλήθος επαναλήψεων ή το να ξεπεράσει ένα χρονικό όριο εκτέλεσης.



Εικόνα 3-1: Εικονογραφική αναπαράσταση επαναλαμβανόμενης τοπικής αναζήτησης, πηγή: H.R. Lourenço et al.(2010) [38]

Ο μεταυρετικός αλγόριθμος ILS αποτελείται από 4 βασικά συστατικά:

- Κατασκευή αρχικής λύσης
- Τοπική Αναζήτηση
- Διαταραχή
- Κριτήριο αποδοχής

Algorithm 1 ILS

```
1: procedure ILS
2:     S_0 ← Κατασκεύασε μια αρχική λύση
3:     S_0 ← Εφαρμοσε τοπική αναζήτηση στη λύση S_0
4:     while Η συνθήκη δεν έχει ικανοποιηθεί do
5:         S_1 ← Διατάραξε την S_0 με βαση την ιστορία
6:         S_2 ← Εφάρμοσε τοπική αναζήτηση στη λύση S_1
7:         S_0 ← Διαλεξε την S_0 ή την S_2 με βάση κάποιο κριτήριο
```

3.1.1 Κατασκευή αρχικής λύσης

Υπάρχουν 2 βασικές τεχνικές κατασκευής της αρχικής λύσης:

- Κατασκευή τυχαίας λύσης
- Εφαρμογή ενός άπληστου ευρετικού αλγορίθμου

Σύμφωνα με τους H.R. Lourenço et al.(2010) [38], η χρήση ενός άπληστου αλγορίθμου για την κατασκευή της αρχικής λύσης σε συνδυασμό με την τοπική αναζήτηση του ILS συνήθως οδηγεί σε καλύτερες λύσεις και στη μείωση του χρόνου εκτέλεσης του συνολικού αλγορίθμου καθώς χρειάζονται λιγότερα βήματα βελτίωσης. Η διαφορά αυτή γίνεται πιο αισθητή ειδικά σε περιπτώσεις όπου ο χρόνος εκτέλεσης είναι περιορισμένος, καθώς όσο περνάει η ώρα, η διαφορά της ποιότητας των λύσεων μεταξύ των δύο εναλλακτικών, μειώνεται.

3.1.2 Τοπική Αναζήτηση

Η ποιότητα της τοπικής αναζήτησης επηρεάζει άμεσα την ποιότητα του ILS. Συνήθως προτιμάται μια ισχυρή τοπική αναζήτηση, αλλά εάν ο χρόνος εκτέλεσης είναι περιορισμένος, μερικές φορές προτιμάται η συχνότερη εφαρμογή μιας χειρότερης και συντομότερης τοπικής αναζήτησης από μία ισχυρότερη και πιο αργή. Σημαντικό ρόλο για την λήψη αυτής της απόφασης είναι η αναλογία κέρδους και χρόνου εκτέλεσης μεταξύ των δύο επιλογών. Για παράδειγμα, για το TSP η τεχνική 3-opt είναι λίγο πιο αργή από την 2-opt, αλλά η βελτίωση της ποιότητας των λύσεων αξίζει τον επιπλέον χρόνο της CPU. Παρ' όλα αυτά, η τεχνική 4-opt αν και δίνει ελαφρώς καλύτερες λύσεις από την 3-opt, συνήθως είναι O(n) φορές πιο αργή (όπου n ο αριθμός των πόλεων) οπότε δεν προτιμάται.

Επίσης μπορεί να υπάρχουν ορισμένα πλεονεκτήματα στο να επιτρέπει η τοπική αναζήτηση και χειρότερες λύσεις, όπως στον αλγόριθμο Simulated Annealing ή στον Tabu Search. Για παράδειγμα, σύμφωνα με τους H.R. Lourenço et al.(2010) [38] , στο πρόβλημα job-shop scheduling problem (JSSP), η ενσωμάτωση του Tabu Search αλγορίθμου στον ILS αντί της τοπικής αναζήτησης, δίνει καλύτερα αποτελέσματα.

3.1.3 Διαταραχή

Στο βήμα της διαταραχής, ο ILS ξεφεύγει από το τοπικό βέλτιστο στο οποίο έχει καταλήξει από την τοπική αναζήτηση. Σε κάθε πρόβλημα βελτιστοποίησης, η τεχνική Διαταραχής μπορεί να είναι διαφορετική. Για παράδειγμα, για το Πρόβλημα Περιοδεύοντος Πωλητή (TSP), η τεχνική διαταραχής μπορεί είναι η αφαίρεση διαδρομών από μία λύση.

Η ένταση της διαταραχής παίζει καθοριστικό ρόλο για την απόδοση του αλγορίθμου. Εάν η διαταραχή είναι πολύ μικρή, τότε υπάρχει ο κίνδυνος να οδηγείται ο αλγόριθμος σε κύκλους καθώς οι κόμβοι που θα αφαιρούνται από τη διαταραχή θα προστίθενται πίσω κατά την τοπική αναζήτηση και ο αλγόριθμος θα καταλήγει πάλι στο ίδιο τοπικό βέλτιστο, περιορίζοντας έτσι σημαντικά τη διαφορετικότητα των λύσεων. Εάν όμως η διαταραχή είναι πολύ έντονη τότε υπάρχει ο κίνδυνος ο αλγόριθμος να ξεκινάει σε κάθε επανάληψη από τυχαίες αρχικές λύσεις, κάτι που θα οδηγήσει πιθανότατα στην εύρεση λιγότερο ποιοτικών λύσεων καθώς και στην αύξηση του χρόνου εκτέλεσης της επόμενης τοπικής αναζήτησης. Μια καλή ιδέα για τη ρύθμιση της έντασης της Διαταραχής, είναι να προσαρμόζεται ντετερμινιστικά κατά τη διάρκεια του αλγορίθμου.

3.1.4 Κριτήριο αποδοχής

Το κριτήριο αποδοχής καθορίζει εάν σε μια επανάληψη του ILS θα γίνει αποδεκτή η λύση που προέκυψε από την τοπική αναζήτηση. Ουσιαστικά, ρυθμίζει την ισορροπία μεταξύ δύο στρατηγικών αναζήτησης καινούριων λύσεων, της εντατικοποίησης και της διαφοροποίησης. Οι δύο ακραίες επιλογές για το κριτήριο αποδοχής είναι:

- Αποδοχή μόνο καλύτερων λύσεων (ευνοεί την εντατικοποίηση της αναζήτησης)
- Αποδοχή οποιασδήποτε λύσης (ευνοεί τη διαφοροποίηση της αναζήτησης)

Φυσικά υπάρχουν και ενδιάμεσες επιλογές, όπως για παράδειγμα στον αλγόριθμο Προσομοιωμένης Ανόπτησης (Simulated annealing, SA). Στον αλγόριθμο SA, μία λύση S*’ μπορεί να γίνει αποδεκτή ως διάδοχος της λύσης S* ακόμα και αν είναι χειρότερη, με πιθανότητα $p = \exp\{\left(C(s^*) - C\left(s^{*'}\right)\right)/T\}$ όπου C(s*) το σκορ μίας λύσης s* και T μια μεταβλητή θερμοκρασία που μειώνεται όσο τρέχει ο αλγόριθμος. Μία άλλη ενδιάμεση επιλογή είναι η εναλλαγή μεταξύ των δύο στρατηγικών αναζήτησης με βάση το ιστορικό των λύσεων, δηλαδή, εάν η στρατηγική εντατικοποίησης δε βελτιώνει περαιτέρω τις λύσεις, ο αλγόριθμος να μπορεί να αλλάξει προσωρινά στρατηγική.

3.2 Υλοποίηση Επαναλαμβανόμενης Τοπικής Αναζήτησης για το TOPTW

Πριν αναλυθεί ο υλοποιηθείς αλγόριθμος, πρέπει πρώτα να διευκρινιστούν τα χαρακτηριστικά του κάθε κόμβου.

Για κάθε κόμβο i, είναι εξαρχής γνωστές οι εξής πληροφορίες(σταθερές):

- οι συντεταγμένες του (x_i,y_i)
- το κέρδος της επίσκεψης ($profit_i$)
- η χρονική διάρκεια της επίσκεψης ($visitDur_i$)

- η ώρα έναρξης λειτουργίας του ($openTime_i$)
- η ώρα παύσης λειτουργίας του ($closeTime_i$)

Παρ΄ όλα αυτά για την υλοποίηση του αλγορίθμου, για κάθε κόμβο i χρειάζονται κάποιες πρόσθετες πληροφορίες(μεταβλητές) οι οποίες ενημερώνονται κατά την εκτέλεση του αλγορίθμου:

- η ώρα άφιξης ($arrTime_i$)
- η διάρκεια αναμονής ($wait_i$)
- η ώρα έναρξης της επίσκεψης ($startOfVisit_i$)
- η ώρα αναχώρησης ($depTime_i$)
- η διάρκεια συνολικής χρονικής ολίσθησης λόγω της εισαγωγής του i ($shift_i$)
- ο μέγιστος χρόνος που μπορεί να παραταθεί η έναρξη της επίσκεψης ($maxShift_i$)

Επίσης είναι προφανές πως πρέπει να υπολογιστούν εξαρχής οι αποστάσεις μεταξύ όλων των κόμβων. Στη παρούσα εργασία η ευκλείδεια απόσταση από τον κόμβο i στον κόμβο j, αντιπροσωπεύει και το χρόνο ταξιδιού ($travelTime_{ij}$) μεταξύ αυτών των κόμβων. Ακόμα, πρέπει να αναφερθεί πως για την υλοποίηση του αλγόριθμου, οι κόμβοι οργανώνονται σε 2 δομές δεδομένων, Unvisited και Walks. Η πρώτη δομή δεδομένων είναι μια λίστα των κόμβων που δεν έχουν συμπεριληφθεί ακόμη στη στις διαδρομές, ενώ η δομή δεδομένων Walks είναι ένα σύνολο από λίστες κόμβων που αναπαριστούν τις διαδρομές. Η σειρά των δεδομένων στη λίστα Unvisited δεν παίζει κάποιο ρόλο, ενώ οι σειρές των κόμβων στη δομή Walks είναι σημαντικές για την απεικόνιση των τροχιών.

3.2.1 Βήμα Εισαγωγής

Στο βήμα εισαγωγής γίνεται προσπάθεια εισαγωγής των Unvisited κόμβων στις διαδρομές. Για να εισαχθεί ένας καινούριος κόμβος σε κάποια από τις διαδρομές, πρέπει να πληροί ορισμένα κριτήρια.

Πρώτα απ’ όλα, όπως είναι φυσικό, για να εισαχθεί ένας κόμβος j ανάμεσα στους κόμβους i και k στη διαδρομή m, θα πρέπει η ώρα αναχώρησης από τον κόμβο j ($depTime_j$) να είναι πριν από την ώρα παύσης λειτουργίας του ($depTime_j \leq closeTime_j$). Αρχικά, λοιπόν, υπολογίζεται το $depTime_j$ και ελέγχεται εάν ικανοποιεί την παραπάνω συνθήκη. Εάν την ικανοποιεί, τότε η επίσκεψη στον κόμβο j καθίσταται εφικτή όσον αφορά το ωράριο λειτουργίας του. Παρόλα αυτά, επειδή με την εισαγωγή του j θα μεταβληθεί η ώρα άφιξης του k αλλά και ίσως των κόμβων που έπονται μετά τον k, πρέπει να επαληθευτεί πως δεν καθιστά αδύνατη την επίσκεψη σε οποιονδήποτε από τους επερχόμενους κόμβους. Για το σκοπό αυτό, έχει οριστεί αναδρομικά σε κάθε κόμβο της διαδρομής m μια τιμή MaxShift έτσι ώστε να είναι γνωστό το πόσο μπορεί να παραταθεί η επίσκεψη στον εκάστοτε κόμβο χωρίς να προκαλέσει δυσχέρεια στη συνέχεια της διαδρομής. Σε περίπτωση, λοιπόν, που δεν δημιουργείται πρόβλημα με την εισαγωγή του j στην ακμή i → k τότε υπολογίζεται η τιμή $Shift_j$ για αυτό το σημείο εισαγωγής.

Στο σημείο αυτό, αξίζει να αναφερθεί πως θα μπορούσε να προστεθεί ένας ακόμα έλεγχος που δεν περιλαμβάνεται στον αλγόριθμο των Vansteenwegen et al. [6]. Όπως εξηγήθηκε παραπάνω, μια θέση εισαγωγής A για έναν κόμβο i θεωρείται καλύτερη από μία θέση εισαγωγής

Β εάν ισχύει Shift[iA] < Shift[iB]. Παρ΄ όλα αυτά, στην αρχή του αλγορίθμου, που οι διαδρομές είναι ακόμα κενές, ο κάθε κόμβος της λίστας Unvisited που εξετάζεται έχει το ίδιο Shift και για όλες τις διαδρομές. Έτσι, με τη παραπάνω λογική, κάθε φορά ο πρώτος κόμβος προς εισαγωγή θα εισάγεται στη πρώτη διαδρομή. Για το λόγο αυτό, στη περίπτωση που ισχύει Shift{iA} = Shift{iB}, ίσως να ανατίθεται μια πιθανότητα εισαγωγής στη θέση Β, της τάξεως 50%. Η παραπάνω αλλαγή προσδίδει τυχαιότητα στο αλγόριθμο αφαιρώντας του την ντετερμινιστική του ιδιότητα και καθιστώντας τον στοχαστικό καθώς τα αποτελέσματα πλέον ποικίλλουν εμφανίζοντας διάφορες χαμηλότερες αλλά και υψηλότερες τιμές κέρδους από το συνηθισμένο.

$$arrTime_j = depTime_i + TT_{ij} \quad (3.1)$$

$$wait_j = \max(0, openTime_j - arrTime_j) \quad (3.2)$$

$$startOfVisit_j = arrTime_j + wait_j \quad (3.3)$$

$$Shift_j = TT_{ij} + wait_j + visitDur_j + TT_{jk} - TT_{ik} \quad (3.4)$$

$$MaxShift_j = \min(closeTime_j - depTime_j, wait_k + MaxShift_k) \quad (3.5)$$

Η παραπάνω διαδικασία επαναλαμβάνεται για κάθε υποψήφιο προς εισαγωγή κόμβο, και ελέγχεται η εισαγωγή του σε κάθε τροχιά της κάθε διαδρομής. Έτσι, για κάθε κόμβο υπολογίζεται η καλύτερη θέση εισαγωγής στις διαδρομές. Έπειτα πρέπει να αποφασιστεί το ποιος κόμβος θα προστεθεί στις διαδρομές. Για το λόγο αυτό, για κάθε υποψήφιο προς εισαγωγή κόμβο i υπολογίζεται μία τιμή $ratio_i$.

$$ratio_i = \frac{profit_i^2}{minShift_i} \quad (3.6)$$

όπου $minShift_i$ είναι το Shift της καλύτερης θέσης εισαγωγής του κόμβου i στις διαδρομές. Ο κόμβος με το μεγαλύτερο ratio επιλέγεται για την επόμενη εισαγωγή.

Αφότου εισαχθεί ένας κόμβος i στη διαδρομή m, είναι προφανές πως η ώρα άφιξης στον κόμβο i+1 θα μετατοπιστεί. Σε περίπτωση που η άφιξη στον κόμβο i+1 δεν επηρεάσει την ώρα έναρξης της επίσκεψης στον i+1 κόμβο, κάτι που θα συμβεί εάν προηγουμένως ίσχυε $wait_{i+1} \neq 0$ και συνεχίσει να ισχύει παρά τη προσθήκη του i, τότε η ώρα άφιξης στον κόμβο i+2 δεν θα αλλάξει. Με το σκεπτικό αυτό, μετά από κάθε εισαγωγή ενός κόμβου i σε μια διαδρομή m, για κάθε κόμβο που ακολουθεί μέχρι τον κόμβο n που επηρεάζεται η τιμή του $startOfVisit_n$ή μέχρι τον τελικό κόμβο N πρέπει να ενημερώνονται οι τιμές arrTime, wait, startOfVisit, depTime και Shift.

Επίσης, όπως έγινε αντιληπτό παραπάνω, η τιμή MaxShift είναι αναγκαία για την έλεγχο εφικτότητας μιας εισαγωγής. Όμως, επειδή όπως αναφέρθηκε προηγουμένως η τιμή depTime μερικών κόμβων μεταβάλλεται, και η τιμή MaxShift εξαρτάται από την τιμή depTime (σχέση 3.6), πρέπει ξεκινώντας από τον κόμβο ν που δεν επηρεάστηκε η τιμή του startOfVisit(n) ή από τον τελικό κόμβο N και προς τον αρχικό κόμβο «0» να ενημερωθεί και η τιμή της MaxShift τους.

Η παραπάνω διαδικασία επαναλαμβάνεται έως ότου να μην είναι δυνατή κάποια άλλη εισαγωγή στις διαδρομές, δηλαδή μέχρι η λίστα των κόμβων προς εισαγωγή είτε να είναι κενή

είτε μέχρι όλοι κόμβοι που την αποτελούν να προκαλούν δυσχέρεια σε κάθε διαδρομή m με τη εισαγωγή τους σε αυτή.

Algorithm 2 Insert

```
1: procedure INSERT
2:     for Κάθε κόμβο στη λίστα Unvisited do
3:         Βρες τη θέση εισαγωγής με το μικρότερο Shift
4:         Υπολόγισε το ratio
5:     Εισήγαγε τον κόμβο (j) με το μεγαλύτερο ratio στη καταλληλότερη θέση
6:     Ενημέρωσε τα arrTime_j, wait_j, startOfVisit_j, depTime_j, MaxShift_j
7:     for Κάθε κόμβο μετά τον j μέχρι τον n do
8:         Ενημέρωσε τα arrTime, wait, startOfVisit, depTime, Shift
9:     for Κάθε κόμβο από τον 0 μέχρι τον n do
10:        Ενημέρωσε το MaxShift
```

3.2.2 Βήμα Διαταραχής

Στο προηγούμενο βήμα, ουσιαστικά εφαρμόσθηκε η τεχνική της τοπικής αναζήτησης και με «άπληστο» τρόπο κατασκευάστηκε μία λύση S0. Παρ' όλα αυτά, επειδή πιθανότατα η λύση που παράχθηκε είναι μόνο τοπικά βέλτιστη, ακολουθεί το βήμα της διαταραχής. Στο βήμα αυτό, η λύση που παράχθηκε προηγουμένως υπόκειται σε μια διαδικασία αφαίρεσης κόμβων από κάθε διαδρομή έτσι ώστε να ξεφύγει από το τοπικό βέλτιστο.

Η διαδικασία αυτή χρειάζεται 2 παραμέτρους, την S και την R. Η πρώτη αντιπροσωπεύει τον κόμβο από τον οποίο θα ξεκινήσει η διαδικασία αφαίρεσης (πρώτο, δεύτερο, τρίτο, κλπ.) και η δεύτερη το πλήθος των συνεχόμενων κόμβων που πρόκειται να αφαιρεθούν ξεκινώντας από τον S. Η αφαίρεση αυτή θα εφαρμοσθεί σε κάθε διαδρομή της λύσης που παράχθηκε στο προηγούμενο βήμα με τα ίδια S και R. Έστω routeLen(m) το μήκος μιας διαδρομής m. Εάν σε κάποια επανάληψη προκύψει ότι S + R > routeLen(m) τότε η αφαίρεση συνεχίζεται από τον πρώτο κόμβο της διαδρομής m.

Μετά από την αφαίρεση των κόμβων είναι προφανές πως πρέπει να ενημερωθούν οι μεταβλητές των κόμβων που παρέμειναν στις διαδρομές. Οπότε επαναλαμβάνεται η διαδικασία ενημέρωσης των arrTime, wait, startOfVisit, depTime, Shift και MaxShift που περιεγράφηκε στο τέλος του προηγούμενου βήματος.

Algorithm 3 Shake

```
1: procedure SHAKE
2:     for Κάθε διαδρομή do
3:         Αφαίρεσε τους κόμβους από τον S μέχρι και τον S+R-1
4:         for Κάθε κόμβο μετά τον S+R do
5:             Ενημέρωσε τα arrTime,wait,startOfVisit,depTime,Shift
6:         end for
7:         for Κάθε κόμβο από τον 0 μέχρι τον n do
8:             Ενημέρωσε το MaxShift
9:         end for
10:    end for
11: end procedure
```

3.2.3 Ευρετικός Αλγόριθμος Επαναλαμβανόμενης Τοπικής Αναζήτησης

Μετά από κάθε βήμα Εισαγωγής, προκύπτει μία λύση S’, η οποία συγκρίνεται με τη βέλτιστη μέχρι στιγμής λύση S. Στη περίπτωση που η λύση S’ είναι καλύτερη από την S, ως βέλτιστη θεωρείται πλέον η S’. Στην αντίθετη περίπτωση αυξάνεται η μεταβλητή timesNotImproved κατά 1. Ακολουθεί το βήμα Διαταραχής κατά το οποίο αφαιρούνται κόμβοι επιτρέποντας στη λύση S να ξεφύγει από πιθανό τοπικό βέλτιστο σημείο. Επίσης χρειάζεται μια ρύθμιση στις μεταβλητές S και R, οι οποίες αναφέρθηκαν στο βήμα διαταραχής, έτσι ώστε να είναι βέβαιο πως όλοι οι κόμβοι που εισήλθαν αρχικά στην λύση θα έχουν αφαιρεθεί τουλάχιστον μία φορά μέχρι το τερματισμό. Ο αλγόριθμος τερματίζεται όταν οι λύσεις που προκύψουν για maxTimesNotImproved συνεχόμενες φορές είναι χειρότερες από τη βέλτιστη λύση.

Algorithm 4 Ευρετικός Αλγόριθμος ILS

```
1:  procedure ILS HEURISTIC(maxTimesNotImproved)
2:      S ← 1
3:      R ← 1
4:      timesNotImproved ← 0
5:      while timesNotImproved < maxTimesNotImproved do
6:          Εκτέλεσε την Εισαγωγή
7:          if S καλύτερη από τη BestSolution then
8:              BestSolution ← S
9:              R ← 1
10:             timesNotImproved ← 0
11:         else
12:             timesNotImproved ← timesNotImproved+1
13:         Εκτέλεσε την Διαταραχή
14:         S ← S+R
15:         R ← R+1
16:         if S ⩾ μήκος μικρότερης διαδρομής then
17:             S ← S - μήκος μικρότερης διαδρομής
18:         if R > μήκος μεγαλύτερης διαδρομής/2 then
19:             R ← 1
```

4. Διαχωρισμός Τοπικής Αναζήτησης

Όπως προαναφέρθηκε στο 3ο Κεφάλαιο, ένα από τα σημαντικότερα στοιχεία της Επαναλαμβανόμενης Τοπικής Αναζήτησης είναι φυσικά η Τοπική Αναζήτηση. Μάλιστα είναι και το πιο χρονοβόρο καθώς κατά τη διάρκεια της πραγματοποιούνται πολλοί έλεγχοι που είναι από τις πιο χρονοβόρες πράξεις για έναν επεξεργαστή.

Στόχος της τρέχουσας εργασίας, είναι να βελτιώσει την ταχύτητα του ILS. Η διαδικασία αυτή μπορεί να περιγραφεί με 3 βασικά βήματα:

1. Διαχωρισμός των n Unvisited κόμβων σε S υπογραφήματα
2. Σειριακή εφαρμογή Τοπικής Αναζήτησης σε κάθε υπογράφημα
3. Σειριακή εφαρμογή Διαταραχής σε κάθε υπογράφημα

Η προσπάθεια αυτή εγείρει διάφορα προβλήματα σε κάθε βήμα. Όσον αφορά το πρώτο βήμα, υπενθυμίζεται πως η τρέχουσα εργασία αντιμετωπίζει το TOPTW και τα χρονικά παράθυρα περιορίζουν αρκετά τον τρόπο με τον οποίο θα διαχωριστεί το γράφημα. Για παράδειγμα, εάν ο διαχωρισμός γίνει με μοναδικό κριτήριο την τοποθεσία των κόμβων με τη βοήθεια κάποιου αλγορίθμου (k-means), τότε υπάρχει η πιθανότητα να προκύψει μια συστάδα κόμβων με αταίριαστα χρονικά παράθυρα που θα παρήγαγε μια χαμηλής αξίας διαδρομή. Αντίθετα, εάν το μοναδικό κριτήριο ήταν τα χρονικά παράθυρα, τότε μπορεί να προέκυπταν συστάδες με κόμβους διάσπαρτους μεταξύ τους, και λύσεις χαμηλής αξίας.

Όσον αφορά το δεύτερο βήμα, αποτελεί πρόβλημα το γεγονός ότι δεν υπάρχει αρχικός και τελικός κόμβος σε κάθε κλάση για να εφαρμοσθεί απευθείας ο ILS που περιεγράφηκε στο Κεφάλαιο 3. Προφανώς, δεν είναι σοφό να θεωρηθεί ο ουδέτερος σταθμός (depot) του πρωτότυπου προβλήματος ως σταθμός για όλα τα υποπροβλήματα, καθώς κάθε διαδρομή σε κάθε υπογράφημα θα ξεκινούσε και θα τελείωνε στο ίδιο σημείο με αποτέλεσμα να αγνοούνται κόμβοι μακριά από αυτό και να περιορίζεται σημαντικά ο χώρος των λύσεων. Οι αρχικές ιδέες για να αντιμετωπιστεί το πρόβλημα αυτό ήταν δύο:

- Ανάθεση τους κεντροειδούς της συστάδας ως αρχικός και τελικός σταθμός της συστάδας αυτής
- Ανάθεση ενός κόμβου από την περίμετρο της συστάδας, ως αρχικός και τελικός σταθμός της συστάδας αυτής

Επίσης, μια ιδέα για την αντιμετώπιση του προβλήματος αυτού, ήταν να θεωρηθεί το κάθε υπό-πρόβλημα ως Πρόβλημα Προσανατολισμού με επιλογή Ξενοδοχείων (Orienteering Problem with Hotel Selection - OPHS), στο οποίο οι σταθμοί δεν είναι προκαθορισμένοι, καθώς υπάρχει ένα σύνολο ξενοδοχείων από το οποίο επιλέγεται ένας αρχικός και ένας τελικός κόμβος. Για την τρέχουσα περίπτωση, θα μπορούσε να παράγεται ένα αντίγραφο της λίστας Unvisited ως σύνολο ξενοδοχείων για κάθε υπογράφημα, αλλά φυσικά με μηδενικό κέρδος και χρονική διάρκεια επίσκεψης.

Για τη συνέχεια του Κεφαλαίου θα χρησιμοποιείται ο όρος “λύση διαστήματος” για να περιγράψει μία λύση ενός υπογραφήματος. Επίσης, οι κόμβοι, όπως θα αναλυθεί στην πορεία, χωρίζονται με βάση τα χρονικά διαστήματα που έχουν καθοριστεί στη φάση αρχικοποίησης των

χρονικών υποδιαστημάτων (Ενότητα 4.1), οπότε οι έννοιες υποδιάστημα, υποπρόβλημα και υπογράφημα είναι παρεμφερείς και θα χρησιμοποιούνται συχνά.

Μια λύση διαστήματος περιλαμβάνει τα εξής:

- Μια λίστα Unvisited που περιέχει τους κόμβους που δεν έχουν μπει ακόμη στις διαδρομές του διαστήματος
- Ένα διάνυσμα λιστών Walks που αναπαριστούν τις διαδρομές της λύσης του διαστήματος
    - Η ώρα αναχώρησης (depTime) του πρώτου κόμβου κάθε λίστας Walk αναπαριστά την ώρα έναρξης της διαδρομής
    - Η ώρα κλεισίματος (closeTime) του τελευταίου κόμβου κάθε λίστας Walk αναπαριστά την μέγιστη ώρα λήξης της διαδρομής.

Το άθροισμα των κερδών των κόμβων του διανύσματος Walks αποδίδει το συνολικό σκορ της λύσης.

