

Πανεπιστήμιο Πειραιώς – Τμήμα Πληροφορικής

Πρόγραμμα Μεταπτυχιακών Σπουδών

«Προηγμένα Συστήματα Πληροφορικής, Προηγμένες Τεχνολογίες Ανάπτυξης Λογισμικού»

**Μεταπτυχιακή Διατριβή**

|  |  |
| --- | --- |
| Τίτλος Διατριβής | Διαχωρισμένη Τοπική Αναζήτηση για το Πρόβλημα Ομαδικού Προσανατολισμού με Χρονικά Παράθυρα |
| Ονοματεπώνυμο Φοιτητή | Ευστάθιος Καψιώτης |
| Πατρώνυμο | Ηλίας |
| Αριθμός Μητρώου | ΜΠΣΠ17030 |
| Επιβλέπων | Χαράλαμπος Κωνσταντόπουλος, Βαθμίδα |

Ευχαριστίες

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

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 την 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 6](#_Toc128268080)

[2. Το Πρόβλημα Προσανατολισμού 8](#_Toc128268082)

[2.1 Το πρόβλημα Προσανατολισμού με Χρονικά Παράθυρα (OPTW) 9](#_Toc128268083)

[2.2 Το Χρονικά Εξαρτώμενο Πρόβλημα Προσανατολισμού (TDOP) 10](#_Toc128268084)

[2.3 Το πρόβλημα Ομαδικού Προσανατολισμού (TOP) 14](#_Toc128268085)

[2.4 Το πρόβλημα του Ομαδικού Προσανατολισμού με Χρονικά Παράθυρα 16](#_Toc128268086)

[2.5 Το Πρόβλημα Χρονικά Εξαρτώμενου Ομαδικού Προσανατολισμού με Χρονικά Παράθυρα (TDTOPTW) 20](#_Toc128268087)

[3. Αλγόριθμος Επίλυσης TOPTW 22](#_Toc128268088)

[3.1 Τεχνική Επαναλαμβανόμενης Τοπικής Αναζήτησης 22](#_Toc128268089)

[3.2 Υλοποίηση Επαναλαμβανόμενης Τοπικής Αναζήτησης στο TOPTW 23](#_Toc128268090)

[3.2.1 Βήμα Εισαγωγής 23](#_Toc128268091)

[3.2.2 Βήμα Διαταραχής 25](#_Toc128268092)

[3.2.3 Ευρετικός Αλγόριθμος Επαναλαμβανόμενης Τοπικής Αναζήτησης 26](#_Toc128268093)

[4. Διαχωρισμός Τοπικής Αναζήτησης 28](#_Toc128268094)

[4.1 Αρχικοποίηση των χρονικών υπο-διαστημάτων 30](#_Toc128268095)

[4.2 Διαχωρισμός των Unvisited κόμβων 31](#_Toc128268096)

[4.3 Διαχωρισμένη Τοπική Αναζήτηση 31](#_Toc128268097)

[4.3.1 Προσθήκη τελικών κόμβων 33](#_Toc128268098)

[4.3.2 Προσθήκη αρχικών κόμβων 35](#_Toc128268099)

[4.4 Υπερχείλιση Διαδρομών 38](#_Toc128268100)

[5. Πειραματικά Αποτελέσματα 40](#_Toc128268101)

[5.1.1 Instance pr10 44](#_Toc128268102)

[6. Βιβλιογραφία 45](#_Toc128268103)

# Εισαγωγή

Το Πρόβλημα Προσανατολισμού (Orienteering Problem, OP) αναφέρθηκε για πρώτη φορά από τον Tsiligirides (1984) 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) και σαν το Πρόβλημα της Μέγιστης Συλλογής (Maximum Collection Problem, Kataoka & Morito, 1988).

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):

* 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 TTDP Vansteenwegen & Oudheudsen (2007). 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) 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 τα προβλήματα OP, VRP καθώς είναι συγγενικά και αναφέρονται διάφορες αλγοριθμικές προσεγγίσεις που έχουν προταθεί. Στο Κεφάλαιο 3 παρουσιάζεται ο αλγόριθμος ILS (Vansteenwegen et al. 2009) 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.

# 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 κ.α. 2011)[;] χρησιμοποιώντας τις εξής μεταβλητές:

* N ο αριθμός των κόμβων (1, 2, ..., N) με αρχικό κόμβο s = 1 και τελικό t = N
* η τιμή κέρδους της επίσκεψης στον κόμβο i
* το κόστος μετακίνησης από τον κόμβο i στον κόμβο j
* εάν η επίσκεψη στον κόμβο i ακολουθείται από την επίσκεψη στον κόμβο j, ειδάλλως

Χρησιμοποιώντας λοιπόν τους παραπάνω συμβολισμούς προκύπτουν οι παρακάτω σχέσεις:

|  |  |  |
| --- | --- | --- |
|  |  | (2.1) |
|  |  | (2.2) |
|  |  | (2.3) |
|  |  | (2.4) |
|  |  | (2.5) |
|  |  | (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 Ν 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 Ν 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 . Οι σχέσεις 5 και 6 εμποδίζουν την ύπαρξη υπό-διαδρομών (Miller, Tucker, & Zermin 1960)[;].

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) ή μη κατευθυνόμενο (Tsiligirides 1984[;] , Bansal κ.α. 2004 [;])
* Έχει προκαθοριστεί ένας αρχικός κόμβος αλλά όχι ένας τελικός (rooted OP) (Arkin κ.α. 1998[;], Chen and Har-Peled 2006[;]). Το rooted OP αποτελεί ευκολότερο πρόβλημα από το κλασσικό OP
* Δεν έχει καθοριστεί ούτε αρχικός ούτε τελικός κόμβος (unrooted OP) (Gendreau κ.α. 1998)[;, ;]. Αποτελεί ευκολότερη περίπτωση από αυτή του rooted OP

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 κ.α. (1987), Laporte και Martello (1990) 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 κ.α. (1998), Fischetti κ.α. (1998) και branch-and-bound Laporte and Martello (1990), Ramesh κ.α. (1992). 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), 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)), 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.

## 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(2014) 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). Ο αλγόριθμος πυροδοτώντας έναν παλμό από τον αρχικό κόμβο και ωθώντας τον προς τον τελικό , 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 (Duque et.al(2014)).

Οι Gunawan et al.(2015a) προτείνουν έναν αλγόριθμο Επαναλαμβανόμενης Τοπικής Αναζήτησης (ILS) παρόμοιο με τον ILS των Vansteenwegen et al. (2009) που θα αναλυθεί περαιτέρω στο 3ο Κεφάλαιο. Ο αλγόριθμος των Gunawan et al.(2015a) 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. διαφέρει από τον ILS των Vansteenwegen et al. 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.

## Το Χρονικά Εξαρτώμενο Πρόβλημα Προσανατολισμού (TDOP)

Οι Fomin and Lingas (2002) 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 Α στο σημείο Β δεν είναι σταθερός καθ’ όλη τη διάρκεια της ημέρας όπως θεωρείται στο OP αλλά εξαρτάται από την ώρα αναχώρησης από το Α. 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) περιέγραψαν ένα μοντέλο Μεικτού Ακέραιου Προγραμματισμού (MIP) με βάση το MIP για το OP (Vansteenwegen, Souffriau, & Van Oudheusden, (2011)).

* εάν ο χρήστης διατρέχει το τόξο i → j και η ώρα αναχώρησης από τον i είναι μέσα στο χρονικό διάστημα t, ειδάλλως
* η ώρα αναχώρησης στο χρονικό διάστημα t όταν διατρέχεται το i → j
* ο συντελεστής κλίσης του γραμμικού χρονικά εξαρτώμενου χρόνου ταξιδιού
* ο συντελεστής παρεμπόδισης του γραμμικού χρονικά εξαρτώμενου χρόνου ταξιδιού
* το κάτω όριο του χρονικού διαστήματος t για την ακμή i → j
* ο αριθμός των χρονικών διαστημάτων για την ακμή i → j
* κέρδος του κόμβου i
* ο μέγιστος συνολικός χρόνος ταξιδιού
* Ο αρχικός κόμβος είναι ο 1 και ο τελικός είναι ο Ν

|  |  |  |
| --- | --- | --- |
|  |  | (2.7) |
|  |  | (2.8) |
|  |  | (2.9) |
|  |  | (2.10) |
|  |  | (2.11) |
|  |  | (2.12) |
|  |  | (2.13) |
|  |  | (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 Ν. 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 . 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 μαζί με τους αντίστοιχους συντελεστές και . Με βάση αυτούς τους συντελεστές, μπορεί να υπολογιστεί ο εκάστοτε χρόνος ταξιδιού

Επίσης με βάση τους χρόνους ταξιδιού και το σύνολο των χρονοθυρίδων () τα θ και η μπορούν να υπολογιστούν ως εξής:

Οι Verbeeck et al.(2014a) 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 συνολικό χρόνο ολίσθησης της τροχιάς. Πιο συγκεκριμένα σε περίπτωση που η ακμή A → C περιλαμβάνει κεντρικούς δρόμους με μεγάλη πιθανότητα σύγχυσης ενώ οι ακμές A → B και B → C περιλαμβάνουν πιο ερημικούς δρόμους με μικρότερη πιθανότητα σύγχυσης, ίσως σε ώρα αιχμής να ισχύει . 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) ανέπτυξαν το δικό τους μοντέλο Ακέραιου Γραμμικού Προγραμματισμού για το TDOP. Χρησιμοποιώντας τους παρακάτω συμβολισμούς:

* εάν το ταξίδι από τον κόμβο i προς τον κόμβο j αρχίζει στη χρονική περίοδο t, ειδάλλως
* η τιμή κέρδους του κόμβου i
* ο συνολικός διαθέσιμος χρόνος
* ο χρόνος ταξιδιού της ακμής i → j που ξεκίνησε κατά τη χρονική περίοδο t προκύπτουν οι σχέσεις

|  |  |  |
| --- | --- | --- |
|  |  | (2.15) |
|  |  | (2.16) |
|  |  | (2.17) |
|  |  | (2.18) |
|  |  | (2.19) |
|  |  | (2.20) |
|  |  | (2.21) |
|  |  | (2.22) |
|  |  | (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) 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 () και η αργότερη ώρα αναχώρησης () 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.

## 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) ως εξής:

* k ο αριθμός των διαφορετικών διαδρομών
* εάν η επίσκεψη στον κόμβο i ακολουθείται από την επίσκεψη στον κόμβο j στη διαδρομή m ειδάλλως
* εάν πραγματοποιείται επίσκεψη στον κόμβο i στη διαδρομή m, ειδάλλως
* η θέση της επίσκεψης στον κόμβο i στη διαδρομή m

Χρησιμοποιώντας τους παραπάνω συμβολισμούς προκύπτουν οι παρακάτω σχέσεις:

|  |  |  |
| --- | --- | --- |
|  |  | (2.24) |
|  |  | (2.25) |
|  |  | (2.26) |
|  |  | (2.27) |
|  |  | (2.28) |
|  |  | (2.29) |
|  |  | (2.30) |
|  |  | (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 Ν 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).

Οι Hao Tang, Elise Miller-Hooks (2005) προτείνουν ένα μεταευρετικό αλγόριθμο αναζήτησης Tabu για την επίλυση του TOP.

Οι Liangjun Ke et al. (2007) 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) 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) 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) 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) 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 αρχικές λύσεις , όπου το αντιπροσωπεύει το πλήθος των σημείων έναρξης του MSA. Για κάθε αρχική λύση παράγεται μια γειτονιά λύσεων από την οποία επιλέγεται κάποια νέα λύση . Εάν η υπερτερεί στην τιμή κέρδους από την , τότε η πρώτη αντικαθιστά τη δεύτερη. Εάν όμως μειονεκτεί, τότε της ανατίθεται μία πιθανότητα αντικατάστασης της που είναι αντιστρόφως ανάλογη με τη διαφορά των 2 λύσεων καθώς όσο καλύτερη είναι η από την , 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. (2014) 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

## 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:

* εάν στη διαδρομή d η επίσκεψη στον κόμβο i προηγείται από την επίσκεψη στον j, ειδάλλως
* εάν πραγματοποιείται επίσκεψη στον κόμβο i στη διαδρομή d, ειδάλλως
* η χρονική στιγμή έναρξης της επίσκεψης στον κόμβο i στη διαδρομή d
* μια σταθερά M

Χρησιμοποιώντας τους παραπάνω συμβολισμούς προκύπτουν οι σχέσεις:

|  |  |  |
| --- | --- | --- |
|  |  | (2.32) |
|  |  | (2.33) |
|  |  | (2.34) |
|  |  | (2.35) |
|  |  | (2.36) |
|  |  | (2.37) |
|  |  | (2.38) |
|  |  | (2.39) |
|  |  | (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 . Οι σχέσεις 2.38 και 2.39 περιορίζουν την έναρξη της επίσκεψης στο χρονικό παράθυρο του εκάστοτε κόμβου.

Οι Montemanni & Gambardella (2009) 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 (pheromone trail) η οποία διαθέτει τη πληροφορία σχετικά με το πόσο «καλή» υπήρξε στο παρελθόν η ακμή i – j.
  + Ο βαθμός επιθυμίας (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) 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) προτείνουν την προσάρτηση των διαδρομών σε μία ενιαία θεωρώντας τον αρχικό και τελικό κόμβο ίδιους.

Οι Vansteenwegen et al. (2009) 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 (). Στη συνέχεια, υπολογίζεται η τιμή για κάθε κόμβο i. Ο κόμβος με τη μεγαλύτερο 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) 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) 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 η 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) 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 (). 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 . 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 (στην αρχή του αλγορίθμου), ισχύει ότι . Η clusterParameter μειώνεται στη πορεία του αλγορίθμου μέχρι να φτάσει το 1 όπου πλέον θα ισχύει .

Οι Hu & Lim (2014) 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.

## 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) χρησιμοποίησαν το TDOPTW για να μοντελοποιήσουν το TTDP και ανέπτυξαν τρεις ευρετικούς αλγορίθμους, τον TDCSCRoutes, τον SlackCSCRoutes και τον AvgCSCRoutes.

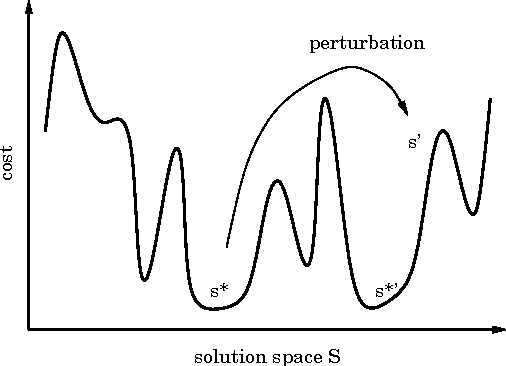
* Time Dependent CSCRoutes (TDCSCRoutes):  
  Ο TDCSCRoutes διαφέρει στο βήμα εισαγωγής από τον CSCRoutes που περιεγράφηκε στην υποενότητα του TOPTW καθώς τροποποιείται έτσι ώστε να μπορεί να χειριστεί χρονικά εξαρτημένα κόστη ακμών. Οπότε το προπαρασκευαστικό στάδιο ομαδοποίησης των κόμβων με τη χρήση 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 , όπου ο όρος συμβολίζει το μέγιστο επιτρεπτό χρόνο έναρξης της επίσκεψης στον κόμβο i έτσι ώστε να μην επηρεάζει την εγκυρότητα της διαδρομής. Όσο μικρότερο είναι το , 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 .
* 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.

# Αλγόριθμος Επίλυσης TOPTW

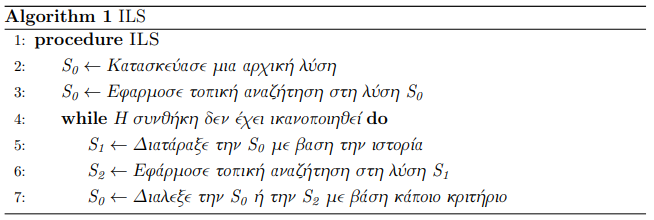
Για τη παρούσα εργασία, ο αλγόριθμος που χρησιμοποιήθηκε για την επίλυση του TOPTW είναι ο Iterated Local Search (Vansteenwegen et al. (2009)) ο οποίος βασίζεται στη τεχνική της Επαναλαμβανόμενης Τοπικής Αναζήτησης.

## Τεχνική Επαναλαμβανόμενης Τοπικής Αναζήτησης

Η διαφορά της Επαναλαμβανόμενης Τοπικής Αναζήτησης από την Τοπική Αναζήτηση είναι πως η δεύτερη μπορεί να οδηγήσει σε τοπικό βέλτιστο, κάτι που αποφεύγεται με την πρώτη καθώς προκαλεί μία διαταραχή στην εκάστοτε τοπικά βέλτιστη λύση.



Πιο αναλυτικά, στην Επαναλαμβανόμενη Τοπική Αναζήτηση κατασκευάζεται μια αρχική λύση S0, η οποία βελτιώνεται μέχρι συναντηθεί ένα τοπικό βέλτιστο. Έπειτα, λαμβάνοντας υπόψιν την ιστορία της διαδικασίας, προκαλείται 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 λύση. 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.

Οι (Vansteenwegen et al. (2009)) προσάρμοσαν στον αλγόριθμο τους την παραπάνω τεχνική για την επίλυση του TOPTW. Παρακάτω παρουσιάζεται η υλοποίηση του ILS για τους σκοπούς της παρούσας εργασίας.

## Υλοποίηση Επαναλαμβανόμενης Τοπικής Αναζήτησης στο TOPTW

Ο αλγόριθμος ILS αποτελείται ουσιαστικά από 2 βήματα, την Εισαγωγή (Τοπική Αναζήτηση) και τη Διαταραχή. Όμως πριν αναλυθούν ενδελεχώς, πρέπει πρώτα να διευκρινιστούν τα χαρακτηριστικά του κάθε κόμβου.

Έτσι λοιπόν, για κάθε κόμβο i, είναι εξαρχής γνωστές οι εξής πληροφορίες(σταθερές):

* οι συντεταγμένες του (,)
* το κέρδος της επίσκεψης ()
* η χρονική διάρκεια της επίσκεψης ()
* η ώρα έναρξης λειτουργίας του ()
* η ώρα παύσης λειτουργίας του ()

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 ()
* η διάρκεια αναμονής ()
* η ώρα έναρξης της επίσκεψης ()
* η ώρα αναχώρησης ()
* η διάρκεια συνολικής κατανάλωσης χρόνου λόγω της εισαγωγής του i ()
* ο μέγιστος χρόνος που μπορεί να παραταθεί η έναρξη της επίσκεψης ()

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 () μεταξύ αυτών των κόμβων. Ακόμα, πρέπει να αναφερθεί πως για την υλοποίηση του αλγόριθμου, οι κόμβοι οργανώνονται σε 2 λίστες Unvisited και Walk. Στη πρώτη είναι καταχωρημένοι οι κόμβοι που δεν έχουν εισαχθεί ακόμα στη διαδρομή, ενώ στη δεύτερη οι κόμβοι που έχουν εισαχθεί.

### Βήμα Εισαγωγής

Στο βήμα εισαγωγής γίνεται προσπάθεια εισαγωγής των κόμβων, που βρίσκονται στη λίστα Unvisited, 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 (), εφόσον εισαχθεί μετά τον i, να μην είναι αργότερα από την ώρα παύσης λειτουργίας του κόμβου j. Αρχικά λοιπόν υπολογίζεται το 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. Όπως εξηγήθηκε παραπάνω, μια θέση εισαγωγής A για έναν κόμβο i θεωρείται καλύτερη από μία θέση εισαγωγής B 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. Επίσης πρέπει να συνυπολογιστεί πως ένας κόμβος έχει περισσότερες πιθανότητες να εισαχθεί σε μία διαδρομή με 3,4 κόμβους παρά σε μία διαδρομή με 1 κόμβο καθώς στη πρώτη υπάρχουν περισσότερες θέσεις εισαγωγής. 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.

|  |  |  |
| --- | --- | --- |
|  |  | (3.1) |
|  |  | (3.2) |
|  |  | (3.3) |
|  |  | (3.4) |
|  |  | (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 .

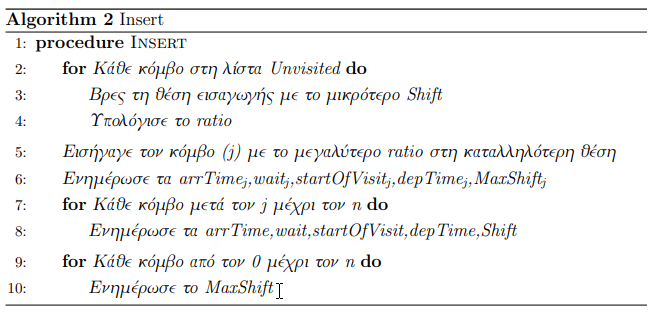
|  |  |  |
| --- | --- | --- |
|  |  | (3.6) |

όπου 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 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 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 Ν 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.