Η διαχωρισμένη Τοπική Αναζήτηση αλλάζει τον ILS ως εξής:

Algorithm 5 Διαχωρισμένη Επαναλαμβανόμενη Τοπική Αναζήτηση

```
procedure SPLIT ILS(maxTimesNotImproved, numOfIntervals)
    pool ← pois
    timesNotImproved ← 0
    intervals ← Αρχικοποίηση χρονικών υποδιαστημάτων(numOfIntervals)
    h ← Αρχικοποίηση ιστορικού καταλληλότητας(pool, intervals)
    ac ← Αρχικοποίηση ενεργητικότητας κόμβων(pool, intervals)
    pS ← Αρχικοποίηση λύσεων διαστημάτων(numOfIntervals)
    sS ← Αρχικοποίηση παραμέτρων διαταραχής(numOfIntervals)
    while timesNotImproved < maxTimesNotImproved do
        pS ← Διαχωρισμός των unvisited κόμβων(pool, h, ac)
        pS ← Διαχωρισμένη τοπική αναζήτηση(pS)
        score ← Συλλογή σκορ(pS)
        if score > bestScore then
            bS ← pS
            for all sS do
                R ← 1
            end for
            timesNotImproved ← 0
        else
            timesNotImproved ← timesNotImproved + 1
        end if
        pS ← Διαχωρισμένη διαταραχή(pS, sS)
    end while
end procedure
```

- Η μεταβλητή intervals είναι ένα διάνυσμα που κρατάει τα χρονικά διαστήματα στα οποία χωρίζεται το πρόβλημα. Για παράδειγμα, εάν η παράμετρος S έχει τη τιμή 2, και το χρονικό περιθώριο του πρωτότυπου προβλήματος είναι timebudget = [0-1000], τότε μπορεί προκύψουν δύο καινούρια υπο-προβλήματα με διαστήματα [0-500] και [500-1000]. Το άθροισμα των παραγόμενων διαστημάτων, ισούται πάντα με τη συνολική διάρκεια του αρχικού προβλήματος. Στην ενότητα 4.1 περιγράφεται η διαδικασία καθορισμού των παραγόμενων χρονικών υποδιαστημάτων.
- Η μεταβλητή activities είναι ένα map που κρατάει την ενεργή διάρκεια του κάθε σημείου ενδιαφέροντος (POI) σε κάθε υποδιάστημα που έχει προκύψει από την αρχικοποίηση των χρονικών υποδιαστημάτων.
- Η μεταβλητή history είναι ένα map που κρατάει ένα ιστορικό "καταλληλότητας" για κάθε κόμβο σχετικά με κάθε διάστημα. Εάν ένας κόμβος είναι ενεργός σε πολλά διαστήματα, τότε κρατείται στο map history ένα ιστορικό του κόμβου ως προς κάθε διάστημα. Το ιστορικό αυτό κρατάει τις φορές που ο κόμβος μπήκε ως Unvisited σε ένα διάστημα, και τις φορές που προστέθηκε σε κάποια από τις διαδρομές της λύσης του διαστήματος. Η ανάθεση ενός κόμβου σε κάποιο υποδιάστημα με βάση το ιστορικό του, αναλύεται περαιτέρω στην Ενότητα 4.2.
- Η μεταβλητή processSolutions είναι ένα διάνυσμα που κρατάει τις λύσεις διαστημάτων που προκύπτουν κατά τη διάρκεια του αλγορίθμου. Σε περίπτωση που προκύψει ένα καλύτερο σκορ από τη διαχωρισμένη τοπική αναζήτηση (SplitSearch), τα processSolutions αποθηκεύονται σε ένα καινούριο διάνυσμα bestSolutions.
- Η μεταβλητή shakeSettings είναι ένα διάνυσμα που κρατάει τις τιμές S και R για κάθε λύση διαστήματος. Η διαδικασία διαταραχής εφαρμόζεται σε κάθε λύση διαστήματος ξεχωριστά. Για το λόγο αυτό, για κάθε λύση διαστήματος απαιτούνται διαφορετικά S και R . Το S αναπαριστά τη θέση από όπου θα ξεκινήσει η αφαίρεση των κόμβων σε κάθε διαδρομής μιας λύσης διαστήματος, και το R αναπαριστά τον αριθμό των κόμβων που θα αφαιρεθούν. Ο τρόπος ρύθμισης των S και R περιγράφεται στην Ενότητα 4.5.

4.1 Αρχικοποίηση των χρονικών υποδιαστημάτων

Ο πιο απλός τρόπος για να οριοθετηθούν S χρονικά υποδιαστήματα είναι να χωριστεί το χρονικό απόθεμα (timeBudget) του αρχικού προβλήματος ισόποσα σε ακριβώς S διαστήματα. Για παράδειγμα, εάν το αρχικό πρόβλημα έχει ένα timeBudget=[0-1000] και S=4 τότε θα προκύψουν 4 υποδιαστήματα με διάρκεια 250 χρονικών μονάδων: [0-250], [250-500], [500-750] και [750-1000].

Παρ' όλα αυτά, όπως θα αναλυθεί περαιτέρω και στην επόμενη Ενότητα (4.2), η διάρκεια ενεργητικότητας ενός κόμβου στα υποδιαστήματα παίζει σημαντικό ρόλο για την ανάθεση του σε ένα από αυτά. Εάν λοιπόν, δε ληφθεί υπόψιν η ενεργητικότητα των κόμβων στην οριοθέτηση των υποδιαστημάτων, υπάρχει ο κίνδυνος να καταλήξει μια μεγάλη μερίδα των κόμβων σε κάποιο από τα υποδιαστήματα. Εάν φυσικά, σε μια ακραία περίπτωση, όλοι οι κόμβοι καταλήξουν σε ένα υποδιάστημα, τότε το μόνο που θα έχει επιτευχθεί πρακτικά είναι να έχει μειωθεί το χρονικό απόθεμα του πρωτότυπου προβλήματος.

Για το σκοπό αυτό, υλοποιήθηκε ένας απλός ευρετικός που προσπαθεί να οριοθετήσει τα χρονικά διαστήματα μειώνοντας, επαναλαμβανόμενα, τα όρια του διαστήματος με τους περισσότερους κόμβους.

Πιο συγκεκριμένα, αρχικά ορίζεται η αντικειμενική συνάρτηση:

$$minimize\ P = maxCount - minCount \quad (4.1)$$

όπου maxCount αριθμός των κόμβων στο υποδιάστημα με τους περισσότερους κόμβους, και minCount ο αριθμός των κόμβων στο υποδιάστημα με τους λιγότερους κόμβους. Ο υλοποιηθείς αλγόριθμος προσπαθεί να μειώσει τη διαφορά αυτή μειώνοντας το υποδιάστημα στο οποίο αντιστοιχεί το maxCount. Παρ ‘όλα αυτά τα ενδιάμεσα διαστήματα έχουν δυο μεταβλητά όρια, ένα αριστερό και ένα δεξί. Οπότε πρέπει να καθοριστεί ο τρόπος με τον οποίο θα μειωθεί το υποδιάστημα, δηλαδή πόσο θα μεταβληθεί το αριστερό του όριο και πόσο το δεξί.

Έστω ένα υποδιάστημα $interval_i$ στο οποίο αντιστοιχεί το maxCount, $duration_i$ η διάρκεια του $interval_i$ και $count_i$ το πλήθος των κόμβων που αντιστοιχεί στο $interval_i$($count_i = maxCount$). Το $interval_i$ θα μειωθεί κατά $reduce_i = a * duration_i$ όπου α μια σταθερά με τιμή 0.2. Εάν η μείωση του διαστήματος $interval_i$ οδηγήσει σε μείωση του P (σχέση 4.1), τότε τα νέα όρια των διαστημάτων θεωρούνται τα καινούρια βέλτιστα και η διαδικασία επαναλαμβάνεται. Ο αλγόριθμος σταματάει όταν η τιμή του P δε μειωθεί για N συνεχόμενες φορές. Για τα πειραματικά αποτελέσματα του Κεφαλαίου 5, το N είχε τη τιμή 25.

4.2 Διαχωρισμός των Unvisited κόμβων

Η διαδικασία διαχωρισμού των Unvisited κόμβων στα S υπο-προβλήματα/διαστήματα γίνεται με βάση το ιστορικό καταλληλότητας (history) και την ενεργητικότητα (activities) του κάθε κόμβου σε κάθε διάστημα.

Την πρώτη φορά που θα κληθεί η συνάρτηση SplitUnvisited, η λίστα pool θα περιέχει όλους τους Unvisited κόμβους του προβλήματος, ενώ οι διαδρομές των processSolutions θα είναι άδειες. Ο εκάστοτε κόμβος θα ανατεθεί στο διάστημα όπου έχει το μεγαλύτερο σκορ. Το σκορ ενός κόμβου i σε ένα διάστημα k υπολογίζεται ως εξής:

$$score_{ik} = activityRatio_{ik} \cdot \frac{S_{ik}}{N_{ik}} \quad (4.2)$$

- $activityRatio_{ik}$: Απεικονίζει το ποσοστό της ενεργής διάρκειας του κόμβου i στο διάστημα k σε σχέση με τη συνολική ενεργή διάρκειά του. Εάν ο κόμβος i είναι ανενεργός καθ’ όλη τη διάρκεια του διαστήματος k, τότε προφανώς το $activityRatio_{ik}$ άρα και το $score_{ik}$ θα έχουν τιμή 0 σε κάθε επανάληψη. Οπότε, ο κόμβος i δεν θα εξεταστεί ποτέ για εισαγωγή στο διάστημα k.
- N_{ik}: Απεικονίζει τις φορές που ο κόμβος i επιλέχθηκε για το διάστημα k. Για τους κόμβους που είναι ενεργοί σε πολλά διαστήματα, επιδιώκεται να εξεταστεί η εισαγωγή τους, σε όσο το δυνατόν περισσότερα. Εάν λοιπόν, το N_{ik} συνεχίσει να αυξάνεται, το $score_{ik}$ θα μειώνεται και θα δοθεί η δυνατότητα να εξεταστεί η εισαγωγή του κόμβου i και στα υπόλοιπα διαστήματα που είναι ενεργός.
- S_{ik}: Απεικονίζει τις φορές που ο κόμβος i μπήκε στη λύση του διαστήματος k.

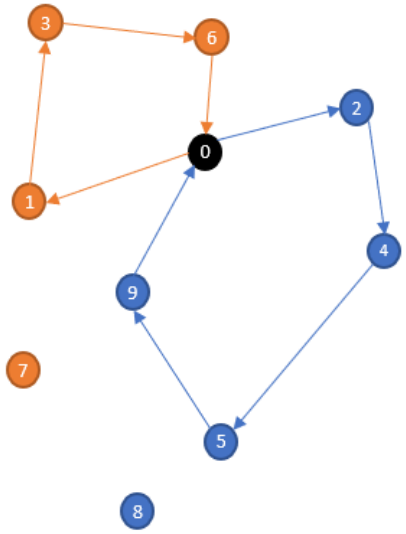
Ο λόγος S_{ik}/N_{ik} εκφράζει την καταλληλότητα του κόμβου i ως προς το διάστημα k.

4.3 Διαχωρισμένη Τοπική Αναζήτηση

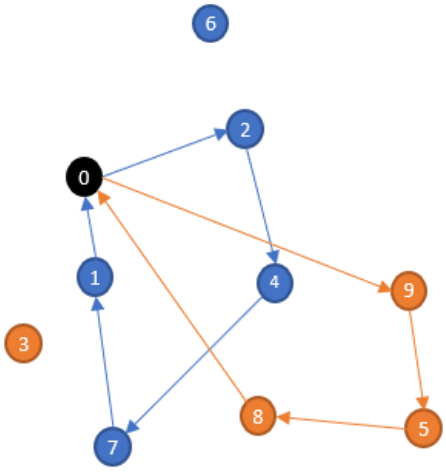
Όπως προαναφέρθηκε και στην αρχή του Κεφαλαίου, ένα σημαντικό θέμα που προκύπτει όταν χωριστεί το γράφημα, είναι το ποιος ή το ποιοι κόμβοι θα θεωρηθούν ως αφετηρία (sd) και τερματισμός (ed) του κάθε υποπροβλήματος.

Η πιο απλή υλοποίηση είναι να χρησιμοποιηθεί η αφετηρία και ο τερματισμός του πρωτότυπου προβλήματος ως αφετηρία και τερματισμός για κάθε υποπρόβλημα. Προφανώς η λύση αυτή δεν είναι αποδοτική, καθώς περιορίζεται σημαντικά ο χώρος των λύσεων. Στόχος κάθε διαδρομής θα είναι να καταλήγει πίσω στο depot οπότε οι κόμβοι που βρίσκονται σχετικά μακριά από αυτό, θα είναι δυσκολότερο να επιλεχθούν, λαμβάνοντας υπόψιν πάντα πως το χρονικό παράθυρο κάθε υπο-προβλήματος είναι μικρότερο ή ίσο του αρχικού προβλήματος. Επίσης, σε περίπτωση που οι κόμβοι του επόμενου υποπροβλήματος βρίσκονται μακριά από το τερματικό σταθμό, οι διαδρομές του τρέχοντος υποπροβλήματος θα έχουν οδηγηθεί χωρίς λόγο πίσω στην αφετηρία, καταλήγοντας έτσι σε χειρότερες λύσεις.

Έστω λοιπόν ένα πρόβλημα TOPTW με αφετηρία έναν κόμβο sd και τερματισμό έναν κόμβο ed το οποίο θα χωριστεί σε S υποπροβλήματα TOPTW. Οι διαδρομές του πρώτου υποπροβλήματος ($toptw_0$) θα ξεκινούν όλες από τον κόμβο sd που είναι ορισμένος στο πρωτότυπο πρόβλημα. Για κάθε άλλο υποπρόβλημα $toptw_i$ με $i \in [1, S-1]$, προστίθενται στις διαδρομές τεχνητές αφετηρίες όπως περιγράφεται στην υποενότητα 4.3.2. Επίσης για όλα τα υποπροβλήματα κατασκευάζονται τελικοί στόχοι, οι οποίοι δεν λειτουργούν ως τερματικοί σταθμοί για τις διαδρομές. Η διαδικασία αυτή αναλύεται λεπτομερώς στην υποενότητα 4.3.1. Η προεργασία που αναλύεται στις ακόλουθες υποενότητες, και η φάση εισαγωγής(τοπική αναζήτηση) που αναλύθηκε στην υποενότητα 3.2.1, εφαρμόζεται σειριακά σε κάθε υποπρόβλημα.



Εικόνα 4-1: Σε αυτό το παράδειγμα η επιστροφή στην αφετηρία σε κάθε υποπρόβλημα δεν είναι τόσο χρονοβόρα καθώς οι κόμβοι των 2 υποπροβλημάτων είναι αντισυμμετρικοί ως προς τον αρχικό κόμβο (0).



Εικόνα 4-2: Σε αυτό το παράδειγμα η επιστροφή στον αρχικό κόμβο σε κάθε υποπρόβλημα και η μετάβαση στους κόμβους του δεύτερου υποπροβλήματος καταλαμβάνει μεγάλα χρονικά διαστήματα.

4.3.1 Προσθήκη τελικών στόχων

Ένας τρόπος καθορισμού τελικού κόμβου για ένα υποπρόβλημα $toptw_i$, είναι να υπολογισθεί το σταθμισμένο κεντροειδές του επόμενου υποπροβλήματος ($toptw_{i+1}$) το οποίο είναι ένα τεχνητό σημείο που προκύπτει από τους Unvisited κόμβους λαμβάνοντας υπόψιν και τα κέρδη τους. Με αυτό τον τρόπο, η κάθε διαδρομή ενός υποπροβλήματος θα οδηγείται προς το πιο κερδοφόρο κέντρο βάρους του επόμενου. Η υλοποίηση αυτή όμως έχει αρκετά μειονεκτήματα, σχεδιαστικά και προγραμματιστικά.

- Στην αρχή του προγράμματος, για τα έτοιμα στιγμιότυπα εισόδου (π.χ. Cordeau) χρησιμοποιούνται οι ευκλείδειες αποστάσεις μεταξύ όλων των κόμβων του γραφήματος

ως χρόνοι ταξιδιού. Δηλαδή, εάν το στιγμιότυπο του προβλήματος έχει 100 κόμβους, τότε θα αρχικοποιηθεί ένας δισδιάστατος πίνακας travelTimes μεγέθους 100*100=10000 θέσεων. Ο πίνακας αυτός χρησιμοποιείται στη φάση της τοπικής αναζήτησης για την κατασκευή των διαδρομών. Όταν υπολογίζεται το σταθμισμένο κέντρο του επόμενου υποπροβλήματος, προστίθεται ως τελικός κόμβος στο τρέχον εξεταζόμενο υποπρόβλημα. Καθίσταται, λοιπόν, σαφές πως ο κόμβος αυτός και οι αποστάσεις του από τους υπόλοιπους πρέπει να προστεθούν στον πίνακα travelTimes καθώς πλέον ο καινούριος αυτός κόμβος αποτελεί μέρος του τρέχοντος υποπροβλήματος. Για να γίνει αυτό θα πρέπει να υπολογισθεί η απόστασή του από τα υπόλοιπα 100 σημεία. Παρ' όλα αυτά στο τρέχον υποπρόβλημα $toptw_i$ μπορεί να υπάρχουν μόνο 20 κόμβοι οπότε οι υπόλοιποι 80 υπολογισμοί είναι αχρείαστοι. Αυτό φυσικά μπορεί να αποφευχθεί εάν κατασκευάζεται κάθε φορά ένας μικρότερος πίνακας για κάθε υποπρόβλημα με βάση πάντα τον πρωτότυπο πίνακα travelTimes.
Επίσης οι Unvisited κόμβοι, όπως εξηγείται και στην Ενότητα 4.2, μπορούν να αλλάξουν υποπρόβλημα κατά τη διάρκεια του αλγορίθμου με βάση το ιστορικό καταλληλότητας. Οπότε σε κάθε επανάληψη, πρέπει να υπολογιστούν καινούρια σταθμισμένα κέντρα καθώς τα παλιά πλέον πιθανότατα να είναι παρωχημένα καθώς οι λίστες Unvisited μπορεί να αλλάζουν σε κάθε επανάληψη.

- Μετά από την Τοπική Αναζήτηση, αφότου γεμίσει η κάθε διαδρομή, εφαρμόζεται η Διαταραχή όπου αφαιρούνται κόμβοι και δημιουργούντα χρονικά κενά, όπου μετά την ενημέρωση των χρόνων άφιξης, αναχώρησης κ.λπ., μετατοπίζονται προς το τέλος των διαδρομών αφήνοντας χώρο για καινούριες εισαγωγές στην επόμενη επανάληψη. Παρ' όλα αυτά, δεν εγγυάται κανείς πως η εισαγωγή ενός κόμβου στο τέλος μιας διαδρομής μπορεί να είναι εφικτή, καθώς μπορεί ο χρόνος ταξιδιού από τον τελευταίο κόμβο της τρέχουσας διαδρομής προς το σταθμισμένο κεντροειδές του επόμενου υποπροβλήματος να είναι μεγαλύτερο από το μέγεθος του διαθέσιμου χρονικού παραθύρου. Μία λύση για αυτήν την περίπτωση είναι ο υπολογισμός ενός ενδιάμεσου κόμβου όπου θα είναι δυνατή η άφιξη σε αυτόν.
  Για παράδειγμα, έστω ένα πρόβλημα OPTW (μόνο μία διαδρομή) με χρονικό απόθεμα $timeBudget = [0 - 1000]$, χωρισμένο σε δύο διαστήματα/προβλήματα $optw_a$ και $optw_b$ με χρονικά παράθυρα $timeBudget_a = [0 - 500]$ και $timeBudget_b = [500 - 1000]$ αντίστοιχα. Έστω πως ο τελευταίος κόμβος της διαδρομής του $optw_a$ είναι ο z με ώρα αναχώρησης $depTime_z = 420$ και $maxShift_z = 80$ και έστω cnext το σταθμισμένο κεντροειδές του $optw_b$ με χρονική απόσταση από το z 100 μονάδες χρόνου ($travelTime_{z \to cnext} = 100$). Ο τελικός κόμβος ed της διαδρομής του $optw_a$ υπολογίζεται από τις σχέσεις:

$$t = \frac{maxShift_z}{travelTime_{z \to cnext}} \quad \textbf{(4.3)}$$

$$(x, y)_{ed} = ((1 - t) \cdot x_z + t \cdot x_{cnext}, (1 - t) \cdot y_z + t \cdot y_{cnext}) \quad \textbf{(4.4)}$$

Από τη διαδικασία αυτή, θα προκύψει ο τελικός κόμβος ed με χρόνο άφιξης και αναχώρησης $arrTime_{ed} = depTime_{ed} = 500$ και μηδενική διάρκεια επίσκεψης ($visitDuration = 0$). Παρ' όλο, που η διαδρομή πλέον είναι έγκυρη, ουσιαστικά δεν υπάρχει πλέον διαθέσιμος χρόνος για άλλες εισαγωγές, εκτός από τους χρόνους

αναμονής και μάλιστα ακριβώς πριν από την φάση της Τοπικής Αναζήτησης. Πρέπει λοιπόν να προστεθεί μια παράμετρος α στη σχέση 4.3 που θα ρυθμίζει το ποσοστό του διαθέσιμου χρόνου που θα καταλαμβάνει η εισαγωγή του ed. Εάν στο παραπάνω παράδειγμα τεθεί α=0.5, τότε θα προκύψει ένας κόμβος ed με χρόνο άφιξης $arrTime_{ed} = 460$ οπότε θα υπάρξει διαθέσιμος χώρος και για άλλες εισαγωγές στην Τοπική Αναζήτηση.

Για την αντιμετώπιση των παραπάνω προβλημάτων, εν τέλει θεωρήθηκε ένας άλλος τρόπος κατά τον οποίο δεν προστίθεται κάποιος τελικός κόμβος. Όπως αναφέρθηκε και στο Κεφάλαιο 2, ένα Πρόβλημα Προσανατολισμού μπορεί να έχει σταθερή αφετηρία και τερματισμό, σταθερή αφετηρία χωρίς τερματισμό (rooted) ή να μην είναι γνωστή ούτε η αφετηρία ούτε ο τερματισμός. Όλα τα υποπροβλήματα λοιπόν αντιμετωπίζονται ως rooted TOPTW δηλαδή οι διαδρομές είναι ανοικτές και είναι δυνατή η εισαγωγή κόμβων ακόμα και στο τέλος των διαδρομών.

Φυσικά, η μετατροπή των υποπροβλημάτων σε rooted TOPTW μπορεί να χειροτερέψει την ποιότητα των λύσεων καθώς όπως είναι μη αποδοτικό το να γυρίσει μια διαδρομή πίσω στην αφετηρία, άλλο τόσο είναι και το να προστίθενται κόμβοι χωρίς να λαμβάνεται καθόλου υπόψιν το που βρίσκονται οι κόμβοι του επόμενου υποπροβλήματος. Στη χειρότερη περίπτωση, οι κόμβοι του επόμενου προβλήματος μπορεί να βρίσκονται εντελώς αντίθετα από την κατεύθυνση που ακολουθεί η τρέχουσα διαδρομή. Για το λόγο αυτό, χρειαζόταν ένας τρόπος να οδηγηθεί η τρέχουσα διαδρομή προς τη σωστή κατεύθυνση αλλά χωρίς την εισαγωγή ενός τελικού κόμβου, που θα δέσμευε σημαντικό χρόνο από την επίσκεψη κερδοφόρων κόμβων.

Όπως αναλύθηκε στο Κεφάλαιο 3, σε κάθε επανάληψη της Τοπικής Αναζήτησης του ILS, υπολογίζεται η καλύτερη θέση εισαγωγής για κάθε κόμβο, δηλαδή η θέση με το μικρότερο minShift και στη συνέχεια επιλέγεται ο κόμβος με το μεγαλύτερο $ratio_i = \frac{profit_i^2}{minShift_i}$. Η χρονική ολίσθηση της εισαγωγής ενός κόμβου j μεταξύ των κόμβων i και k υπολογίζεται από τη σχέση:

$$shift_j = travelTime_{i\to j} + wait_j + visitDur_j + travelTime_{j\to k} - travelTime_{i\to k} \quad (4.5)$$

Η εισαγωγή όμως ενός κόμβου στο τέλος της διαδρομής σημαίνει πως δεν υπάρχει κόμβος k. Άρα η παραπάνω σχέση για θέσεις εισαγωγής στο τέλος των διαδρομών μετατρέπεται ως εξής:

$$shift_j = travelTime_{i\to j} + wait_j + visitDur_j \quad (4.6)$$

Η θέση εισαγωγής, που δίνει το καλύτερο σκορ θεωρείται η βέλτιστη θέση για τον κόμβο j. Στον αλγόριθμο των Vansteenwegen et al. (2009) [6] το σκορ εξαρτάται μόνο από τη χρονική ολίσθηση που προκαλεί η εισαγωγή. Επειδή όμως χρειάζεται να ληφθεί υπόψιν και το σταθμισμένο κεντροειδές του επόμενου υποπροβλήματος (cnext), προστέθηκε ένας ακόμα παράγοντας που είναι η απόσταση του εξεταζόμενου κόμβου προς το cnext:

$$posScore_j = shift_j + distance(j, cnext) \quad (4.7)$$

Όμως δεν έχει νόημα να λαμβάνεται υπόψιν η απόσταση προς το cnext όταν εξετάζεται η εισαγωγή ενός κόμβου στην αρχή της τρέχουσας διαδρομής. Οπότε, η τελική σχέση που προκύπτει είναι η εξής:

$$posScore_j = shift_j + distance(j, cnext) \cdot (p/t) \quad (4.8)$$

Όπου p είναι το index της θέσης της εξεταζόμενης εισαγωγής στη διαδρομή και t ο συνολικός αριθμός των πιθανών θέσεων της τρέχουσας διαδρομής. Για παράδειγμα, εάν σε μια διαδρομή υπάρχουν 10 θέσεις εισαγωγής, τότε στην πρώτη θέση, η βαρύτητα της απόστασης του j προς τον cnext κόμβο θα είναι $\frac{1}{10}$, στη δεύτερη $\frac{2}{10}$ κ.ο.κ. , μέχρι την τελική όπου θα είναι $\frac{10}{10}$. Η θέση εισαγωγής, λοιπόν, με τη μικρότερη τιμή του posScore θα θεωρηθεί ως η βέλτιστη θέση εισαγωγής για τον κόμβο j.

4.3.2 Προσθήκη αρχικών κόμβων

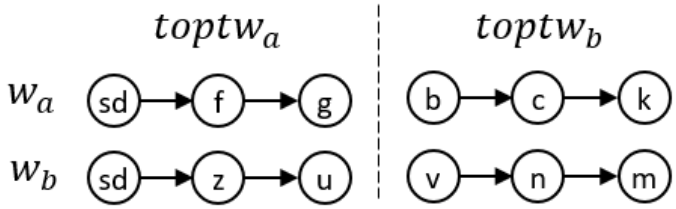
Στην προσθήκη αρχικών κόμβων επιλέγεται ουσιαστικά ο τελευταίος κόμβος που προέκυψε από την φάση κατασκευής του προηγούμενου διαστήματος. Και σε αυτή τη διαδικασία υπάρχουν όμως προβλήματα που πρέπει να επιλυθούν.

Έστω $z_{i,j}$ ο τελευταίος κόμβος μιας διαδρομής $R_{i,j}$ με $i \in [1, m]$ και $j \in [1, s]$ και έστω πως τη χρονική στιγμή t, ο αλγόριθμος εξετάζει το διάστημα I_{j+1}. Αρχικά θα θεωρηθεί ως υποψήφιος αρχικός κόμβος ($c_{i,j+1}$) της διαδρομής $R_{i,j+1}$ ένας κλώνος του κόμβου $z_{i,j}$ του διαστήματος I_j. Για να είναι έγκυρη η εισαγωγή του κόμβου $c_{i,j+1}$ στην αρχή του $R_{i,j+1}$ θα πρέπει να ισχύουν οι δύο παρακάτω προϋποθέσεις:

- Δεν παραβιάζονται οι χρόνοι του $c_{i,j+1}$
- Η ολίσθηση του χρόνου προς τα δεξιά λόγω της εισαγωγής του $c_{i,j+1}$ δεν παραβιάζει τους χρονικούς περιορισμούς των κόμβων που έπονται

Ο κόμβος $c_{i,j+1}$ θεωρείται ουδέτερος κόμβος, καθώς έχει μηδενική διάρκεια επίσκεψης και το χρονικό του παράθυρο είναι ίσο με το χρονικό παράθυρο του διαστήματος I_{j+1}. Οπότε είναι πρακτικά αδύνατο να παραβιαστούν οι χρονικοί περιορισμοί του $c_{i,j+1}$. Παρ ’όλα αυτά το $c_{i,j+1}$ διατηρεί τις συντεταγμένες του $z_{i,j}$ οπότε ο χρόνος ταξιδιού από τον $c_{i,j+1}$ προς τον τρέχοντα αρχικό κόμβο της διαδρομής $R_{i,j+1}$ θα προκαλέσει μια ολίσθηση των χρόνων άφιξης, αναχώρησης κ.λπ. των υπόλοιπων κόμβων της διαδρομής $R_{i,j+1}$ προς τα δεξιά. Εάν λοιπόν πράγματι παραβιάζεται κάποιος χρονικός περιορισμούς από τους επακόλουθους κόμβους, τότε αφαιρείται ο αρχικός κόμβος της διαδρομής $R_{i,j}$ και μεταφέρεται στη λίστα Unvisited του διαστήματος I_{j+1} .

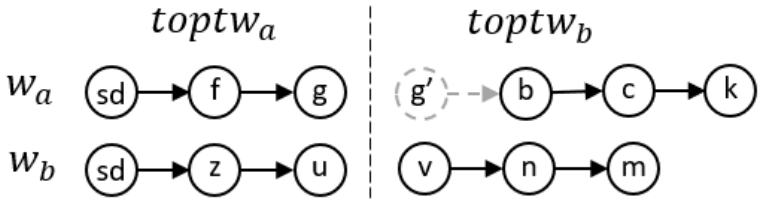
Έστω λοιπόν ένα πρόβλημα TOPTW που έχει χωριστεί σε 2 διαστήματα A και B και μια χρονική στιγμή t στην οποία ο αλγόριθμος έχει ήδη κατασκευάσει δύο διαδρομές και ετοιμάζεται για άλλη μια Τοπική Αναζήτηση στο υποπρόβλημα B.



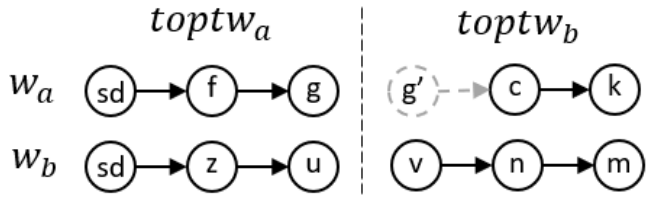
Η προεργασία που περιεγράφηκε παραπάνω, εφαρμόζεται σε κάθε διαδρομή του υποπροβλήματος σειριακά. Οπότε αρχικά θα εξεταστεί η διαδρομή w_a του προβλήματος $toptw_b$.

Αρχικά, ως αφετηρία κάθε τροχιάς του προβλήματος $toptw_b$, τοποθετείται ένας κλώνος του τελευταίου κόμβου της αντίστοιχης τροχιάς του προηγούμενου υποπροβλήματος, δηλαδή στο συγκεκριμένο παράδειγμα, οι κόμβοι g και u για τις τροχιές w_a και w_b αντίστοιχα. Έστω g' ο κλώνος του κόμβου g και u' ο κλώνος του κόμβου u:

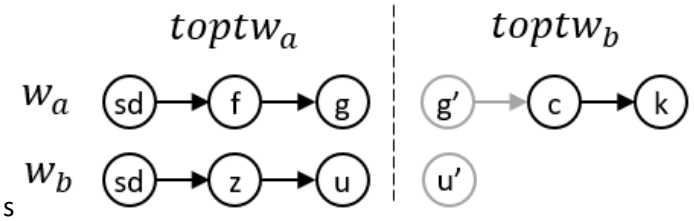
- $id_{g'} = dummy\_start\_depot$
- $visitDuration_{g'} = waitDuration_{g'} = 0$
- $arrTime_{g'} = depTime_{g'} = timeWindow_b.openTime$
- $timeWindow_{g'} = timeWindow_b$



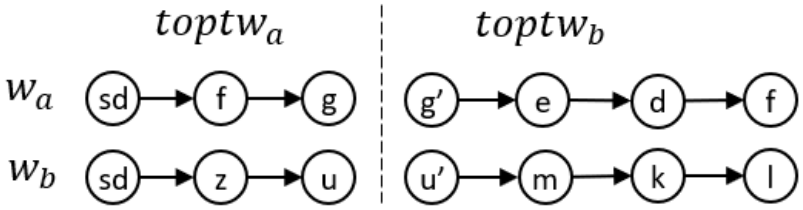
Εφόσον για την ώρα αναχώρησης του g' έχει οριστεί πως $depTime_{g'} = timeWindow_B.openTime$, η εισαγωγή του g' δεν παραβιάζει το χρονικό του παράθυρο. Παρ' όλα αυτά πρέπει να εξεταστεί εάν η εισαγωγή του g' προκαλεί κάποιο πρόβλημα στη συνέχεια της διαδρομής. Εάν όντως προκαλεί, τότε αφαιρείται ο πρώτος κόμβος της διαδρομής, δηλαδή στο συγκεκριμένο παράδειγμα ο κόμβος b.



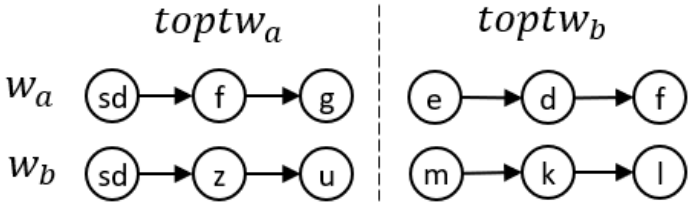
Αυτό θα συνεχιστεί μέχρι να είναι έγκυρη η διαδρομή $Walk_B$ με την εισαγωγή του g'. Όταν πλέον η εισαγωγή του g' είναι έγκυρη, τότε θα εισαχθεί στην αρχή της διαδρομής και η διαδικασία αυτή θα επαναληφθεί για τη διαδρομή w_b. Ακόμα και αν αφαιρεθούν όλοι οι κόμβοι από μία διαδρομή και μείνει μόνο ο ουδέτερος τεχνητός κόμβος του προηγούμενου διαστήματος, ο αλγόριθμος μπορεί να το διαχειριστεί καθώς όπως αναφέρθηκε και στην υποενότητα 4.3.1, εξετάζεται ακόμα και η θέση μετά τον τελευταίο κόμβο ως θέση εισαγωγής. Οπότε ο ελάχιστος αριθμός κόμβων που μπορεί να έχει μια διαδρομή είναι 1.



Αφότου τελειώσει αυτή η προεργασία στις διαδρομές του προβλήματος B, θα ακολουθήσει η διαδικασία της Τοπικής Αναζήτησης από την οποία προκύπτουν δύο νέες διαδρομές για το $toptw_b$.



Όταν τελειώσει η Τοπική Αναζήτηση, αφαιρούνται οι αρχικοί τεχνητοί κόμβοι των διαδρομών και ενημερώνονται οι χρόνοι όλων των κόμβων.



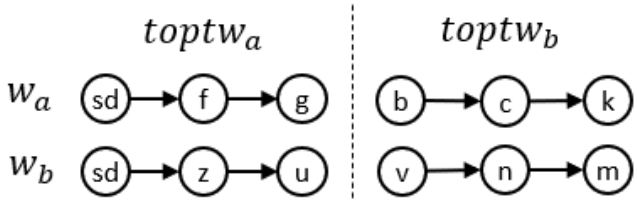
Υπενθυμίζεται πως είναι επιτρεπτές οι αφίξεις στους κόμβους πριν τα χρονικά τους παράθυρα αλλά όχι μετά. Οπότε, η ολίσθηση των χρόνων άφιξης, αναχώρησης κ.λπ. των κόμβων προς τα πίσω δε μπορεί να δημιουργήσει μη έγκυρες διαδρομές.

4.4 Υπερχείλιση και διόρθωση διαδρομών

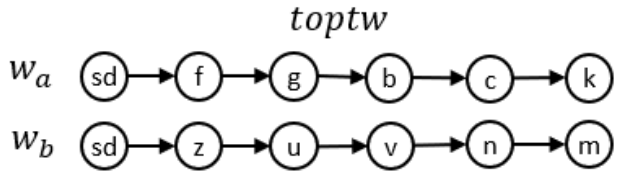
Όπως αναλύθηκε και στην ενότητα 4.3.1, σε κάθε υποδιάστημα, για κάθε διαδρομή θεωρείται ένας τελικός στόχος που είναι το σταθμισμένο κέντρο της αντίστοιχης διαδρομής του επόμενου υποδιαστήματος. Στο τελευταίο υποδιάστημα, ως στόχος για κάθε διαδρομή θεωρείται ο τελικός κόμβος (ed) του πρωτότυπου προβλήματος. Έτσι κάθε διαδρομή του τελευταίου υποδιαστήματος στοχεύει προς τον κόμβο ed αλλά δεν καταλήγει σε αυτόν. Το ζητούμενο του αλγορίθμου για το TOPTW που μελετά η παρούσα εργασία, είναι να δημιουργήσει m ολοκληρωμένες διαδρομές με αρχή και τέλος τους κόμβους sd και ed αντίστοιχα. Οπότε πρέπει να εξεταστεί εάν όταν ενωθούν οι διαδρομές των υποδιαστημάτων μεταξύ τους, και προστεθεί και ο τελικός κόμβος ed σε κάθε διαδρομή, ότι οι διαδρομές θα παραμείνουν έγκυρες. Επειδή η κάθε διαδρομή ενός υποδιαστήματος έχει τροποποιηθεί πριν την φάση της κατασκευής έτσι ώστε να είναι συμβατή με την αντίστοιχη διαδρομή του προηγούμενου υποδιαστήματος, η ενοποίηση των διαδρομών δεν θα προκαλέσει κάποιο πρόβλημα στους χρονικούς τους περιορισμούς . Όμως η αναγκαστική

εισαγωγή του ed στο τέλος των διαδρομών μπορεί να προκαλέσει κάποια διαδρομή να υπερβεί το χρονικό της απόθεμα.

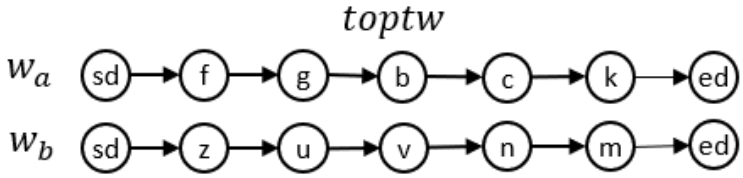
Για παράδειγμα, έστω ένα πρόβλημα TOPTW με χρονικό παράθυρο $timeBudget = [0 - 1000]$, χωρισμένο σε δύο υποπροβλήματα $toptw_a$ και $toptw_b$ με χρονικά παράθυρα $timeBudget_a = [0 - 500]$ και $timeBudget_b = [500 - 1000]$ αντίστοιχα. Από την τελευταία φάση της τοπικής αναζήτησης έχουν προκύψει οι εξής διαδρομές:



Επίσης έστω πως για τις ώρες αναχώρησης από τους τελευταίους κόμβους των διαδρομών του $toptw_b$ ισχύει πως $depTime_k = 970$ και $depTime_m = 950$. Όπως αναφέρθηκε στην υποενότητα 4.3.2, πριν την φάση της τοπικής αναζήτησης για το $toptw_b$, προστέθηκαν τεχνητοί αρχικοί κόμβοι g' και u' για τις δυο διαδρομές οι οποίοι μετά την φάση της κατασκευής αφαιρέθηκαν. Επειδή όμως ισχύει $depTime_{g'} \geq depTime_g$ και $depTime_{u'} \geq depTime_u$ και οι διαδρομές w_a και w_b του $toptw_b$ ήταν έγκυρες έχοντας ως αρχικούς κόμβους τους g' και u', σίγουρα θα είναι και έγκυρες όταν έπονται των κόμβων g και u. Από την ενοποίηση λοιπόν των διαστημάτων προκύπτουν οι εξής διαδρομές:



Για την καινούρια ώρα αναχώρησης από τους κόμβους k και m, ισχύει πως $depTime'_k \leq depTime_k \leq 1000$ και $depTime'_m \leq depTime_m \leq 1000$ οπότε οι διαδρομές είναι έγκυρες, όσον αφορά τουλάχιστον τα χρονικά αποθέματά τους. Έστω λοιπόν πως $depTime'_k = 950$ και $depTime'_m = 900$. Με την προσθήκη των τελικών κόμβων προκύπτουν οι διαδρομές:



Εάν $travelTime_{k \to ed} > 50$ ή $travelTime_{m \to ed} > 100$ τότε η λύση δεν είναι έγκυρη διότι κάποια από τις διαδρομές υπερβαίνει το χρονικό απόθεμα τους προβλήματος. Για το λόγο αυτό, μετά από κάθε τοπική αναζήτηση, εξετάζεται εάν μετά την ένωση των υποδιαστημάτων και την προσθήκη τελικών κόμβων προκύπτουν ανέφικτες διαδρομές. Εάν μια διαδρομή υπερβαίνει το χρονικό της περιθώριο, τότε αφαιρείται ο προτελευταίος κόμβος (πριν τον κόμβο ed) μέχρι η διαδρομή να γίνει έγκυρη.

4.5 Διαχωρισμένη Διαταραχή

Η διαταραχή όπως και η τοπική αναζήτηση, πραγματοποιείται σε κάθε υποπρόβλημα σειριακά. Όπως περιεγράφηκε και στο Κεφάλαιο 3, η διαταραχή γίνεται κυκλικά σε κάθε διαδρομή, δηλαδή στην πρώτη επανάληψη του ILS, θα αφαιρεθεί ένας κόμβος (R=1) ξεκινώντας από την πρώτη θέση της κάθε διαδρομής (S=1), ενώ στις επόμενες επαναλήψεις θα αφαιρούνται όλο και περισσότεροι κόμβοι ξεκινώντας από μεγαλύτερες θέσεις των διαδρομών. Η διαταραχή σε κάθε υποπρόβλημα γίνεται ανεξάρτητα από τα υπόλοιπα, καθώς κάθε υποπρόβλημα έχει τις δικές του παραμέτρους S και R.

Οι παράμετροι S_i και R_i κάθε υποδιαστήματος i, αυξάνονται μετά από κάθε διαταραχή ως εξής:

- Το S_i αυξάνεται κατά R_i
- Το R_i αυξάνεται κατά 1

Επίσης, οι παράμετροι S_i και R_i επαναφέρονται στις αρχικές τους τιμές στις εξής περιπτώσεις:

- Εάν $S_i \geq minWalkSize_i$ τότε $S_i = 1$
- Εάν $R_i = \max ToRemove_i$ τότε $R_i = 1$

Όπου $minWalkSize_i$ είναι το μήκος της μικρότερης διαδρομής του υποδιαστήματος i. Η μεταβλητή $maxToRemove_i$ υπολογίζεται από τη σχέση:

$$maxToRemove_i = numOfVisits_i/2 * numOfWalks$$

Όπου $numOfVisits_i$ ο αριθμός των κόμβων στις διαδρομές του υποδιαστήματος i και $numOfWalks$ ο αριθμός των διαδρομών.

5. Πειραματικά Αποτελέσματα

Οι υπολογισμοί έγιναν σε ένα t2.medium μηχάνημα της Amazon με επεξεργαστή 3.3 GHz Intel Xeon Scalable και μνήμη RAM 4GB. Για όλα τα παρακάτω παραδείγματα, η παράμετρος MAX_TIMES_NOT_IMPROVED ισούται με 150, δηλαδή ο αλγόριθμος Επαναλαμβανόμενης Τοπικής Αναζήτησης για κάθε παράδειγμα τερματίστηκε όταν η παραγόμενη λύση δεν βελτιώθηκε για 150 διαδοχικές επαναλήψεις.

5.1 Σύγκριση αποτελεσμάτων για διαφορετικά S

Η παράμετρος S καθορίζει τον αριθμό των υποπροβλημάτων που θα διαχωριστεί το αρχικό γράφημα. Εάν S=1, ο αλγόριθμος θα τρέξει ως ένας κανονικός αλγόριθμος ILS για ένα rooted TOPTW πρόβλημα. Σε κάθε πίνακα που ακολουθεί και για κάθε στιγμιότυπο εισόδου συμπεριλαμβάνεται το σκορ και ο χρόνος εκτέλεσης του ILS, που υλοποιήθηκε για τους σκοπούς της τρέχουσας εργασίας, με $S \in \{1,2,3,4\}$, το αντίστοιχο σκορ του ILS των Vansteenwegen et al. (2009) [6], και το σκορ της καλύτερης γνωστής λύσης με βάση τα πειραματικά αποτελέσματα των Karabalut et al.(2020) [39].

Πίνακας 5-1: Πειραματικά αποτελέσματα για τα στιγμιότυπα εισόδου των Cordeau, Gendreau και Laporte (m = 1)

	BK	ILS (2009)	S=1			S=2				S=3				S=4			
Name	Score	Score	Score		CPU(s)	Score		CPU(s)		Score		CPU(s)		Score		CPU(s)	
			Value	Gap %(BK)	Value	Value	Gap (%)	Value	Gap (%)	Value	Gap (%)	Value	Gap (%)	Value	Gap (%)	Value	Gap (%)
pr01	308	304	242	21.43	0.088	281	-16.12	0.087	1.14	274	-13.22	0.08	9.09	229	5.37	0.074	15.91
pr02	404	385	375	7.18	0.222	328	12.53	0.172	22.52	322	14.13	0.171	22.97	295	21.33	0.21	5.41
pr03	394	384	376	4.57	0.37	366	2.66	0.531	-43.51	350	6.91	0.288	22.16	309	17.82	0.3	18.92
pr04	489	447	478	2.25	1.095	447	6.49	0.492	55.07	411	14.02	0.436	60.18	433	9.41	0.423	61.37
pr05	595	576	524	11.93	1.13	511	2.48	0.639	43.45	486	7.25	1.119	0.97	504	3.82	1.143	-1.15
pr06	590	538	574	2.71	1.131	543	5.4	0.931	17.68	518	9.76	0.752	33.51	460	19.86	0.679	39.96
pr07	298	291	261	12.42	0.141	251	3.83	0.123	12.77	229	12.26	0.123	12.77	251	3.83	0.125	11.35
pr08	463	463	447	3.46	0.508	389	12.98	0.332	34.65	417	6.71	0.35	31.1	355	20.58	0.282	44.49
pr09	493	461	424	14	0.822	416	1.89	0.699	14.96	333	21.46	0.585	28.83	322	24.06	0.427	48.05
pr10	594	539	520	12.46	1.188	519	0.19	1.413	-18.94	472	9.23	0.659	44.53	440	15.38	0.637	46.38
pr11	353	330	319	9.63	0.109	308	3.45	0.118	-8.26	274	14.11	0.083	23.85	285	10.66	0.09	17.43
pr12	442	431	424	4.07	0.371	418	1.42	0.246	33.69	407	4.01	0.533	-43.67	403	4.95	0.449	-21.02
pr13	467	450	444	4.93	0.475	377	15.09	0.384	19.16	376	15.32	0.315	33.68	391	11.94	0.398	16.21
pr14	567	482	510	10.05	1.005	480	5.88	0.875	12.94	434	14.9	0.448	55.42	450	11.76	0.465	53.73
pr15	708	638	661	6.64	1.491	597	9.68	1.021	31.52	580	12.25	0.635	57.41	553	16.34	0.549	63.18
pr16	674	559	596	11.57	3.491	553	7.21	2.016	42.25	533	10.57	0.82	76.51	509	14.6	0.898	74.28
pr17	362	346	341	5.8	0.16	320	6.16	0.174	-8.75	285	16.42	0.124	22.5	258	24.34	0.129	19.38
pr18	539	479	447	17.07	0.492	507	-13.42	0.347	29.47	435	2.68	0.307	37.6	390	12.75	0.308	37.4
pr19	562	499	468	16.73	1.253	427	8.76	0.617	50.76	428	8.55	0.898	28.33	384	17.95	0.462	63.13
pr20	667	570	610	8.55	2.248	586	3.93	1.833	18.46	534	12.46	0.655	70.86	556	8.85	0.895	60.19

Πίνακας 5-2: Πειραματικά αποτελέσματα για τα στιγμιότυπα εισόδου των Cordeau, Gendreau και Laporte (m = 2)

	BK	ILS (2009)	S=1			S=2				S=3				S=4			
Name	Score	Score	Score		CPU(s)	Score		CPU(s)		Score		CPU(s)		Score		CPU(s)	
			Value	Gap %(BK)	Value	Value	Gap (%)	Value	Gap (%)	Value	Gap (%)	Value	Gap (%)	Value	Gap (%)	Value	Gap (%)
pr01	502	471	451	10.16	0.18	441	2.22	0.083	53.89	441	2.22	0.122	32.22	427	5.32	0.135	25
pr02	715	660	670	6.29	0.576	620	7.46	0.359	37.67	597	10.9	0.331	42.53	576	14.03	0.262	54.51
pr03	742	714	673	9.3	0.881	639	5.05	0.518	41.2	663	1.49	0.567	35.64	600	10.85	0.337	61.75
pr04	926	863	799	13.71	1.108	790	1.13	0.91	17.87	807	-1	0.596	46.21	745	6.76	0.542	51.08
pr05	1101	1011	1018	7.54	6.089	870	14.54	1.618	73.43	852	16.31	1.641	73.05	770	24.36	0.81	86.7
pr06	1076	997	1009	6.23	3.48	987	2.18	1.456	58.16	933	7.53	1.264	63.68	943	6.54	1.011	70.95
pr07	566	552	541	4.42	0.261	498	7.95	0.167	36.02	444	17.93	0.134	48.66	517	4.44	0.18	31.03
pr08	834	796	776	6.95	0.858	727	6.31	0.68	20.75	705	9.15	0.387	54.9	647	16.62	0.347	59.56
pr09	909	867	843	7.26	4.332	738	12.46	1.636	62.23	716	15.07	0.778	82.04	726	13.88	0.615	85.8

pr10	1134	1004	1016	10.41	2.843	961	5.41	1.719	39.54	955	6	1.218	57.16	908	10.63	1.097	61.41
pr11	566	542	525	7.24	0.111	502	4.38	0.082	26.13	456	13.14	0.082	26.13	473	9.9	0.081	27.03
pr12	774	727	700	9.56	0.942	690	1.43	0.445	52.76	665	5	0.236	74.95	655	6.43	0.261	72.29
pr13	843	757	771	8.54	2.423	737	4.41	0.525	78.33	693	10.12	0.382	84.23	681	11.67	0.44	81.84
pr14	1017	925	964	5.21	2.432	908	5.81	1.491	38.69	862	10.58	0.664	72.7	725	24.79	0.524	78.45
pr15	1220	1126	1086	10.98	2.526	1043	3.96	1.664	34.13	1028	5.34	1.026	59.38	959	11.69	0.981	61.16
pr16	1231	1110	1101	10.56	4.315	1030	6.45	2.122	50.82	984	10.63	1.125	73.93	954	13.35	1.811	58.03
pr17	652	624	587	9.97	0.228	567	3.41	0.175	23.25	518	11.75	0.164	28.07	503	14.31	0.159	30.26
pr18	953	877	825	13.43	0.91	878	-6.42	0.712	21.76	807	2.18	0.481	47.14	736	10.79	0.387	57.47
pr19	1034	955	969	6.29	2.738	818	15.58	1.91	30.24	772	20.33	0.831	69.65	739	23.74	0.613	77.61
pr20	1241	1056	1109	10.64	5.299	1084	2.25	1.884	64.45	996	10.19	1.162	78.07	999	9.92	1.296	75.54

Πίνακας 5-3: Πειραματικά αποτελέσματα για τα στιγμιότυπα εισόδου των Cordeau, Gendreau και Laporte (m = 3)

	BK	ILS (2009)	S=1			S=2				S=3				S=4			
Name	Score	Score	Score		CPU(s)	Score		CPU(s)		Score		CPU(s)		Score		CPU(s)	
			Value	Gap %(BK)	Value	Value	Gap (%)	Value	Gap (%)	Value	Gap (%)	Value	Gap (%)	Value	Gap (%)	Value	Gap (%)
pr01	622	598	598	3.86	0.173	540	9.7	0.122	29.48	553	7.53	0.085	50.87	518	13.38	0.099	42.77
pr02	943	899	858	9.01	1.467	834	2.8	0.332	77.37	808	5.83	0.287	80.44	764	10.96	0.292	80.1
pr03	1010	946	959	5.05	2.331	884	7.82	0.784	66.37	740	22.84	0.436	81.3	800	16.58	0.411	82.37
pr04	1294	1195	1178	8.96	2.845	1163	1.27	2.289	19.54	1128	4.24	0.79	72.23	1148	2.55	0.986	65.34
pr05	1482	1356	1314	11.34	2.917	1295	1.45	1.39	52.35	1284	2.28	1.996	31.57	1247	5.1	1.333	54.3
pr06	1514	1376	1401	7.46	4.848	1340	4.35	2.962	38.9	1344	4.07	1.667	65.61	1328	5.21	1.394	71.25
pr07	744	713	689	7.39	0.328	653	5.22	0.204	37.8	643	6.68	0.166	49.39	629	8.71	0.284	13.41
pr08	1139	1082	1047	8.08	0.981	1027	1.91	0.779	20.59	970	7.35	0.519	47.09	937	10.51	0.482	50.87
pr09	1282	1144	1138	11.23	2.568	1162	-2.11	2.104	18.07	1025	9.93	1.188	53.74	1052	7.56	1.006	60.83
pr10	1573	1473	1495	4.96	8.727	1322	11.57	1.938	77.79	1381	7.63	1.788	79.51	1284	14.11	1.553	82.2
pr11	654	632	630	3.67	0.111	617	2.06	0.092	17.12	580	7.94	0.078	29.73	563	10.63	0.102	8.11
pr12	1002	902	923	7.88	0.78	883	4.33	0.31	60.26	835	9.53	0.327	58.08	797	13.65	0.268	65.64
pr13	1152	1046	1063	7.73	1.167	1021	3.95	0.942	19.28	914	14.02	0.544	53.38	942	11.38	0.57	51.16
pr14	1372	1197	1247	9.11	1.838	1190	4.57	0.988	46.25	1095	12.19	0.71	61.37	1131	9.3	1.133	38.36
pr15	1659	1488	1534	7.53	4.203	1449	5.54	1.606	61.79	1425	7.11	1.297	69.14	1379	10.1	1.61	61.69
pr16	1668	1478	1508	9.59	8.306	1468	2.65	5.722	31.11	1466	2.79	3.028	63.54	1333	11.6	1.69	79.65
pr17	841	808	792	5.83	0.367	787	0.63	0.237	35.42	699	11.74	0.195	46.87	668	15.66	0.29	20.98
pr18	1282	1165	1181	7.88	2.424	1117	5.42	0.835	65.55	1003	15.07	0.52	78.55	970	17.87	0.542	77.64
pr19	1417	1238	1292	8.82	4.421	1254	2.94	1.948	55.94	1191	7.82	1.328	69.96	1137	12	0.947	78.58
pr20	1690	1514	1534	9.23	4.669	1509	1.63	2.531	45.79	1454	5.22	2.805	39.92	1376	10.3	1.268	72.84

Πίνακας 5-4: Πειραματικά αποτελέσματα για τα στιγμιότυπα εισόδου των Cordeau, Gendreau και Laporte (m = 4)