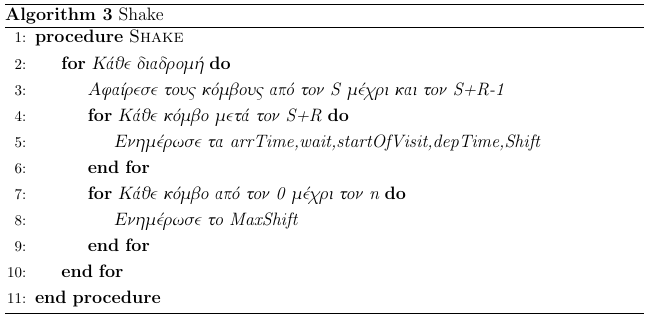


### Βήμα Διαταραχής

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.

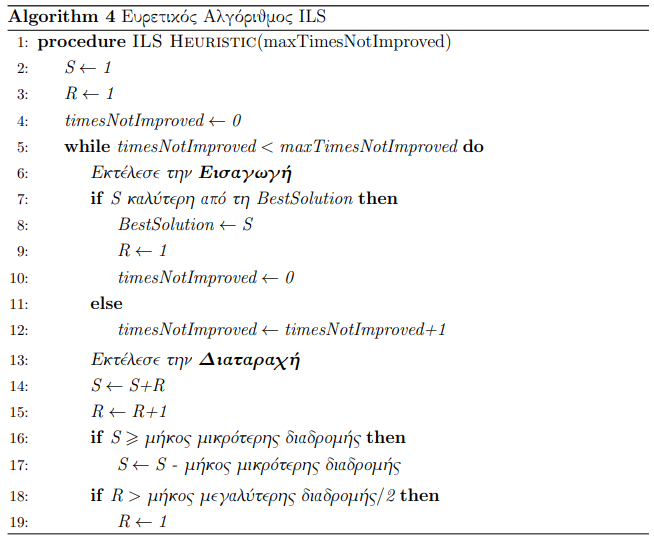
Η διαδικασία αυτή χρειάζεται 2 παραμέτρους, την S και την R. Η πρώτη αντιπροσωπεύει τον κόμβο από τον οποίο θα ξεκινήσει η διαδικασία αφαίρεσης (πρώτο, δεύτερο, τρίτο, κλπ.) και η δεύτερη το πλήθος των συνεχόμενων κόμβων που πρόκειται να αφαιρεθούν ξεκινώντας από τον S. Η αφαίρεση αυτή θα εφαρμοσθεί 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.



### 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.



# 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 παρούσας εργασίας, είναι να βελτιώσει την ταχύτητα του ILS. Η διαδικασία αυτή μπορεί να περιγράφει με 3 βασικά βήματα:

1. Διαχωρισμός των n Unvisited κόμβων σε m υπο -προβλήματα
2. Σειριακή εφαρμογή Τοπικής Αναζήτησης σε κάθε υπο-πρόβλημα
3. Σειριακή εφαρμογή Διαταραχής σε κάθε υπο -πρόβλημα

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 γινόταν με μοναδικό κριτήριο την τοποθεσία των κόμβων με τη βοήθεια κάποιου αλγορίθμου (k-means), τότε υπάρχει η πιθανότητα να προέκυπτε μια συστάδες 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 διαδρομές χαμηλής αξίας.

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

* Ανάθεση τους κεντροειδούς της συστάδας ως σταθμός της συστάδας αυτής
* Ανάθεση ενός κόμβου από την περίμετρο της συστάδας, ως σταθμός της συστάδας αυτής

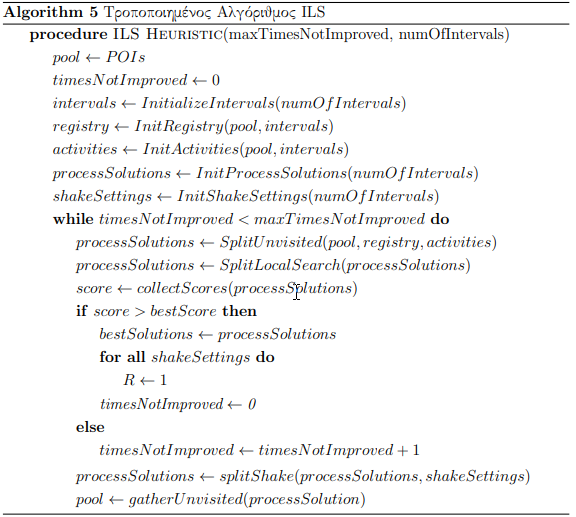
Επίσης, μια ιδέα για την αντιμετώπιση του προβλήματος αυτού, ήταν να θεωρηθεί το κάθε υπό-πρόβλημα ως Πρόβλημα Προσανατολισμού με επιλογή Ξενοδοχείων (Orienteering Problem with Hotel Selection - OPHS), στο οποίο τα depots δεν είναι προκαθορισμένα, καθώς υπάρχει ένα σύνολο ξενοδοχείων από το οποίο επιλέγεται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

Για τη συνέχεια της ενότητας θα χρησιμοποιείται ο όρος Solution για να περιγράφει ένα στιγμιότυπο ενός προβλήματος αλλά και της λύσης του. Ένα Solution περιλαμβάνει τα εξής:

* Μια λίστα Unvisited που περιέχει τους κόμβους που δεν έχουν μπει ακόμη στη διαδρομή
* Μια λίστα Walk που περιέχει τους κόμβους που έχουν ήδη μπει στη διαδρομή
  + Η ώρα αναχώρησης (depTime) του πρώτου κόμβου της λίστας Walk αναπαριστά την ώρα έναρξης της διαδρομής
  + Η ώρα κλεισίματος (closeTime) του τελευταίου κόμβου της λίστας Walk αναπαριστά την μέγιστη ώρα λήξης της διαδρομής.

Βλέποντας, λοιπόν, κάποιος τις πληροφορίες ενός Solution, μπορεί όχι μόνο να δει την λύση ενός προβλήματος αλλά και το ίδιο το πρόβλημα. Η ένωση των λιστών Walk και Unvisited, η ώρα αναχώρησης του πρώτου κόμβου της λίστας Walk και η ώρα κλεισίματος του τελευταίου κόμβου της λίστας Walk αναπαριστούν ένα στιγμιότυπο ενός προβλήματος Προσανατολισμού με χρονικά παράθυρα. Η λίστα Walk από μόνη της μπορεί να θεωρηθεί ως η λύση του προβλήματος αυτού, ενώ το άθροισμα των κερδών των κόμβων της λίστας Walk, μπορεί να δώσει το σκορ της λύσης.

Η διαχωρισμένη Τοπική Αναζήτηση αλλάζει τον ILS ως εξής:



* Η μεταβλητή intervals είναι ένα διάνυσμα που κρατάει τα χρονικά διαστήματα στα οποία χωρίζεται το πρόβλημα. Για παράδειγμα, εάν το numOfIntervals έχει τη τιμή 2, και το χρονικό διάστημα του αρχικού 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. Παρακάτω περιγράφεται η διαδικασία καθορισμού των παραγόμενων χρονικών διαστημάτων.
* Η μεταβλητή activities είναι ένα map που κρατάει την ενεργή διάρκεια του κάθε σημείου ενδιαφέροντος (POI) σε κάθε διάστημα.
* Η μεταβλητή registry είναι ένα map που κρατάει ένα ιστορικό καταλληλότητας για κάθε κόμβο σχετικά με κάθε διάστημα.
* Η μεταβλητή processSolutions είναι ένα διάνυσμα που κρατάει τα Solutions που προκύπτουν κατά τη διάρκεια του αλγορίθμου. Σε περίπτωση που προκύψει ένα καλύτερο σκορ από τη SplitSearch διαδικασία, τα processSolutions αποθηκεύονται σε ένα καινούριο διάνυσμα bestSolutions.
* Η μεταβλητή shakeSettings είναι ένα διάνυσμα που κρατάει τις τιμές S και R για κάθε Solution. Η διαδικασία διαταραχής εφαρμόζεται σε κάθε Solution ξεχωριστά. Για το λόγο αυτό, για κάθε Solution απαιτούνται διαφορετικά S και R . Το S αναπαριστά την θέση από όπου θα ξεκινήσει η αφαίρεση των κόμβων σε κάθε διαδρομή του εκάστοτε Solution, και το R αναπαριστά τον αριθμό των κόμβων που θα αφαιρεθούν. Τα S και R, ρυθμίζονται ανάλογα με το μέγεθος της μικρότερης διαδρομής της εκάστοτε λύσης και είναι ανεξάρτητα για κάθε Solution.

## Αρχικοποίηση των χρονικών υπο-διαστημάτων

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

Παρ’ όλα αυτά, όπως θα αναλυθεί περαιτέρω και στην επόμενη υποενότητα (4.2), η διάρκεια ενεργητικότητας ενός κόμβου σε κάθε χρονικό υποδιάστημα παίζει σημαντικό ρόλο για την ανάθεση του σε ένα από αυτά. Με βάση λοιπόν το προηγούμενο παράδειγμα, υπάρχει ο κίνδυνος οι περισσότεροι ή και όλοι οι Unvisited κόμβοι να ανατεθούν σε ένα συγκεκριμένο χρονικό υποδιάστημα. Προς αποφυγήν αυτού, τα διαστήματα πρέπει να οριοθετηθούν λαμβάνοντας υπόψιν τα χρονικά παράθυρα των κόμβων έτσι ώστε στη συνέχεια, οι κόμβοι να διαμοιραστούν όσο το δυνατό πιο δίκαια στα χρονικά υποδιαστήματα, τουλάχιστον για την πρώτη επανάληψη του ILS.

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

## Διαχωρισμός των Unvisited κόμβων

Η διαδικασία διαχωρισμού των Unvisited κόμβων στα n υπο-προβλήματα/διαστήματα γίνεται με βάση το ιστορικό καταλληλότητας (registry) 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:

|  |  |  |
| --- | --- | --- |
|  |  | (4.1) |

* : 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 άρα και το θα έχουν τιμή 0 σε κάθε επανάληψη. Οπότε, ο κόμβος i δεν θα εξεταστεί ποτέ για εισαγωγή στο διάστημα k.
* : Απεικονίζει το πλήθος των φορών 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 συνεχίσει να αυξάνεται, το θα μειώνεται και θα δοθεί η δυνατότητα να εξεταστεί η εισαγωγή του κόμβου i και στα υπόλοιπα διαστήματα που είναι ενεργός.
* : Απεικονίζει το πλήθος των φορών που ο κόμβος i μπήκε στη λύση του διαστήματος k.

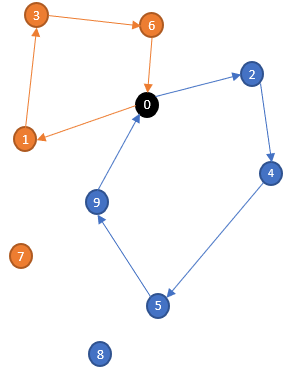
Ο λόγος εκφράζει την καταλληλότητα του κόμβου i ως προς το διάστημα k.

## Διαχωρισμένη Τοπική Αναζήτηση

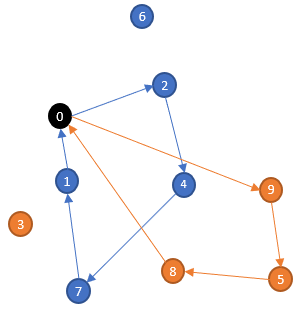
Όπως προαναφέρθηκε και στην αρχή του Κεφαλαίου, ένα σημαντικό θέμα που προκύπτει όταν χωρίζουμε το γράφημα, είναι το ποιος ή το ποιοι κόμβοι θα θεωρηθούν ως startDepot και endDepot του κάθε υπο-προβλήματος.

Η πιο απλή υλοποίηση είναι να χρησιμοποιηθεί το αρχικό depot του προβλήματος ως αρχικό και τελικό depot για κάθε υπο-πρόβλημα. 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 υπο-προβλήματος βρίσκονται μακριά από το depot, η τρέχουσα διαδρομή θα έχει οδηγηθεί χωρίς λόγο πίσω στην αφετηρία, καταλήγοντας έτσι σε χειρότερες λύσεις.

Έστω λοιπόν ένα πρόβλημα OPTW με ένα startDepot sd, ένα endDepot ed και ένα timebudget = [s, e], το οποίο θα χωριστεί σε n υπο-προβλήματα OPTW. Η διαδρομή του πρώτου υπο-προβλήματος () θα ξεκινάει από το sd ενώ η διαδρομή του τελευταίου υπο-προβλήματος () θα τελειώνει στο ed. Παρακάτω περιγράφεται η διαδικασία που συμπληρώνει τον αρχικό κόμβο για κάθε διαδρομή υπο-προβλήματος optw > 0 και τον τελικό κόμβο για κάθε διαδρομή υπο-προβλήματος optw < n − 1. Επειδή τα υπο-προβλήματα επιλύονται σειριακά, πρώτα υπολογίζεται ο τελικός κόμβος του πρώτου υπο-προβλήματος, οπότε θα αναλυθεί πρώτα η προσθήκη τελικών κόμβων.



Εικόνα 4‑1: 1ο Παράδειγμα χρήσης της αφετηρίας του αρχικού προβλήματος ως αρχικό και τελικό σταθμό για κάθε υποπρόβλημα. Σε αυτό το παράδειγμα η επιστροφή στην αφετηρία δεν είναι τόσο χρονοβόρα καθώς οι κόμβοι των 2 υποπροβλημάτων είναι συμμετρικοί ως προς τον αρχικό κόμβο (0).



Εικόνα 4‑2: 2ο παράδειγμα χρήσης της αφετηρίας του αρχικού προβλήματος ως αρχικό και τελικό σταθμό για κάθε υποπρόβλημα. Σε αυτό το παράδειγμα η επιστροφή στον αρχικό κόμβο και η μετάβαση στους κόμβους του δεύτερου υποπροβλήματος καταλαμβάνει μεγάλα χρονικά διαστήματα

### Προσθήκη τελικών κόμβων

Ένας τρόπος εύρεσης τελικού κόμβου για το υπο-πρόβλημα , ήταν να υπολογισθεί το σταθμισμένο κέντρο του επόμενου υπο-προβλήματος . Τα βάρη των κόμβων είναι τα κέρδη τους. Με αυτό τον τρόπο, η κάθε διαδρομή ενός υποπροβλήματος θα οδηγούνταν προς το πιο κερδοφόρο κέντρο βάρους του επόμενου υποπροβλήματος. Η υλοποίηση αυτή όμως έχει αρκετά μειονεκτήματα, σχεδιαστικά και προγραμματιστικά.

* Στην αρχή του προγράμματος, για τα έτοιμα παραδείγματα (π.χ. Cordeau) υπολογίζονται οι ευκλείδειες αποστάσεις μεταξύ όλων των κόμβων του γραφήματος. Δηλαδή, εάν το στιγμιότυπο του προβλήματος έχει 100 κόμβους, τότε θα αρχικοποιηθεί ένας δισδιάστατος πίνακας (διάνυσμα διανυσμάτων) travelTimes μεγέθους 100\*100=10000 θέσεων. Ο πίνακας αυτός χρησιμοποιείται στη φάση Κατασκευής ή αλλιώς Τοπικής Αναζήτησης όπου κατασκευάζονται οι διαδρομές. 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 υποπροβλήματος. Για να γίνει αυτό θα πρέπει να υπολογισθεί η απόστασή του από τα υπόλοιπα 100 σημεία. Παρ’ όλα αυτά στο τρέχον υποπρόβλημα 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 μπορεί να αλλάξουν υποπρόβλημα μετά από κάθε επανάληψη του ILS (Ενότητα 4.2) με βάση τον πίνακα καταλληλότητας (registry). Οπότε σε κάθε επανάληψη, πρέπει να υπολογιστούν καινούρια σταθμισμένα κέντρα καθώς τα παλιά πλέον πιθανότατα να είναι παρωχημένα.
* Μετά από την Τοπική Αναζήτηση, αφότου γεμίσει η κάθε διαδρομή, εφαρμόζεται η Διαταραχή όπου αφαιρεί κόμβους δημιουργώντας 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 κέντρο της επόμενης 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 (μία διαδρομή) με χρονικό παράθυρο , χωρισμένο σε δύο διαστήματα/προβλήματα και με χρονικά παράθυρα και αντίστοιχα. Έστω πως ο τελευταίος κόμβος της διαδρομής του είναι ο z με ώρα αναχώρησης και και έστω cnext το σταθμισμένο κέντρο του με . Ο τελικός κόμβος ed της διαδρομής του θα υπολογιζόταν από τις σχέσεις:

|  |  |  |
| --- | --- | --- |
|  |  | (4.2) |
|  |  | (4.3) |

Από τη διαδικασία αυτή, θα προέκυπτε ο τελικός κόμβος ed με χρόνο άφιξης και αναχώρησης και μηδενική διάρκεια επίσκεψης . Παρ’ όλο, που η διαδρομή πλέον είναι έγκυρη, ουσιαστικά δεν υπάρχει πλέον διαθέσιμος χρόνος για άλλες εισαγωγές, εκτός από τους χρόνους αναμονής στους 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 οπότε θα υπάρξει διαθέσιμος χώρος και για άλλες εισαγωγές στην Τοπική Αναζήτηση.

Για την αντιμετώπιση των παραπάνω προβλημάτων, εν τέλει θεωρήθηκε ένας καινούριος 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 (directed), 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 καθώς είναι δυνατή η εισαγωγή κόμβων 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 . Το minShift παίζει έναν ρόλο στην επιλογή του κόμβου που θα εισαχθεί αλλά λιγότερο σημαντικό από το profit. Το shift της εισαγωγής ενός κόμβου j μεταξύ των κόμβων i και k υπολογίζεται από τη σχέση:

|  |  |  |
| --- | --- | --- |
|  |  | (4.4) |

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:

|  |  |  |
| --- | --- | --- |
|  |  | (4.5) |

Επειδή όμως χρειάζεται να ληφθεί υπόψιν και η επόμενη λύση, προστέθηκε ένας ακόμα παράγοντας που είναι η απόσταση του εξεταζόμενου κόμβου προς το cnext της επόμενης λύσης.

|  |  |  |
| --- | --- | --- |
|  |  | (4.6) |

Ακόμα και αυτή η λύση όμως είναι δεν απολύτως ικανοποιητική, καθώς ουσιαστικά η φόρμουλα που υπολογίζει το shift των εισαγωγών στις ενδιάμεσες θέσεις είναι διαφορετική από αυτή που χρησιμοποιείται στις τελικές. Όμως δεν έχει και νόημα να λαμβάνεται υπόψιν η απόσταση προς το cnext όταν εξετάζεται την εισαγωγή ενός κόμβου στην αρχή της τρέχουσας διαδρομής. Οπότε, η τελική σχέση που προκύπτει είναι η εξής:

|  |  |  |
| --- | --- | --- |
|  |  | (4.7) |

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 , στη δεύτερη κ.ο.κ. , μέχρι την τελική όπου θα είναι .

### 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.

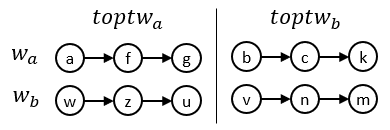
Έστω ο τελευταίος κόμβος μιας διαδρομής με και και έστω πως τη χρονική στιγμή t, ο αλγόριθμος εξετάζει το διάστημα . Αρχικά θα θεωρηθεί ως υποψήφιος αρχικός κόμβος () της διαδρομής ένας κλώνος του κόμβου του διαστήματος . Για να είναι έγκυρη η εισαγωγή του κόμβου στην αρχή του θα πρέπει να ισχύουν οι δύο παρακάτω προϋποθέσεις:

* Δεν παραβιάζονται οι χρόνοι του
* Η ολίσθηση του χρόνου προς τα δεξιά λόγω της εισαγωγής του δεν παραβιάζει τους χρονικούς περιορισμούς των κόμβων που έπονται

Ο κόμβος θεωρείται ουδέτερος κόμβος, καθώς έχει μηδενική διάρκεια επίσκεψης και το χρονικό του παράθυρο είναι ίσο με το χρονικό παράθυρο του διαστήματος . Οπότε είναι πρακτικά αδύνατο να παραβιαστούν οι χρονικοί περιορισμοί του .

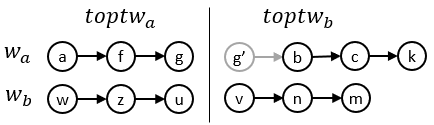
Παρ ’όλα αυτά το διατηρεί τις συντεταγμένες του οπότε ο χρόνος ταξιδιού από τον προς τον τρέχων αρχικό κόμβο της διαδρομής θα προκαλέσει μια ολίσθηση των χρόνων άφιξης, αναχώρησης κ.λπ. των υπόλοιπων κόμβων της διαδρομής προς τα δεξιά. Εάν λοιπόν πράγματι παραβιάζεται κάποιος χρονικός περιορισμούς από τους επακόλουθους κόμβους, τότε αφαιρείται ο αρχικός κόμβος της διαδρομής και μεταφέρεται στη λίστα Unvisited του διαστήματος .

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

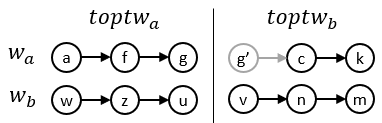


Η προεργασία που περιεγράφηκε παραπάνω, εφαρμόζεται σε κάθε διαδρομή σειριακά. Οπότε αρχικά θα εξεταστεί η διαδρομή του προβλήματος .