	BK	ILS (2009)	S=1			S=2				S=3				S=4			
Name	Score	Score	Score		CPU(s)	Score		CPU(s)		Score		CPU(s)		Score		CPU(s)	
			Value	Gap %(BK)	Value	Value	Gap (%)	Value	Gap (%)	Value	Gap (%)	Value	Gap (%)	Value	Gap (%)	Value	Gap (%)
pr01	657	644	649	1.22	0.184	605	6.78	0.08	56.52	610	6.01	0.142	22.83	583	10.17	0.117	36.41
pr02	1079	1014	1001	7.23	0.964	969	3.2	0.347	64	947	5.39	0.288	70.12	947	5.39	0.57	40.87
pr03	1246	1162	1118	10.27	1.408	1062	5.01	1.019	27.63	1028	8.05	0.658	53.27	1030	7.87	0.972	30.97
pr04	1585	1452	1483	6.44	7.67	1435	3.24	1.322	82.76	1366	7.89	1.15	85.01	1374	7.35	0.901	88.25
pr05	1844	1665	1640	11.06	4.069	1627	0.79	1.968	51.63	1532	6.59	1.298	68.1	1545	5.79	1.109	72.75
pr06	1886	1696	1695	10.13	12.217	1694	0.06	2.842	76.74	1683	0.71	2.076	83.01	1588	6.31	1.456	88.08
pr07	876	840	821	6.28	0.414	786	4.26	0.23	44.44	760	7.43	0.181	56.28	712	13.28	0.263	36.47
pr08	1385	1267	1286	7.15	1.585	1205	6.3	0.663	58.17	1154	10.26	0.56	64.67	1126	12.44	0.498	68.58
pr09	1619	1460	1417	12.48	2.929	1394	1.62	1.447	50.6	1393	1.69	1.7	41.96	1408	0.64	1.233	57.9
pr10	1943	1782	1784	8.18	7.365	1729	3.08	2.834	61.52	1690	5.27	1.948	73.55	1623	9.02	2.987	59.44
pr11	657	654	654	0.46	0.087	654	0	0.106	-21.84	640	2.14	0.094	-8.05	618	5.5	0.164	-88.51
pr12	1132	1041	1067	5.74	0.664	1025	3.94	0.682	-2.71	989	7.31	0.324	51.2	940	11.9	0.317	52.26
pr13	1386	1263	1269	8.44	1.755	1238	2.44	1.092	37.78	1176	7.33	0.66	62.39	1117	11.98	0.449	74.42
pr14	1674	1528	1529	8.66	2.402	1501	1.83	2.487	-3.54	1399	8.5	1.158	51.79	1427	6.67	1.409	41.34
pr15	2065	1818	1824	11.67	6.725	1815	0.49	5.457	18.86	1746	4.28	1.685	74.94	1606	11.95	1.215	81.93
pr16	2065	1889	1861	9.88	8.132	1829	1.72	4.968	38.91	1718	7.68	2.979	63.37	1645	11.61	2.97	63.48
pr17	934	889	894	4.28	0.769	868	2.91	0.186	75.81	799	10.63	0.315	59.04	800	10.51	0.322	58.13
pr18	1539	1352	1425	7.41	2.123	1372	3.72	1.211	42.96	1266	11.16	1.033	51.34	1245	12.63	1.339	36.93
pr19	1760	1560	1613	8.35	5.337	1534	4.9	3.632	31.95	1490	7.63	1.708	68	1449	10.17	0.975	81.73

pr20	2062	1846	1979	4.03	9.077	1873	5.36	3.908	56.95	1792	9.45	1.966	78.34	1720	13.09	2.129	76.55

Πίνακας 5-5: Πειραματικά αποτελέσματα για τα στιγμιότυπα εισόδου των Solomon (m=1)

	BK	ILS (2009)	S=1			S=2				S=3				S=4			
Name	Score	Score	Score		CPU(s)	Score		CPU(s)		Score		CPU(s)		Score		CPU(s)	
			Value	Gap %(BK)	Value	Value	Gap (%)	Value	Gap (%)	Value	Gap (%)	Value	Gap (%)	Value	Gap (%)	Value	Gap (%)
c102	360	360	230	36.11	0.188	220	4.35	0.165	12.23	210	8.7	0.165	12.23	200	13.04	0.183	2.66
c103	400	390	350	12.5	0.221	310	11.43	0.169	23.53	300	14.29	0.176	20.36	260	25.71	0.18	18.55
c104	420	400	370	11.9	0.194	310	16.22	0.172	11.34	340	8.11	0.199	-2.58	310	16.22	0.192	1.03
c105	340	340	260	23.53	0.173	280	-7.69	0.177	-2.31	260	0	0.179	-3.47	250	3.85	0.184	-6.36
c106	340	340	290	14.71	0.181	280	3.45	0.165	8.84	270	6.9	0.175	3.31	240	17.24	0.177	2.21
c107	370	360	310	16.22	0.18	290	6.45	0.162	10	280	9.68	0.172	4.44	280	9.68	0.179	0.56
c108	370	370	330	10.81	0.197	310	6.06	0.176	10.66	290	12.12	0.169	14.21	290	12.12	0.184	6.6
c109	380	380	350	7.89	0.19	340	2.86	0.172	9.47	300	14.29	0.176	7.37	290	17.14	0.185	2.63
c201	870	840	770	11.49	0.253	770	0	0.189	25.3	770	0	0.197	22.13	780	-1.3	0.184	27.27
c202	930	910	830	10.75	0.28	830	0	0.21	25	840	-1.2	0.209	25.36	810	2.41	0.242	13.57
c203	960	940	890	7.29	0.268	880	1.12	0.249	7.09	900	-1.12	0.224	16.42	910	-2.25	0.194	27.61
c204	980	950	920	6.12	0.275	910	1.09	0.214	22.18	880	4.35	0.227	17.45	870	5.43	0.2	27.27
c205	910	900	820	9.89	0.247	830	-1.22	0.189	23.48	840	-2.44	0.2	19.03	850	-3.66	0.179	27.53
c206	930	910	880	5.38	0.299	880	0	0.203	32.11	860	2.27	0.193	35.45	870	1.14	0.285	4.68
c207	930	910	860	7.53	0.285	870	-1.16	0.226	20.7	850	1.16	0.181	36.49	850	1.16	0.186	34.74
c208	950	930	910	4.21	0.277	900	1.1	0.198	28.52	870	4.4	0.191	31.05	880	3.3	0.227	18.05
r101	198	182	143	27.78	0.153	103	27.97	0.152	0.65	126	11.89	0.162	-5.88	112	21.68	0.167	-9.15
r102	286	286	213	25.52	0.175	239	-12.21	0.164	6.29	221	-3.76	0.178	-1.71	230	-7.98	0.197	-12.57
r103	293	286	248	15.36	0.182	262	-5.65	0.182	0	231	6.85	0.177	2.75	241	2.82	0.185	-1.65
r104	303	297	254	16.17	0.182	282	-11.02	0.232	-27.47	252	0.79	0.242	-32.97	214	15.75	0.193	-6.04
r105	247	247	195	21.05	0.171	195	0	0.175	-2.34	175	10.26	0.168	1.75	163	16.41	0.171	0
r106	293	293	265	9.56	0.202	251	5.28	0.182	9.9	254	4.15	0.229	-13.37	257	3.02	0.198	1.98
r107	299	288	275	8.03	0.192	265	3.64	0.191	0.52	235	14.55	0.18	6.25	214	22.18	0.18	6.25
r108	308	297	254	17.53	0.182	281	-10.63	0.19	-4.4	249	1.97	0.311	-70.88	254	0	0.293	-60.99
r109	277	276	239	13.72	0.182	251	-5.02	0.283	-55.49	246	-2.93	0.409	-124.73	216	9.62	0.181	0.55
r110	284	281	255	10.21	0.197	256	-0.39	0.185	6.09	225	11.76	0.17	13.71	223	12.55	0.178	9.64
r111	297	295	259	12.79	0.193	260	-0.39	0.178	7.77	235	9.27	0.167	13.47	200	22.78	0.345	-78.76
r112	298	295	274	8.05	0.212	269	1.82	0.163	23.11	244	10.95	0.192	9.43	243	11.31	0.178	16.04
r201	797	788	765	4.02	0.355	736	3.79	0.199	43.94	757	1.05	0.225	36.62	715	6.54	0.226	36.34
r202	930	880	855	8.06	0.546	877	-2.57	0.465	14.84	792	7.37	0.216	60.44	844	1.29	0.204	62.64
r203	1028	980	986	4.09	0.747	926	6.09	0.286	61.71	919	6.8	0.539	27.84	925	6.19	0.305	59.17
r204	1093	1073	1055	3.48	0.562	980	7.11	0.253	54.98	983	6.82	0.264	53.02	981	7.01	0.234	58.36
r205	953	931	885	7.14	0.319	906	-2.37	0.222	30.41	856	3.28	0.259	18.81	862	2.6	0.238	25.39
r206	1032	996	961	6.88	0.5	964	-0.31	0.374	25.2	958	0.31	0.676	-35.2	917	4.58	0.209	58.2
r207	1077	1038	1032	4.18	0.736	998	3.29	0.226	69.29	995	3.59	0.408	44.57	944	8.53	0.361	50.95
r208	1118	1069	1080	3.4	0.645	1018	5.74	0.311	51.78	1027	4.91	0.245	62.02	1036	4.07	0.393	39.07
r209	961	926	907	5.62	0.522	910	-0.33	0.31	40.61	870	4.08	0.237	54.6	878	3.2	0.208	60.15
r210	1000	958	913	8.7	0.347	931	-1.97	0.332	4.32	897	1.75	0.274	21.04	897	1.75	0.217	37.46
r211	1051	1023	1001	4.76	0.409	962	3.9	0.278	32.03	960	4.1	0.295	27.87	985	1.6	0.201	50.86
rc101	219	219	193	11.87	0.171	176	8.81	0.176	-2.92	173	10.36	0.17	0.58	163	15.54	0.178	-4.09
rc102	266	259	236	11.28	0.177	205	13.14	0.166	6.21	196	16.95	0.181	-2.26	213	9.75	0.181	-2.26
rc103	266	265	226	15.04	0.206	221	2.21	0.184	10.68	217	3.98	0.173	16.02	165	26.99	0.18	12.62
rc104	301	297	241	19.93	0.18	234	2.9	0.169	6.11	211	12.45	0.17	5.56	217	9.96	0.195	-8.33
rc105	244	221	203	16.8	0.172	165	18.72	0.172	0	202	0.49	0.17	1.16	187	7.88	0.19	-10.47
rc106	252	239	225	10.71	0.19	197	12.44	0.165	13.16	210	6.67	0.204	-7.37	184	18.22	0.172	9.47
rc107	277	274	257	7.22	0.19	240	6.61	0.171	10	216	15.95	0.164	13.68	197	23.35	0.181	4.74
rc108	298	288	278	6.71	0.194	256	7.91	0.193	0.52	223	19.78	0.31	-59.79	211	24.1	0.187	3.61
rc201	795	780	771	3.02	0.303	762	1.17	0.355	-17.16	756	1.95	0.197	34.98	665	13.75	0.184	39.27
rc202	938	882	856	8.74	0.445	857	-0.12	0.277	37.75	792	7.48	0.266	40.22	872	-1.87	0.326	26.74
rc203	1003	960	946	5.68	0.458	915	3.28	0.374	18.34	903	4.55	0.241	47.38	892	5.71	0.341	25.55
rc204	1140	1117	1099	3.6	0.357	1009	8.19	0.32	10.36	1023	6.92	0.381	-6.72	982	10.65	0.245	31.37
rc205	859	840	819	4.66	0.377	813	0.73	0.251	33.42	796	2.81	0.24	36.34	716	12.58	0.201	46.68
rc206	899	860	841	6.45	0.369	832	1.07	0.227	38.48	828	1.55	0.224	39.3	813	3.33	0.225	39.02
rc207	983	926	903	8.14	0.331	889	1.55	0.228	31.12	900	0.33	0.402	-21.45	839	7.09	0.3	9.37

rc208	1057	1037	973	7.95	0.429	941	3.29	0.23	46.39	986	-1.34	0.258	39.86	911	6.37	0.21	51.05

Πίνακας 5-6: Πειραματικά αποτελέσματα για τα στιγμιότυπα εισόδου των Solomon (m=2)

	BK	ILS (2009)	S=1			S=2				S=3				S=4			
Name	Score	Score	Score		CPU(s)	Score		CPU(s)		Score		CPU(s)		Score		CPU(s)	
			Value	Gap %(BK)	Value	Value	Gap (%)	Value	Gap (%)	Value	Gap (%)	Value	Gap (%)	Value	Gap (%)	Value	Gap (%)
c102	660	650	360	45.45	0.299	360	0	0.21	29.77	330	8.33	0.203	32.11	320	11.11	0.222	25.75
c103	720	700	580	19.44	0.57	530	8.62	0.223	60.88	520	10.34	0.226	60.35	460	20.69	0.248	56.49
c104	760	750	640	15.79	0.402	610	4.69	0.248	38.31	580	9.38	0.243	39.55	550	14.06	0.267	33.58
c105	640	640	470	26.56	0.257	490	-4.26	0.221	14.01	450	4.26	0.224	12.84	420	10.64	0.229	10.89
c106	620	620	500	19.35	0.27	480	4	0.21	22.22	470	6	0.221	18.15	400	20	0.222	17.78
c107	670	670	560	16.42	0.279	550	1.79	0.211	24.37	510	8.93	0.218	21.86	490	12.5	0.224	19.71
c108	680	670	590	13.24	0.309	580	1.69	0.228	26.21	540	8.47	0.217	29.77	500	15.25	0.235	23.95
c109	720	710	640	11.11	0.326	610	4.69	0.215	34.05	560	12.5	0.245	24.85	500	21.88	0.228	30.06
c201	1460	1400	1290	11.64	0.427	1280	0.78	0.248	41.92	1310	-1.55	0.249	41.69	1330	-3.1	0.219	48.71
c202	1470	1430	1370	6.8	0.525	1350	1.46	0.442	15.81	1360	0.73	0.391	25.52	1300	5.11	0.328	37.52
c203	1480	1430	1410	4.73	0.563	1360	3.55	0.342	39.25	1400	0.71	0.329	41.56	1360	3.55	0.251	55.42
c204	1490	1460	1420	4.7	0.814	1410	0.7	0.438	46.19	1370	3.52	0.439	46.07	1370	3.52	0.22	72.97
c205	1470	1450	1410	4.08	0.482	1430	-1.42	0.378	21.58	1400	0.71	0.234	51.45	1380	2.13	0.236	51.04
c206	1480	1440	1440	2.7	0.543	1430	0.69	0.342	37.02	1410	2.08	0.23	57.64	1390	3.47	0.211	61.14
c207	1490	1450	1430	4.03	0.838	1440	-0.7	0.403	51.91	1420	0.7	0.255	69.57	1390	2.8	0.217	74.11
c208	1490	1460	1460	2.01	0.763	1450	0.68	0.362	52.56	1430	2.05	0.244	68.02	1420	2.74	0.249	67.37
r101	349	330	275	21.2	0.204	217	21.09	0.2	1.96	257	6.55	0.2	1.96	186	32.36	0.199	2.45
r102	508	508	461	9.25	0.358	408	11.5	0.215	39.94	411	10.85	0.221	38.27	355	22.99	0.242	32.4
r103	522	513	468	10.34	0.358	439	6.2	0.239	33.24	400	14.53	0.235	34.36	414	11.54	0.279	22.07
r104	552	539	506	8.33	0.32	470	7.11	0.249	22.19	387	23.52	0.279	12.81	414	18.18	0.243	24.06
r105	453	430	351	22.52	0.249	333	5.13	0.215	13.65	329	6.27	0.215	13.65	320	8.83	0.224	10.04
r106	529	529	438	17.2	0.29	444	-1.37	0.256	11.72	416	5.02	0.242	16.55	382	12.79	0.233	19.66
r107	538	529	474	11.9	0.331	461	2.74	0.228	31.12	427	9.92	0.255	22.96	424	10.55	0.259	21.75
r108	560	549	513	8.39	0.314	485	5.46	0.253	19.43	429	16.37	0.218	30.57	423	17.54	0.455	-44.9
r109	506	498	453	10.47	0.329	407	10.15	0.227	31	414	8.61	0.221	32.83	395	12.8	0.29	11.85
r110	525	515	456	13.14	0.324	432	5.26	0.218	32.72	409	10.31	0.229	29.32	407	10.75	0.222	31.48
r111	544	535	490	9.93	0.333	479	2.24	0.262	21.32	460	6.12	0.219	34.23	383	21.84	0.225	32.43
r112	544	515	491	9.74	0.351	469	4.48	0.223	36.47	399	18.74	0.22	37.32	424	13.65	0.232	33.9
r201	1256	1231	1192	5.1	0.784	1148	3.69	0.298	61.99	1170	1.85	0.467	40.43	1132	5.03	0.254	67.6
r202	1348	1270	1300	3.56	0.668	1308	-0.62	0.775	-16.02	1219	6.23	0.214	67.96	1252	3.69	0.499	25.3
r203	1418	1377	1345	5.15	0.588	1355	-0.74	0.283	51.87	1300	3.35	0.215	63.44	1307	2.83	0.213	63.78
r204	1458	1440	1431	1.85	0.523	1424	0.49	0.247	52.77	1396	2.45	0.195	62.72	1410	1.47	0.337	35.56
r205	1386	1338	1324	4.47	0.576	1339	-1.13	0.373	35.24	1315	0.68	0.378	34.38	1271	4	0.258	55.21
r206	1450	1401	1380	4.83	0.422	1378	0.14	0.288	31.75	1349	2.25	0.275	34.83	1399	-1.38	0.334	20.85
r207	1458	1428	1417	2.81	0.407	1417	0	0.614	-50.86	1370	3.32	0.265	34.89	1407	0.71	0.516	-26.78
r208	1458	1458	1456	0.14	0.298	1451	0.34	0.183	38.59	1429	1.85	0.382	-28.19	1436	1.37	0.4	-34.23
r209	1414	1345	1357	4.03	0.442	1331	1.92	0.416	5.88	1331	1.92	0.456	-3.17	1334	1.69	0.376	14.93
r210	1427	1365	1358	4.84	0.624	1367	-0.66	0.332	46.79	1326	2.36	0.388	37.82	1328	2.21	0.259	58.49
r211	1458	1422	1435	1.58	0.552	1431	0.28	0.402	27.17	1386	3.41	0.23	58.33	1374	4.25	0.343	37.86
rc101	427	427	378	11.48	0.257	356	5.82	0.217	15.56	311	17.72	0.211	17.9	294	22.22	0.22	14.4
rc102	660	650	457	9.5	0.29	403	11.82	0.252	13.1	392	14.22	0.234	19.31	395	13.57	0.233	19.66
rc103	720	700	464	11.45	0.367	431	7.11	0.26	29.16	426	8.19	0.22	40.05	376	18.97	0.239	34.88
rc104	760	750	520	9.57	0.313	437	15.96	0.24	23.32	446	14.23	0.217	30.67	360	30.77	0.274	12.46
rc105	640	640	384	20	0.253	328	14.58	0.224	11.46	325	15.36	0.218	13.83	352	8.33	0.236	6.72
rc106	620	620	422	12.63	0.287	407	3.55	0.214	25.44	383	9.24	0.213	25.78	324	23.22	0.217	24.39
rc107	670	670	484	9.36	0.328	476	1.65	0.231	29.57	430	11.16	0.203	38.11	353	27.07	0.299	8.84
rc108	680	670	517	7.01	0.312	455	11.99	0.295	5.45	377	27.08	0.239	23.4	403	22.05	0.225	27.88
rc201	720	710	1294	6.57	0.643	1249	3.48	0.273	57.54	1265	2.24	0.251	60.96	1245	3.79	0.209	67.5
rc202	1460	1400	1436	5.03	1.321	1383	3.69	0.287	78.27	1372	4.46	0.219	83.42	1349	6.06	0.4	69.72
rc203	1470	1430	1502	7.97	0.512	1471	2.06	0.445	13.09	1454	3.2	0.665	-29.88	1439	4.19	0.428	16.41
rc204	1480	1430	1650	3.85	0.615	1621	1.76	0.535	13.01	1556	5.7	0.234	61.95	1564	5.21	0.517	15.93
rc205	1490	1460	1362	6.58	0.692	1373	-0.81	0.406	41.33	1348	1.03	0.305	55.92	1283	5.8	0.333	51.88
rc206	1470	1450	1451	6.51	0.972	1427	1.65	0.297	69.44	1421	2.07	0.388	60.08	1343	7.44	0.229	76.44
rc207	1480	1440	1547	3.25	0.76	1487	3.88	0.344	54.74	1495	3.36	0.475	37.5	1391	10.08	0.488	35.79

rc208	1490	1450	1633	3.49	1.012	1615	1.1	0.292	71.15	1574	3.61	0.719	28.95	1507	7.72	0.328	67.59

Πίνακας 5-7: Πειραματικά αποτελέσματα για τα στιγμιότυπα εισόδου των Solomon (m=3)

	BK	ILS (2009)	S=1			S=2				S=3				S=4			
Name	Score	Score	Score		CPU(s)	Score		CPU(s)		Score		CPU(s)		Score		CPU(s)	
			Value	Gap %(BK)	Value	Value	Gap (%)	Value	Gap (%)	Value	Gap (%)	Value	Gap (%)	Value	Gap (%)	Value	Gap (%)
c102	920	890	400	56.52	0.351	400	0	0.241	31.34	400	0	0.245	30.2	380	5	0.271	22.79
c103	990	960	720	27.27	0.462	700	2.78	0.295	36.15	670	6.94	0.324	29.87	610	15.28	0.308	33.33
c104	1030	1010	870	15.53	0.462	810	6.9	0.278	39.83	760	12.64	0.249	46.1	730	16.09	0.382	17.32
c105	870	840	680	21.84	0.378	670	1.47	0.313	17.2	630	7.35	0.281	25.66	560	17.65	0.312	17.46
c106	870	840	710	18.39	0.413	650	8.45	0.276	33.17	640	9.86	0.265	35.84	540	23.94	0.266	35.59
c107	910	900	780	14.29	0.401	790	-1.28	0.308	23.19	710	8.97	0.269	32.92	690	11.54	0.269	32.92
c108	920	900	800	13.04	0.506	800	0	0.291	42.49	760	5	0.279	44.86	720	10	0.281	44.47
c109	970	950	880	9.28	0.573	820	6.82	0.277	51.66	760	13.64	0.281	50.96	670	23.86	0.264	53.93
c201	1810	1750	1750	3.31	0.401	1670	4.57	0.257	35.91	1610	8	0.247	38.4	1720	1.71	0.233	41.9
c202	1810	1750	1690	6.63	0.542	1670	1.18	0.513	5.35	1680	0.59	0.383	29.34	1640	2.96	0.244	54.98
c203	1810	1760	1710	5.52	0.42	1710	0	0.552	-31.43	1690	1.17	0.411	2.14	1680	1.75	0.301	28.33
c204	1810	1780	1740	3.87	0.326	1720	1.15	0.441	-35.28	1690	2.87	0.243	25.46	1680	3.45	0.211	35.28
c205	1810	1770	1750	3.31	0.666	1760	-0.57	0.276	58.56	1720	1.71	0.222	66.67	1740	0.57	0.186	72.07
c206	1810	1770	1760	2.76	0.349	1750	0.57	0.256	26.65	1750	0.57	0.296	15.19	1740	1.14	0.239	31.52
c207	1810	1810	1790	1.1	0.915	1780	0.56	0.319	65.14	1730	3.35	0.203	77.81	1730	3.35	0.241	73.66
c208	1810	1810	1810	0	0.415	1760	2.76	0.242	41.69	1750	3.31	0.204	50.84	1730	4.42	0.195	53.01
r101	484	481	412	14.88	0.275	318	22.82	0.255	7.27	370	10.19	0.237	13.82	270	34.47	0.23	16.36
r102	694	685	565	18.59	0.383	523	7.43	0.275	28.2	492	12.92	0.282	26.37	440	22.12	0.412	-7.57
r103	747	720	650	12.99	0.443	623	4.15	0.332	25.06	540	16.92	0.345	22.12	530	18.46	0.306	30.93
r104	778	765	709	8.87	0.533	693	2.26	0.425	20.26	554	21.86	0.265	50.28	577	18.62	0.379	28.89
r105	620	609	527	15	0.363	462	12.33	0.278	23.42	468	11.2	0.26	28.37	441	16.32	0.261	28.1
r106	729	719	651	10.7	0.441	615	5.53	0.349	20.86	533	18.13	0.273	38.1	521	19.97	0.265	39.91
r107	760	747	674	11.32	0.407	651	3.41	0.347	14.74	565	16.17	0.319	21.62	552	18.1	0.43	-5.65
r108	797	790	712	10.66	0.455	681	4.35	0.313	31.21	598	16.01	0.356	21.76	618	13.2	0.29	36.26
r109	710	699	639	10	0.395	605	5.32	0.3	24.05	533	16.59	0.271	31.39	535	16.28	0.293	25.82
r110	737	711	668	9.36	0.423	640	4.19	0.395	6.62	618	7.49	0.268	36.64	597	10.63	0.266	37.12
r111	774	764	707	8.66	0.402	630	10.89	0.274	31.84	587	16.97	0.26	35.32	571	19.24	0.299	25.62
r112	776	758	707	8.89	0.476	656	7.21	0.312	34.45	583	17.54	0.327	31.3	597	15.56	0.277	41.81
r201	1442	1408	1384	4.02	0.409	1377	0.51	0.415	-1.47	1375	0.65	0.262	35.94	1343	2.96	0.272	33.5
r202	1458	1443	1443	1.03	0.535	1434	0.62	0.336	37.2	1423	1.39	0.451	15.7	1415	1.94	0.176	67.1
r203	1458	1458	1458	0	0.308	1458	0	0.19	38.31	1458	0	0.473	-53.57	1447	0.75	0.207	32.79
r204	1458	1458	1458	0	0.149	1458	0	0.155	-4.03	1445	0.89	0.166	-11.41	1449	0.62	0.15	-0.67
r205	1458	1458	1458	0	0.214	1458	0	0.172	19.63	1457	0.07	0.187	12.62	1446	0.82	0.183	14.49
r206	1458	1458	1458	0	0.19	1458	0	0.14	26.32	1458	0	0.308	-62.11	1458	0	0.158	16.84
r207	1458	1458	1458	0	0.157	1458	0	0.142	9.55	1458	0	0.144	8.28	1458	0	0.139	11.46
r208	1458	1458	1458	0	0.112	1458	0	0.149	-33.04	1458	0	0.129	-15.18	1458	0	0.141	-25.89
r209	1458	1458	1458	0	0.214	1458	0	0.261	-21.96	1458	0	0.161	24.77	1458	0	0.179	16.36
r210	1458	1458	1458	0	0.203	1458	0	0.171	15.76	1458	0	0.141	30.54	1458	0	0.166	18.23
r211	1458	1458	1458	0	0.164	1458	0	0.103	37.2	1458	0	0.112	31.71	1458	0	0.126	23.17
rc101	621	604	514	17.23	0.361	525	-2.14	0.281	22.16	459	10.7	0.251	30.47	452	12.06	0.269	25.48
rc102	714	698	625	12.46	0.387	586	6.24	0.275	28.94	560	10.4	0.29	25.06	513	17.92	0.274	29.2
rc103	764	747	701	8.25	0.607	654	6.7	0.279	54.04	623	11.13	0.291	52.06	503	28.25	0.618	-1.81
rc104	835	822	788	5.63	0.594	639	18.91	0.29	51.18	617	21.7	0.324	45.45	549	30.33	0.413	30.47
rc105	682	654	598	12.32	0.361	497	16.89	0.316	12.47	474	20.74	0.275	23.82	469	21.57	0.267	26.04
rc106	706	678	632	10.48	0.423	613	3.01	0.299	29.31	552	12.66	0.265	37.35	481	23.89	0.34	19.62
rc107	773	745	704	8.93	0.411	688	2.27	0.3	27.01	630	10.51	0.274	33.33	530	24.72	0.338	17.76
rc108	795	757	744	6.42	0.54	651	12.5	0.312	42.22	587	21.1	0.263	51.3	587	21.1	0.257	52.41
rc201	1698	1625	1625	4.3	0.692	1604	1.29	0.3	56.65	1578	2.89	0.276	60.12	1516	6.71	0.21	69.65
rc202	1724	1686	1665	3.42	0.394	1659	0.36	0.301	23.6	1654	0.66	0.537	-36.29	1632	1.98	0.296	24.87
rc203	1724	1724	1714	0.58	0.312	1709	0.29	0.625	-100.32	1701	0.76	0.224	28.21	1704	0.58	0.32	-2.56
rc204	1724	1724	1724	0	0.18	1724	0	0.165	8.33	1724	0	0.163	9.44	1716	0.46	0.172	4.44
rc205	1719	1659	1655	3.72	1.111	1659	-0.24	0.493	55.63	1605	3.02	0.376	66.16	1566	5.38	0.207	81.37
rc206	1724	1708	1714	0.58	0.424	1706	0.47	0.449	-5.9	1692	1.28	0.249	41.27	1685	1.69	0.235	44.58
rc207	1724	1713	1712	0.7	0.251	1712	0	0.343	-36.65	1709	0.18	0.326	-29.88	1689	1.34	0.18	28.29

rc208	1724	1724	1724	0	0.193	1724	0	0.152	21.24	1724	0	0.162	16.06	1724	0	0.324	-67.88

Πίνακας 5-8: Πειραματικά αποτελέσματα για τα στιγμιότυπα εισόδου των Solomon (m=4)