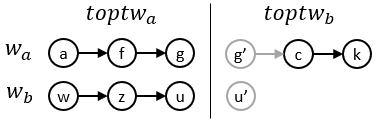
Αρχικά, ως αφετηρία κάθε τροχιάς του προβλήματος , τοποθετείται ένας κλώνος του τελευταίου κόμβου της αντίστοιχης τροχιάς του προηγούμενου υποπροβλήματος, δηλαδή στο συγκεκριμένο παράδειγμα, οι κόμβοι g και u για τις τροχιές και αντίστοιχα. Έστω g’ ο κλώνος του κόμβου g και u’ ο κλώνος του κόμβου u:



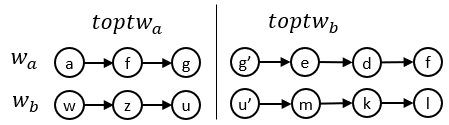
Εφόσον για την ώρα αναχώρησης του g’ έχει οριστεί πως , η εισαγωγή του g’ δεν παραβιάζει το χρονικό του παράθυρο. Παρ’ όλα αυτά πρέπει να εξεταστεί εάν η εισαγωγή του g’ προκαλεί κάποιο πρόβλημα στη συνέχεια της διαδρομής. Εάν όντως προκαλεί, τότε αφαιρείται ο πρώτος αμέσως επόμενος κόμβος της διαδρομής,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



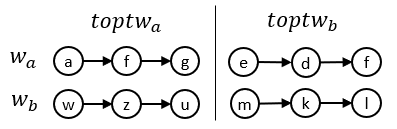
Αυτό θα συνεχιστεί μέχρι να είναι έγκυρη η διαδρομή με την εισαγωγή του g’. Όταν πλέον η εισαγωγή του g’ είναι έγκυρη, η διαδικασία αυτή θα επαναληφθεί για τη διαδρομή . Ακόμα και αν αφαιρεθούν όλοι οι κόμβοι από μία διαδρομή και μείνει μόνο ο ουδέτερος τεχνητός κόμβος του προηγούμενου διαστήματος, ο 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 Β, θα ακολουθήσει η διαδικασία της Τοπικής Αναζήτησης από την οποία προκύπτουν δύο νέες διαδρομές για το .



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

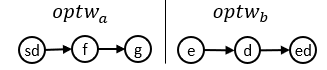


Υπενθυμίζεται πως είναι επιτρεπτές οι αφίξεις στους κόμβους πριν τα χρονικά τους παράθυρα αλλά όχι μετά. Οπότε, η ολίσθηση των χρόνων άφιξης, αναχώρησης κ.λπ. των κόμβων προς τα πίσω δε μπορεί να δημιουργήσει μη έγκυρες διαδρομές.

## Υπερχείλιση Διαδρομών

Η αρχικοποίηση του διανύσματος των επιμέρους λύσεων (processSolutions) γίνεται προσθέτοντας σε κάθε διαδρομή του πρώτου διαστήματος την αφετηρία του πρωτότυπου προβλήματος. Δεν προστίθεται όμως ο τερματικός σταθμός του πρωτότυπου προβλήματος, στις διαδρομές του τελευταίου διαστήματος. Αυτό συμβαίνει διότι, θα δημιουργούταν πρόβλημα στην αφαίρεση των κόμβων κατά την προσθήκη αφετηρίας στο τελευταίο διάστημα, όπως περιεγράφηκε στην υποενότητα 4.3.

Για παράδειγμα, έστω ένα πρόβλημα OPTW χωρισμένο σε δύο διαστήματα:



Από τις προηγούμενες επαναλήψεις έχουν κατασκευαστεί ήδη δύο διαδρομές για τα δυο ξεχωριστά υποπροβλήματα. Το sd είναι η αφετηρία του πρωτότυπου προβλήματος, και ο ed είναι ο τερματισμός του πρωτότυπου προβλήματος.

Όταν ο αλγόριθμος φτάσει στο υποπρόβλημα , θα χρησιμοποιήσει έναν κλώνο του κόμβου g τον g’, ως αφετηρία για τη διαδρομή του . Εάν, η εισαγωγή του g’ δεν είναι δυνατή τότε θα αφαιρεθεί ο πρώτος κόμβος του δηλαδή στη συγκεκριμένη περίπτωση.

# Πειραματικά Αποτελέσματα















Οι υπολογισμοί έγιναν σε ένα t2.medium μηχάνημα της Amazon με επεξεργαστή 3.3 GHz Intel Xeon Scalable και μνήμη RAM 4GB.

Table 2: m=1 Cordeau

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | BK | ILS(2009) | S=1 |  |  | S=2 |  |  | S=3 |  |  | S=4 |  |  |
|  | Score | Score | Score | CPU(s) | Gap (%) | Score | CPU(s) | Gap (%) | Score | CPU(s) | Gap (%) | Score | CPU(s) | Gap (%) |
| pr01 | 308 | 304 | 242 | 0.068 | 20.39 | 281 | 0.067 | 7.57 | 274 | 0.061 | 9.87 | 229 | 0.058 | 24.67 |
| pr02 | 404 | 385 | 375 | 0.171 | 2.6 | 328 | 0.138 | 14.81 | 322 | 0.131 | 16.36 | 295 | 0.166 | 23.38 |
| pr03 | 394 | 384 | 376 | 0.289 | 2.08 | 366 | 0.401 | 4.69 | 350 | 0.222 | 8.85 | 309 | 0.241 | 19.53 |
| pr04 | 489 | 447 | 478 | 0.951 | -6.94 | 447 | 0.373 | 0 | 411 | 0.332 | 8.05 | 433 | 0.339 | 3.13 |
| pr05 | 595 | 576 | 524 | 0.934 | 9.03 | 511 | 0.83 | 11.28 | 486 | 0.848 | 15.63 | 504 | 0.916 | 12.5 |
| pr06 | 590 | 538 | 574 | 0.952 | -6.69 | 543 | 0.77 | -0.93 | 518 | 0.616 | 3.72 | 460 | 0.519 | 14.5 |
| pr07 | 298 | 291 | 261 | 0.111 | 10.31 | 251 | 0.114 | 13.75 | 229 | 0.093 | 21.31 | 251 | 0.098 | 13.75 |
| pr08 | 463 | 463 | 447 | 0.437 | 3.46 | 389 | 0.288 | 15.98 | 417 | 0.266 | 9.94 | 355 | 0.217 | 23.33 |
| pr09 | 493 | 461 | 424 | 0.646 | 8.03 | 416 | 0.578 | 9.76 | 333 | 0.494 | 27.77 | 322 | 0.326 | 30.15 |
| pr10 | 594 | 539 | 520 | 0.943 | 3.53 | 519 | 1.088 | 3.71 | 472 | 0.506 | 12.43 | 440 | 0.483 | 18.37 |
| pr11 | 353 | 330 | 319 | 0.094 | 3.33 | 308 | 0.093 | 6.67 | 274 | 0.065 | 16.97 | 285 | 0.07 | 13.64 |
| pr12 | 442 | 431 | 424 | 0.285 | 1.62 | 418 | 0.191 | 3.02 | 407 | 0.41 | 5.57 | 403 | 0.346 | 6.5 |
| pr13 | 467 | 450 | 444 | 0.383 | 1.33 | 377 | 0.295 | 16.22 | 376 | 0.243 | 16.44 | 391 | 0.305 | 13.11 |
| pr14 | 567 | 482 | 510 | 0.798 | -5.81 | 480 | 0.665 | 0.41 | 434 | 0.328 | 9.96 | 450 | 0.355 | 6.64 |
| pr15 | 708 | 638 | 661 | 1.19 | -3.61 | 597 | 0.833 | 6.43 | 580 | 0.526 | 9.09 | 553 | 0.42 | 13.32 |
| pr16 | 674 | 559 | 596 | 2.636 | -6.62 | 553 | 1.751 | 1.07 | 533 | 0.663 | 4.65 | 509 | 0.68 | 8.94 |
| pr17 | 362 | 346 | 341 | 0.139 | 1.45 | 320 | 0.14 | 7.51 | 285 | 0.102 | 17.63 | 258 | 0.101 | 25.43 |
| pr18 | 539 | 479 | 447 | 0.374 | 6.68 | 507 | 0.28 | -5.85 | 435 | 0.239 | 9.19 | 390 | 0.235 | 18.58 |
| pr19 | 562 | 499 | 468 | 0.957 | 6.21 | 427 | 0.502 | 14.43 | 428 | 0.717 | 14.23 | 384 | 0.42 | 23.05 |
| pr20 | 667 | 570 | 610 | 1.756 | -7.02 | 586 | 1.661 | -2.81 | 534 | 0.535 | 6.32 | 556 | 0.9 | 2.46 |

Table 3: m=2 Cordeau

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | BK | ILS(2009) | S=1 |  |  | S=2 |  |  | S=3 |  |  | S=4 |  |  |
|  | Score | Score | Score | CPU(s) | Gap (%) | Score | CPU(s) | Gap (%) | Score | CPU(s) | Gap (%) | Score | CPU(s) | Gap (%) |
| pr01 | 502 | 471 | 451 | 0.142 |  | 441 | 0.067 |  | 441 | 0.105 |  | 427 | 0.108 |  |
| pr02 | 715 | 660 | 670 | 0.477 |  | 620 | 0.285 |  | 597 | 0.274 |  | 576 | 0.208 |  |
| pr03 | 742 | 714 | 673 | 0.694 |  | 639 | 0.415 |  | 663 | 0.449 |  | 600 | 0.267 |  |
| pr04 | 926 | 863 | 799 | 0.909 |  | 790 | 0.717 |  | 807 | 0.472 |  | 745 | 0.432 |  |
| pr05 | 1101 | 1011 | 1018 | 4.763 |  | 870 | 1.297 |  | 852 | 1.277 |  | 770 | 0.595 |  |
| pr06 | 1076 | 997 | 1009 | 2.683 |  | 987 | 1.157 |  | 933 | 0.973 |  | 943 | 0.852 |  |
| pr07 | 566 | 552 | 541 | 0.207 |  | 498 | 0.133 |  | 444 | 0.107 |  | 517 | 0.15 |  |
| pr08 | 834 | 796 | 776 | 0.683 |  | 727 | 0.539 |  | 705 | 0.305 |  | 647 | 0.283 |  |
| pr09 | 909 | 867 | 843 | 3.756 |  | 738 | 1.3 |  | 716 | 0.667 |  | 726 | 0.498 |  |
| pr10 | 1134 | 1004 | 1016 | 2.308 |  | 961 | 1.679 |  | 955 | 0.991 |  | 908 | 0.941 |  |
| pr11 | 566 | 542 | 525 | 0.087 |  | 502 | 0.065 |  | 456 | 0.067 |  | 473 | 0.065 |  |
| pr12 | 774 | 727 | 700 | 0.719 |  | 690 | 0.344 |  | 665 | 0.191 |  | 655 | 0.217 |  |
| pr13 | 843 | 757 | 771 | 1.935 |  | 737 | 0.452 |  | 693 | 0.308 |  | 681 | 0.423 |  |
| pr14 | 1017 | 925 | 964 | 2.106 |  | 908 | 1.227 |  | 862 | 0.56 |  | 725 | 0.461 |  |
| pr15 | 1220 | 1126 | 1086 | 2.322 |  | 1043 | 1.368 |  | 1028 | 0.872 |  | 959 | 0.873 |  |
| pr16 | 1231 | 1110 | 1101 | 4.775 |  | 1030 | 1.632 |  | 984 | 0.989 |  | 954 | 1.437 |  |
| pr17 | 652 | 624 | 587 | 0.228 |  | 567 | 0.136 |  | 518 | 0.135 |  | 503 | 0.127 |  |
| pr18 | 953 | 877 | 825 | 0.88 |  | 878 | 0.557 |  | 807 | 0.39 |  | 736 | 0.305 |  |
| pr19 | 1034 | 955 | 969 | 2.379 |  | 818 | 1.508 |  | 772 | 0.677 |  | 739 | 0.478 |  |
| pr20 | 1241 | 1056 | 1109 | 4.784 |  | 1084 | 1.529 |  | 996 | 0.976 |  | 999 | 0.987 |  |

Table 4: m=3 Cordeau

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | BK | ILS(2009) | S=1 |  |  | S=2 |  |  | S=3 |  |  | S=4 |  |  |
|  | Score | Score | Score | CPU(s) | Gap (%) | Score | CPU(s) | Gap (%) | Score | CPU(s) | Gap (%) | Score | CPU(s) | Gap (%) |
| pr01 | 622 | 598 | 598 | 0.148 |  | 540 | 0.109 |  | 553 | 0.071 |  | 518 | 0.083 |  |
| pr02 | 943 | 899 | 858 | 1.276 |  | 834 | 0.286 |  | 808 | 0.244 |  | 764 | 0.275 |  |
| pr03 | 1010 | 946 | 959 | 1.847 |  | 884 | 0.696 |  | 740 | 0.38 |  | 800 | 0.386 |  |
| pr04 | 1294 | 1195 | 1178 | 2.144 |  | 1163 | 2.06 |  | 1128 | 0.737 |  | 1148 | 0.972 |  |
| pr05 | 1482 | 1356 | 1314 | 2.283 |  | 1295 | 1.186 |  | 1284 | 2.039 |  | 1247 | 1.241 |  |
| pr06 | 1514 | 1376 | 1401 | 4.127 |  | 1340 | 2.91 |  | 1344 | 1.466 |  | 1328 | 1.29 |  |
| pr07 | 744 | 713 | 689 | 0.289 |  | 653 | 0.179 |  | 643 | 0.14 |  | 629 | 0.256 |  |
| pr08 | 1139 | 1082 | 1047 | 0.722 |  | 1027 | 0.684 |  | 970 | 0.444 |  | 937 | 0.467 |  |
| pr09 | 1282 | 1144 | 1138 | 1.921 |  | 1162 | 1.864 |  | 1025 | 1.009 |  | 1052 | 0.996 |  |
| pr10 | 1573 | 1473 | 1495 | 7.248 |  | 1322 | 1.843 |  | 1381 | 1.651 |  | 1284 | 1.524 |  |
| pr11 | 654 | 632 | 630 | 0.098 |  | 617 | 0.075 |  | 580 | 0.065 |  | 563 | 0.09 |  |
| pr12 | 1002 | 902 | 923 | 0.668 |  | 883 | 0.266 |  | 835 | 0.274 |  | 797 | 0.254 |  |
| pr13 | 1152 | 1046 | 1063 | 1.011 |  | 1021 | 0.832 |  | 914 | 0.472 |  | 942 | 0.551 |  |
| pr14 | 1372 | 1197 | 1247 | 1.542 |  | 1190 | 0.853 |  | 1095 | 0.624 |  | 1131 | 1.112 |  |
| pr15 | 1659 | 1488 | 1534 | 3.358 |  | 1449 | 1.426 |  | 1425 | 1.124 |  | 1379 | 1.541 |  |
| pr16 | 1668 | 1478 | 1508 | 7.029 |  | 1468 | 5.069 |  | 1466 | 3.161 |  | 1333 | 1.641 |  |
| pr17 | 841 | 808 | 792 | 0.289 |  | 787 | 0.2 |  | 699 | 0.204 |  | 668 | 0.258 |  |
| pr18 | 1282 | 1165 | 1181 | 1.943 |  | 1117 | 0.742 |  | 1003 | 0.516 |  | 970 | 0.503 |  |
| pr19 | 1417 | 1238 | 1292 | 3.628 |  | 1254 | 1.717 |  | 1191 | 1.281 |  | 1137 | 0.899 |  |
| pr20 | 1690 | 1514 | 1534 | 3.815 |  | 1509 | 2.263 |  | 1454 | 2.684 |  | 1376 | 1.184 |  |

Table 5: m=4 Cordeau

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | BK | ILS(2009) | S=1 |  |  | S=2 |  |  | S=3 |  |  | S=4 |  |  |
|  | Score | Score | Score | CPU(s) | Gap (%) | Score | CPU(s) | Gap (%) | Score | CPU(s) | Gap (%) | Score | CPU(s) | Gap (%) |
| pr01 | 657 | 644 | 649 | 0.16 |  | 605 | 0.063 |  | 610 | 0.118 |  | 583 | 0.092 |  |
| pr02 | 1079 | 1014 | 1001 | 0.854 |  | 969 | 0.261 |  | 947 | 0.221 |  | 947 | 0.501 |  |
| pr03 | 1246 | 1162 | 1118 | 1.336 |  | 1062 | 0.766 |  | 1028 | 0.503 |  | 1030 | 0.735 |  |
| pr04 | 1585 | 1452 | 1483 | 6.365 |  | 1435 | 1.053 |  | 1366 | 0.982 |  | 1374 | 0.682 |  |
| pr05 | 1844 | 1665 | 1640 | 3.046 |  | 1627 | 1.616 |  | 1532 | 1.046 |  | 1545 | 0.872 |  |
| pr06 | 1886 | 1696 | 1695 | 9.975 |  | 1694 | 2.419 |  | 1683 | 1.732 |  | 1588 | 1.295 |  |
| pr07 | 876 | 840 | 821 | 0.331 |  | 786 | 0.189 |  | 760 | 0.145 |  | 712 | 0.213 |  |
| pr08 | 1385 | 1267 | 1286 | 1.236 |  | 1205 | 0.521 |  | 1154 | 0.448 |  | 1126 | 0.411 |  |
| pr09 | 1619 | 1460 | 1417 | 2.312 |  | 1394 | 1.105 |  | 1393 | 1.373 |  | 1408 | 1.024 |  |
| pr10 | 1943 | 1782 | 1784 | 6.455 |  | 1729 | 2.131 |  | 1690 | 1.415 |  | 1623 | 2.392 |  |
| pr11 | 657 | 654 | 654 | 0.07 |  | 654 | 0.08 |  | 640 | 0.075 |  | 618 | 0.126 |  |
| pr12 | 1132 | 1041 | 1067 | 0.514 |  | 1025 | 0.517 |  | 989 | 0.302 |  | 940 | 0.242 |  |
| pr13 | 1386 | 1263 | 1269 | 1.372 |  | 1238 | 0.822 |  | 1176 | 0.507 |  | 1117 | 0.347 |  |
| pr14 | 1674 | 1528 | 1529 | 1.907 |  | 1501 | 2.072 |  | 1399 | 0.875 |  | 1427 | 1.055 |  |
| pr15 | 2065 | 1818 | 1824 | 5.379 |  | 1815 | 4.531 |  | 1746 | 1.268 |  | 1606 | 0.998 |  |
| pr16 | 2065 | 1889 | 1861 | 6.693 |  | 1829 | 3.696 |  | 1718 | 2.363 |  | 1645 | 2.26 |  |
| pr17 | 934 | 889 | 894 | 0.589 |  | 868 | 0.143 |  | 799 | 0.253 |  | 800 | 0.257 |  |
| pr18 | 1539 | 1352 | 1425 | 1.635 |  | 1372 | 0.945 |  | 1266 | 0.839 |  | 1245 | 1.067 |  |
| pr19 | 1760 | 1560 | 1613 | 4.137 |  | 1534 | 3.08 |  | 1490 | 1.397 |  | 1449 | 0.803 |  |
| pr20 | 2062 | 1846 | 1979 | 7.107 |  | 1873 | 2.999 |  | 1792 | 1.503 |  | 1720 | 1.722 |  |

Table 6: m=1 Solomon

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | BK | ILS(2009) | S=1 |  |  | S=2 |  |  | S=3 |  |  | S=4 |  |  |
|  | Score | Score | Score | CPU(s) | Gap (%) | Score | CPU(s) | Gap (%) | Score | CPU(s) | Gap (%) | Score | CPU(s) | Gap (%) |
| c101 | 320 | 320 | 0 | 0.119 |  | 0 | 0.102 |  | 0 | 0.114 |  | 0 | 0.125 |  |
| c102 | 360 | 360 | 230 | 0.145 |  | 220 | 0.128 |  | 210 | 0.137 |  | 200 | 0.148 |  |
| c103 | 400 | 390 | 350 | 0.171 |  | 310 | 0.135 |  | 300 | 0.144 |  | 260 | 0.155 |  |
| c104 | 420 | 400 | 370 | 0.15 |  | 310 | 0.131 |  | 340 | 0.157 |  | 310 | 0.154 |  |
| c105 | 340 | 340 | 260 | 0.135 |  | 280 | 0.134 |  | 260 | 0.141 |  | 250 | 0.145 |  |
| c106 | 340 | 340 | 290 | 0.168 |  | 280 | 0.129 |  | 270 | 0.141 |  | 240 | 0.145 |  |
| c107 | 370 | 360 | 310 | 0.137 |  | 290 | 0.125 |  | 280 | 0.134 |  | 280 | 0.167 |  |
| c108 | 370 | 370 | 330 | 0.154 |  | 310 | 0.134 |  | 290 | 0.134 |  | 290 | 0.15 |  |
| c109 | 380 | 380 | 350 | 0.146 |  | 340 | 0.133 |  | 300 | 0.146 |  | 290 | 0.152 |  |
| c201 | 870 | 840 | 770 | 0.198 |  | 770 | 0.144 |  | 770 | 0.156 |  | 780 | 0.149 |  |
| c202 | 930 | 910 | 830 | 0.208 |  | 830 | 0.169 |  | 840 | 0.168 |  | 810 | 0.19 |  |
| c203 | 960 | 940 | 890 | 0.212 |  | 880 | 0.191 |  | 900 | 0.179 |  | 910 | 0.162 |  |
| c204 | 980 | 950 | 920 | 0.212 |  | 910 | 0.163 |  | 880 | 0.179 |  | 870 | 0.164 |  |
| c205 | 910 | 900 | 820 | 0.202 |  | 830 | 0.15 |  | 840 | 0.159 |  | 850 | 0.147 |  |
| c206 | 930 | 910 | 880 | 0.229 |  | 880 | 0.155 |  | 860 | 0.158 |  | 870 | 0.23 |  |
| c207 | 930 | 910 | 860 | 0.226 |  | 870 | 0.174 |  | 850 | 0.139 |  | 850 | 0.149 |  |
| c208 | 950 | 930 | 910 | 0.212 |  | 900 | 0.151 |  | 870 | 0.147 |  | 880 | 0.184 |  |
| r101 | 198 | 182 | 143 | 0.116 |  | 103 | 0.115 |  | 126 | 0.125 |  | 112 | 0.137 |  |
| r102 | 286 | 286 | 213 | 0.134 |  | 239 | 0.125 |  | 221 | 0.134 |  | 230 | 0.164 |  |
| r103 | 293 | 286 | 248 | 0.153 |  | 262 | 0.14 |  | 231 | 0.138 |  | 241 | 0.148 |  |
| r104 | 303 | 297 | 254 | 0.137 |  | 282 | 0.178 |  | 252 | 0.188 |  | 214 | 0.155 |  |
| r105 | 247 | 247 | 195 | 0.134 |  | 195 | 0.134 |  | 175 | 0.128 |  | 163 | 0.138 |  |
| r106 | 293 | 293 | 265 | 0.153 |  | 251 | 0.14 |  | 254 | 0.177 |  | 257 | 0.168 |  |
| r107 | 299 | 288 | 275 | 0.143 |  | 265 | 0.149 |  | 235 | 0.139 |  | 214 | 0.145 |  |
| r108 | 308 | 297 | 254 | 0.133 |  | 281 | 0.143 |  | 249 | 0.237 |  | 254 | 0.243 |  |
| r109 | 277 | 276 | 239 | 0.136 |  | 251 | 0.221 |  | 246 | 0.316 |  | 216 | 0.136 |  |
| r110 | 284 | 281 | 255 | 0.154 |  | 256 | 0.143 |  | 225 | 0.136 |  | 223 | 0.135 |  |
| r111 | 297 | 295 | 259 | 0.146 |  | 260 | 0.138 |  | 235 | 0.131 |  | 200 | 0.265 |  |
| r112 | 298 | 295 | 274 | 0.159 |  | 269 | 0.128 |  | 244 | 0.148 |  | 243 | 0.138 |  |
| r201 | 797 | 788 | 765 | 0.268 |  | 736 | 0.154 |  | 757 | 0.173 |  | 715 | 0.173 |  |
| r202 | 930 | 880 | 855 | 0.424 |  | 877 | 0.372 |  | 792 | 0.163 |  | 844 | 0.157 |  |
| r203 | 1028 | 980 | 986 | 0.589 |  | 926 | 0.228 |  | 919 | 0.415 |  | 925 | 0.233 |  |
| r204 | 1093 | 1073 | 1055 | 0.431 |  | 980 | 0.202 |  | 983 | 0.204 |  | 981 | 0.18 |  |
| r205 | 953 | 931 | 885 | 0.247 |  | 906 | 0.174 |  | 856 | 0.217 |  | 862 | 0.179 |  |
| r206 | 1032 | 996 | 961 | 0.376 |  | 964 | 0.295 |  | 958 | 0.554 |  | 917 | 0.158 |  |
| r207 | 1077 | 1038 | 1032 | 0.571 |  | 998 | 0.179 |  | 995 | 0.358 |  | 944 | 0.273 |  |
| r208 | 1118 | 1069 | 1080 | 0.5 |  | 1018 | 0.25 |  | 1027 | 0.189 |  | 1036 | 0.301 |  |
| r209 | 961 | 926 | 907 | 0.399 |  | 910 | 0.25 |  | 870 | 0.188 |  | 878 | 0.159 |  |
| r210 | 1000 | 958 | 913 | 0.283 |  | 931 | 0.268 |  | 897 | 0.217 |  | 897 | 0.168 |  |
| r211 | 1051 | 1023 | 1001 | 0.314 |  | 962 | 0.222 |  | 960 | 0.231 |  | 985 | 0.159 |  |
| rc101 | 219 | 219 | 193 | 0.129 |  | 176 | 0.142 |  | 173 | 0.136 |  | 163 | 0.134 |  |
| rc102 | 266 | 259 | 236 | 0.132 |  | 205 | 0.136 |  | 196 | 0.143 |  | 213 | 0.138 |  |
| rc103 | 266 | 265 | 226 | 0.164 |  | 221 | 0.147 |  | 217 | 0.14 |  | 165 | 0.142 |  |
| rc104 | 301 | 297 | 241 | 0.13 |  | 234 | 0.136 |  | 211 | 0.139 |  | 217 | 0.151 |  |
| rc105 | 244 | 221 | 203 | 0.124 |  | 165 | 0.137 |  | 202 | 0.135 |  | 187 | 0.147 |  |
| rc106 | 252 | 239 | 225 | 0.138 |  | 197 | 0.136 |  | 210 | 0.16 |  | 184 | 0.133 |  |
| rc107 | 277 | 274 | 257 | 0.138 |  | 240 | 0.135 |  | 216 | 0.129 |  | 197 | 0.14 |  |
| rc108 | 298 | 288 | 278 | 0.141 |  | 256 | 0.15 |  | 223 | 0.246 |  | 211 | 0.147 |  |
| rc201 | 795 | 780 | 771 | 0.236 |  | 762 | 0.279 |  | 756 | 0.152 |  | 665 | 0.142 |  |
| rc202 | 938 | 882 | 856 | 0.379 |  | 857 | 0.23 |  | 792 | 0.214 |  | 872 | 0.249 |  |
| rc203 | 1003 | 960 | 946 | 0.367 |  | 915 | 0.296 |  | 903 | 0.189 |  | 892 | 0.259 |  |
| rc204 | 1140 | 1117 | 1099 | 0.27 |  | 1009 | 0.251 |  | 1023 | 0.304 |  | 982 | 0.188 |  |
| rc205 | 859 | 840 | 819 | 0.361 |  | 813 | 0.196 |  | 796 | 0.187 |  | 716 | 0.159 |  |
| rc206 | 899 | 860 | 841 | 0.271 |  | 832 | 0.178 |  | 828 | 0.179 |  | 813 | 0.178 |  |
| rc207 | 983 | 926 | 903 | 0.262 |  | 889 | 0.18 |  | 900 | 0.372 |  | 839 | 0.237 |  |
| rc208 | 1057 | 1037 | 973 | 0.331 |  | 941 | 0.179 |  | 986 | 0.202 |  | 911 | 0.168 |  |

Table 7: m=2 Solomon

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | BK | ILS(2009) | S=1 |  |  | S=2 |  |  | S=3 |  |  | S=4 |  |  |
|  | Score | Score | Score | CPU(s) | Gap (%) | Score | CPU(s) | Gap (%) | Score | CPU(s) | Gap (%) | Score | CPU(s) | Gap (%) |
| c101 | 590 | 590 | 0 | 0.106 |  | 0 | 0.143 |  | 0 | 0.117 |  | 0 | 0.125 |  |
| c102 | 660 | 650 | 360 | 0.246 |  | 360 | 0.185 |  | 330 | 0.161 |  | 320 | 0.174 |  |
| c103 | 720 | 700 | 580 | 0.489 |  | 530 | 0.176 |  | 520 | 0.179 |  | 460 | 0.19 |  |
| c104 | 760 | 750 | 640 | 0.348 |  | 610 | 0.203 |  | 580 | 0.188 |  | 550 | 0.206 |  |
| c105 | 640 | 640 | 470 | 0.207 |  | 490 | 0.188 |  | 450 | 0.177 |  | 420 | 0.188 |  |
| c106 | 620 | 620 | 500 | 0.226 |  | 480 | 0.169 |  | 470 | 0.175 |  | 400 | 0.176 |  |
| c107 | 670 | 670 | 560 | 0.227 |  | 550 | 0.182 |  | 510 | 0.169 |  | 490 | 0.174 |  |
| c108 | 680 | 670 | 590 | 0.248 |  | 580 | 0.182 |  | 540 | 0.172 |  | 500 | 0.185 |  |
| c109 | 720 | 710 | 640 | 0.261 |  | 610 | 0.173 |  | 560 | 0.193 |  | 500 | 0.179 |  |
| c201 | 1460 | 1400 | 1290 | 0.335 |  | 1280 | 0.201 |  | 1310 | 0.198 |  | 1330 | 0.172 |  |
| c202 | 1470 | 1430 | 1370 | 0.426 |  | 1350 | 0.358 |  | 1360 | 0.303 |  | 1300 | 0.268 |  |
| c203 | 1480 | 1430 | 1410 | 0.44 |  | 1360 | 0.276 |  | 1400 | 0.26 |  | 1360 | 0.203 |  |
| c204 | 1490 | 1460 | 1420 | 0.647 |  | 1410 | 0.353 |  | 1370 | 0.346 |  | 1370 | 0.179 |  |
| c205 | 1470 | 1450 | 1410 | 0.373 |  | 1430 | 0.308 |  | 1400 | 0.185 |  | 1380 | 0.202 |  |
| c206 | 1480 | 1440 | 1440 | 0.428 |  | 1430 | 0.346 |  | 1410 | 0.184 |  | 1390 | 0.172 |  |
| c207 | 1490 | 1450 | 1430 | 0.651 |  | 1440 | 0.35 |  | 1420 | 0.2 |  | 1390 | 0.176 |  |
| c208 | 1490 | 1460 | 1460 | 0.652 |  | 1450 | 0.294 |  | 1430 | 0.19 |  | 1420 | 0.199 |  |
| r101 | 349 | 330 | 275 | 0.163 |  | 217 | 0.162 |  | 257 | 0.156 |  | 186 | 0.161 |  |
| r102 | 508 | 508 | 461 | 0.279 |  | 408 | 0.167 |  | 411 | 0.178 |  | 355 | 0.199 |  |
| r103 | 522 | 513 | 468 | 0.283 |  | 439 | 0.189 |  | 400 | 0.19 |  | 414 | 0.226 |  |
| r104 | 552 | 539 | 506 | 0.263 |  | 470 | 0.198 |  | 387 | 0.218 |  | 414 | 0.198 |  |
| r105 | 453 | 430 | 351 | 0.201 |  | 333 | 0.167 |  | 329 | 0.167 |  | 320 | 0.202 |  |
| r106 | 529 | 529 | 438 | 0.232 |  | 444 | 0.199 |  | 416 | 0.2 |  | 382 | 0.197 |  |
| r107 | 538 | 529 | 474 | 0.268 |  | 461 | 0.179 |  | 427 | 0.206 |  | 424 | 0.206 |  |
| r108 | 560 | 549 | 513 | 0.257 |  | 485 | 0.216 |  | 429 | 0.182 |  | 423 | 0.37 |  |
| r109 | 506 | 498 | 453 | 0.265 |  | 407 | 0.191 |  | 414 | 0.181 |  | 395 | 0.24 |  |
| r110 | 525 | 515 | 456 | 0.26 |  | 432 | 0.172 |  | 409 | 0.187 |  | 407 | 0.177 |  |
| r111 | 544 | 535 | 490 | 0.276 |  | 479 | 0.205 |  | 460 | 0.246 |  | 383 | 0.192 |  |
| r112 | 544 | 515 | 491 | 0.281 |  | 469 | 0.175 |  | 399 | 0.181 |  | 424 | 0.214 |  |
| r201 | 1256 | 1231 | 1192 | 0.688 |  | 1148 | 0.233 |  | 1170 | 0.387 |  | 1132 | 0.237 |  |
| r202 | 1348 | 1270 | 1300 | 0.536 |  | 1308 | 0.609 |  | 1219 | 0.18 |  | 1252 | 0.462 |  |
| r203 | 1418 | 1377 | 1345 | 0.468 |  | 1355 | 0.219 |  | 1300 | 0.182 |  | 1307 | 0.191 |  |
| r204 | 1458 | 1440 | 1431 | 0.427 |  | 1424 | 0.193 |  | 1396 | 0.164 |  | 1410 | 0.305 |  |
| r205 | 1386 | 1338 | 1324 | 0.471 |  | 1339 | 0.291 |  | 1315 | 0.312 |  | 1271 | 0.217 |  |
| r206 | 1450 | 1401 | 1380 | 0.337 |  | 1378 | 0.268 |  | 1349 | 0.238 |  | 1399 | 0.308 |  |
| r207 | 1458 | 1428 | 1417 | 0.322 |  | 1417 | 0.478 |  | 1370 | 0.208 |  | 1407 | 0.658 |  |
| r208 | 1458 | 1458 | 1456 | 0.238 |  | 1451 | 0.143 |  | 1429 | 0.225 |  | 1436 | 0.364 |  |
| r209 | 1414 | 1345 | 1357 | 0.346 |  | 1331 | 0.328 |  | 1331 | 0.213 |  | 1334 | 0.345 |  |
| r210 | 1427 | 1365 | 1358 | 0.496 |  | 1367 | 0.275 |  | 1326 | 0.268 |  | 1328 | 0.263 |  |
| r211 | 1458 | 1422 | 1435 | 0.438 |  | 1431 | 0.327 |  | 1386 | 0.191 |  | 1374 | 0.305 |  |
| rc101 | 427 | 427 | 378 | 0.201 |  | 356 | 0.178 |  | 311 | 0.172 |  | 294 | 0.223 |  |
| rc102 | 505 | 494 | 457 | 0.229 |  | 403 | 0.2 |  | 392 | 0.197 |  | 395 | 0.21 |  |
| rc103 | 524 | 519 | 464 | 0.328 |  | 431 | 0.205 |  | 426 | 0.208 |  | 376 | 0.202 |  |
| rc104 | 575 | 565 | 520 | 0.25 |  | 437 | 0.194 |  | 446 | 0.174 |  | 360 | 0.295 |  |
| rc105 | 480 | 459 | 384 | 0.199 |  | 328 | 0.183 |  | 325 | 0.188 |  | 352 | 0.25 |  |
| rc106 | 483 | 458 | 422 | 0.223 |  | 407 | 0.173 |  | 383 | 0.168 |  | 324 | 0.214 |  |
| rc107 | 534 | 515 | 484 | 0.263 |  | 476 | 0.193 |  | 430 | 0.162 |  | 353 | 0.266 |  |
| rc108 | 556 | 546 | 517 | 0.247 |  | 455 | 0.237 |  | 377 | 0.187 |  | 403 | 0.199 |  |
| rc201 | 1385 | 1305 | 1294 | 0.564 |  | 1249 | 0.262 |  | 1265 | 0.196 |  | 1245 | 0.165 |  |
| rc202 | 1512 | 1461 | 1436 | 1.056 |  | 1383 | 0.231 |  | 1372 | 0.173 |  | 1349 | 0.331 |  |
| rc203 | 1632 | 1573 | 1502 | 0.423 |  | 1471 | 0.367 |  | 1454 | 0.554 |  | 1439 | 0.346 |  |
| rc204 | 1716 | 1656 | 1650 | 0.49 |  | 1621 | 0.441 |  | 1556 | 0.211 |  | 1564 | 0.468 |  |
| rc205 | 1458 | 1381 | 1362 | 0.554 |  | 1373 | 0.33 |  | 1348 | 0.249 |  | 1283 | 0.272 |  |
| rc206 | 1552 | 1495 | 1451 | 0.769 |  | 1427 | 0.244 |  | 1421 | 0.332 |  | 1343 | 0.198 |  |
| rc207 | 1599 | 1531 | 1547 | 0.612 |  | 1487 | 0.284 |  | 1495 | 0.398 |  | 1391 | 0.412 |  |
| rc208 | 1692 | 1606s | 1633 | 0.822 |  | 1615 | 0.236 |  | 1574 | 0.618 |  | 1507 | 0.277 |  |

Table 8: m=3 Solomon

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | BK | ILS(2009) | S=1 |  |  | S=2 |  |  | S=3 |  |  | S=4 |  |  |
|  | Score | Score | Score | CPU(s) | Gap (%) | Score | CPU(s) | Gap (%) | Score | CPU(s) | Gap (%) | Score | CPU(s) | Gap (%) |
| c101 | 810 | 790 | 0 | 0.158 |  | 0 | 0.126 |  | 0 | 0.142 |  | 0 | 0.154 |  |
| c102 | 920 | 890 | 400 | 0.416 |  | 400 | 0.207 |  | 400 | 0.225 |  | 380 | 0.264 |  |
| c103 | 990 | 960 | 720 | 0.57 |  | 700 | 0.247 |  | 670 | 0.29 |  | 610 | 0.285 |  |
| c104 | 1030 | 1010 | 870 | 0.525 |  | 810 | 0.247 |  | 760 | 0.212 |  | 730 | 0.356 |  |
| c105 | 870 | 840 | 680 | 0.31 |  | 670 | 0.277 |  | 630 | 0.232 |  | 560 | 0.293 |  |
| c106 | 870 | 840 | 710 | 0.335 |  | 650 | 0.238 |  | 640 | 0.229 |  | 540 | 0.253 |  |
| c107 | 910 | 900 | 780 | 0.327 |  | 790 | 0.262 |  | 710 | 0.256 |  | 690 | 0.245 |  |
| c108 | 920 | 900 | 800 | 0.498 |  | 800 | 0.249 |  | 760 | 0.235 |  | 720 | 0.257 |  |
| c109 | 970 | 950 | 880 | 0.475 |  | 820 | 0.237 |  | 760 | 0.237 |  | 670 | 0.222 |  |
| c201 | 1810 | 1750 | 1750 | 0.333 |  | 1670 | 0.238 |  | 1610 | 0.23 |  | 1720 | 0.211 |  |
| c202 | 1810 | 1750 | 1690 | 0.464 |  | 1670 | 0.434 |  | 1680 | 0.316 |  | 1640 | 0.204 |  |
| c203 | 1810 | 1760 | 1710 | 0.348 |  | 1710 | 0.481 |  | 1690 | 0.366 |  | 1680 | 0.258 |  |
| c204 | 1810 | 1780 | 1740 | 0.276 |  | 1720 | 0.395 |  | 1690 | 0.212 |  | 1680 | 0.197 |  |
| c205 | 1810 | 1770 | 1750 | 0.584 |  | 1760 | 0.249 |  | 1720 | 0.191 |  | 1740 | 0.21 |  |
| c206 | 1810 | 1770 | 1760 | 0.302 |  | 1750 | 0.233 |  | 1750 | 0.29 |  | 1740 | 0.228 |  |
| c207 | 1810 | 1810 | 1790 | 0.775 |  | 1780 | 0.288 |  | 1730 | 0.18 |  | 1730 | 0.211 |  |
| c208 | 1810 | 1810 | 1810 | 0.339 |  | 1760 | 0.208 |  | 1750 | 0.174 |  | 1730 | 0.203 |  |
| r101 | 484 | 481 | 412 | 0.211 |  | 318 | 0.21 |  | 370 | 0.225 |  | 270 | 0.231 |  |
| r102 | 694 | 685 | 565 | 0.311 |  | 523 | 0.236 |  | 492 | 0.265 |  | 440 | 0.411 |  |
| r103 | 747 | 720 | 650 | 0.642 |  | 623 | 0.284 |  | 540 | 0.318 |  | 530 | 0.293 |  |
| r104 | 778 | 765 | 709 | 0.631 |  | 693 | 0.359 |  | 554 | 0.259 |  | 577 | 0.397 |  |
| r105 | 620 | 609 | 527 | 0.359 |  | 462 | 0.237 |  | 468 | 0.226 |  | 441 | 0.253 |  |
| r106 | 729 | 719 | 651 | 0.523 |  | 615 | 0.303 |  | 533 | 0.25 |  | 521 | 0.229 |  |
| r107 | 760 | 747 | 674 | 0.488 |  | 651 | 0.297 |  | 565 | 0.303 |  | 552 | 0.501 |  |
| r108 | 797 | 790 | 712 | 0.483 |  | 681 | 0.271 |  | 598 | 0.327 |  | 618 | 0.248 |  |
| r109 | 710 | 699 | 639 | 0.408 |  | 605 | 0.265 |  | 533 | 0.245 |  | 535 | 0.256 |  |
| r110 | 737 | 711 | 668 | 0.441 |  | 640 | 0.335 |  | 618 | 0.242 |  | 597 | 0.234 |  |
| r111 | 774 | 764 | 707 | 0.5 |  | 630 | 0.234 |  | 587 | 0.239 |  | 571 | 0.246 |  |
| r112 | 776 | 758 | 707 | 0.572 |  | 656 | 0.261 |  | 583 | 0.307 |  | 597 | 0.236 |  |
| r201 | 1442 | 1408 | 1384 | 0.377 |  | 1377 | 0.358 |  | 1375 | 0.248 |  | 1343 | 0.243 |  |
| r202 | 1458 | 1443 | 1443 | 0.592 |  | 1434 | 0.28 |  | 1423 | 0.414 |  | 1415 | 0.157 |  |
| r203 | 1458 | 1458 | 1458 | 0.359 |  | 1458 | 0.159 |  | 1458 | 0.432 |  | 1447 | 0.199 |  |
| r204 | 1458 | 1458 | 1458 | 0.138 |  | 1458 | 0.13 |  | 1445 | 0.156 |  | 1449 | 0.134 |  |
| r205 | 1458 | 1458 | 1458 | 0.235 |  | 1458 | 0.152 |  | 1457 | 0.173 |  | 1446 | 0.161 |  |
| r206 | 1458 | 1458 | 1458 | 0.189 |  | 1458 | 0.12 |  | 1458 | 0.297 |  | 1458 | 0.136 |  |
| r207 | 1458 | 1458 | 1458 | 0.159 |  | 1458 | 0.126 |  | 1458 | 0.138 |  | 1458 | 0.127 |  |
| r208 | 1458 | 1458 | 1458 | 0.115 |  | 1458 | 0.132 |  | 1458 | 0.132 |  | 1458 | 0.121 |  |
| r209 | 1458 | 1458 | 1458 | 0.21 |  | 1458 | 0.224 |  | 1458 | 0.145 |  | 1458 | 0.156 |  |
| r210 | 1458 | 1458 | 1458 | 0.189 |  | 1458 | 0.147 |  | 1458 | 0.139 |  | 1458 | 0.161 |  |
| r211 | 1458 | 1458 | 1458 | 0.162 |  | 1458 | 0.088 |  | 1458 | 0.103 |  | 1458 | 0.109 |  |
| rc101 | 621 | 604 | 514 | 0.366 |  | 525 | 0.234 |  | 459 | 0.246 |  | 452 | 0.244 |  |
| rc102 | 714 | 698 | 625 | 0.357 |  | 586 | 0.238 |  | 560 | 0.3 |  | 513 | 0.238 |  |
| rc103 | 764 | 747 | 701 | 0.578 |  | 654 | 0.236 |  | 623 | 0.261 |  | 503 | 0.55 |  |
| rc104 | 835 | 822 | 788 | 0.582 |  | 639 | 0.27 |  | 617 | 0.282 |  | 549 | 0.359 |  |
| rc105 | 682 | 654 | 598 | 0.322 |  | 497 | 0.263 |  | 474 | 0.25 |  | 469 | 0.231 |  |
| rc106 | 706 | 678 | 632 | 0.398 |  | 613 | 0.263 |  | 552 | 0.214 |  | 481 | 0.316 |  |
| rc107 | 773 | 745 | 704 | 0.371 |  | 688 | 0.253 |  | 630 | 0.239 |  | 530 | 0.337 |  |
| rc108 | 795 | 757 | 744 | 0.521 |  | 651 | 0.274 |  | 587 | 0.228 |  | 587 | 0.226 |  |
| rc201 | 1698 | 1625 | 1625 | 0.691 |  | 1604 | 0.256 |  | 1578 | 0.252 |  | 1516 | 0.191 |  |
| rc202 | 1724 | 1686 | 1665 | 0.349 |  | 1659 | 0.25 |  | 1654 | 0.451 |  | 1632 | 0.253 |  |
| rc203 | 1724 | 1724 | 1714 | 0.28 |  | 1709 | 0.52 |  | 1701 | 0.193 |  | 1704 | 0.287 |  |
| rc204 | 1724 | 1724 | 1724 | 0.167 |  | 1724 | 0.141 |  | 1724 | 0.132 |  | 1716 | 0.149 |  |
| rc205 | 1719 | 1659 | 1655 | 1.052 |  | 1659 | 0.42 |  | 1605 | 0.313 |  | 1566 | 0.186 |  |
| rc206 | 1724 | 1708 | 1714 | 0.389 |  | 1706 | 0.377 |  | 1692 | 0.211 |  | 1685 | 0.201 |  |
| rc207 | 1724 | 1713 | 1712 | 0.222 |  | 1712 | 0.287 |  | 1709 | 0.28 |  | 1689 | 0.16 |  |
| rc208 | 1724 | 1724 | 1724 | 0.183 |  | 1724 | 0.128 |  | 1724 | 0.14 |  | 1724 | 0.311 |  |