	BK	ILS (2009)	S=1			S=2				S=3				S=4			
Name	Score	Score	Score		CPU(s)	Score		CPU(s)		Score		CPU(s)		Score		CPU(s)	
			Value	Gap %(BK)	Value	Value	Gap (%)	Value	Gap (%)	Value	Gap (%)	Value	Gap (%)	Value	Gap (%)	Value	Gap (%)
c102	1150	1090	400	65.22	0.257	400	0	0.248	3.5	400	0	0.256	0.39	400	0	0.316	-22.96
c103	1210	1150	840	30.58	0.59	810	3.57	0.489	17.12	770	8.33	0.299	49.32	730	13.1	0.333	43.56
c104	1260	1220	1010	19.84	0.593	980	2.97	0.471	20.57	930	7.92	0.302	49.07	900	10.89	0.353	40.47
c105	1070	1030	850	20.56	0.488	810	4.71	0.384	21.31	760	10.59	0.32	34.43	680	20	0.34	30.33
c106	1080	1040	870	19.44	0.494	800	8.05	0.307	37.85	770	11.49	0.37	25.1	690	20.69	0.329	33.4
c107	1120	1100	970	13.39	0.546	950	2.06	0.312	42.86	870	10.31	0.322	41.03	850	12.37	0.322	41.03
c108	1140	1100	1000	12.28	0.686	980	2	0.315	54.08	930	7	0.33	51.9	860	14	0.593	13.56
c109	1190	1180	1080	9.24	0.641	1000	7.41	0.323	49.61	960	11.11	0.366	42.9	900	16.67	0.399	37.75
c201	1810	1810	1790	1.1	0.303	1780	0.56	0.216	28.71	1670	6.7	0.286	5.61	1770	1.12	0.223	26.4
c202	1810	1810	1800	0.55	0.423	1800	0	0.453	-7.09	1760	2.22	0.246	41.84	1740	3.33	0.221	47.75
c203	1810	1810	1810	0	0.259	1790	1.1	0.23	11.2	1780	1.66	0.232	10.42	1780	1.66	0.351	-35.52
c204	1810	1810	1810	0	0.184	1800	0.55	0.174	5.43	1800	0.55	0.367	-99.46	1800	0.55	0.343	-86.41
c205	1810	1810	1810	0	0.219	1810	0	0.155	29.22	1810	0	0.145	33.79	1810	0	0.145	33.79
c206	1810	1810	1810	0	0.201	1810	0	0.149	25.87	1810	0	0.126	37.31	1810	0	0.133	33.83
c207	1810	1810	1810	0	0.212	1810	0	0.139	34.43	1810	0	0.135	36.32	1810	0	0.148	30.19
c208	1810	1810	1810	0	0.205	1810	0	0.152	25.85	1810	0	0.132	35.61	1810	0	0.139	32.2
r101	611	601	510	16.53	0.398	385	24.51	0.319	19.85	446	12.55	0.282	29.15	332	34.9	0.269	32.41
r102	843	807	721	14.47	0.549	627	13.04	0.337	38.62	591	18.03	0.323	41.17	546	24.27	0.45	18.03
r103	928	878	817	11.96	0.695	723	11.51	0.328	52.81	667	18.36	0.354	49.06	648	20.69	0.665	4.32
r104	975	941	867	11.08	0.643	844	2.65	0.406	36.86	695	19.84	0.367	42.92	729	15.92	0.623	3.11
r105	778	735	676	13.11	0.5	580	14.2	0.336	32.8	557	17.6	0.306	38.8	544	19.53	0.313	37.4
r106	906	870	792	12.58	0.709	769	2.9	0.399	43.72	696	12.12	0.324	54.3	673	15.03	0.383	45.98
r107	950	927	874	8	0.968	768	12.13	0.348	64.05	746	14.65	0.348	64.05	719	17.73	0.456	52.89
r108	995	982	888	10.75	0.921	846	4.73	0.335	63.63	755	14.98	0.357	61.24	760	14.41	0.349	62.11
r109	885	866	785	11.3	0.688	747	4.84	0.377	45.2	691	11.97	0.296	56.98	700	10.83	0.662	3.78
r110	915	870	826	9.73	0.884	780	5.57	0.362	59.05	717	13.2	0.621	29.75	718	13.08	0.554	37.33
r111	953	935	877	7.97	0.67	772	11.97	0.357	46.72	707	19.38	0.312	53.43	704	19.73	0.638	4.78
r112	974	939	894	8.21	0.728	855	4.36	0.387	46.84	776	13.2	0.323	55.63	751	16	0.328	54.95
r201	1458	1458	1458	0	0.378	1432	1.78	0.334	11.64	1455	0.21	0.214	43.39	1420	2.61	0.173	54.23
r202	1458	1458	1458	0	0.222	1455	0.21	0.132	40.54	1458	0	0.203	8.56	1458	0	0.133	40.09
r203	1458	1458	1458	0	0.177	1458	0	0.12	32.2	1458	0	0.174	1.69	1458	0	0.156	11.86
r204	1458	1458	1458	0	0.066	1458	0	0.139	-110.61	1458	0	0.155	-134.85	1458	0	0.151	-128.79
r205	1458	1458	1458	0	0.157	1458	0	0.112	28.66	1458	0	0.094	40.13	1458	0	0.116	26.11
r206	1458	1458	1458	0	0.074	1458	0	0.076	-2.7	1458	0	0.118	-59.46	1458	0	0.109	-47.3
r207	1458	1458	1458	0	0.088	1458	0	0.075	14.77	1458	0	0.146	-65.91	1458	0	0.14	-59.09
r208	1458	1458	1458	0	0.058	1458	0	0.122	-110.34	1458	0	0.124	-113.79	1458	0	0.148	-155.17
r209	1458	1458	1458	0	0.104	1458	0	0.12	-15.38	1458	0	0.109	-4.81	1458	0	0.111	-6.73
r210	1458	1458	1458	0	0.162	1458	0	0.098	39.51	1458	0	0.093	42.59	1458	0	0.111	31.48
r211	1458	1458	1458	0	0.059	1458	0	0.097	-64.41	1458	0	0.081	-37.29	1458	0	0.109	-84.75
rc101	811	794	661	18.5	0.47	675	-2.12	0.323	31.28	618	6.51	0.305	35.11	605	8.47	0.318	32.34
rc102	909	881	784	13.75	0.53	731	6.76	0.325	38.68	692	11.73	0.349	34.15	599	23.6	0.456	13.96
rc103	975	947	878	9.95	0.722	773	11.96	0.354	50.97	763	13.1	0.351	51.39	674	23.23	0.52	27.98
rc104	1065	1019	967	9.2	0.602	781	19.23	0.333	44.68	869	10.13	0.321	46.68	833	13.86	0.384	36.21
rc105	875	841	759	13.26	0.526	630	17	0.37	29.66	656	13.57	0.314	40.3	592	22	0.343	34.79
rc106	909	874	813	10.56	0.582	788	3.08	0.347	40.38	729	10.33	0.331	43.13	646	20.54	0.394	32.3
rc107	987	951	912	7.6	0.583	840	7.89	0.389	33.28	803	11.95	0.291	50.09	679	25.55	0.989	-69.64
rc108	1025	998	955	6.83	0.686	854	10.58	0.408	40.52	806	15.6	0.37	46.06	731	23.46	0.334	51.31
rc201	1724	1724	1724	0	0.576	1709	0.87	0.218	62.15	1706	1.04	0.275	52.26	1675	2.84	0.21	63.54
rc202	1724	1724	1724	0	0.248	1724	0	0.215	13.31	1724	0	0.208	16.13	1719	0.29	0.181	27.02
rc203	1724	1724	1724	0	0.145	1724	0	0.167	-15.17	1724	0	0.173	-19.31	1724	0	0.164	-13.1
rc204	1724	1724	1724	0	0.06	1724	0	0.15	-150	1724	0	0.16	-166.67	1721	0.17	0.17	-183.33
rc205	1724	1724	1724	0	0.387	1724	0	0.262	32.3	1715	0.52	0.233	39.79	1684	2.32	0.174	55.04

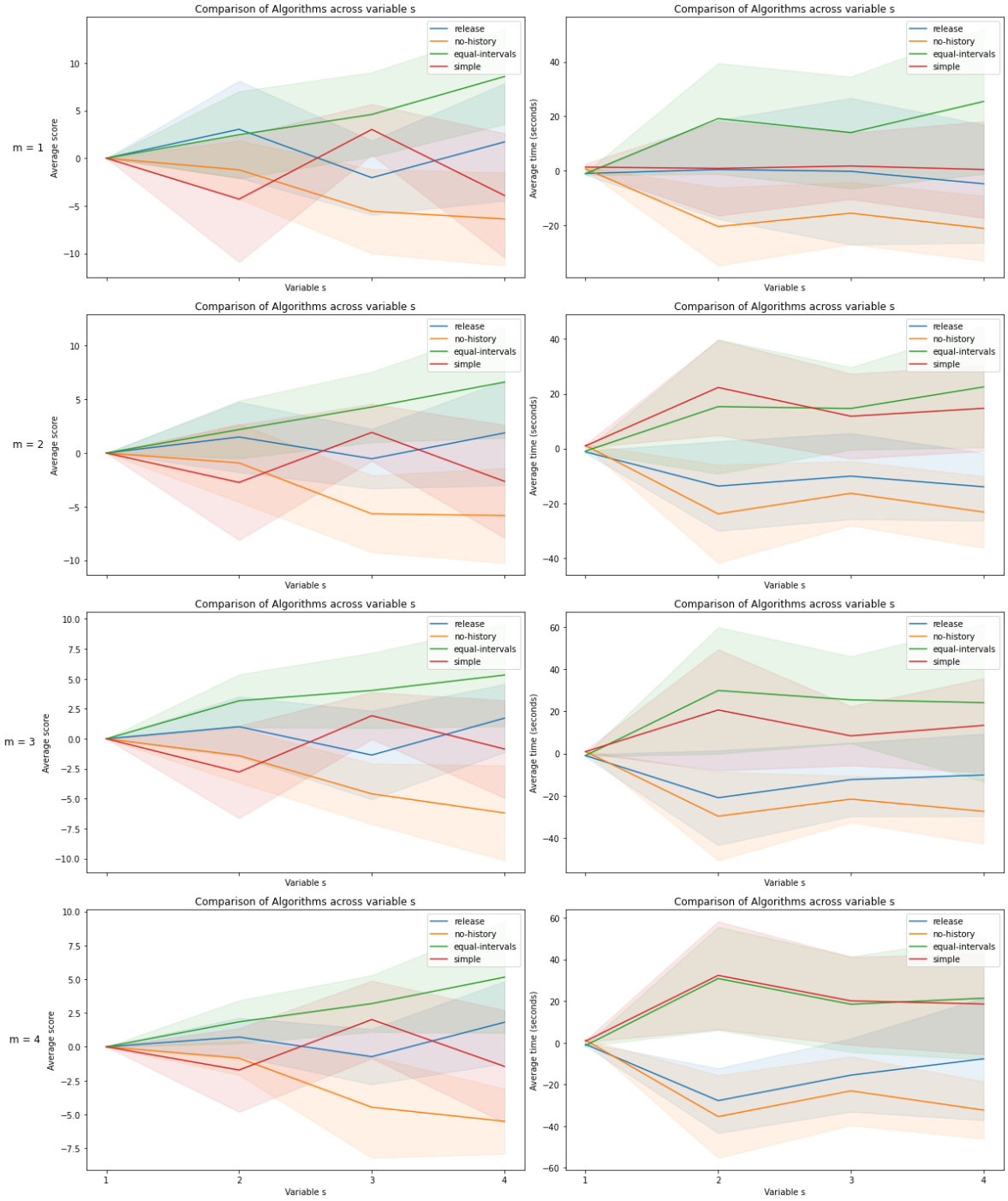
rc206	1724	1724	1724	0	0.181	1724	0	0.157	13.26	1724	0	0.125	30.94	1719	0.29	0.143	20.99
rc207	1724	1724	1724	0	0.187	1724	0	0.143	23.53	1724	0	0.131	29.95	1722	0.12	0.144	22.99
rc208	1724	1724	1724	0	0.073	1724	0	0.082	-12.33	1724	0	0.114	-56.16	1724	0	0.118	-61.64

5.2 Σύγκριση διαφορετικών εκδόσεων του αλγορίθμου

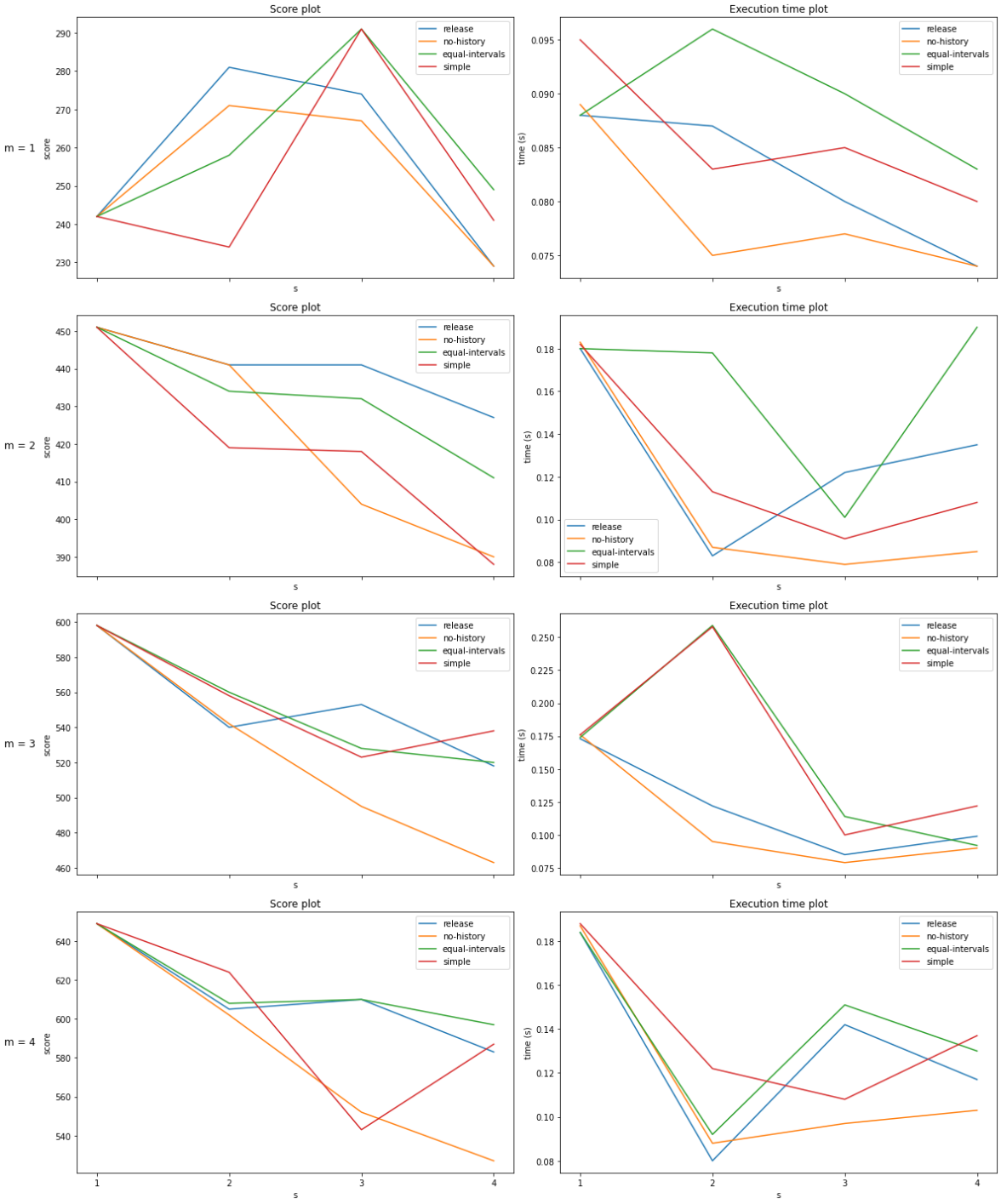
Στο Κεφάλαιο 4, αναλύθηκαν λεπτομερώς οι τροποποιήσεις στον βασικό αλγόριθμο ILS, καθώς και οι κινήσεις που έγιναν για να βελτιώσουν την ταχύτητα του τροποποιημένου αλγορίθμου ή τη βαθμολογία των λύσεων του. Παρακάτω συγκρίνονται σε απόδοση και χρόνο εκτέλεσης 4 διαφορετικές εκδόσεις του αλγορίθμου:

- release: Η τελική έκδοση του αλγορίθμου, η οποία έχει προσαρμοσμένα χρονικά διαστήματα για να είναι διαμοιρασμένοι ισόποσα οι κόμβοι, και χρησιμοποιεί το ιστορικό των κόμβων για να επιλέγει σε ποιο χρονικό διάστημα θα ανατεθούν σε κάθε επανάληψη
- no-history: Η διαφορά με την έκδοση release, είναι ότι δεν κρατάει ιστορικό για της κόμβους
- equal-intervals: Η διαφορά με την έκδοση release είναι πως έχει ίσα χρονικά διαστήματα, όχι προσαρμοσμένα με βάση το πλήθος των κόμβων που της αντιστοιχούν. Για παράδειγμα για s=4, και timeBudget = [0-1000] , θα προκύψουν τα διαστήματα [0-250], [250-500], [500-750] και [750-1000].
- simple: Η έκδοση αυτή είναι συνδυασμός των no-history και equal-intervals καθώς χρησιμοποιεί ίσα χρονικά διαστήματα και δεν χρησιμοποιεί το ιστορικό των κόμβων

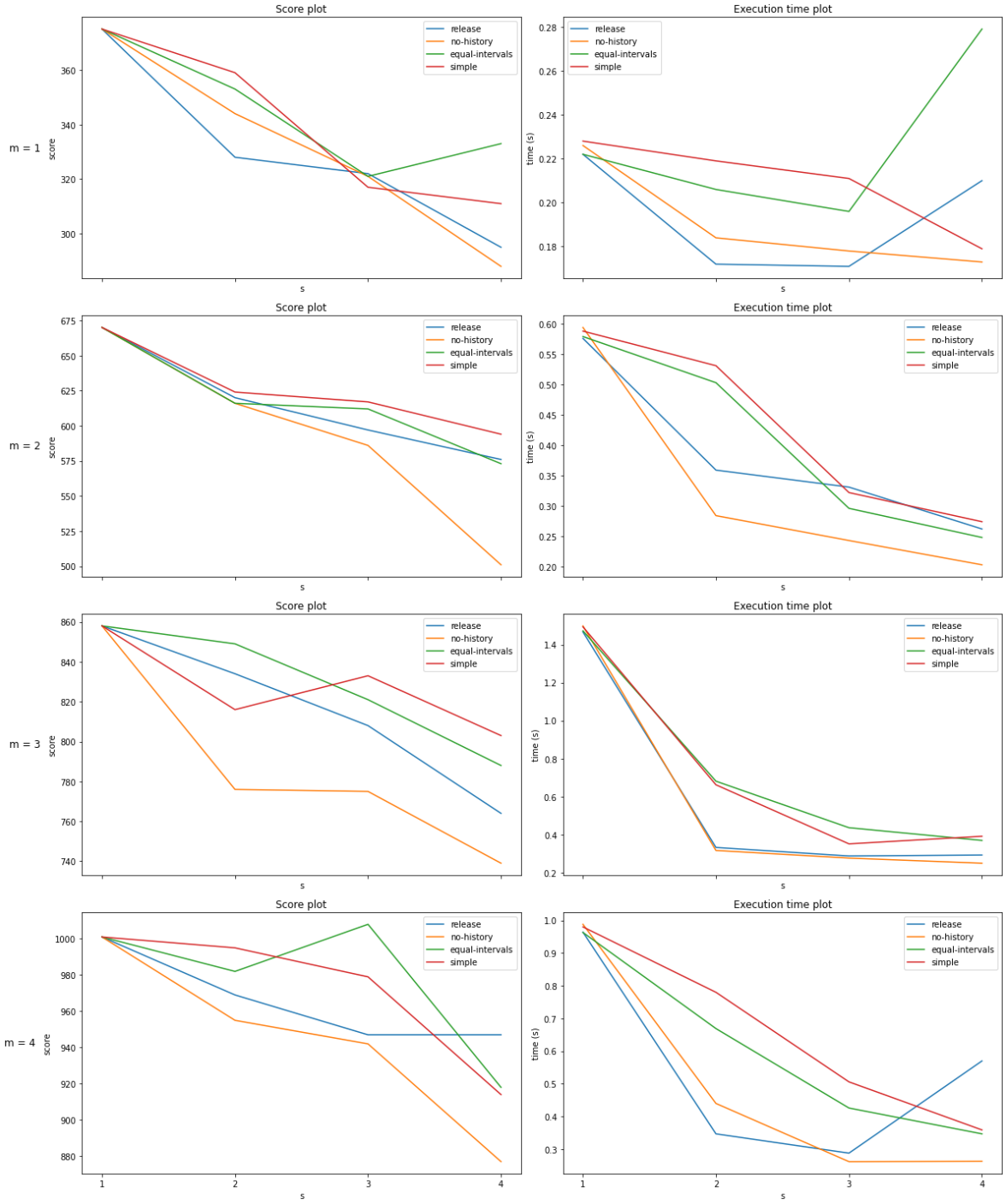
Στην Εικόνα 5-1 συγκρίνονται οι 4 αλγόριθμοι με βάση τους μέσους όρους βαθμολογιών και χρόνων εκτέλεσης για τα στιγμιότυπα εισόδου που είναι βασισμένα στα σύνολα δεδομένων των Cordeau et al. (1997) [40] (pr01-pr20). Σε αυτά τα γραφήματα, η έκδοση release φαίνεται να είναι η δεύτερη καλύτερη σε ποιότητα λύσεων καθώς πρώτη είναι η έκδοση equal-intervals. Σε ταχύτητα εκτέλεσης, η έκδοση equal-intervals είναι στις χειρότερες θέσεις μαζί με την έκδοση simple ενώ η release είναι δεύτερη μετά την έκδοση no-history που είναι όμως η χειρότερη σε ποιότητα λύσεων. Παρόλα αυτά επειδή υπάρχει μεγάλη τυπική απόκλιση (οι χρωματιστές περιοχές γύρω από τους μέσους όρους) στις τιμές των σκορ δεν είναι εύκολο να εξαχθεί κάποιο ασφαλές συμπέρασμα για την υπεροχή κάποιας έκδοσης έναντι των υπολοίπων. Στα γραφήματα χρονικής εκτέλεσης, οι τιμές των release και history φαίνεται να έχουν αρκετή διαφορά από τις τιμές των equal-intervals και simple ειδικά όσο το m αυξάνεται. Στα αρχικά στιγμιότυπα εισόδου, η έκδοση simple ανταγωνίζεται την equal-intervals στην ποιότητα των λύσεων. Παρόλα αυτά ή έκδοση release με την no-history φαίνεται να υπερτερούν από τις άλλες στους χρόνους εκτέλεσης, ειδικά σε παραδείγματα με πολλά δεδομένα εισόδου (π.χ. pr10, pr19, pr20).



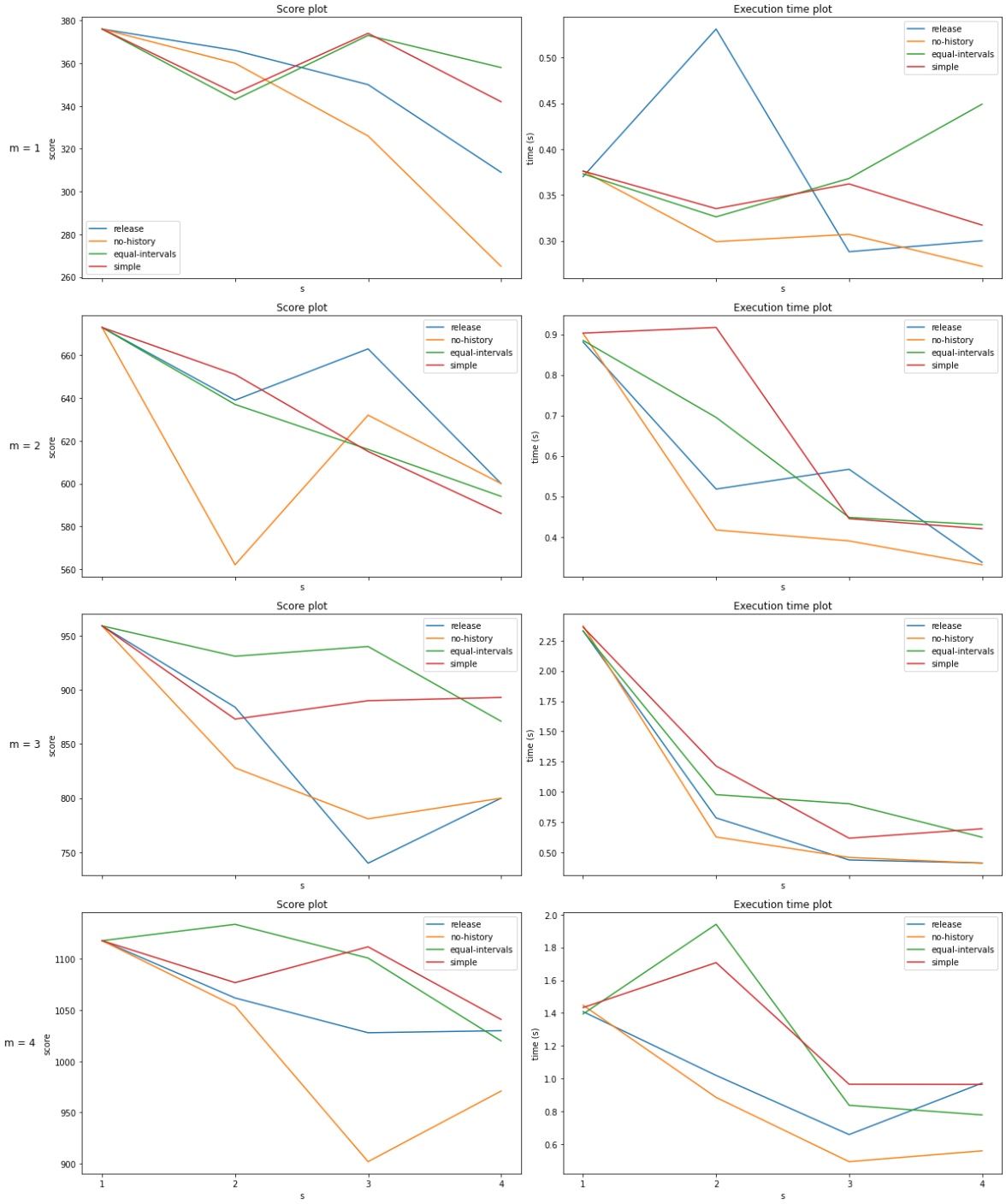
Εικόνα 5-1: Σύγκριση εκδόσεων αλγορίθμου για διαφορετικά s, με τους μέσους όρους βαθμολογιών και χρόνων εκτέλεσης των στιγμιότυπων εισόδου pr01-pr20. Οι χρωματιστές περιοχές απεικονίζουν την τυπική απόκλιση των τιμών.