Table 9: m=4 Solomon

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | BK | ILS(2009) | S=1 |  |  | S=2 |  |  | S=3 |  |  | S=4 |  |  |
|  | Score | Score | Score | CPU(s) | Gap (%) | Score | CPU(s) | Gap (%) | Score | CPU(s) | Gap (%) | Score | CPU(s) | Gap (%) |
| c101 | 1020 | 1000 | 0 | 0.112 |  | 0 | 0.12 |  | 0 | 0.129 |  | 0 | 0.143 |  |
| c102 | 1150 | 1090 | 400 | 0.2 |  | 400 | 0.188 |  | 400 | 0.202 |  | 400 | 0.264 |  |
| c103 | 1210 | 1150 | 840 | 0.508 |  | 810 | 0.373 |  | 770 | 0.228 |  | 730 | 0.257 |  |
| c104 | 1260 | 1220 | 1010 | 0.519 |  | 980 | 0.357 |  | 930 | 0.232 |  | 900 | 0.267 |  |
| c105 | 1070 | 1030 | 850 | 0.421 |  | 810 | 0.289 |  | 760 | 0.244 |  | 680 | 0.261 |  |
| c106 | 1080 | 1040 | 870 | 0.415 |  | 800 | 0.237 |  | 770 | 0.321 |  | 690 | 0.256 |  |
| c107 | 1120 | 1100 | 970 | 0.463 |  | 950 | 0.24 |  | 870 | 0.259 |  | 850 | 0.246 |  |
| c108 | 1140 | 1100 | 1000 | 0.512 |  | 980 | 0.243 |  | 930 | 0.255 |  | 860 | 0.487 |  |
| c109 | 1190 | 1180 | 1080 | 0.523 |  | 1000 | 0.234 |  | 960 | 0.277 |  | 900 | 0.315 |  |
| c201 | 1810 | 1810 | 1790 | 0.252 |  | 1780 | 0.167 |  | 1670 | 0.222 |  | 1770 | 0.174 |  |
| c202 | 1810 | 1810 | 1800 | 0.341 |  | 1800 | 0.345 |  | 1760 | 0.193 |  | 1740 | 0.169 |  |
| c203 | 1810 | 1810 | 1810 | 0.209 |  | 1790 | 0.178 |  | 1780 | 0.182 |  | 1780 | 0.266 |  |
| c204 | 1810 | 1810 | 1810 | 0.155 |  | 1800 | 0.138 |  | 1800 | 0.277 |  | 1800 | 0.266 |  |
| c205 | 1810 | 1810 | 1810 | 0.181 |  | 1810 | 0.132 |  | 1810 | 0.134 |  | 1810 | 0.111 |  |
| c206 | 1810 | 1810 | 1810 | 0.161 |  | 1810 | 0.117 |  | 1810 | 0.113 |  | 1810 | 0.101 |  |
| c207 | 1810 | 1810 | 1810 | 0.173 |  | 1810 | 0.116 |  | 1810 | 0.105 |  | 1810 | 0.157 |  |
| c208 | 1810 | 1810 | 1810 | 0.165 |  | 1810 | 0.121 |  | 1810 | 0.102 |  | 1810 | 0.111 |  |
| r101 | 611 | 601 | 510 | 0.317 |  | 385 | 0.265 |  | 446 | 0.216 |  | 332 | 0.205 |  |
| r102 | 843 | 807 | 721 | 0.422 |  | 627 | 0.299 |  | 591 | 0.255 |  | 546 | 0.339 |  |
| r103 | 928 | 878 | 817 | 0.528 |  | 723 | 0.257 |  | 667 | 0.284 |  | 648 | 0.508 |  |
| r104 | 975 | 941 | 867 | 0.49 |  | 844 | 0.318 |  | 695 | 0.289 |  | 729 | 0.488 |  |
| r105 | 778 | 735 | 676 | 0.375 |  | 580 | 0.262 |  | 557 | 0.242 |  | 544 | 0.246 |  |
| r106 | 906 | 870 | 792 | 0.552 |  | 769 | 0.312 |  | 696 | 0.256 |  | 673 | 0.303 |  |
| r107 | 950 | 927 | 874 | 0.736 |  | 768 | 0.284 |  | 746 | 0.282 |  | 719 | 0.359 |  |
| r108 | 995 | 982 | 888 | 0.743 |  | 846 | 0.262 |  | 755 | 0.285 |  | 760 | 0.297 |  |
| r109 | 885 | 866 | 785 | 0.516 |  | 747 | 0.293 |  | 691 | 0.232 |  | 700 | 0.546 |  |
| r110 | 915 | 870 | 826 | 0.676 |  | 780 | 0.283 |  | 717 | 0.49 |  | 718 | 0.441 |  |
| r111 | 953 | 935 | 877 | 0.547 |  | 772 | 0.283 |  | 707 | 0.247 |  | 704 | 0.503 |  |
| r112 | 974 | 939 | 894 | 0.604 |  | 855 | 0.303 |  | 776 | 0.257 |  | 751 | 0.266 |  |
| r201 | 1458 | 1458 | 1458 | 0.298 |  | 1432 | 0.321 |  | 1455 | 0.17 |  | 1420 | 0.149 |  |
| r202 | 1458 | 1458 | 1458 | 0.177 |  | 1455 | 0.114 |  | 1458 | 0.158 |  | 1458 | 0.104 |  |
| r203 | 1458 | 1458 | 1458 | 0.145 |  | 1458 | 0.099 |  | 1458 | 0.14 |  | 1458 | 0.122 |  |
| r204 | 1458 | 1458 | 1458 | 0.054 |  | 1458 | 0.117 |  | 1458 | 0.129 |  | 1458 | 0.119 |  |
| r205 | 1458 | 1458 | 1458 | 0.132 |  | 1458 | 0.091 |  | 1458 | 0.074 |  | 1458 | 0.091 |  |
| r206 | 1458 | 1458 | 1458 | 0.059 |  | 1458 | 0.063 |  | 1458 | 0.094 |  | 1458 | 0.086 |  |
| r207 | 1458 | 1458 | 1458 | 0.07 |  | 1458 | 0.063 |  | 1458 | 0.117 |  | 1458 | 0.112 |  |
| r208 | 1458 | 1458 | 1458 | 0.047 |  | 1458 | 0.101 |  | 1458 | 0.101 |  | 1458 | 0.117 |  |
| r209 | 1458 | 1458 | 1458 | 0.083 |  | 1458 | 0.094 |  | 1458 | 0.087 |  | 1458 | 0.088 |  |
| r210 | 1458 | 1458 | 1458 | 0.131 |  | 1458 | 0.077 |  | 1458 | 0.077 |  | 1458 | 0.088 |  |
| r211 | 1458 | 1458 | 1458 | 0.045 |  | 1458 | 0.075 |  | 1458 | 0.064 |  | 1458 | 0.087 |  |
| rc101 | 811 | 794 | 661 | 0.432 |  | 675 | 0.25 |  | 618 | 0.243 |  | 605 | 0.26 |  |
| rc102 | 909 | 881 | 784 | 0.422 |  | 731 | 0.249 |  | 692 | 0.276 |  | 599 | 0.365 |  |
| rc103 | 975 | 947 | 878 | 0.572 |  | 773 | 0.265 |  | 763 | 0.28 |  | 674 | 0.404 |  |
| rc104 | 1065 | 1019 | 967 | 0.479 |  | 781 | 0.257 |  | 869 | 0.245 |  | 833 | 0.294 |  |
| rc105 | 875 | 841 | 759 | 0.422 |  | 630 | 0.283 |  | 656 | 0.241 |  | 592 | 0.262 |  |
| rc106 | 909 | 874 | 813 | 0.443 |  | 788 | 0.263 |  | 729 | 0.256 |  | 646 | 0.302 |  |
| rc107 | 987 | 951 | 912 | 0.442 |  | 840 | 0.294 |  | 803 | 0.223 |  | 679 | 0.752 |  |
| rc108 | 1025 | 998 | 955 | 0.519 |  | 854 | 0.311 |  | 806 | 0.282 |  | 731 | 0.255 |  |
| rc201 | 1724 | 1724 | 1724 | 0.436 |  | 1709 | 0.169 |  | 1706 | 0.211 |  | 1675 | 0.162 |  |
| rc202 | 1724 | 1724 | 1724 | 0.19 |  | 1724 | 0.166 |  | 1724 | 0.16 |  | 1719 | 0.141 |  |
| rc203 | 1724 | 1724 | 1724 | 0.11 |  | 1724 | 0.13 |  | 1724 | 0.133 |  | 1724 | 0.168 |  |
| rc204 | 1724 | 1724 | 1724 | 0.046 |  | 1724 | 0.114 |  | 1724 | 0.121 |  | 1721 | 0.129 |  |
| rc205 | 1724 | 1724 | 1724 | 0.299 |  | 1724 | 0.199 |  | 1715 | 0.178 |  | 1684 | 0.138 |  |
| rc206 | 1724 | 1724 | 1724 | 0.137 |  | 1724 | 0.119 |  | 1724 | 0.096 |  | 1719 | 0.126 |  |
| rc207 | 1724 | 1724 | 1724 | 0.144 |  | 1724 | 0.115 |  | 1724 | 0.1 |  | 1722 | 0.127 |  |
| rc208 | 1724 | 1724 | 1724 | 0.056 |  | 1724 | 0.067 |  | 1724 | 0.086 |  | 1724 | 0.1 |  |

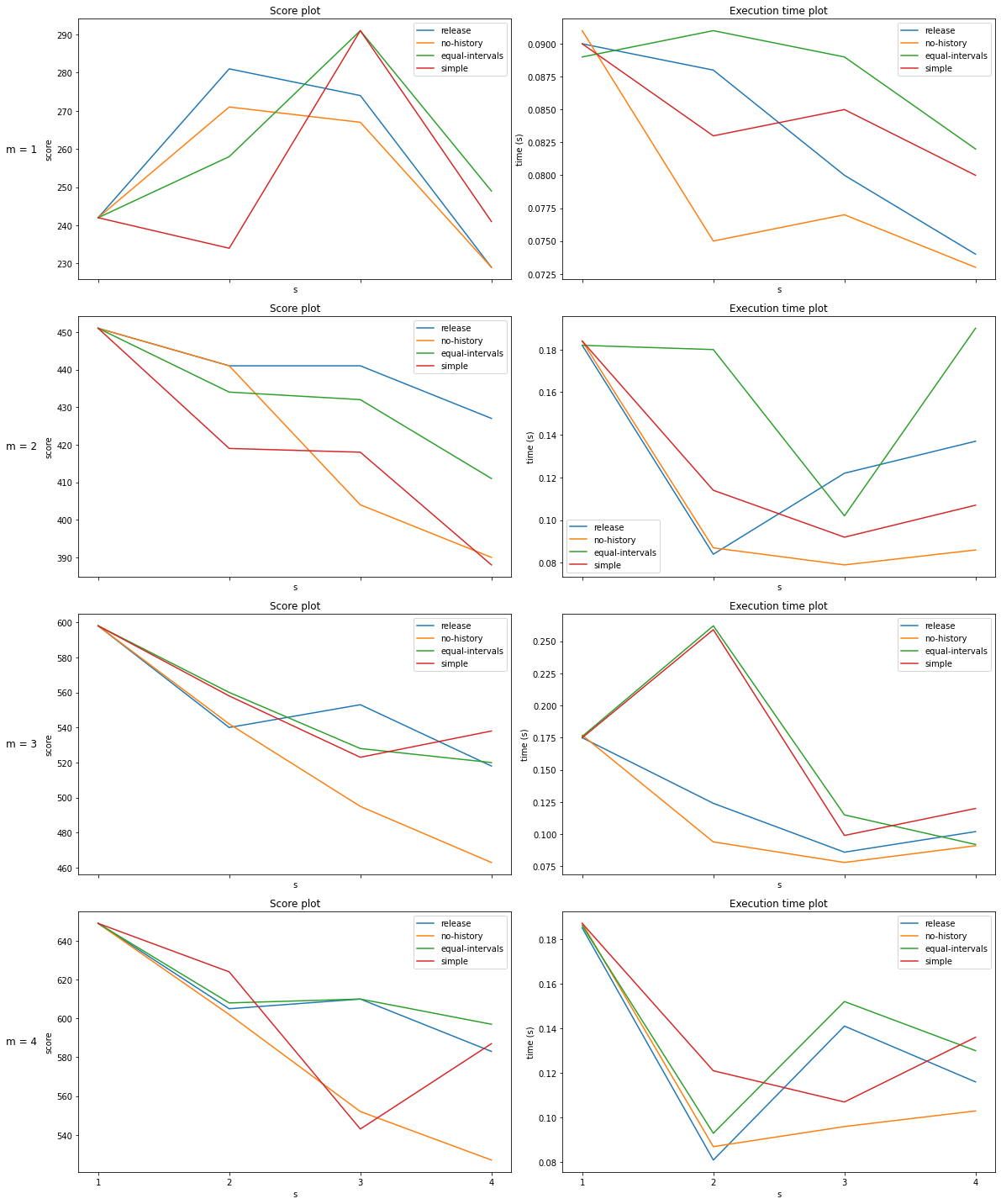


Figure 1: pr01

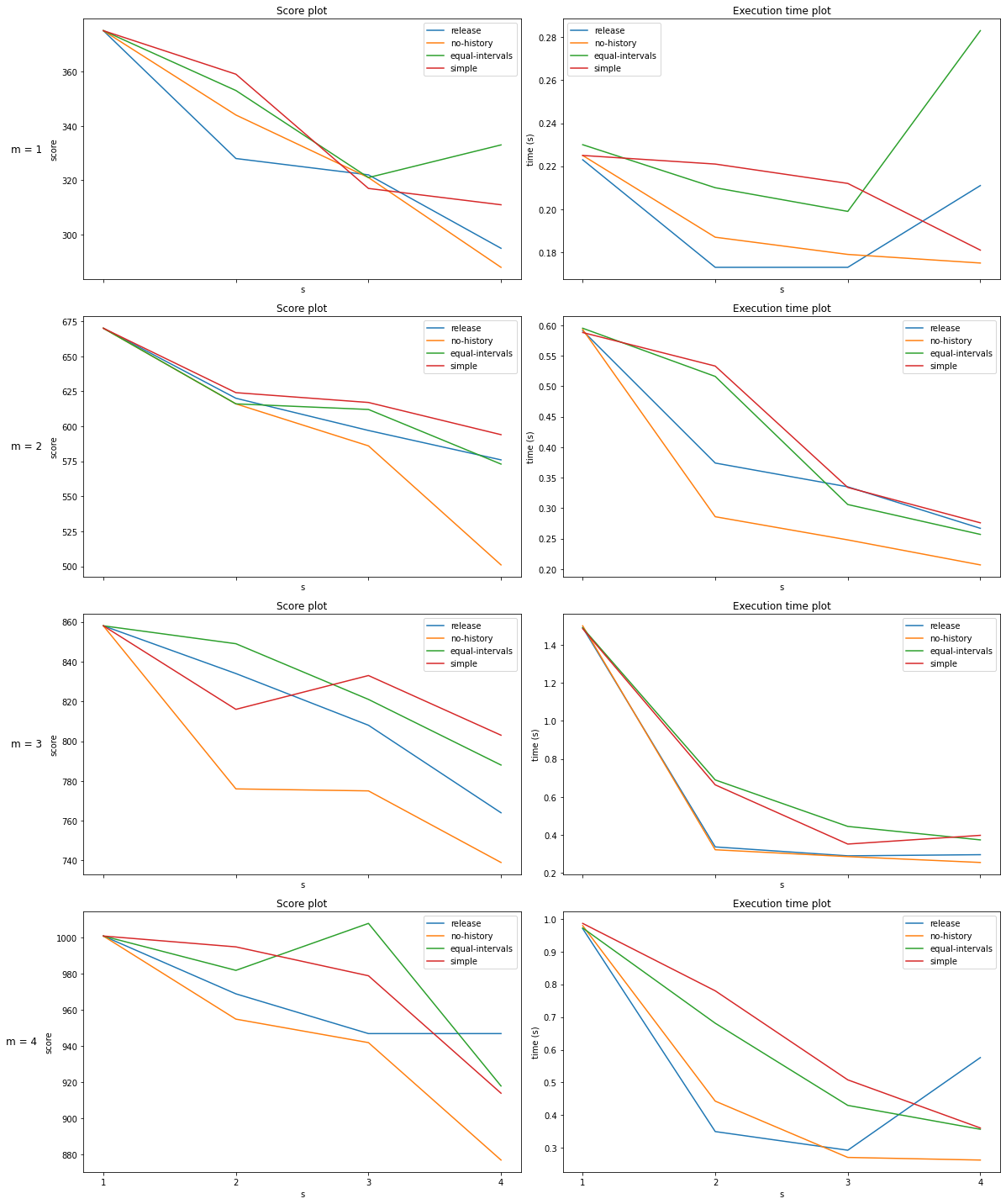


Figure 2: pr02

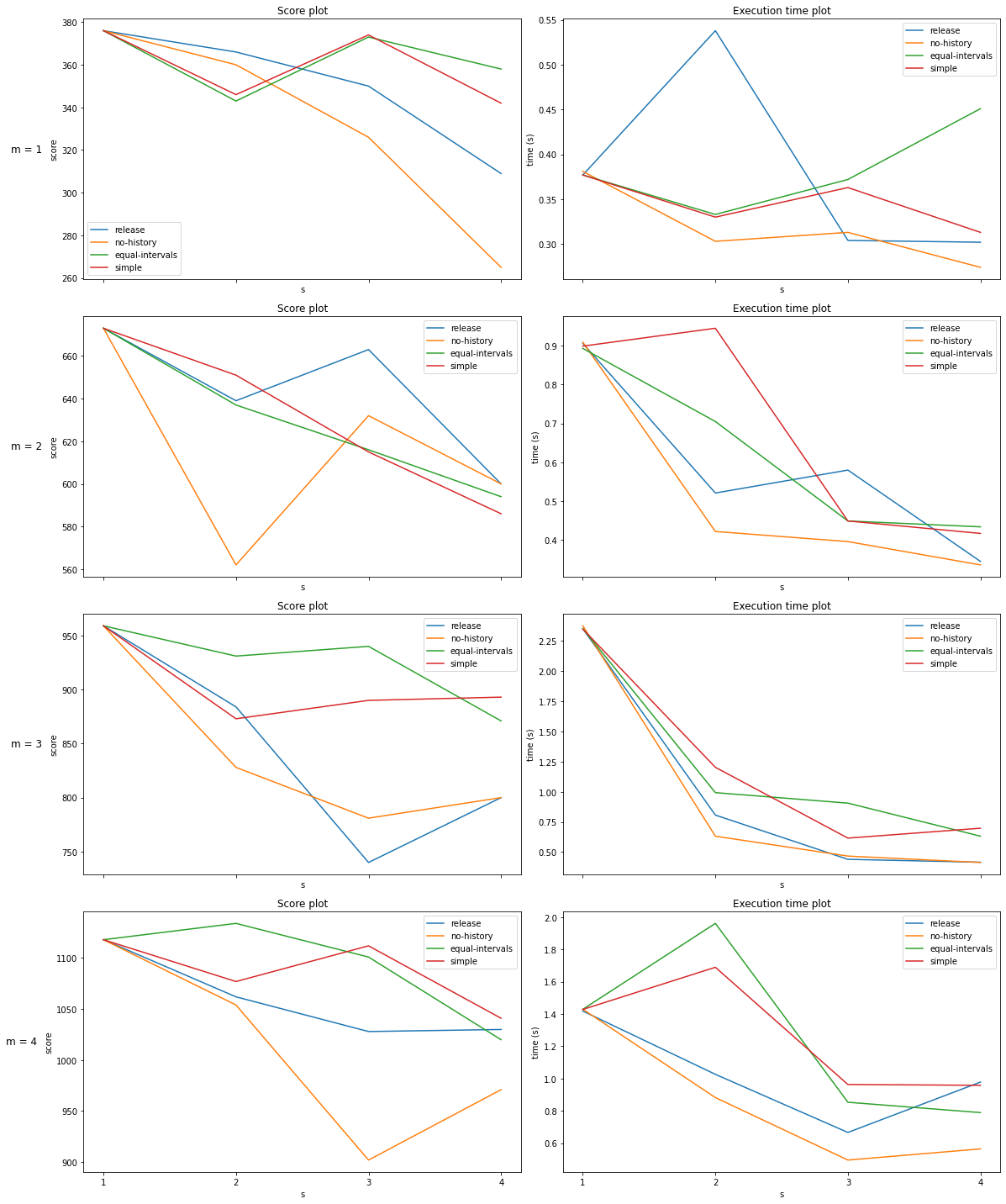


Figure 3: pr03

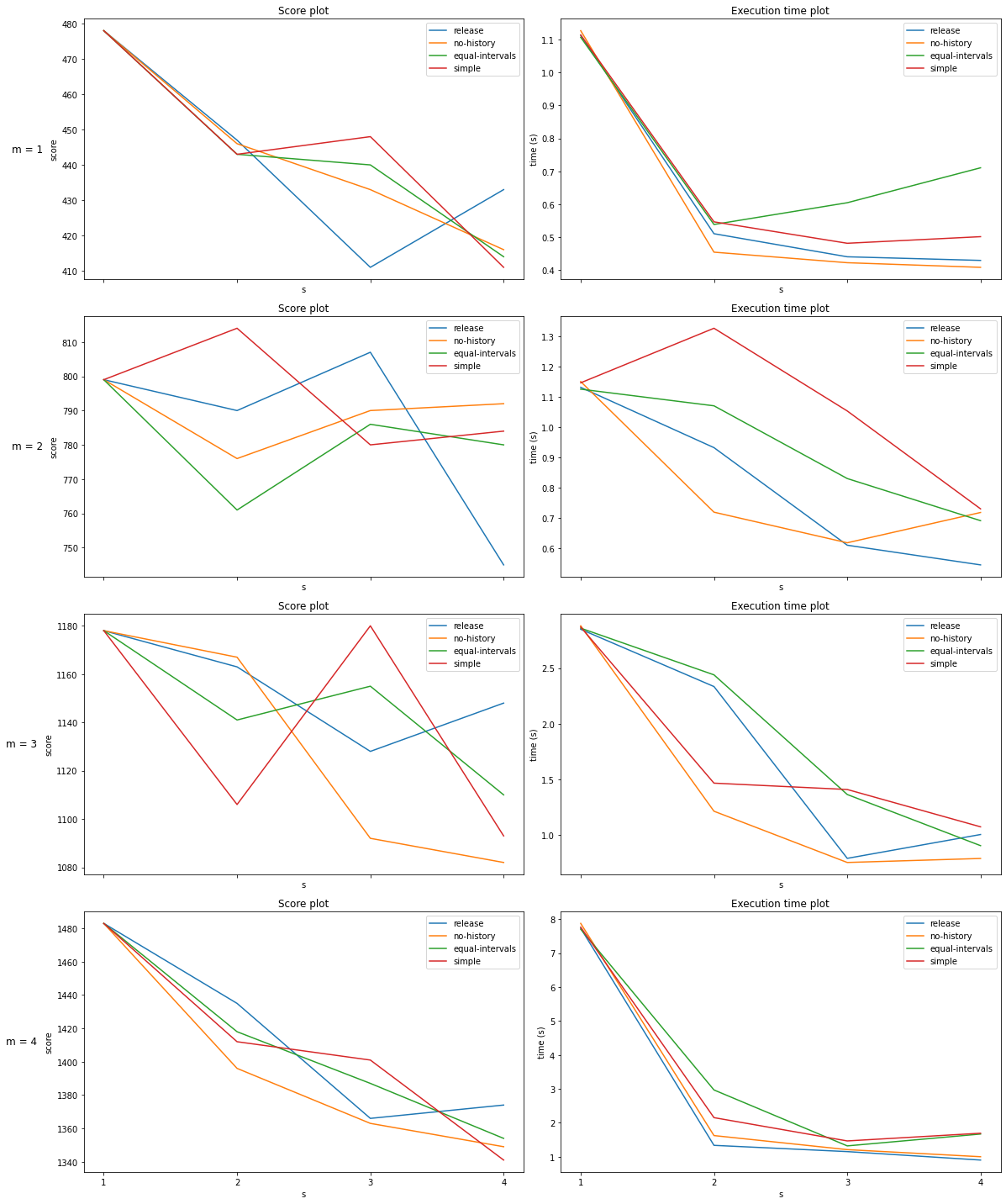


Figure 4: pr04

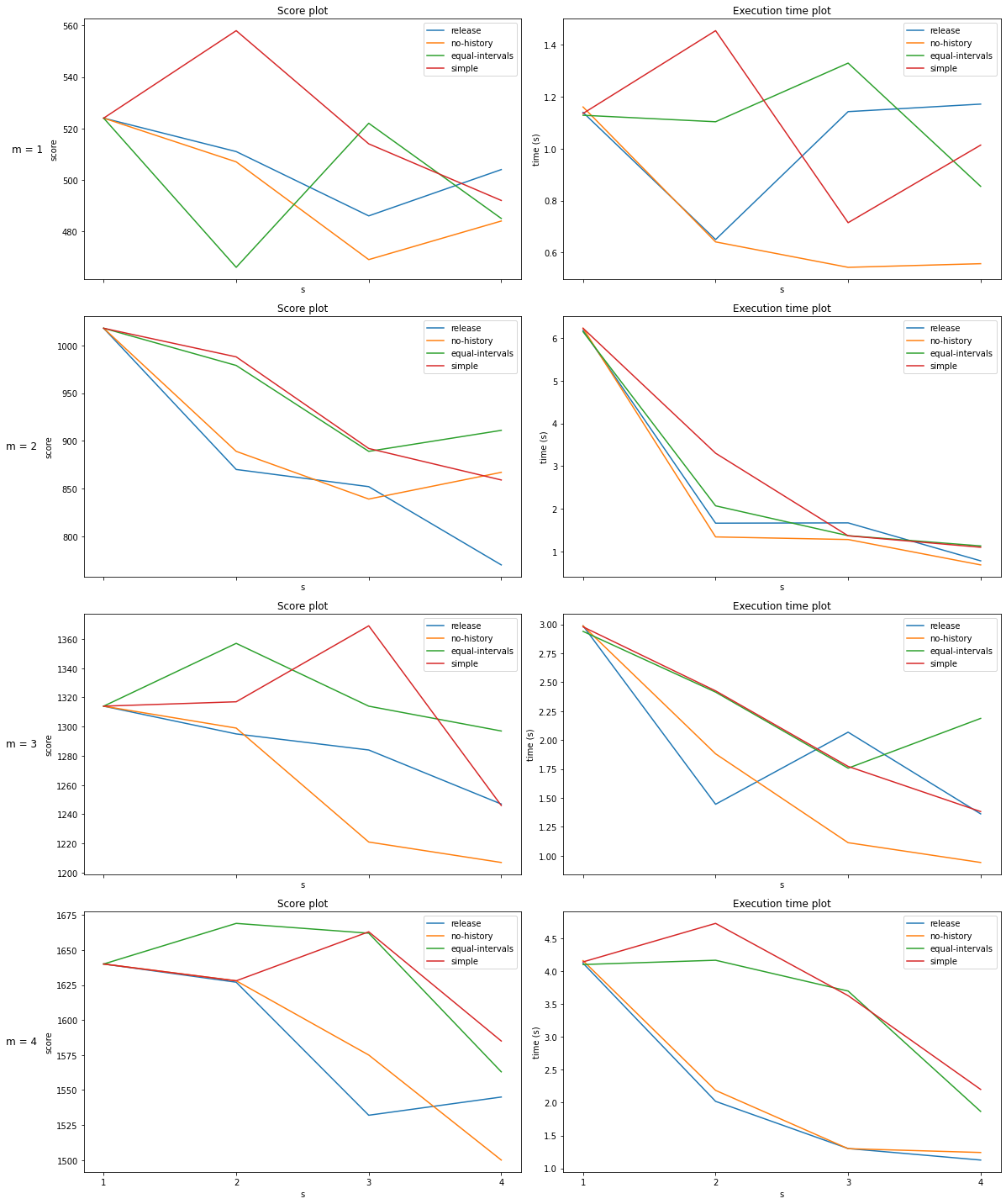


Figure 5: pr05

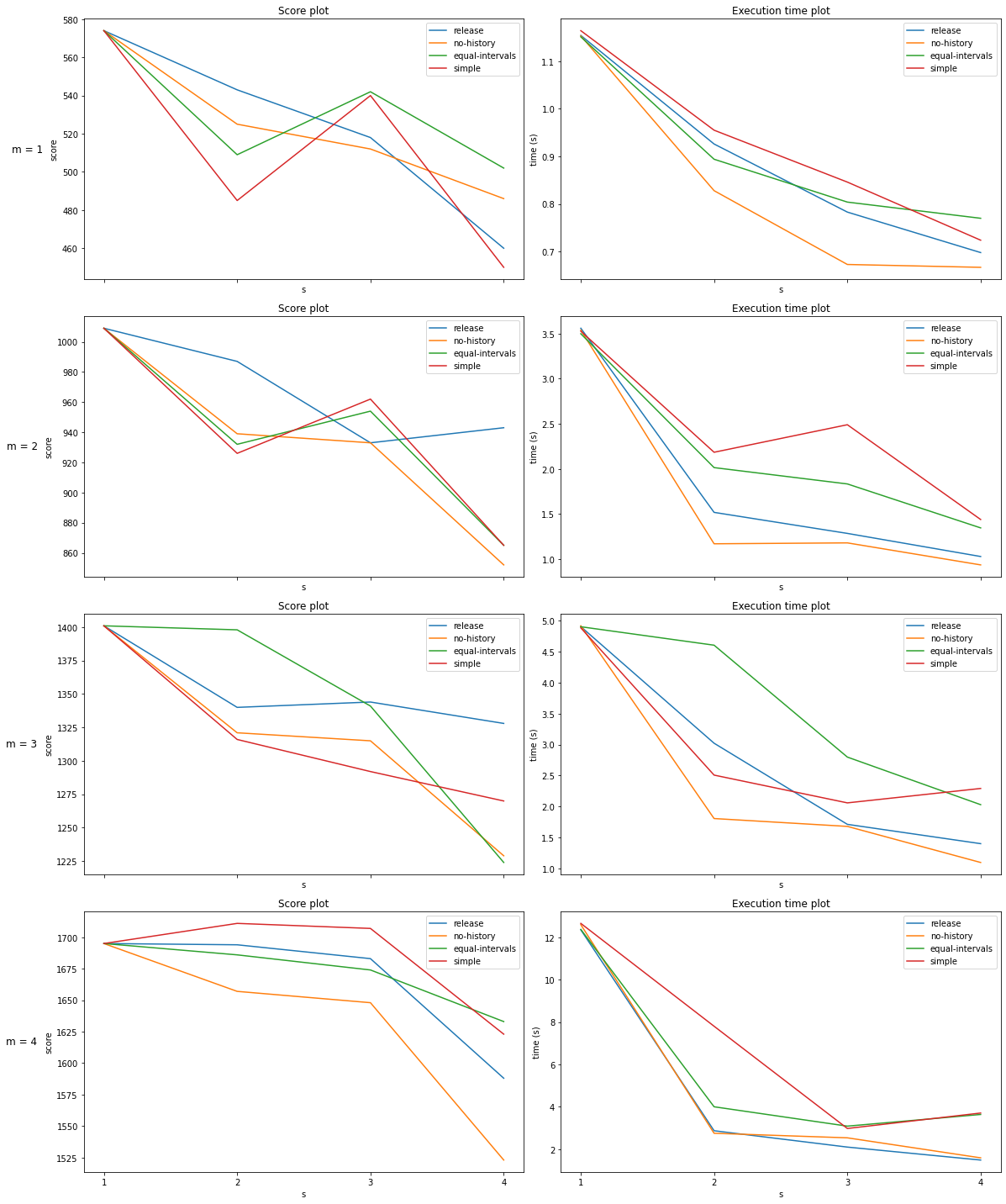


Figure 6: pr06

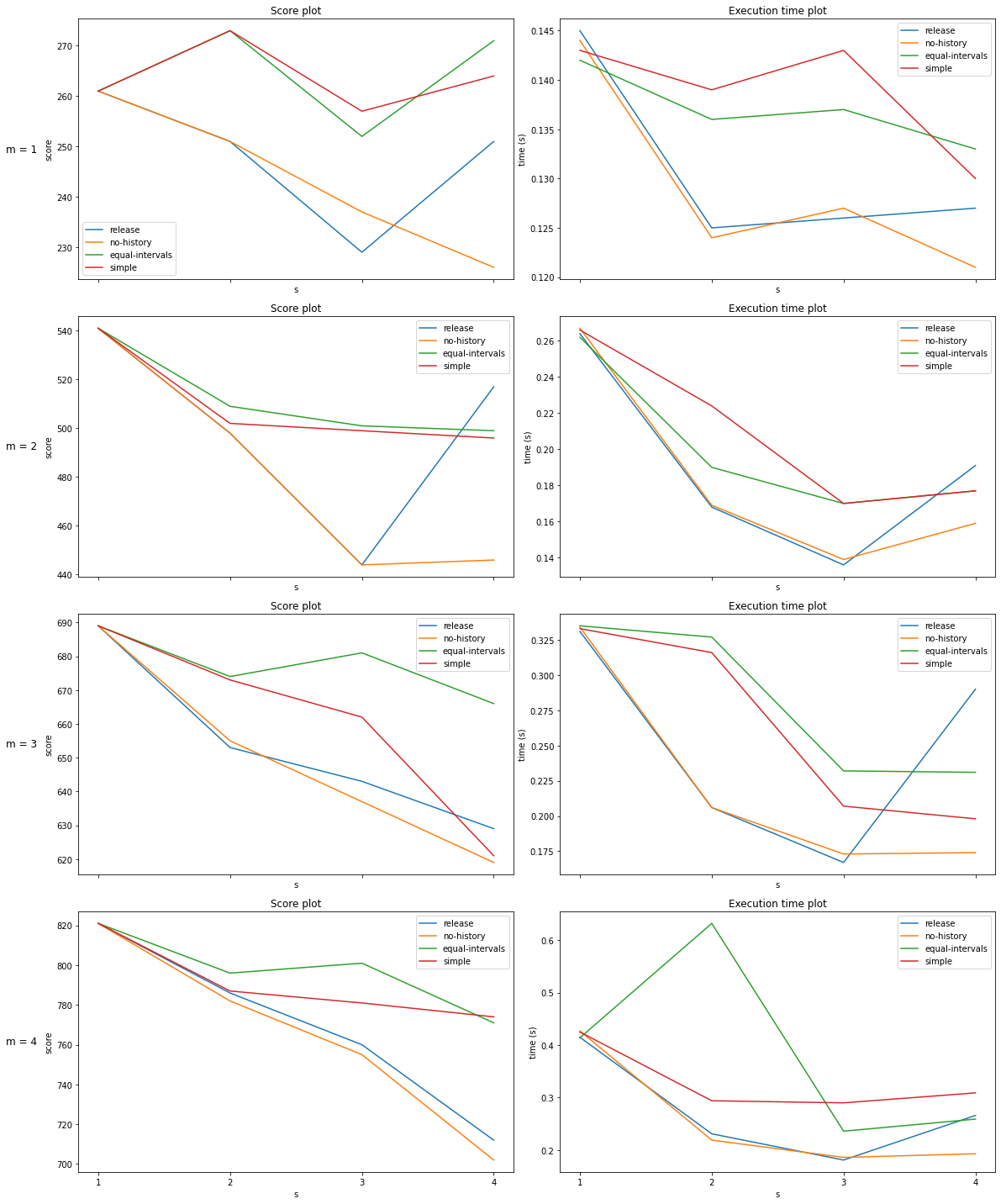