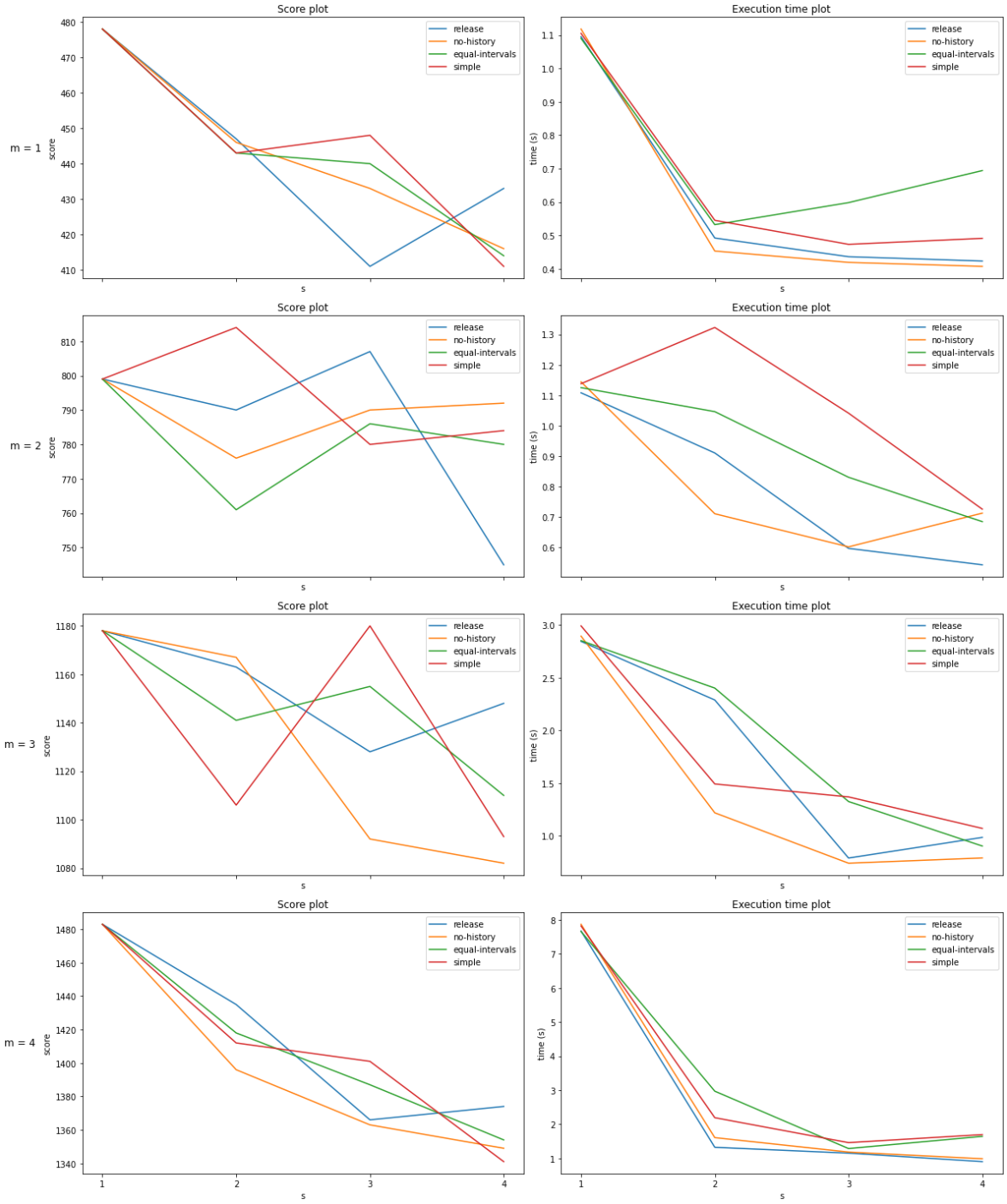
Εικόνα 5-2: Σύγκριση εκδόσεων του αλγορίθμου για το στιγμιότυπο εισόδου pr01 (48 pois)



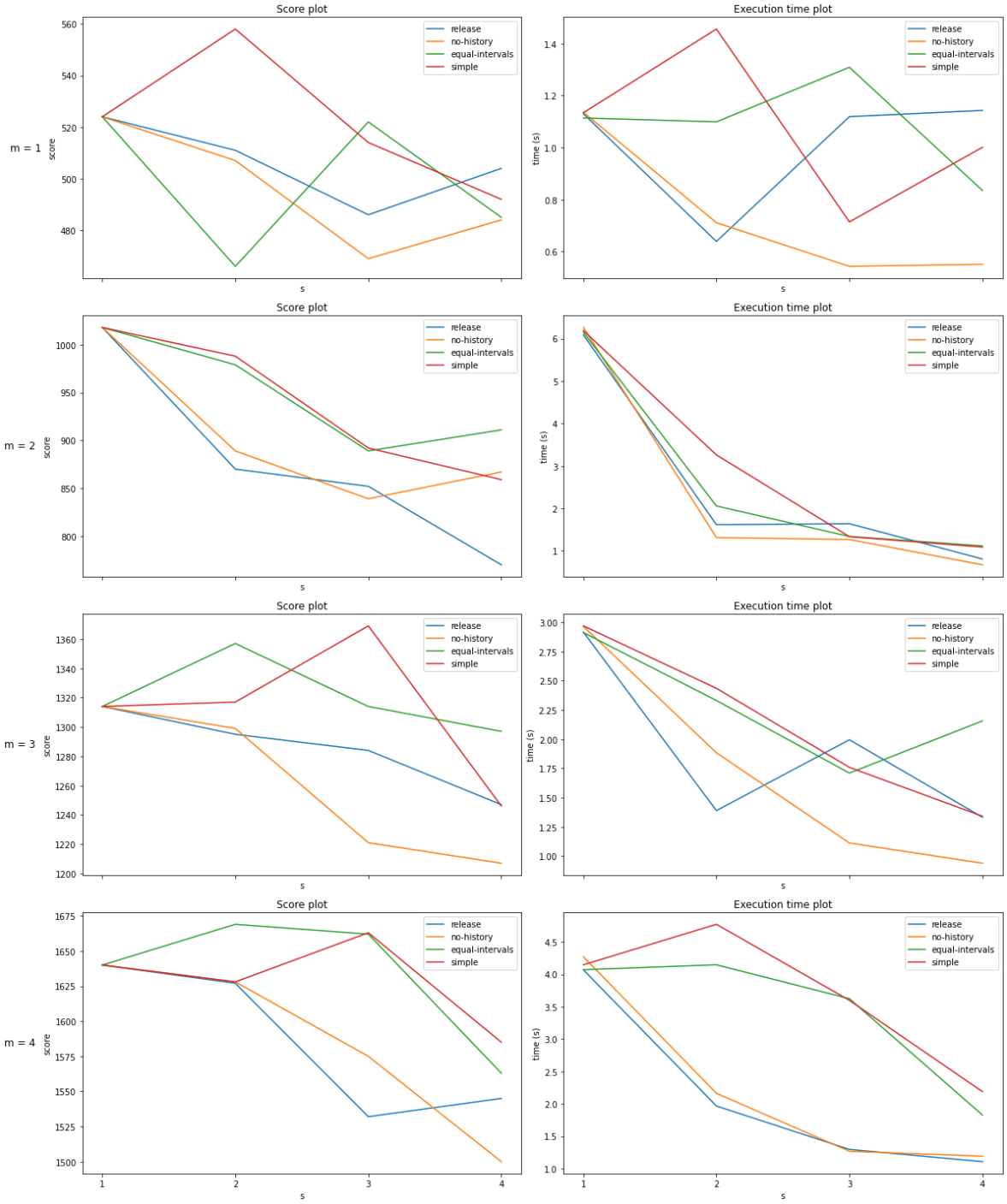
Εικόνα 5-3: Σύγκριση εκδόσεων του αλγορίθμου για το στιγμιότυπο εισόδου pr02 (96 pois)



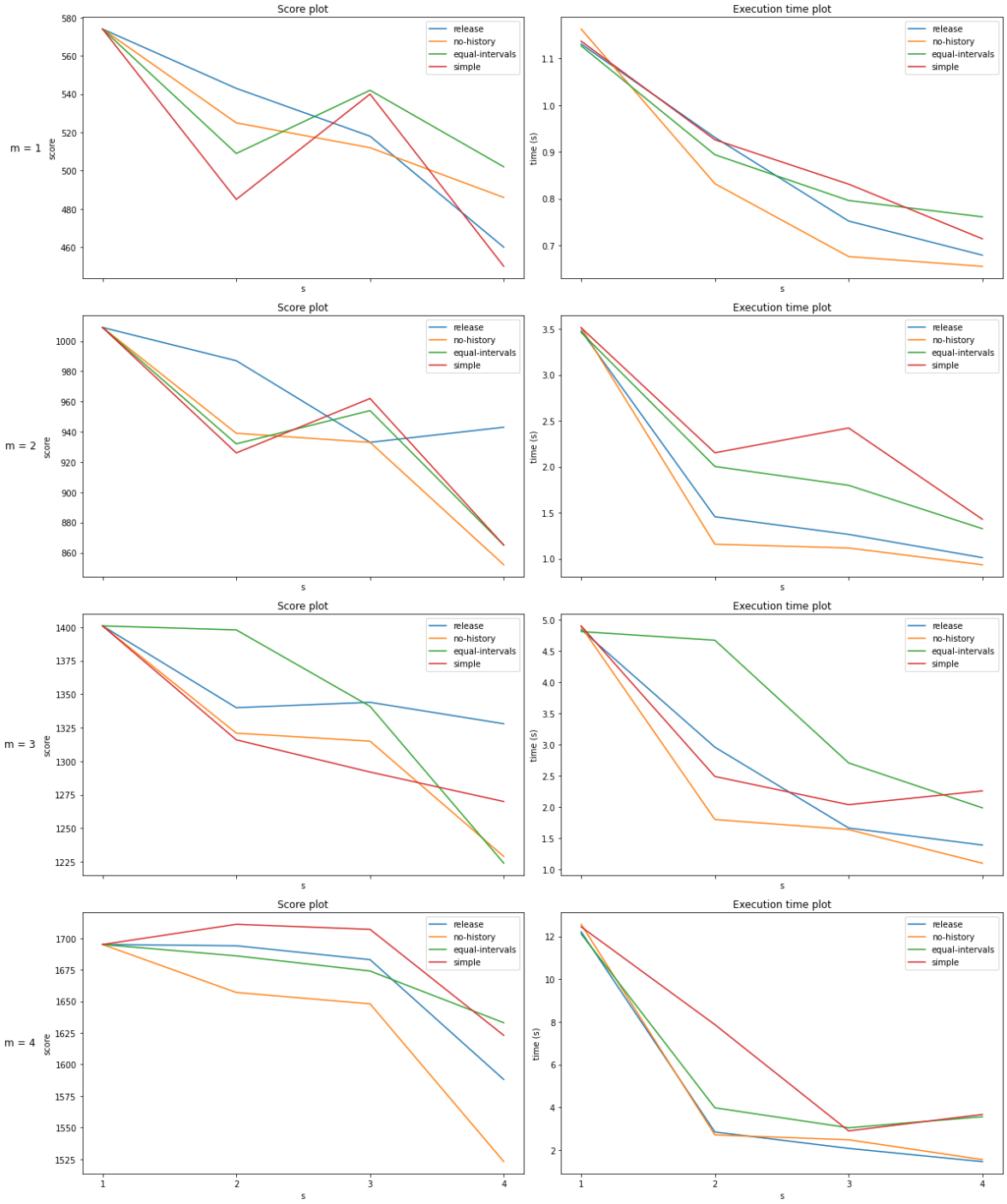
Εικόνα 5-4: Σύγκριση εκδόσεων του αλγορίθμου για το στιγμιότυπο εισόδου pr03 (144 pois)



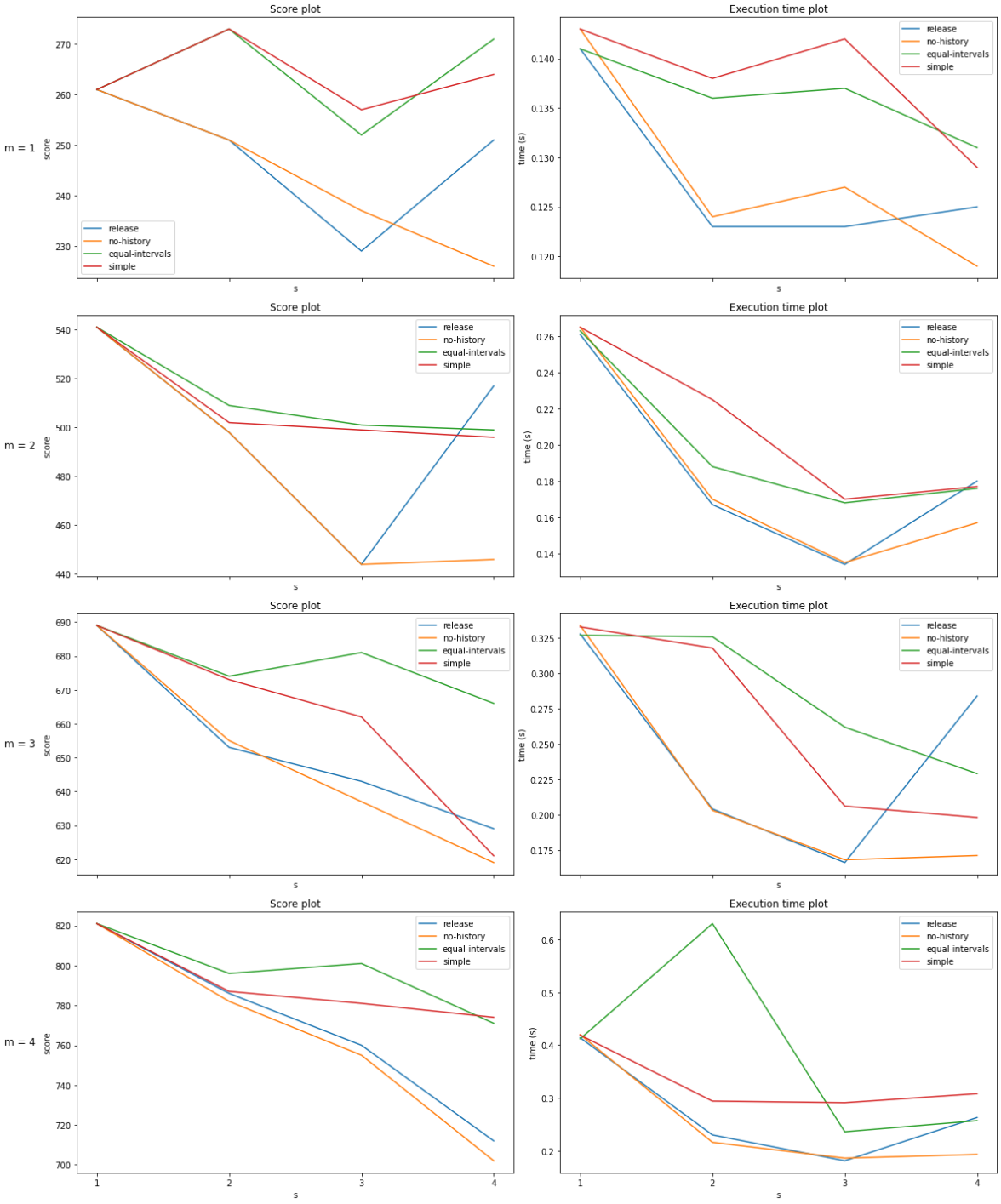
Εικόνα 5-5: Σύγκριση εκδόσεων του αλγορίθμου για το στιγμιότυπο εισόδου pr04 (192 pois)



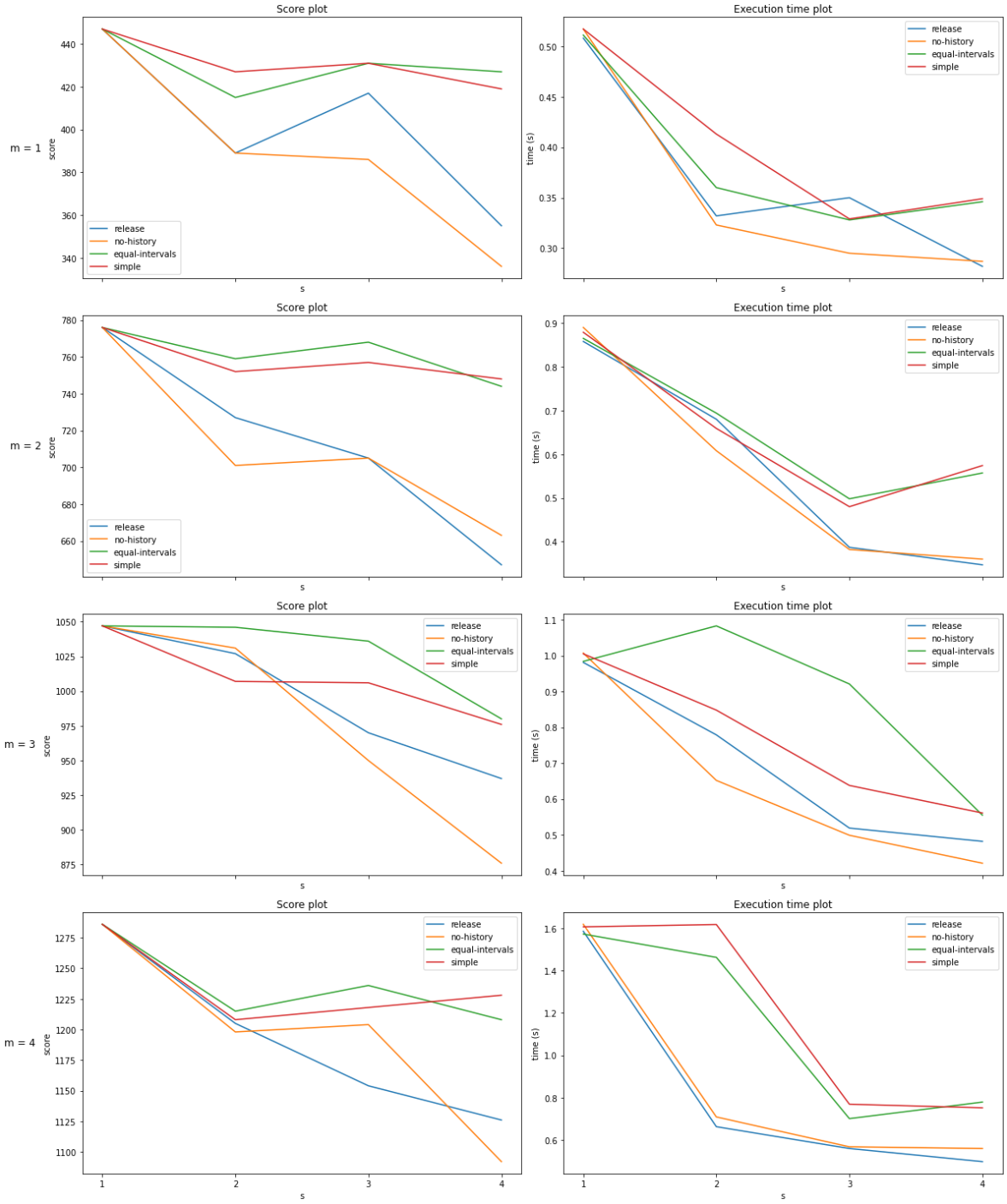
Εικόνα 5-6: Σύγκριση εκδόσεων του αλγορίθμου για το στιγμιότυπο εισόδου pr05 (240 pois)



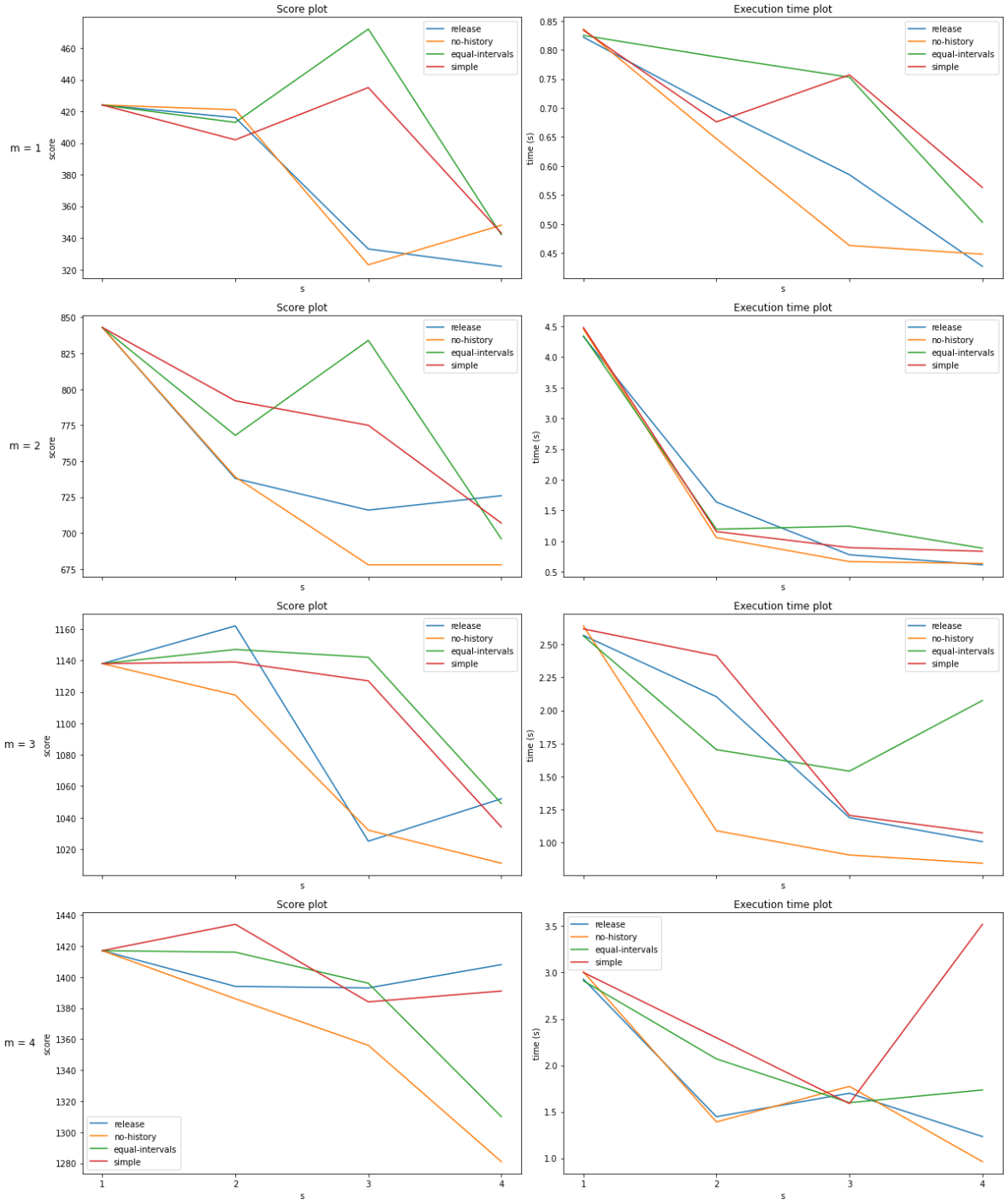
Εικόνα 5-7: Σύγκριση εκδόσεων του αλγορίθμου για το στιγμιότυπο εισόδου pr06 (288 pois)



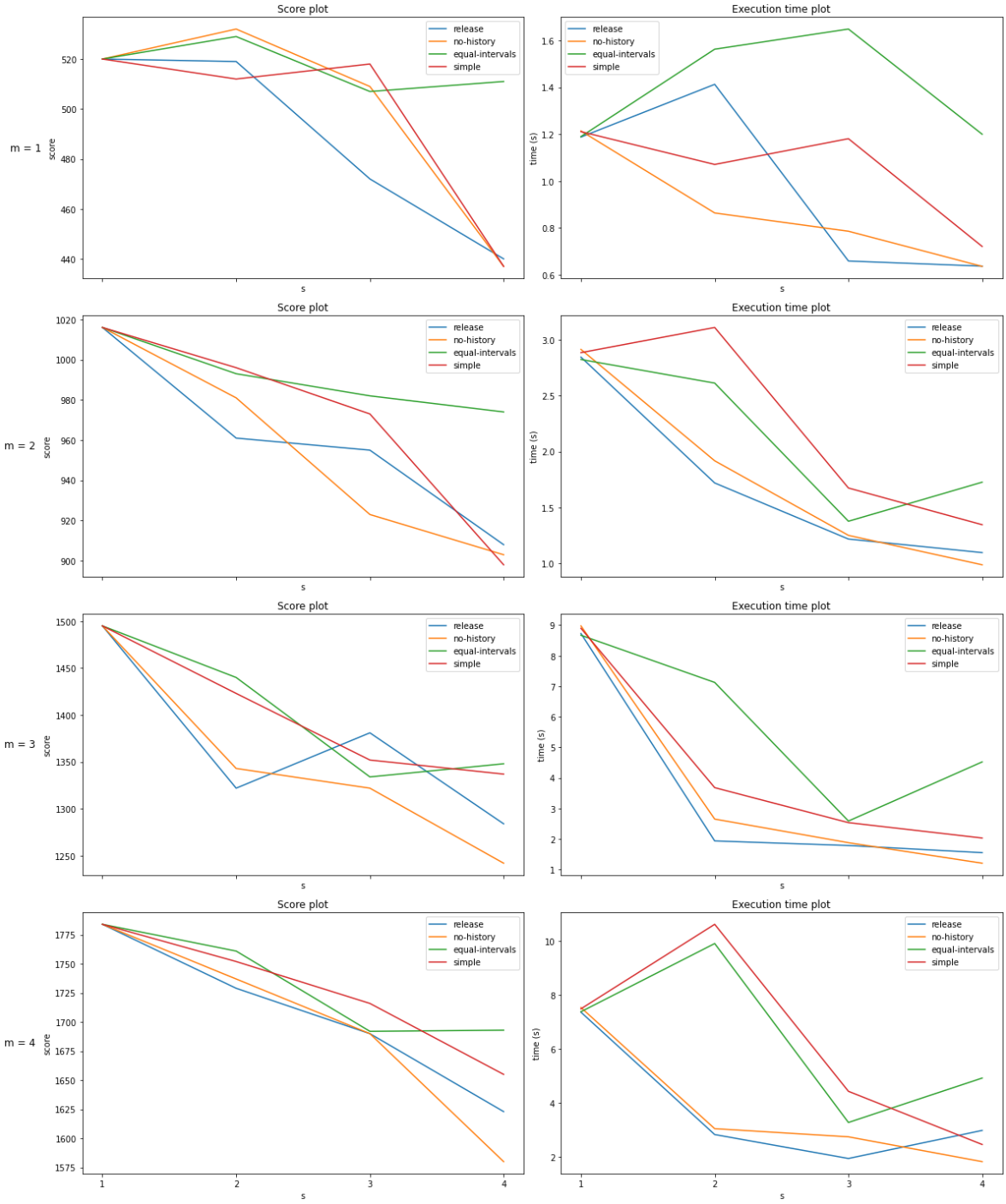
Εικόνα 5-8: Σύγκριση εκδόσεων του αλγορίθμου για το στιγμιότυπο εισόδου pr07 (72 pois)



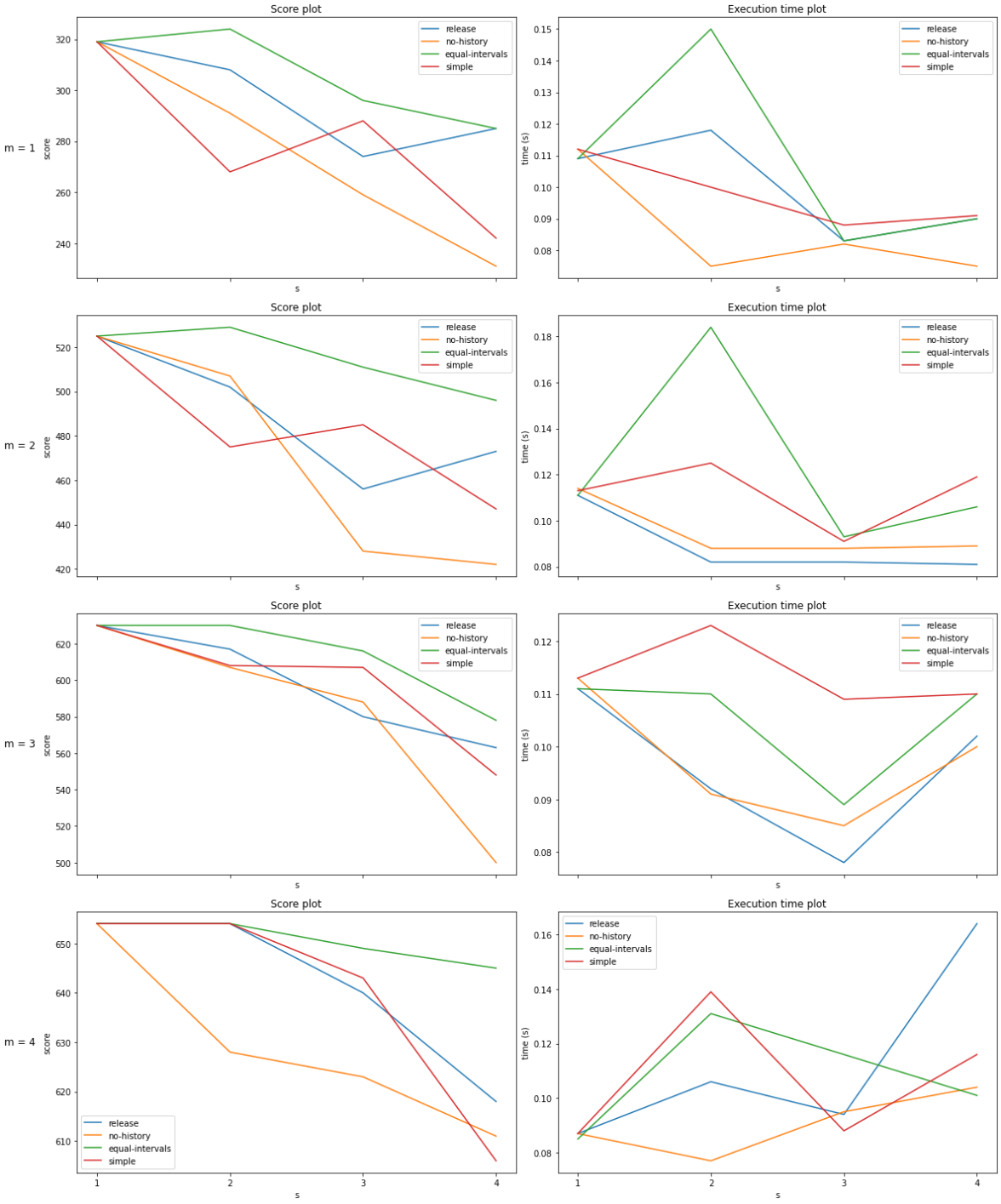
Εικόνα 5-9: Σύγκριση εκδόσεων του αλγορίθμου για το στιγμιότυπο εισόδου pr08 (144 pois)