Figure 7: pr07

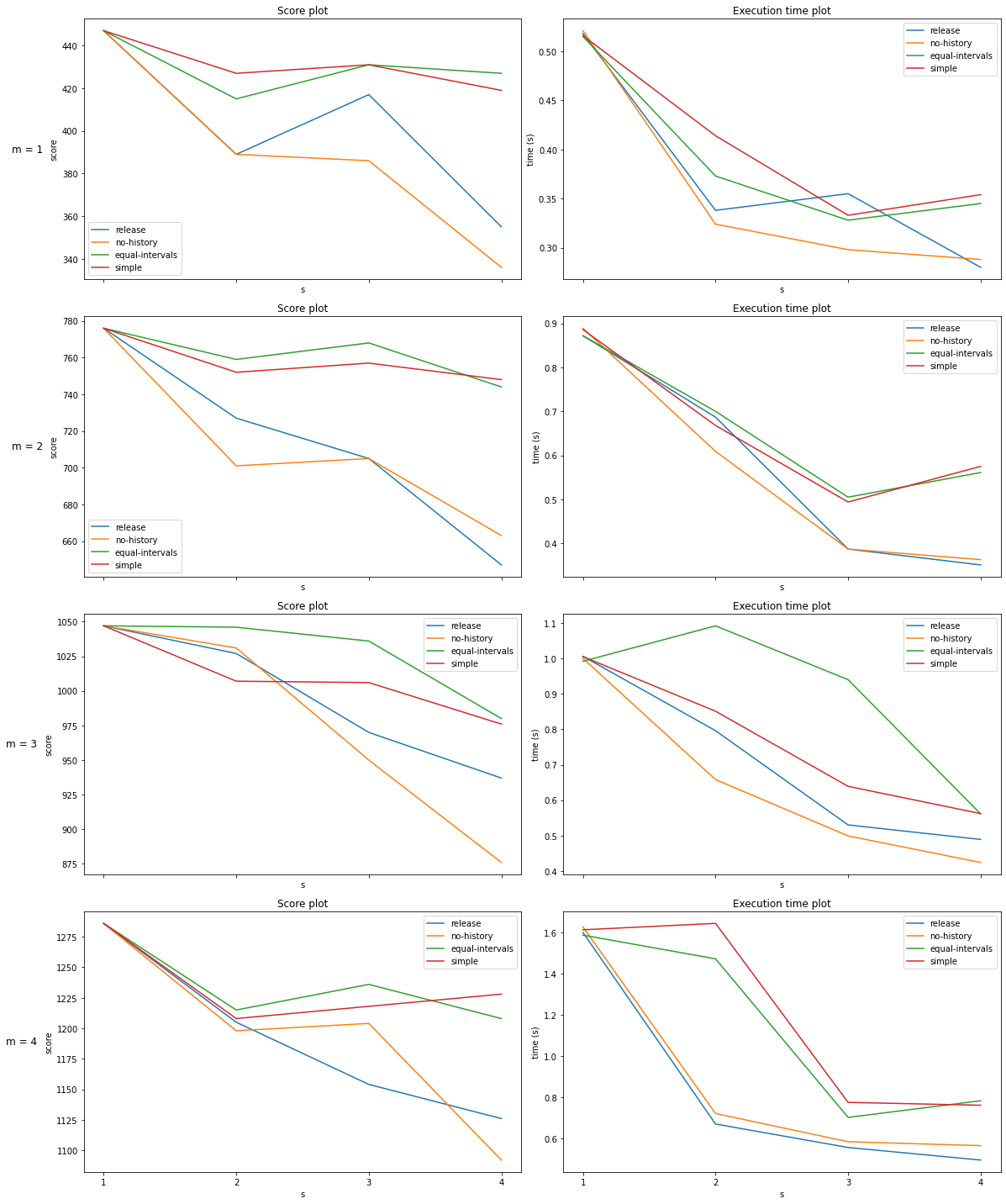


Figure 8: pr08

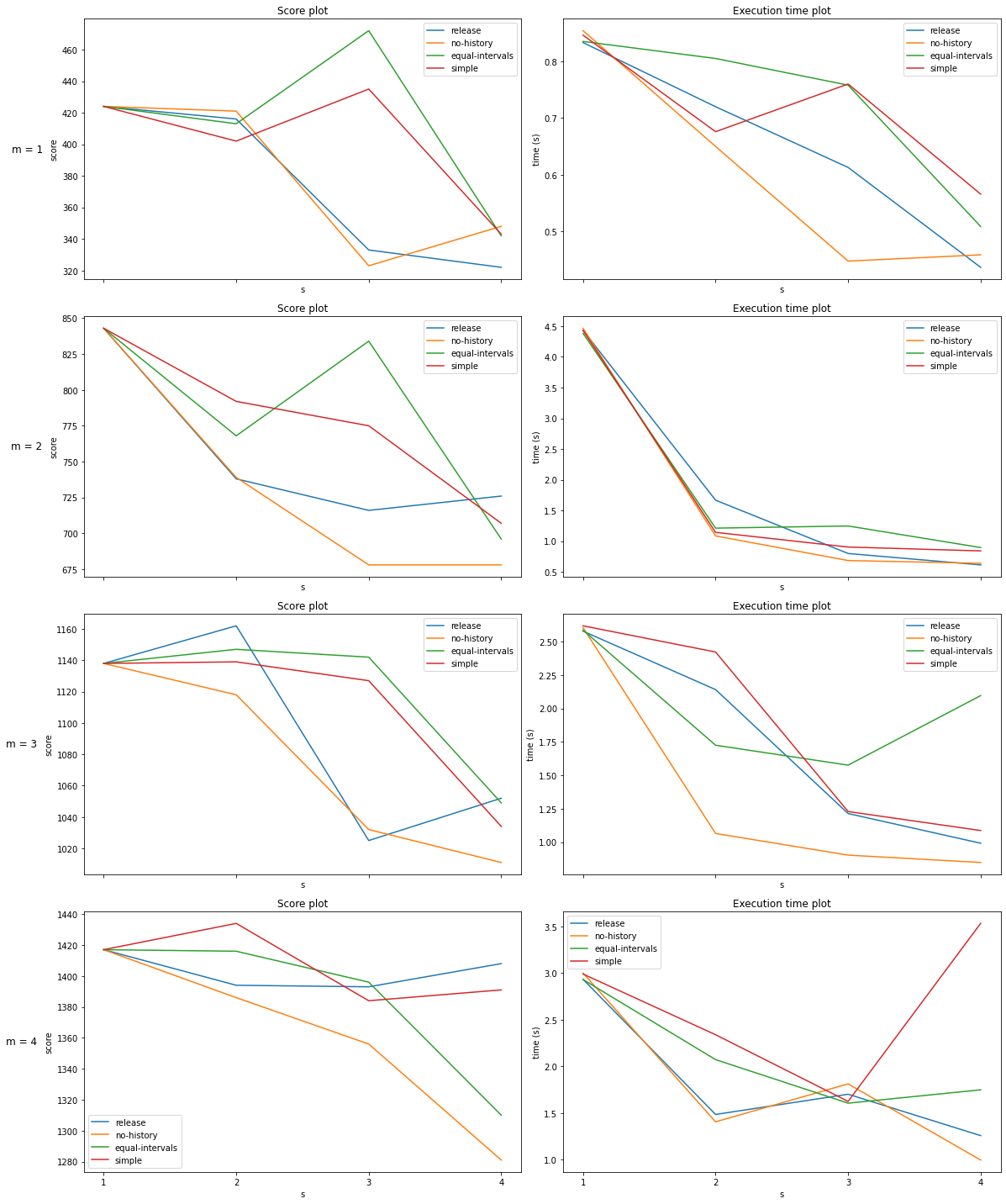


Figure 9: pr09

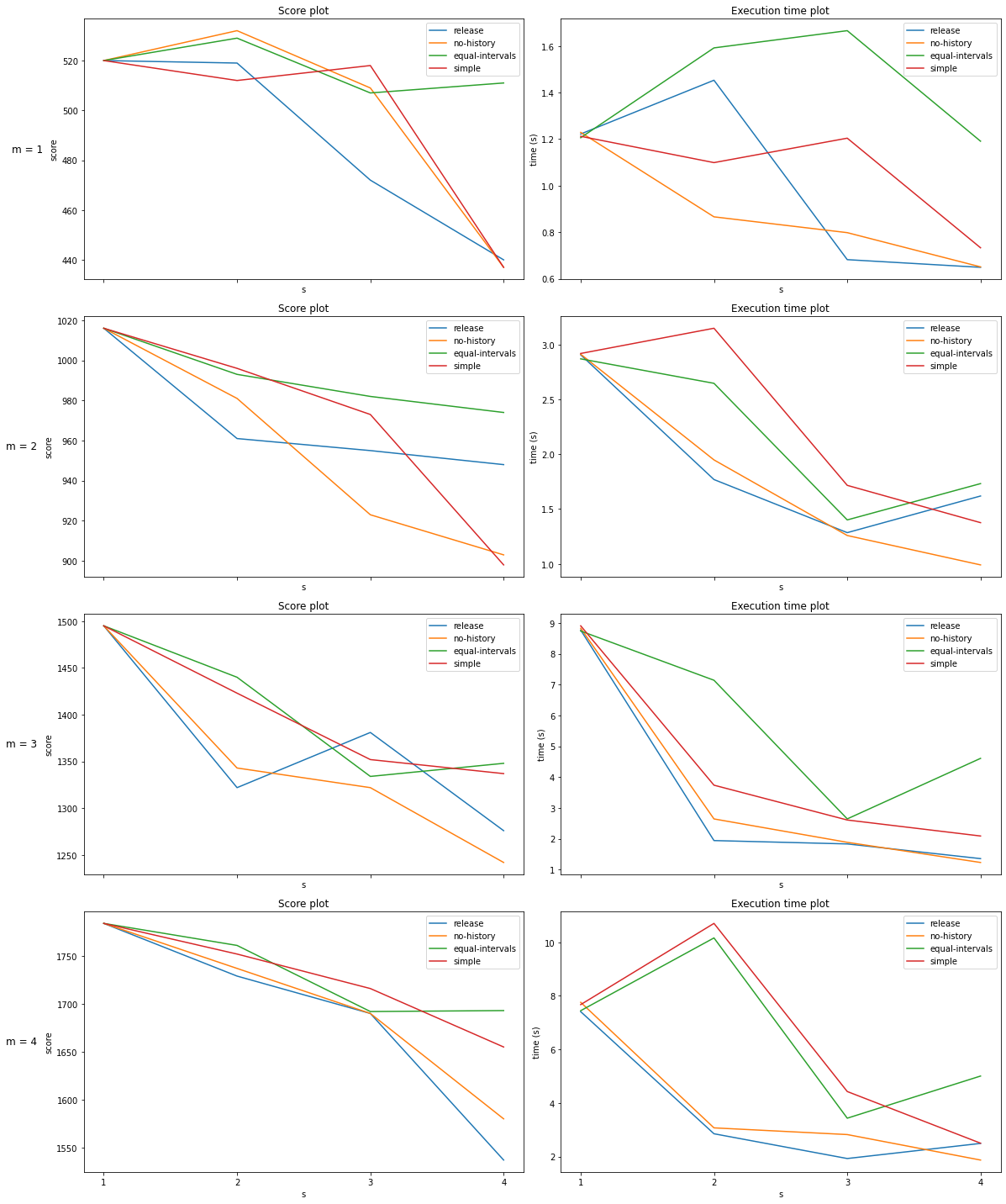


Figure 10: pr10

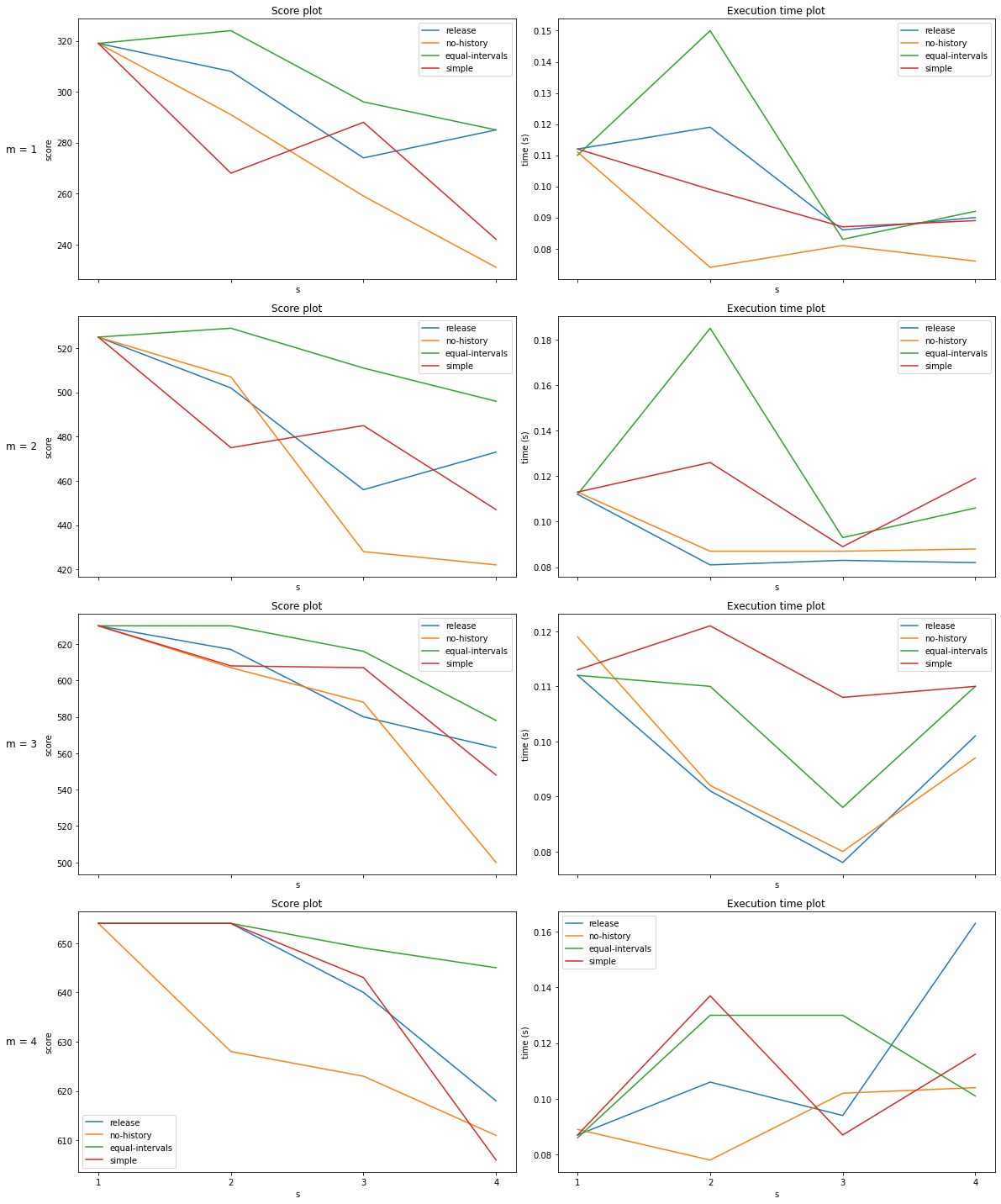


Figure 11: pr11

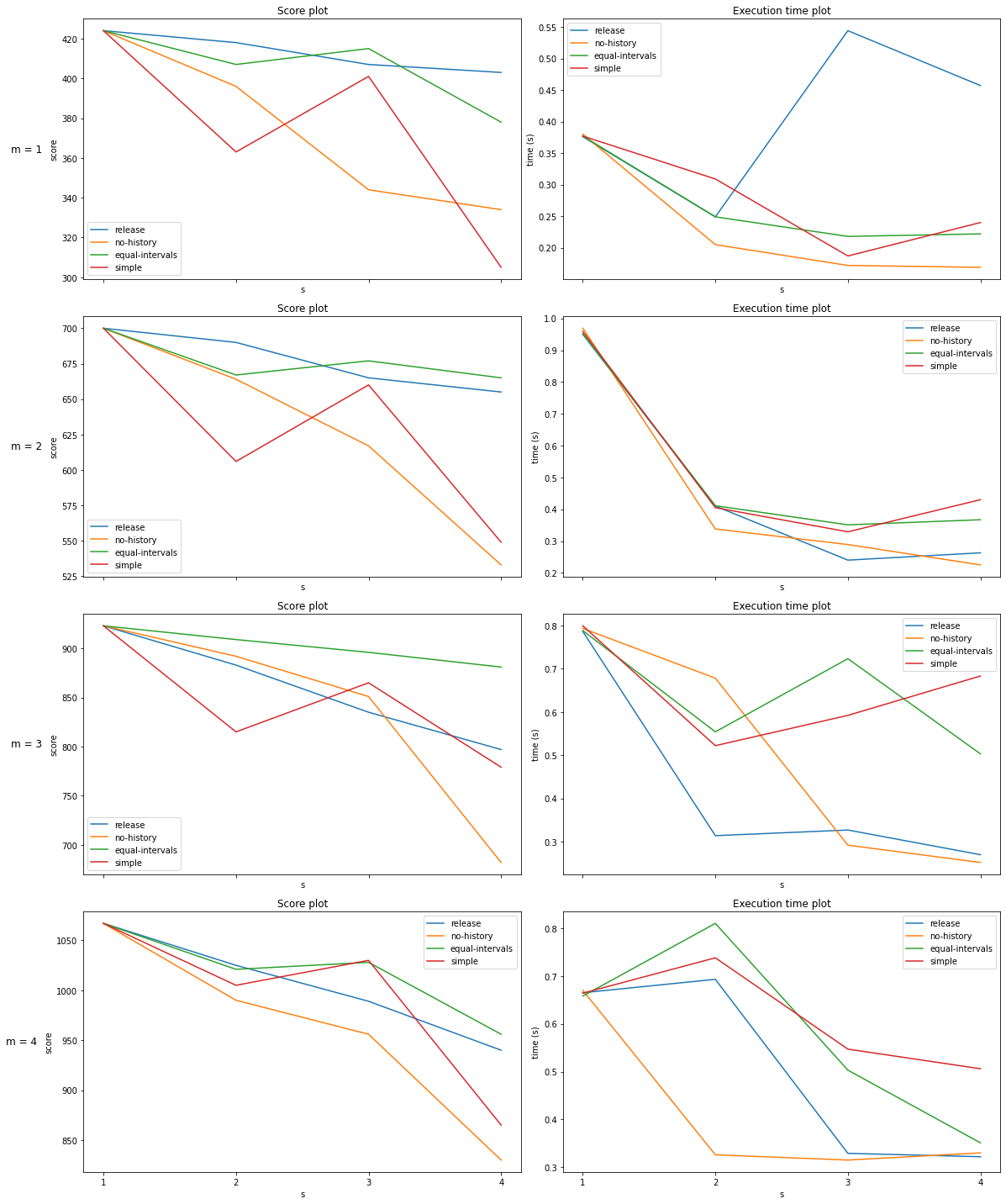


Figure 12: pr12

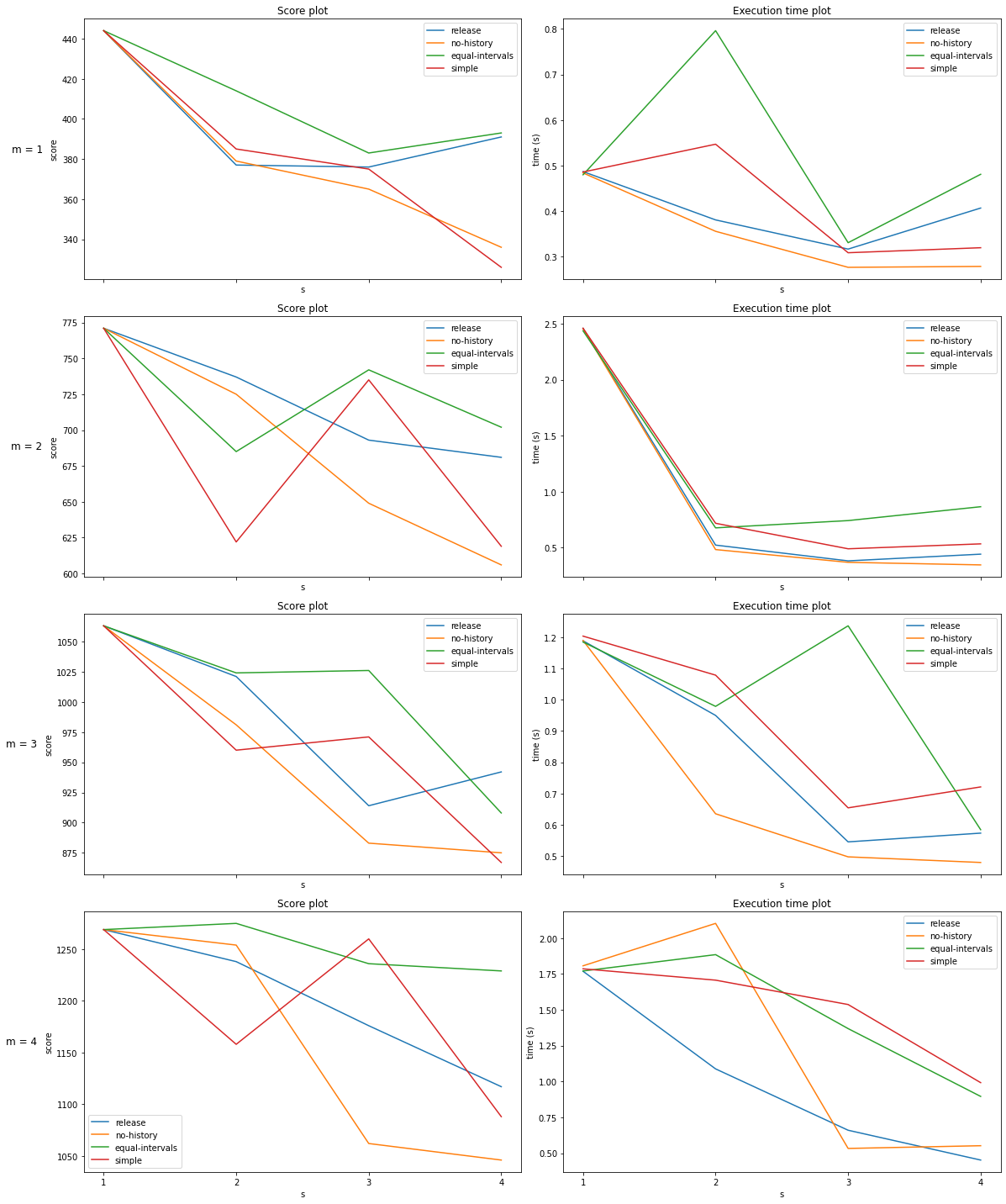