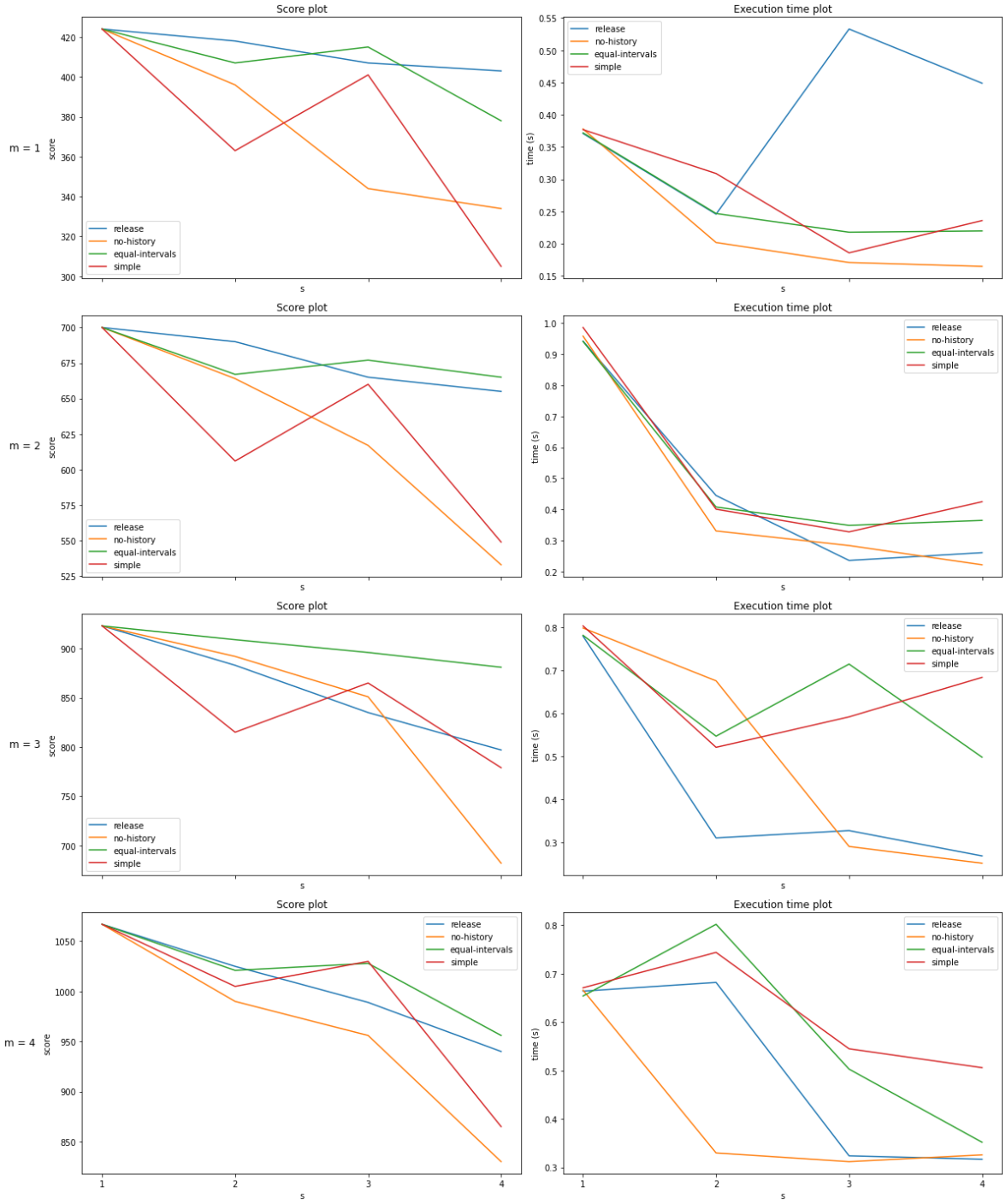
Εικόνα 5-10: Σύγκριση εκδόσεων του αλγορίθμου για το στιγμιότυπο εισόδου pr09 (216 pois)



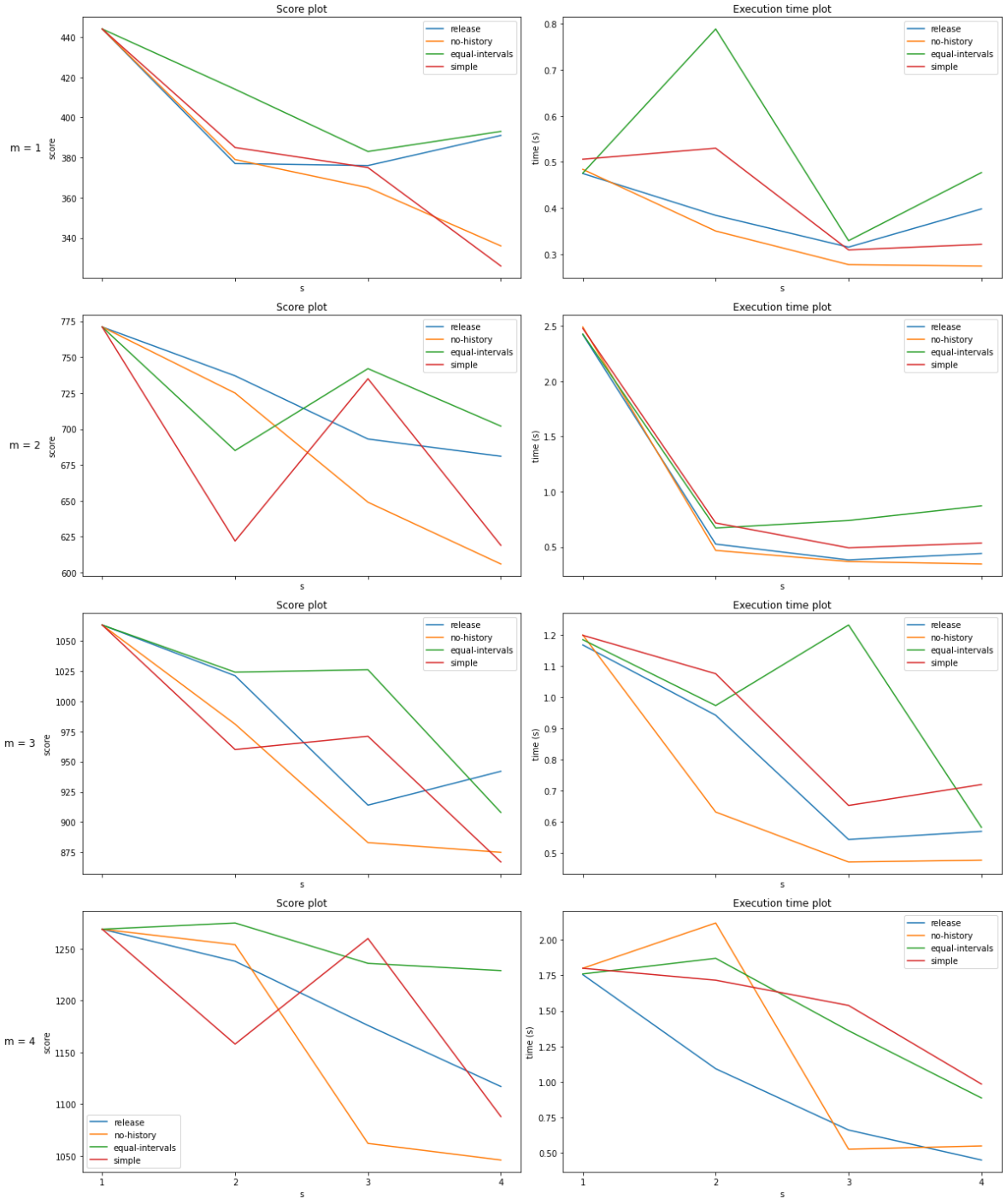
Εικόνα 5-11: Σύγκριση εκδόσεων του αλγορίθμου για το στιγμιότυπο εισόδου pr10 (288 pois)



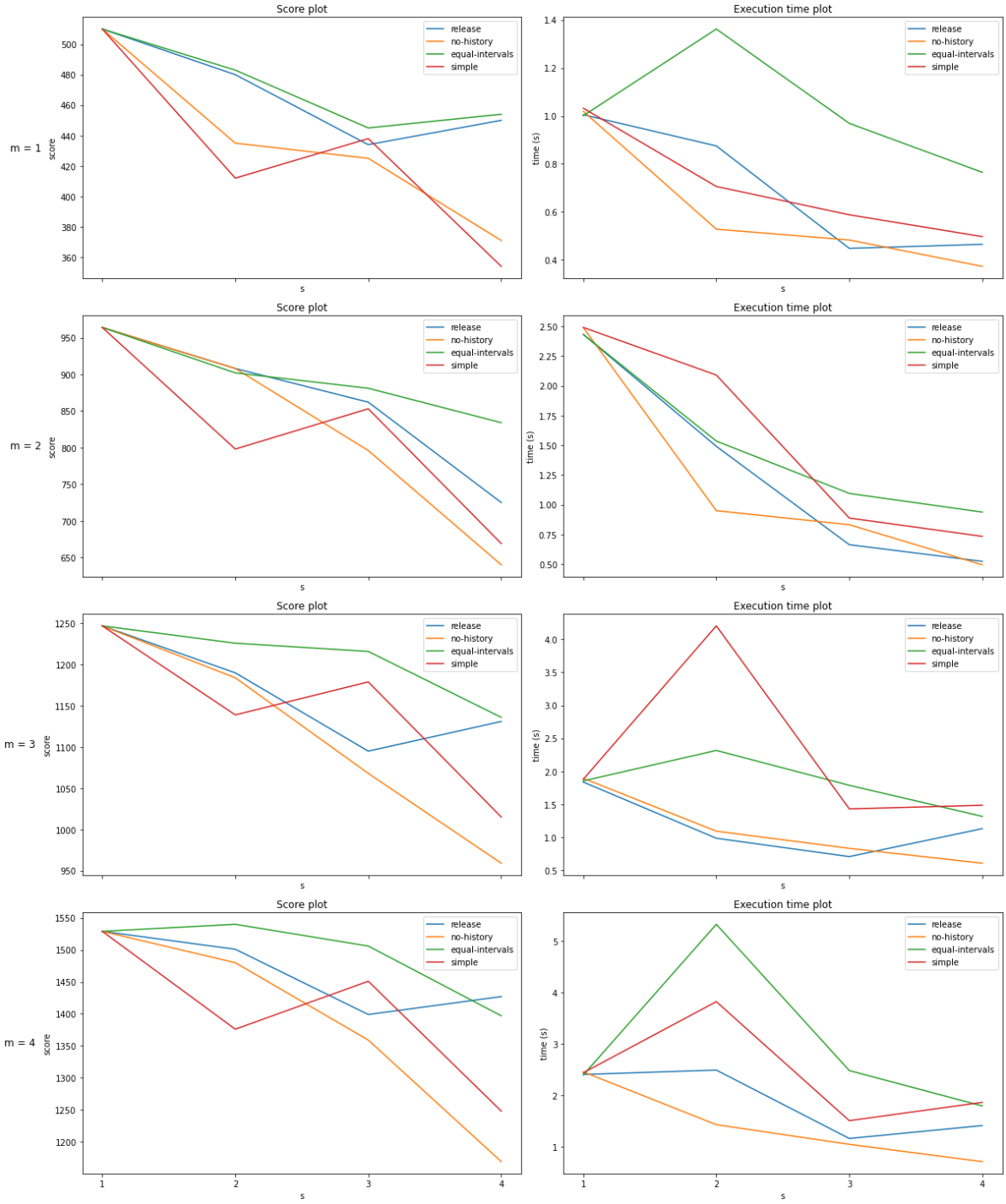
Εικόνα 5-12: Σύγκριση εκδόσεων του αλγορίθμου για το στιγμιότυπο εισόδου pr11 (48 pois)



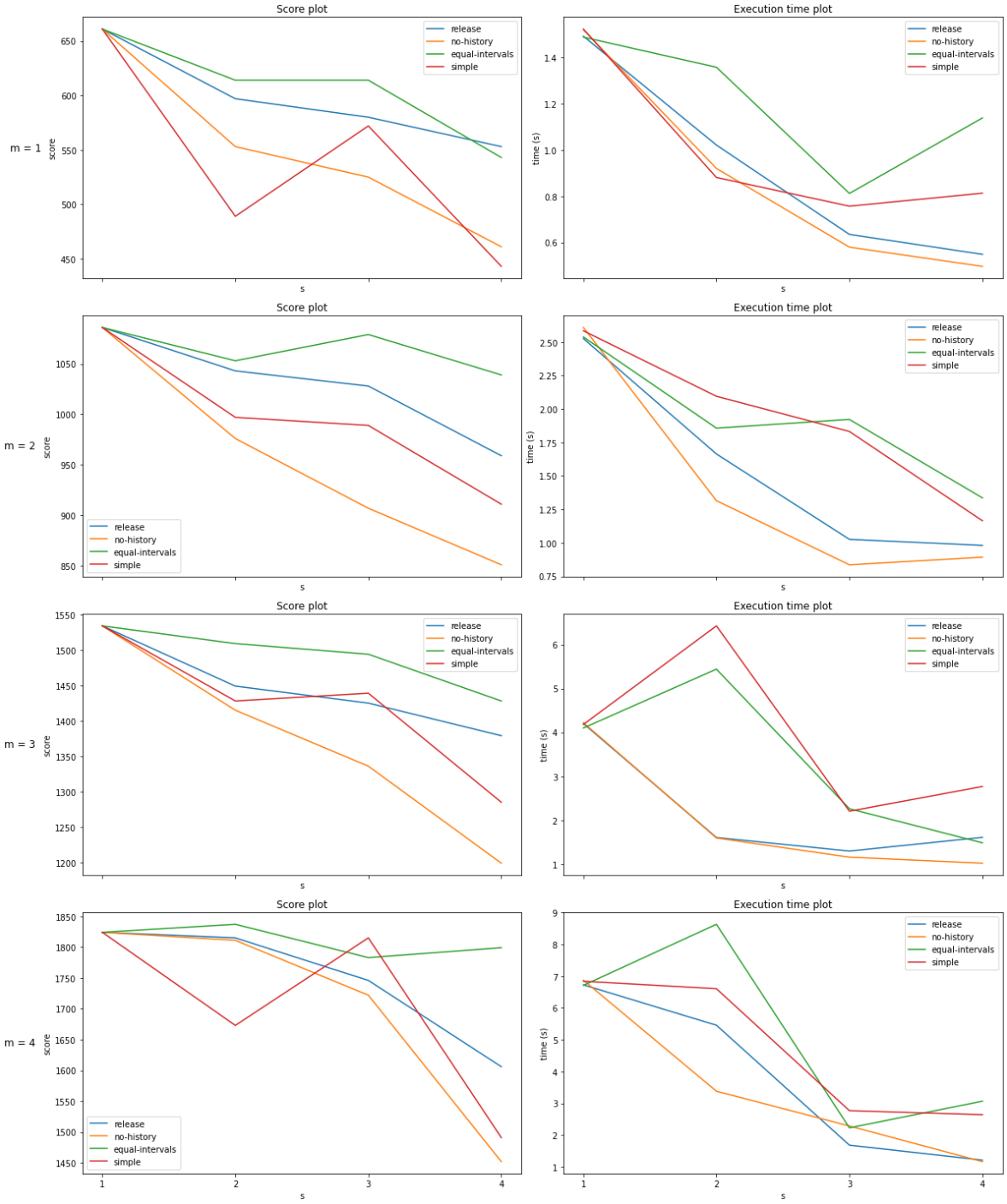
Εικόνα 5-13: Σύγκριση εκδόσεων του αλγορίθμου για το στιγμιότυπο εισόδου pr12 (96 pois)



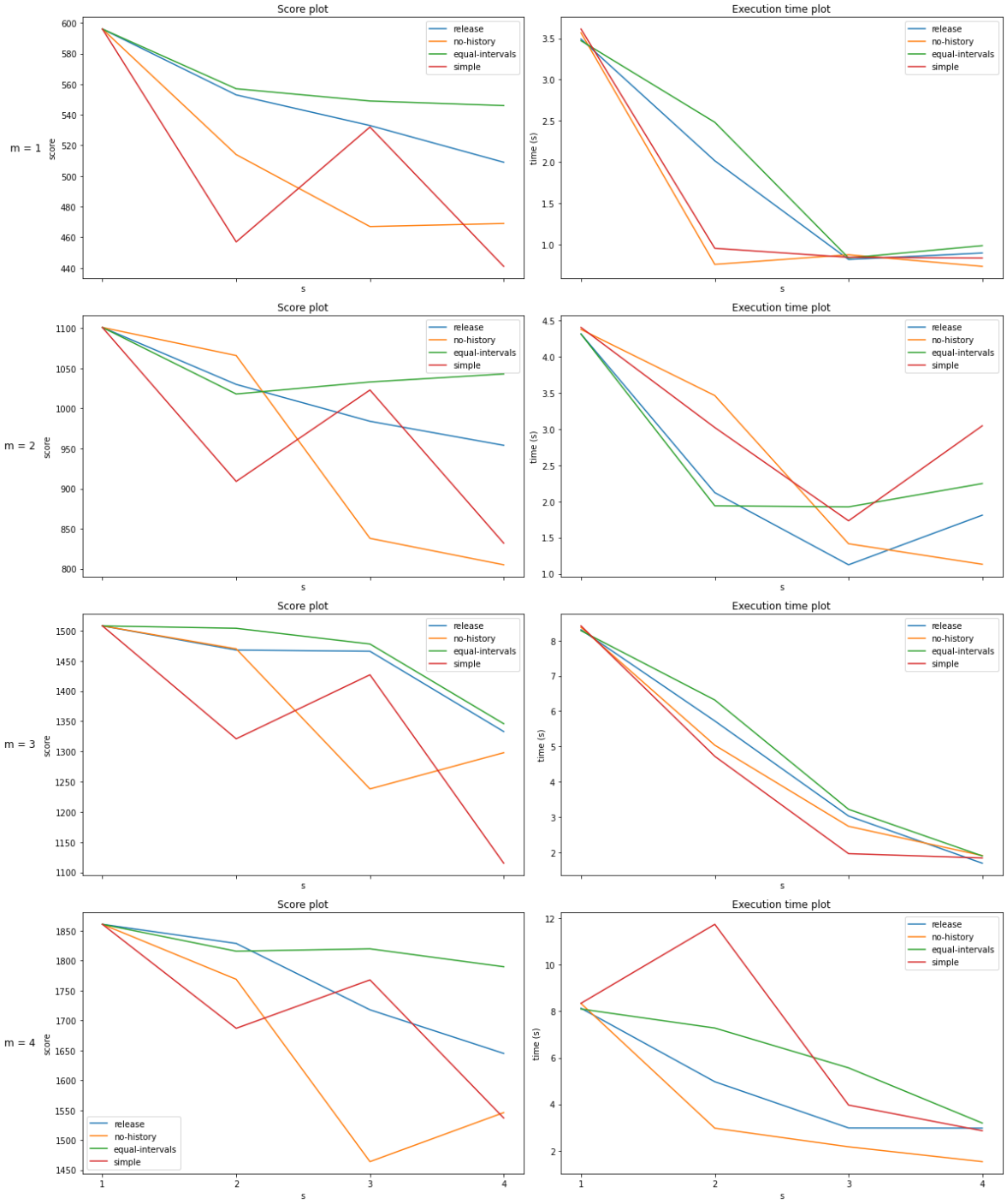
Εικόνα 5-14: Σύγκριση εκδόσεων του αλγορίθμου για το στιγμιότυπο εισόδου pr13 (144 pois)



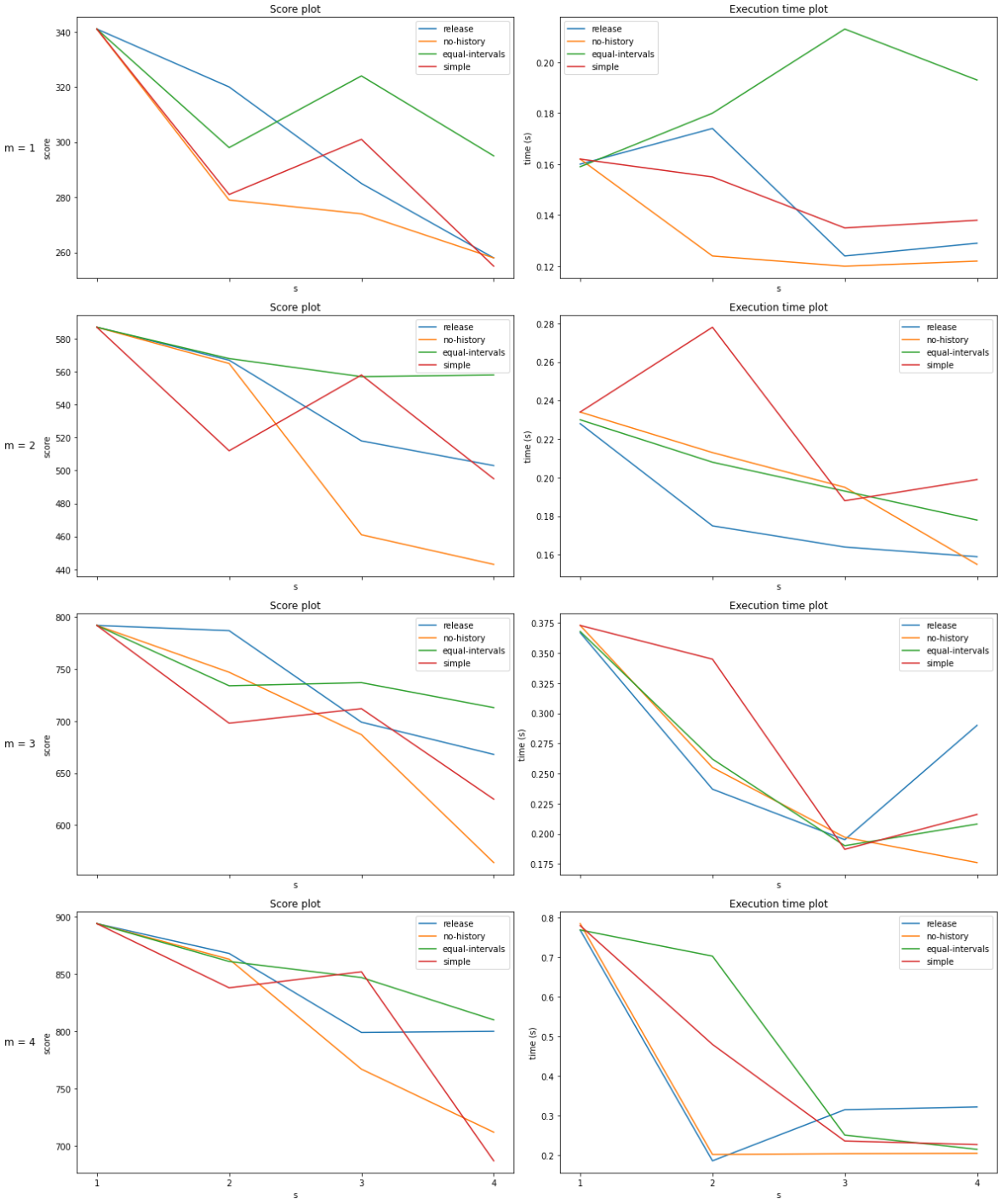
Εικόνα 5-15: Σύγκριση εκδόσεων του αλγορίθμου για το στιγμιότυπο εισόδου pr14 (192 pois)



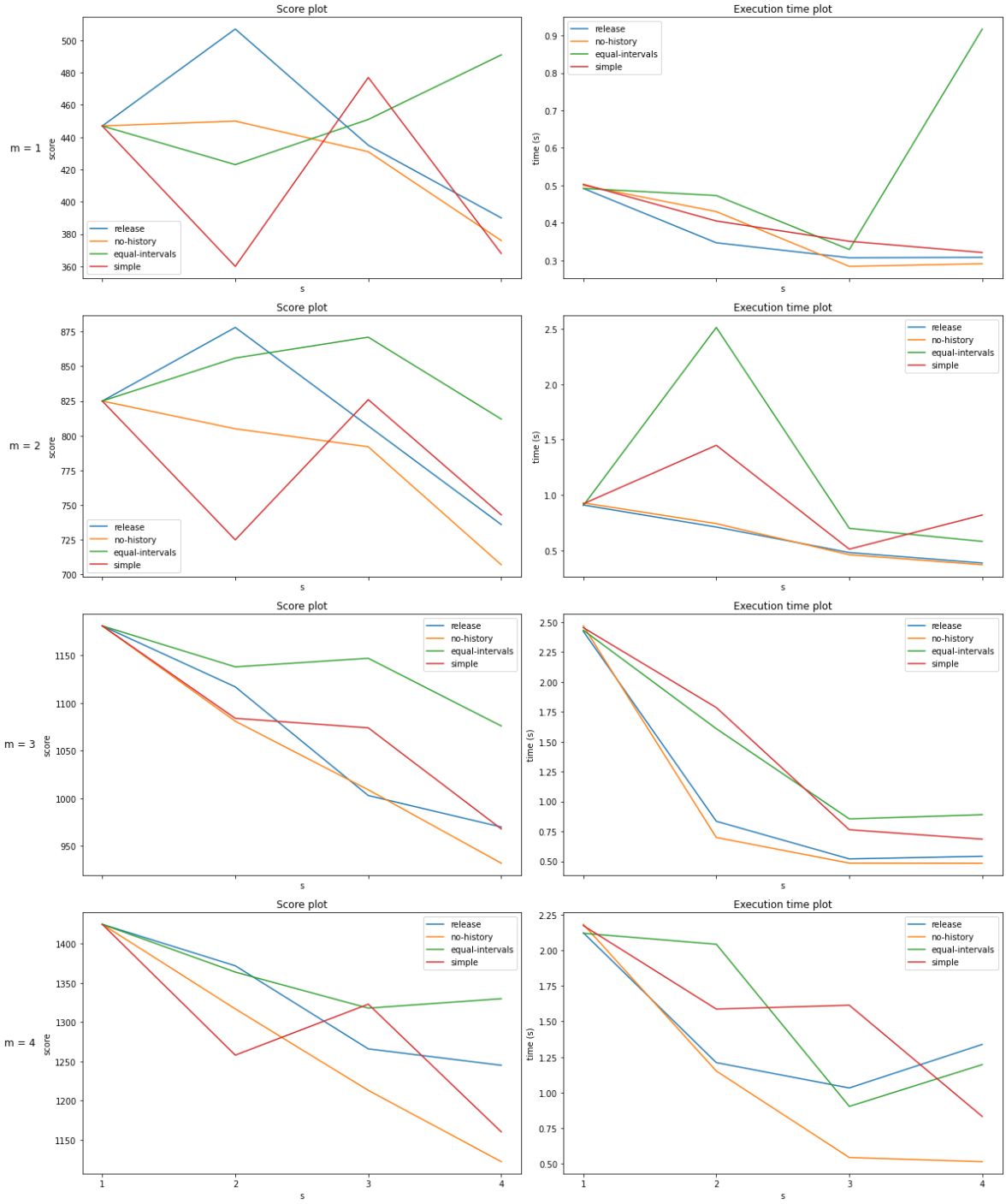
Εικόνα 5-16: Σύγκριση εκδόσεων του αλγορίθμου για το στιγμιότυπο εισόδου pr15 (240 pois)



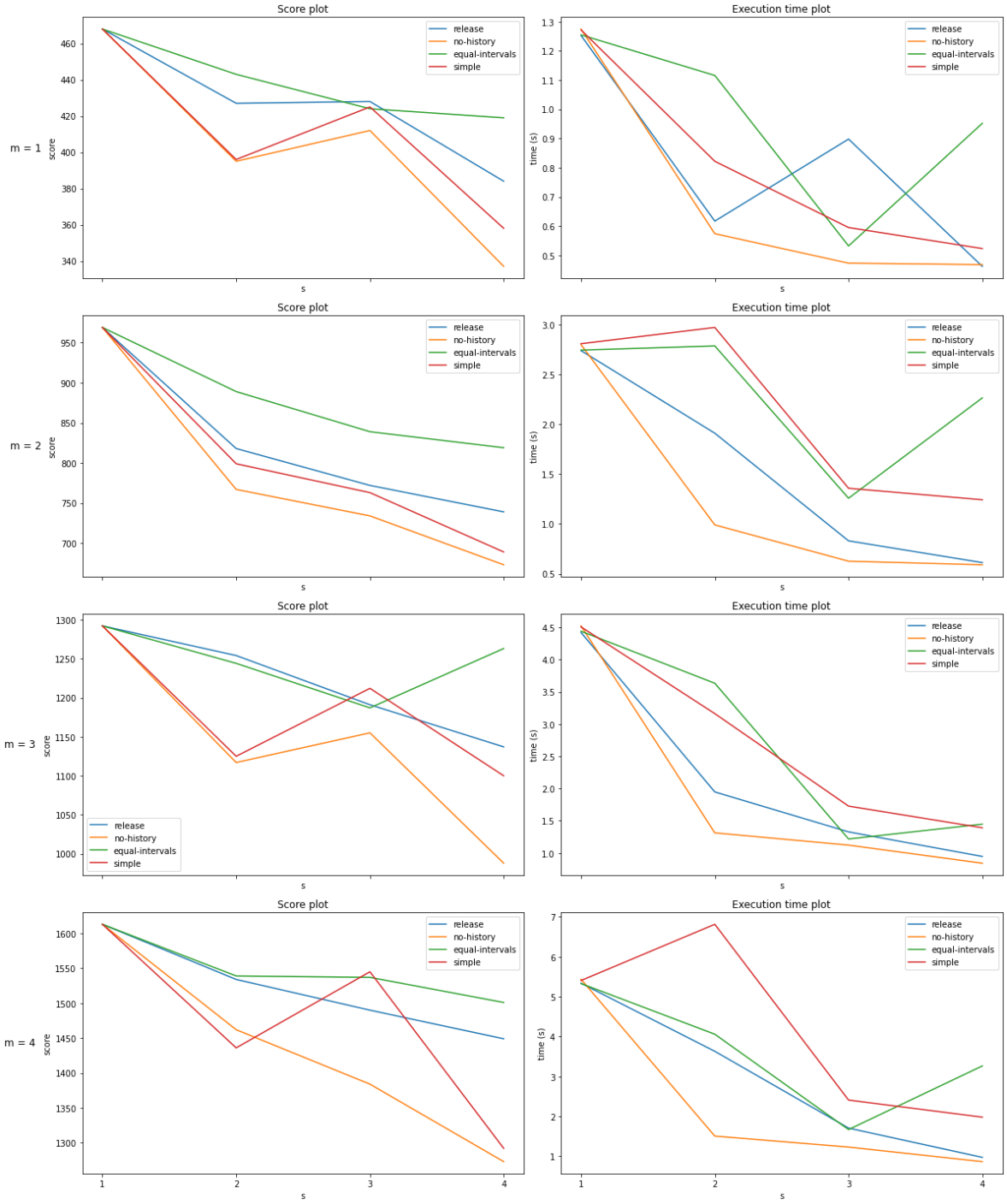
Εικόνα 5-17: Σύγκριση εκδόσεων του αλγορίθμου για το στιγμιότυπο εισόδου pr16 (288 pois)



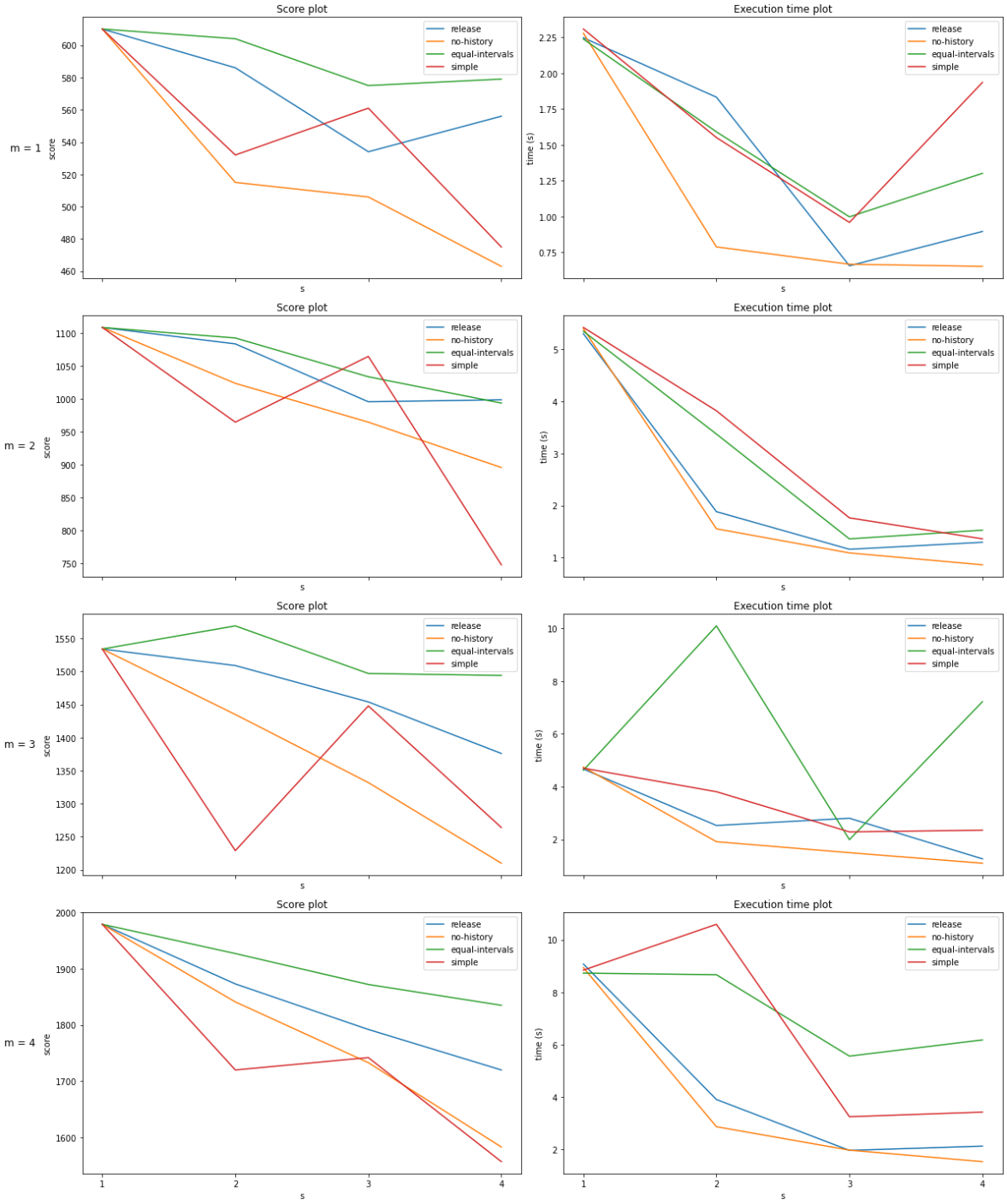
Εικόνα 5-18: Σύγκριση εκδόσεων του αλγορίθμου για το στιγμιότυπο εισόδου pr17 (72 pois)



Εικόνα 5-19: Σύγκριση εκδόσεων του αλγορίθμου για το στιγμιότυπο εισόδου pr18 (144 pois)



Εικόνα 5-20: Σύγκριση εκδόσεων του αλγορίθμου για το στιγμιότυπο εισόδου pr19 (216 pois)



Εικόνα 5-21: Σύγκριση εκδόσεων του αλγορίθμου για το στιγμιότυπο εισόδου pr20 (288 pois)

5.3 Στιγμιότυπο εισόδου της Αθήνας για το TTDP

Όπως αναφέρθηκε και στην Εισαγωγή, μιας από τις πιο σημαντικές εφαρμογές του OP, και πιο συγκεκριμένα του TOPTW, είναι και το Πρόβλημα Σχεδίασης Τουριστικών Διαδρομών (TTDP). Για το λόγο αυτό, δημιουργήθηκε ένα στιγμιότυπου εισόδου με αληθινά σημεία ενδιαφέροντος της Αθήνας που επιλέχθηκαν από την ιστοσελίδα http://index.pois.gr/ . Για την εύρεση των αληθινών διαδρομών μεταξύ των 158 σημείων, με αμάξι, χρησιμοποιήθηκε το OpenTripPlanner, ενώ για τη γραφική αναπαράσταση του στιγμιότυπου και της λύσης χρησιμοποιήθηκε η βιβλιοθήκη LeafletJS. Το στιγμιότυπο εισόδου αποτελείται από 158 κόμβους οι οποίοι ανήκουν σε 8 διαφορετικές κατηγορίες:

Κατηγορία	**Εύρος κέρδους**	**Πλήθος κόμβων**
Μουσεία	40-50	21
Θέατρα	30-40	43
Ελληνική Κουζίνα	25-30	33
Αξιοθέατα	20-25	4
Σινεμά	15-20	5
Εμπορικά Κέντρα	10-15	9
Τέχνες	5-10	42
Ξενοδοχεία	0-0	1

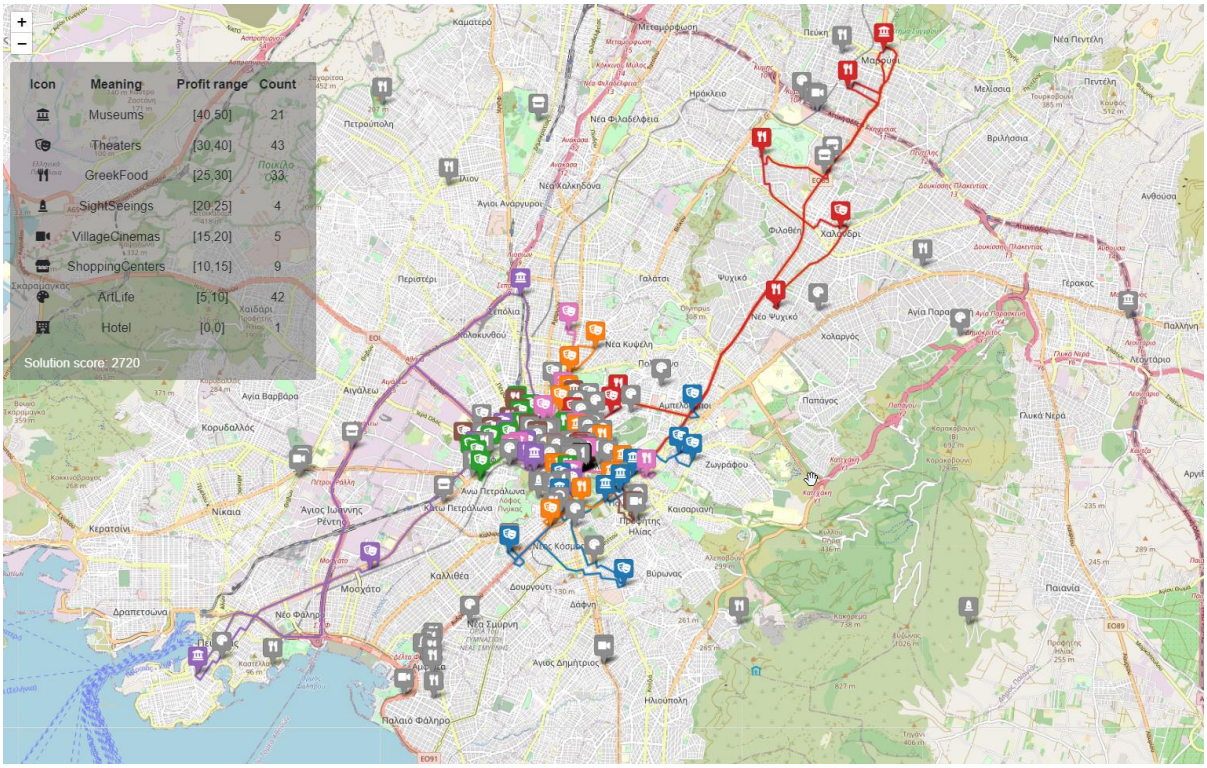
Το κάθε εύρος κέρδους αντιστοιχίστηκε τυχαία σε κάθε κατηγορία. Κάθε κόμβος έλαβε μια τυχαία τιμή κέρδους μέσα στο εύρος κέρδους της κατηγορίας που ανήκει.
Στη συνέχεια κατασκευάστηκαν τα εξής 10 χρονικά παράθυρα:

Ώρα ανοίγματος	**Ώρα κλεισίματος**
0	180
0	360
60	300
180	540
360	720
540	720
540	900
120	360
60	480
120	660

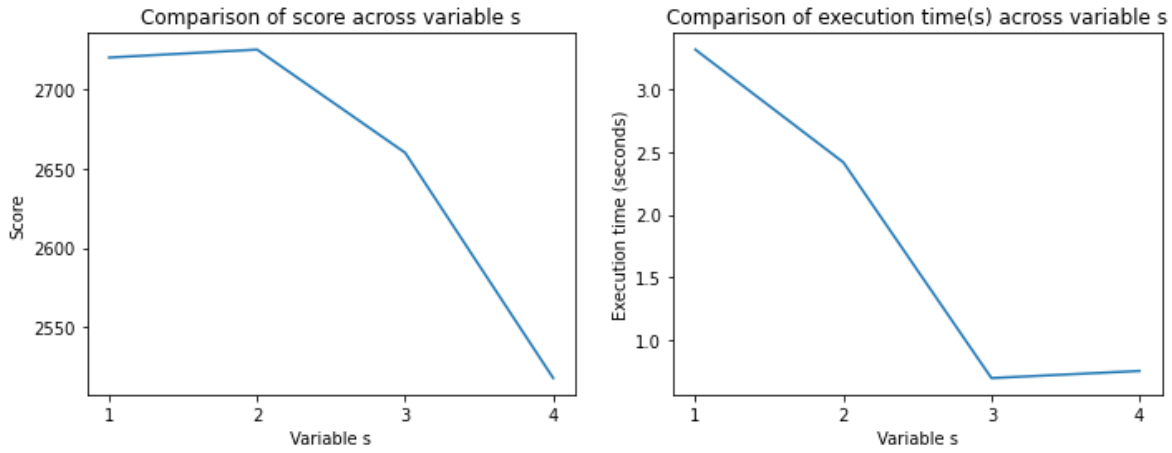
Κάθε 10% των σημείων ενδιαφέροντος πήρε και ένα διαφορετικό χρονικό παράθυρο. Η διάρκεια επίσκεψης κάθε κόμβου εξαρτάται από τη διάρκεια του χρονικού του παραθύρου, καθώς καταλαμβάνει τουλάχιστον το 1/6 της ενεργής διάρκειάς του. Για παράδειγμα, έστω ένας κόμβος

Α, που πήρε το χρονικό παράθυρο {180, 540} .Η χρονική διάρκεια που ο κόμβος Α είναι ενεργός (activityDuration) είναι 360 λεπτά. Η διάρκεια επίσκεψης του Α υπολογίστηκε ως εξής:

$$visitDuration_A = random(\frac{activityDuration_A}{6}, \frac{activityDuration_A}{5}) = random(60,72)$$



Εικόνα 5-22: Γραφική απεικόνιση 7 διαδρομών για το στιγμιότυπο εισόδου της Αθήνας, με S=1



Εικόνα 5-23: Σύγκριση των σκορ και των χρόνων εκτέλεσης για διαφορετικά s

6. Συμπεράσματα

Στην παρούσα εργασία μελετήθηκε το πρόβλημα Προσανατολισμού και οι επεκτάσεις του. Μια από τις πιο σημαντικές εφαρμογές του OP και συγκεκριμένα του TOPTW είναι το Πρόβλημα Σχεδίασης Τουριστικών Διαδρομών. Για το πρόβλημα Προσανατολισμού έχουν υλοποιηθεί πολλοί μεταευρετικοί αλγόριθμοι ένας από του οποίους είναι και ο αλγόριθμος Επαναλαμβανόμενης Τοπικής Αναζήτησης (ILS). Συγκεκριμένα, σκοπός της παρούσης εργασίας ήταν να βελτιώσει την ταχύτητα του αλγορίθμου ILS των Vansteenwegen et al. (2009) χωρίζοντας το αρχικό γράφημα G(V,E) του προβλήματος σε μικρότερα γραφήματα αλλά κρατώντας τις λύσεις σε υψηλό επίπεδο.

Ο τρόπος διαχωρισμού του γραφήματος είναι σημαντικός για την ταχύτητα και την απόδοση του αλγορίθμου. Για την παρούσα εργασία, ο διαχωρισμός έγινε με βάση τα χρονικά παράθυρα των κόμβων χωρίς να λαμβάνεται υπόψιν η τοποθεσία τους. Τα χρονικά υποδιαστήματα οριοθετήθηκαν με στόχο την ισοκατανομή των κόμβων σε αυτά. Ο ίσος διαμοιρασμός των κόμβων στα υποδιαστήματα, μειώνει την ταχύτητα του αλγορίθμου αλλά φαίνεται να μειώνει ελαφρώς και την ποιότητα των λύσεων.

Η οριστική ανάθεση ενός κόμβου σε ένα υποδιάστημα στην εκκίνηση του αλγορίθμου, πιθανότατα να παρήγαγε χαμηλής ποιότητας λύσεις. Για το λόγο αυτό, θεωρήθηκε ένα ιστορικό καταλληλότητας για κάθε κόμβο σχετικά με κάθε υποδιάστημα. Το ιστορικό αυτό ενημερώνεται κατά τη διάρκεια του αλγορίθμου, εκμεταλλευόμενο την επαναληπτικότητα του ILS. Με βάση αυτό το ιστορικό, ένας κόμβος μπορεί να δοκιμαστεί σε πολλά υποδιαστήματα, και να καταλήξει στο πιο κατάλληλο για αυτό. Από τα πειραματικά αποτελέσματα του Κεφαλαίου 5, φαίνεται πως η χρήση του ιστορικού καταλληλότητας βελτιώνει την ποιότητα των λύσεων.

Παρ’ όλα αυτά, σίγουρα υπάρχουν περιθώρια βελτίωσης στις παραπάνω τεχνικές. Για παράδειγμα, στην προεργασία του αλγορίθμου, θα μπορούσε να εφαρμοσθεί ένας απλός αλγόριθμος δημιουργίας συστάδων (π.χ. κ-μέσων) στους κόμβους, έτσι ώστε να σχηματιστούν k συστάδες. Εάν δύο κόμβοι, είναι στην ίδια συστάδα θα μπορούσε να θεωρούνται συγγενικοί κόμβοι. Η συγγένεια των κόμβων, θα μπορούσε να συνυπολογιστεί στο διαχωρισμό των Unvisited κόμβων σε υποδιαστήματα σε κάθε επανάληψη του ILS. Δηλαδή θα θεωρούταν πιο ευνοϊκό για έναν κόμβο, να μπει, ως Unvisited κόμβος, στο υποδιάστημα που έχει τους περισσότερους συγγενικούς κόμβους. Επίσης, στην εξίσωση της συγγένειας, θα μπορούσε να προστεθεί και η συχνότητα συνάντησης δύο κόμβων. Δηλαδή, εάν δύο κόμβοι εμφανίζονται συχνά μαζί στις λύσεις, τότε θα μπορούσε να αυξάνεται ο βαθμός συγγένειας μεταξύ τους.

7. Βιβλιογραφία

[1] T. Tsiligirides, «Heuristic methods applied to orienteering,» *Journal of the Operational Research Society,* pp. 797-809, 1984.

[2] G. L. k. S. Martello, «The selective travelling salesman problem,» *Discrete Applied Mathematics,* pp. 193-207, 1990.

[3] S. K. k. S. Morito, «An algorithm for single constraint maximum collection problem,» *Journal of the Operations Research Society of Japan,* pp. 515-531, 1988.

[4] O. A. M. T. L. P. V. a. W. S. Ander Garcia, «Personalized tourist route generation,» σε *Current Trends in Web Engineering - 10th International Conference on Web Engineering*, Vienna, Austria, 2010.

[5] P. V. a. D. V. Oudheusden, «Research, 209(1):1 10, 2011.,» *The mobile tourist guide: An OR opportunity,* pp. 21-27, 2007.

[6] W. S. G. V. B. a. D. V. O. Pieter Vansteenwegen, «Iterated local search for the team orienteering problem with time windows,» *Computers & Operations Research,* pp. 3281-3290, 2009.

[7] W. S. a. D. V. O. P. Vansteenwegen, «The orienteering problem: A survey,» *European Journal of Operational Research,* pp. 1-10, 2011.

[8] A. W. T. a. R. A. Z. C. E. Miller, «Integer programming formulations and travelling salesman problems,» *Journal of the Association for Computing Machinery,* pp. 326-329, 1960.

[9] V. N. &. R. Ravi, «The Directed Orienteering Problem,» *Algorithmica,* p. 1017–1030, 2011.

[10] A. B. S. C. a. A. M. N. Bansal, «Approximation algorithms for deadline-tsp and vehicle routing with time-windows,» *STOC '04: Proceedings of the thirty-sixth annual ACM symposium on Theory of computing,* pp. 166-174, 2014.

[11] J. S. B. M. a. G. N. E. M. Arkin, «Resource-constrained geometric network optimization,» *Proceedings of the 14th Annual Symposium on Computational Geometry, SCG '98,* pp. 307-316, 1998.

[12] K. C. a. S. Har-Peled, «The orienteering problem in the plane revisited,» *Proceedings of the 22nd Annual Symposium on Computational Geometry, SCG '06,* pp. 247-254, 2006.

[13] G. L. a. F. S. M. Gendreau, « Atabu search heuristic for the undirected selective travelling salesman problem,» *European Journal of Operational Research,* pp. 539-545, 1998.

[14] G. L. a. F. S. Michel Gendreau, «A branch-and-cut algorithm for the undirected selective traveling salesman problem,» *Networks,* pp. 263-273, 1998.

[15] L. L. a. R. V. Bruce L. Golden, «The orienteering problem,» *Naval Research Logistics (NRL),* pp. 307-318, 1987.

[16] J. J. S. G. P. T. Matteo Fischetti, «Solving the Orienteering Problem through Branch-and-Cut,» *INFORMS Journal on Computing,* pp. 133-148, 1998.

[17] Y. S. Y. a. M. K. Ram Ramesh, « An optimal algorithm for the orienteering tour problem,» *ORSA Journal On Computing,* pp. 155-165, 1992.

[18] L. L. A. L. M. Daniel Duque, «Solving the Orienteering Problem with Time Windows via the Pulse Framework,» *Computers & Operations Research,* pp. 168-176, 2015.

[19] A. L. M. Leonardo Lozano, «On an exact method for the constrained shortest path problem,» *Computers & Operations Research,* pp. 378-384, 2013.

[20] H. C. L. &. K. L. Aldy Gunawan, «An Iterated Local Search Algorithm for Solving the Orienteering Problem with Time Windows,» *Evolutionary Computation in Combinatorial Optimization,* p. 61–73, 2015.

[21] A. L. Fedor V. Fomin, «Approximation algorithms for time-dependent orienteering,» *Information Processing Letters,* pp. 57-62, 2002.

[22] K. S. E.-H. A. P. V. C. Verbeeck, «A fast solution method for the time-dependent orienteering problem,» *European Journal of Operational Research,* pp. 419-432, 2014.

[23] W. S. D. V. O. Pieter Vansteenwegen, «The orienteering problem: A survey,» *European Journal of Operational Research,* pp. 1-10, 2011.

[24] Z. Y. k. H. C. L. Aldy Gunawan, «A mathematical model and metaheuristics for time dependent orienteering problem,» *RESEARCH COLLECTION SCHOOL OF COMPUTING AND INFORMATION SYSTEMS,* 2014.

[25] C. A. k. Z. F. Liangjun Ke, «Ants can solve the team orienteering problem,» *Computers & Industrial Engineering,* pp. 648-665, 2008.

[26] P. V. G. V. B. D. V. O. Wouter Souffriau, «A greedy randomised adaptive search procedure for the team orienteering,» σε *EU/MEeting 2008 on metaheuristics for logistics and vehicle routing*, Troyes, France, 2008.

[27] D. C. D. A. M. Hermann Bouly, «A memetic algorithm for the team orienteering problem,» *Applications of Evolutionary Computing,* p. 649–658, 2010.

[28] S.-W. Lin, «Solving the team orienteering problem using effective multi-start simulated annealing,» *Applied Soft Computing,* pp. 1064-1073, 2013.

[29] A. Q. J. A. O. G. A. B. P. &. L. D. João Ferreira, «Solving the Team Orienteering Problem: Developing a Solution Tool Using a Genetic Algorithm Approach,» *Soft Computing in Industrial Applications,* p. 365–375, 2013.

[30] L. G. Roberto Montemanni, «Ant colony system for team orienteering problems with time windows,» *Foundations of Computing and Decision Sciences,* pp. 287-306, 2009.

[31] V. F. Y. Shih-Wei Lin, «A simulated annealing heuristic for the team orienteering problem with time windows,» *European Journal of Operational Research,* pp. 94-107, 2012.

[32] R. M. J. M. R. W. C. Nacima Labadie, «The Team Orienteering Problem with Time Windows: An LP-based Granular Variable Neighborhood Search,» *European Journal of Operational Research,* pp. 15-27, 2012.

[33] C. K. K. M. G. P. &. Y. T. Damianos Gavalas, «Cluster-Based Heuristics for the Team Orienteering Problem with Time Windows,» σε *International Symposium on Experimental Algorithms*, Rome, Italy, 2013.

[34] A. L. Qian Hu, «An iterative three-component heuristic for the team orienteering problem with time windows,» *European Journal of Operational Research,* pp. 276-286, 2014.

[35] J. B. J. R. M.-T. José Ruiz-Meza, «A GRASP to solve the multi-constraints multi-modal team orienteering problem with time windows for groups with heterogeneous preferences,» *Computers & Industrial Engineering,* τόμ. 162, 2021.

[36] C. K. K. M. G. P. N. V. Damianos Gavalas, «Heuristics for the time dependent team orienteering problem: Application to tourist route planning,» *Computers & Operations Research,* pp. 36-50, 2015.

[37] A. D. C. V. A. G. P. V. M. Khodadadian, «Time dependent orienteering problem with time windows and service time dependent profits,» *Computers & Operations Research,* τόμ. Volume 143, 2022.

[38] O. C. M. &. T. S. Helena R. Lourenço, «Iterated Local Search: Framework and Applications,» σε *Handbook of Metaheuristics*, 2010, p. 363–397.

[39] M. F. T. Korhan Karabulut, «An evolution strategy approach to the team orienteering problem with time windows,» *Computers & Industrial Engineering,* τόμ. 139, pp. 106-109, 2020.

[40] J. A. G. M. A. L. G. Cordeau, «A tabu search heuristic for periodic and multi-depot vehicle routing problems,» σε *Networks*, 1997, pp. 105-119.

[41] J. F. C. G. G. a. G. L. Andrea Attanasio, «Parallel tabu search heuristics for the dynamic multi-vehicle dial-a-ride problem,» σε *Parallel Computing*, 2004, pp. 377-387.

[42] D. L. E. a. R. A. S. Abdel Monaem F.M. AbdAllah, «On solving periodic re-optimization dynamic vehicle routing problems,» *Applied Soft Computing,* pp. 1-12, 2017.

[43] G. L. a. F. S. M. Gendreau, «A branch-and-cut algorithm for the undirected selective traveling salesman problem,» *Networks,* pp. 263-273, 1998.

[44] C. T. A. a. Z. R. Miller, «Integer Programming Formulations and Traveling Salesman Problems,» *Journal of the Association for Computing Machinery,* pp. 326-329, 1960.

[45] E. M.-H. Hao Tang, «A TABU search heuristic for the team orienteering problem,» *Computers & Operations Research,* pp. 1379-1407, 2005.

[46] O. C. M. &. T. S. Helena Ramalhinho Lourenço, «Iterated Local Search: Framework and Applications,» σε *Handbook of Metaheuristics*, 2018, pp. 129-168.