Figure 13: pr13

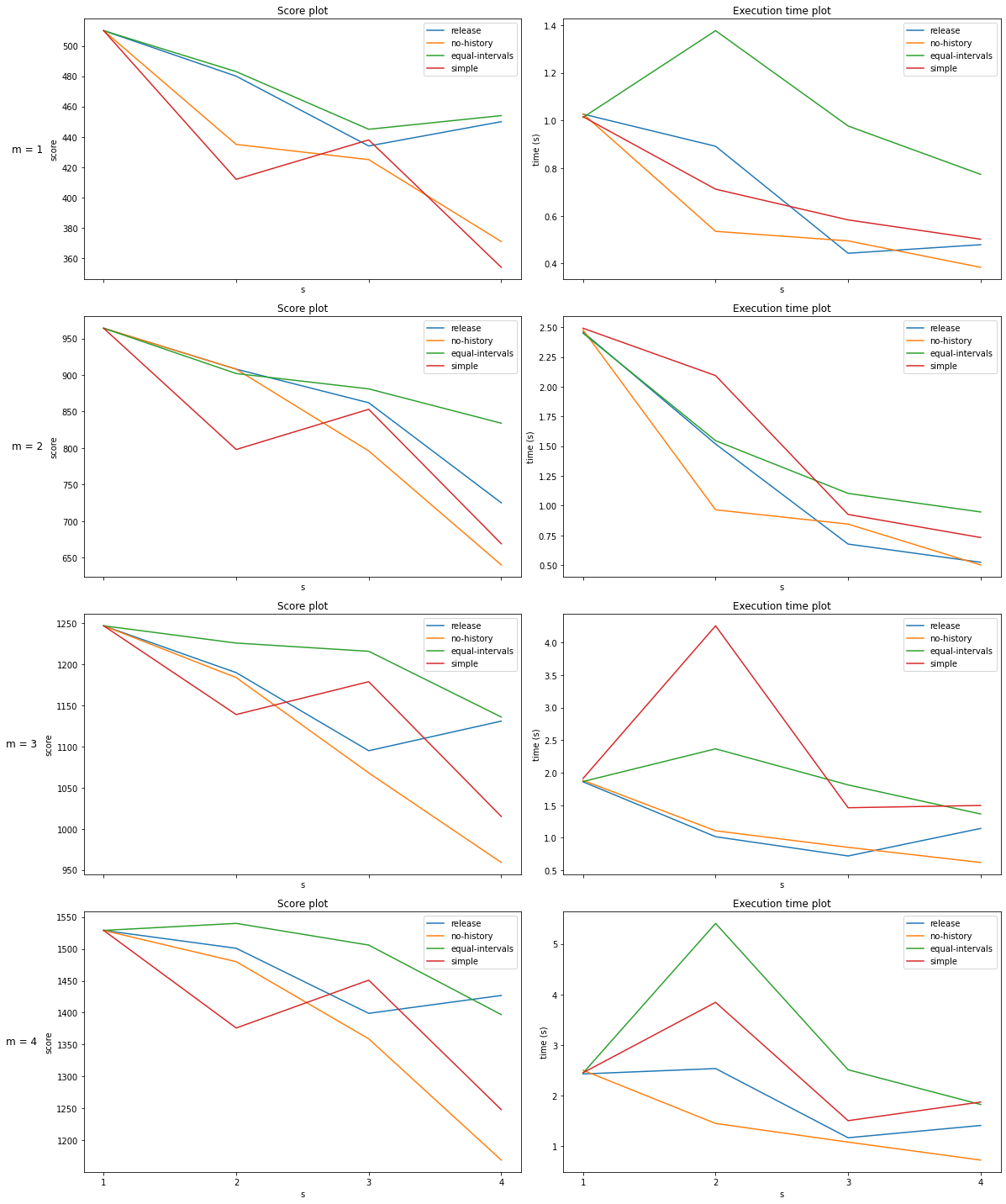


Figure 14: pr14

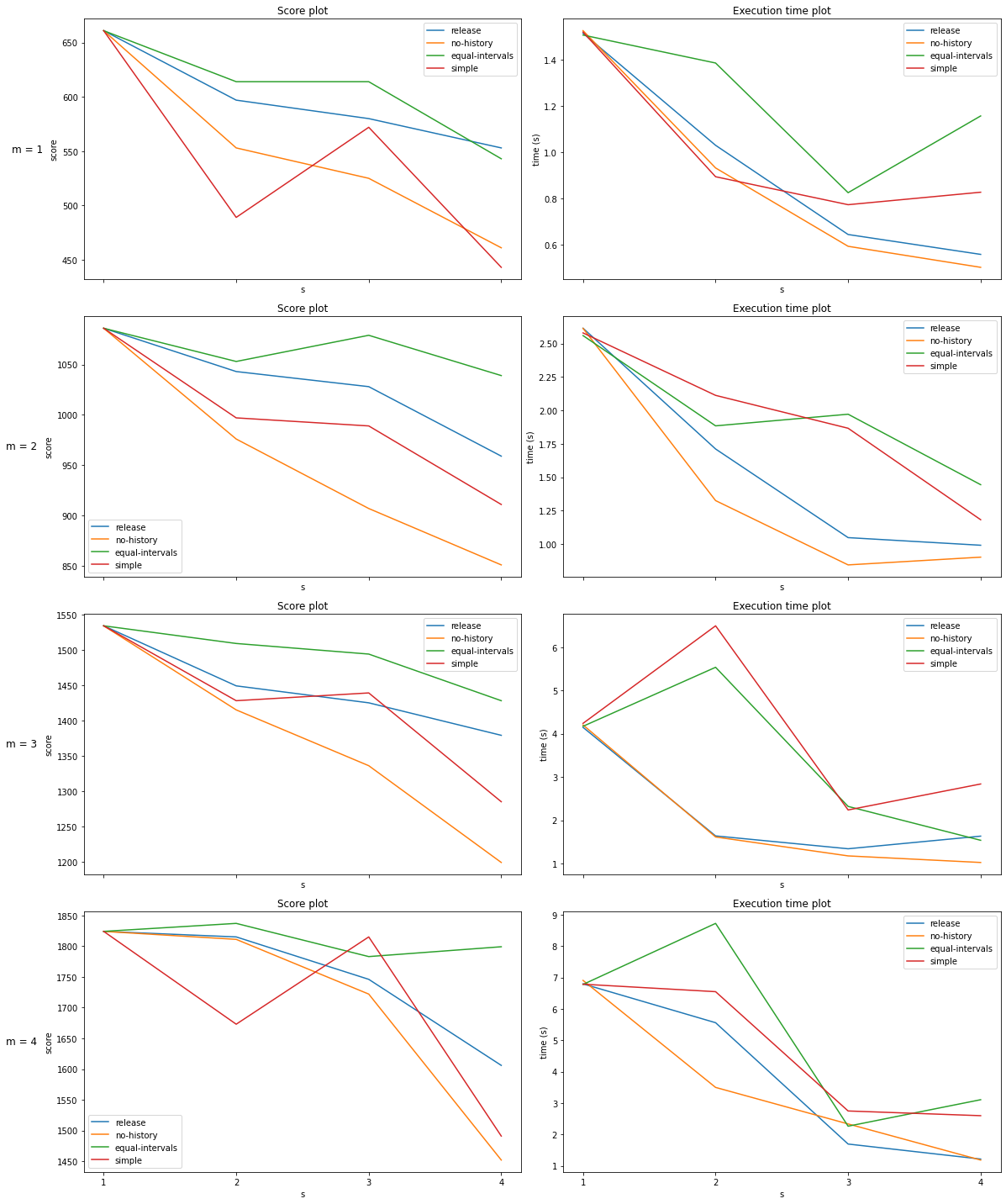


Figure 15: pr15

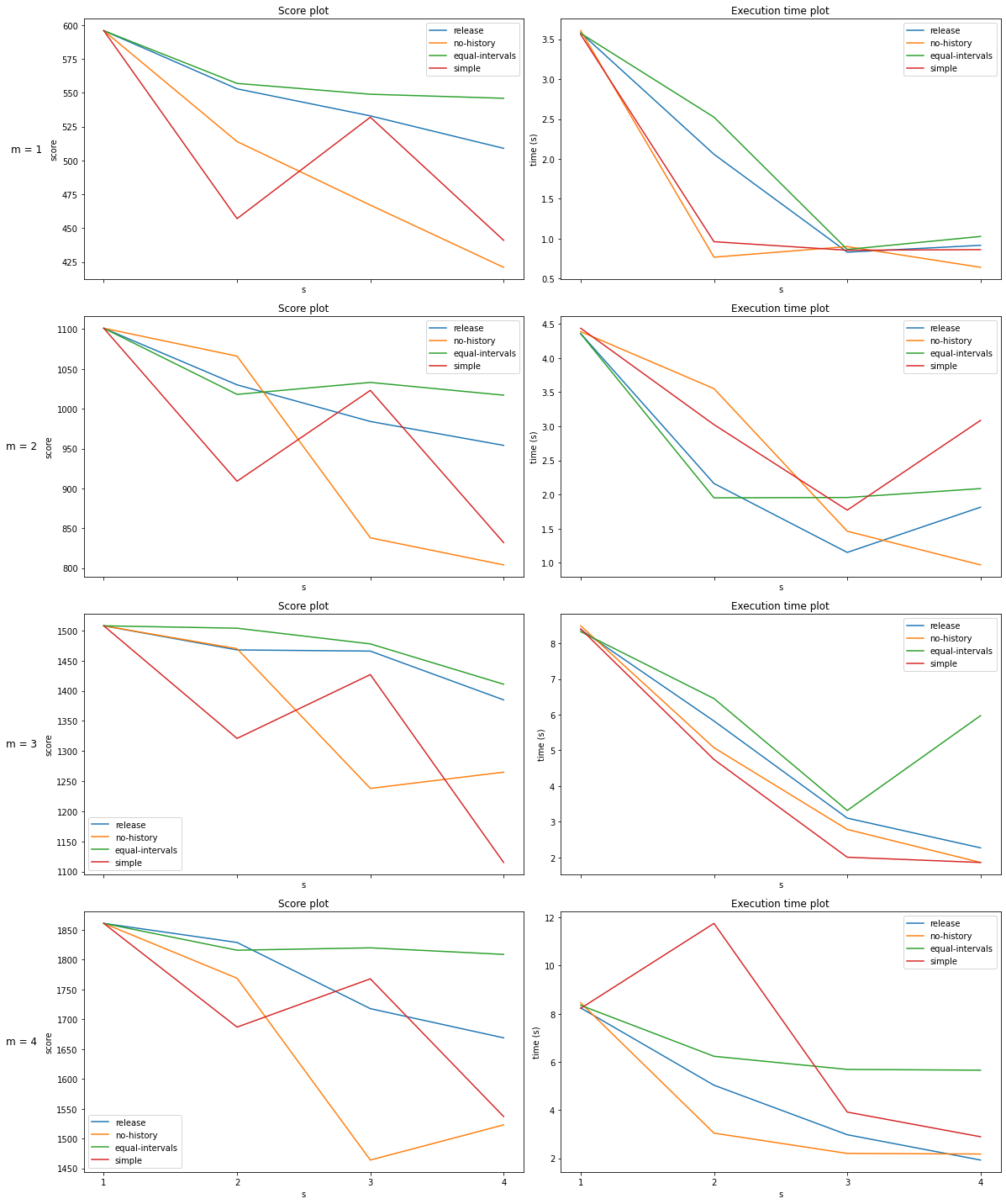


Figure 16: pr16

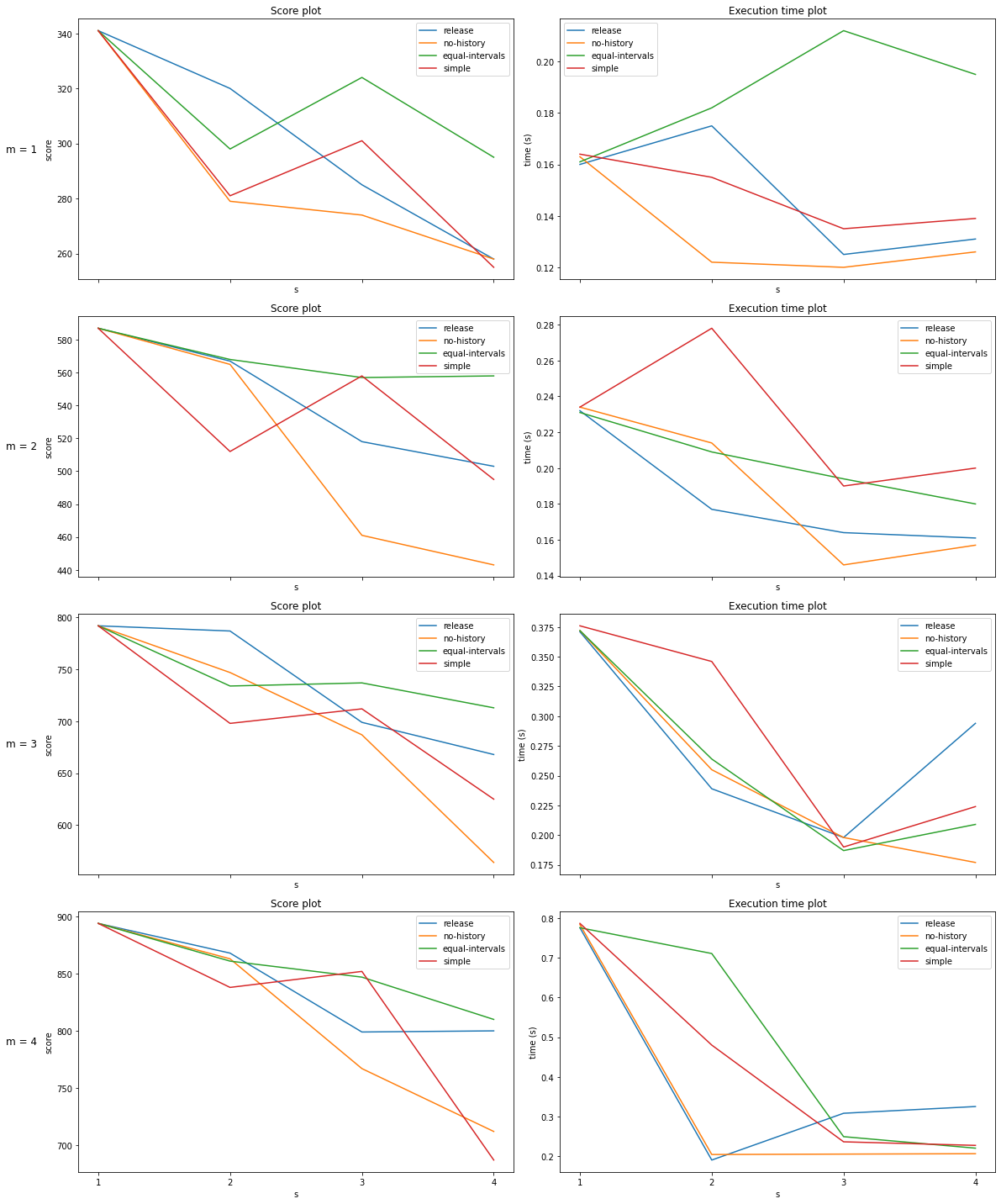


Figure 17: pr17

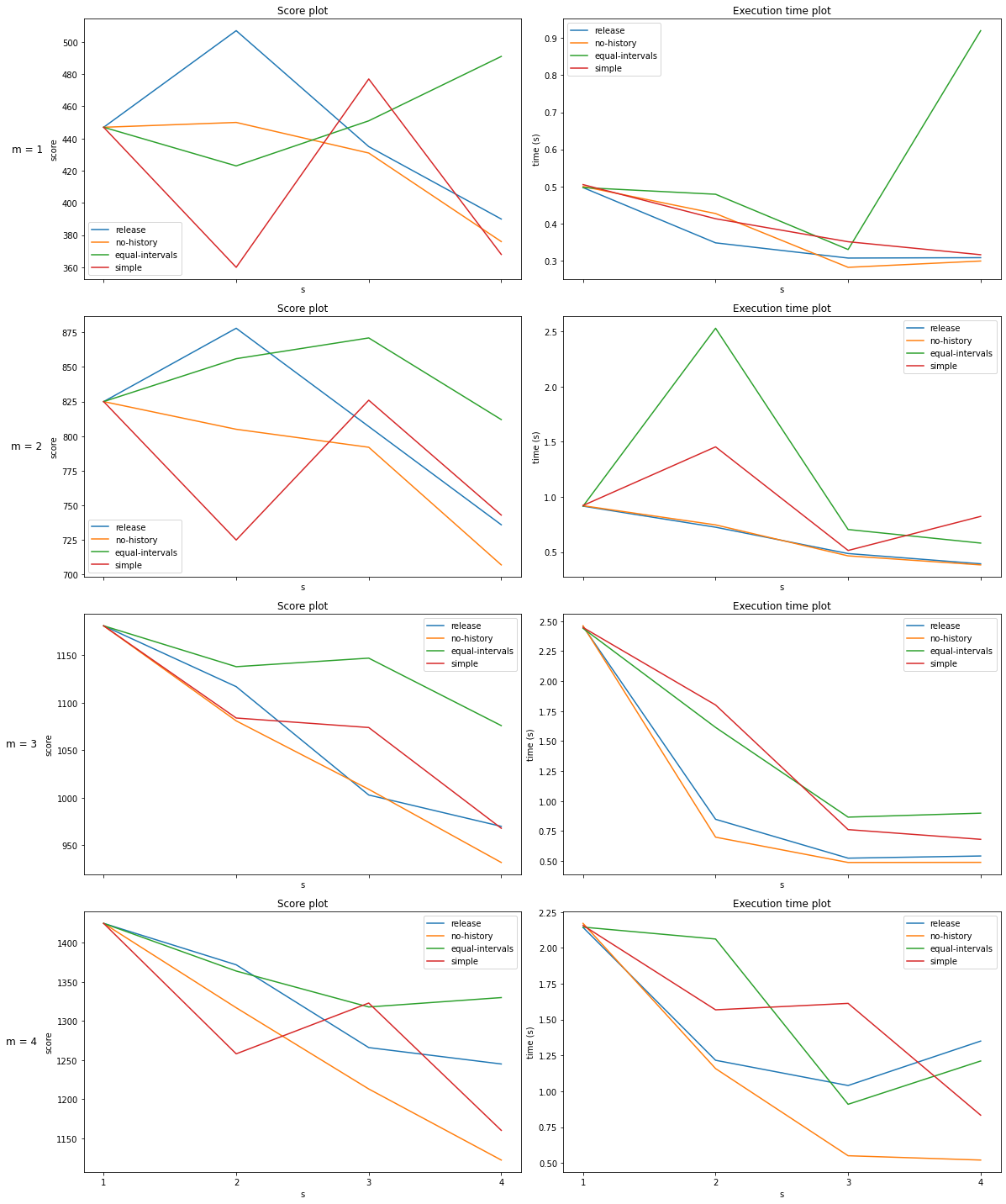


Figure 18: pr18

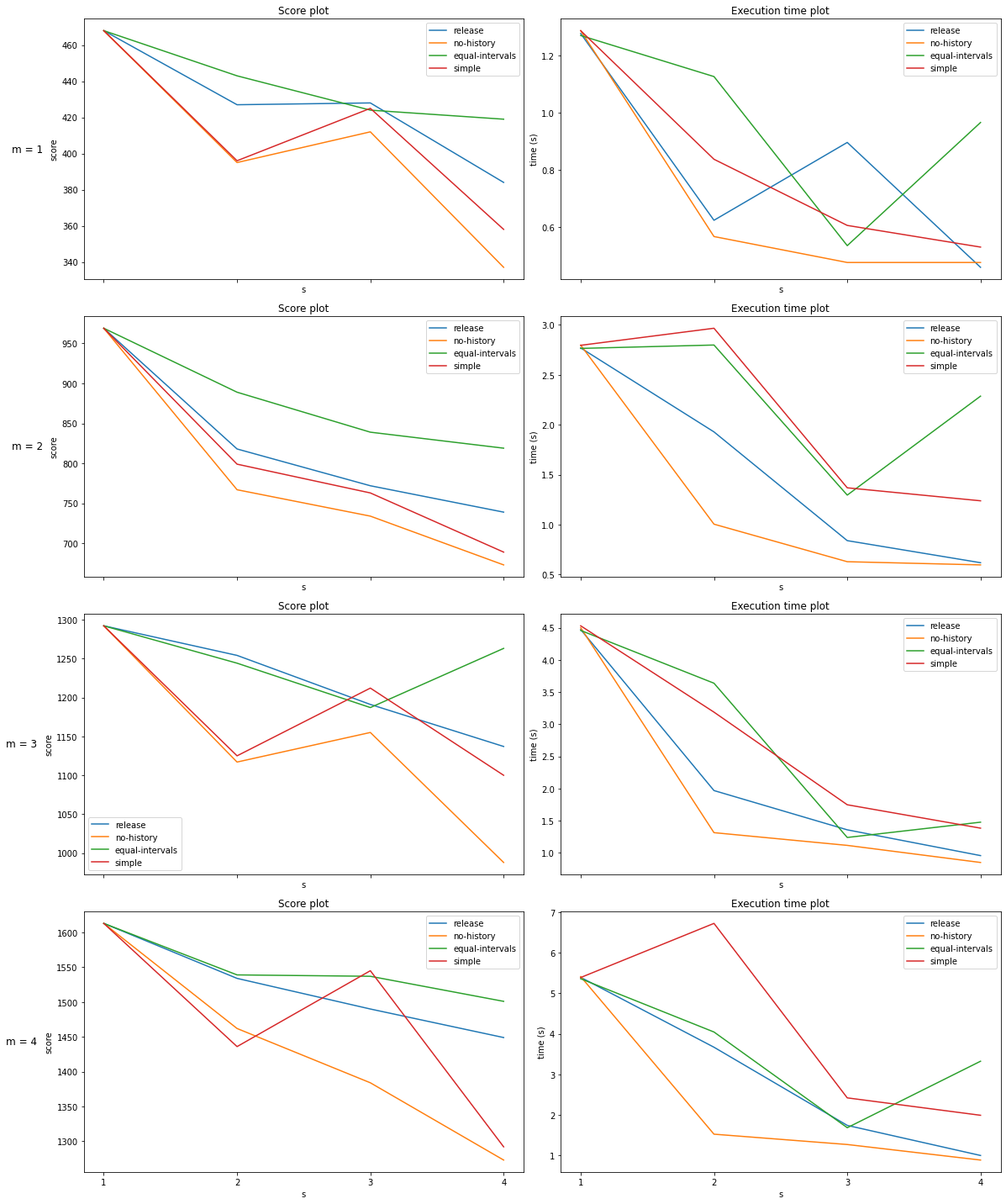


Figure 19: pr19

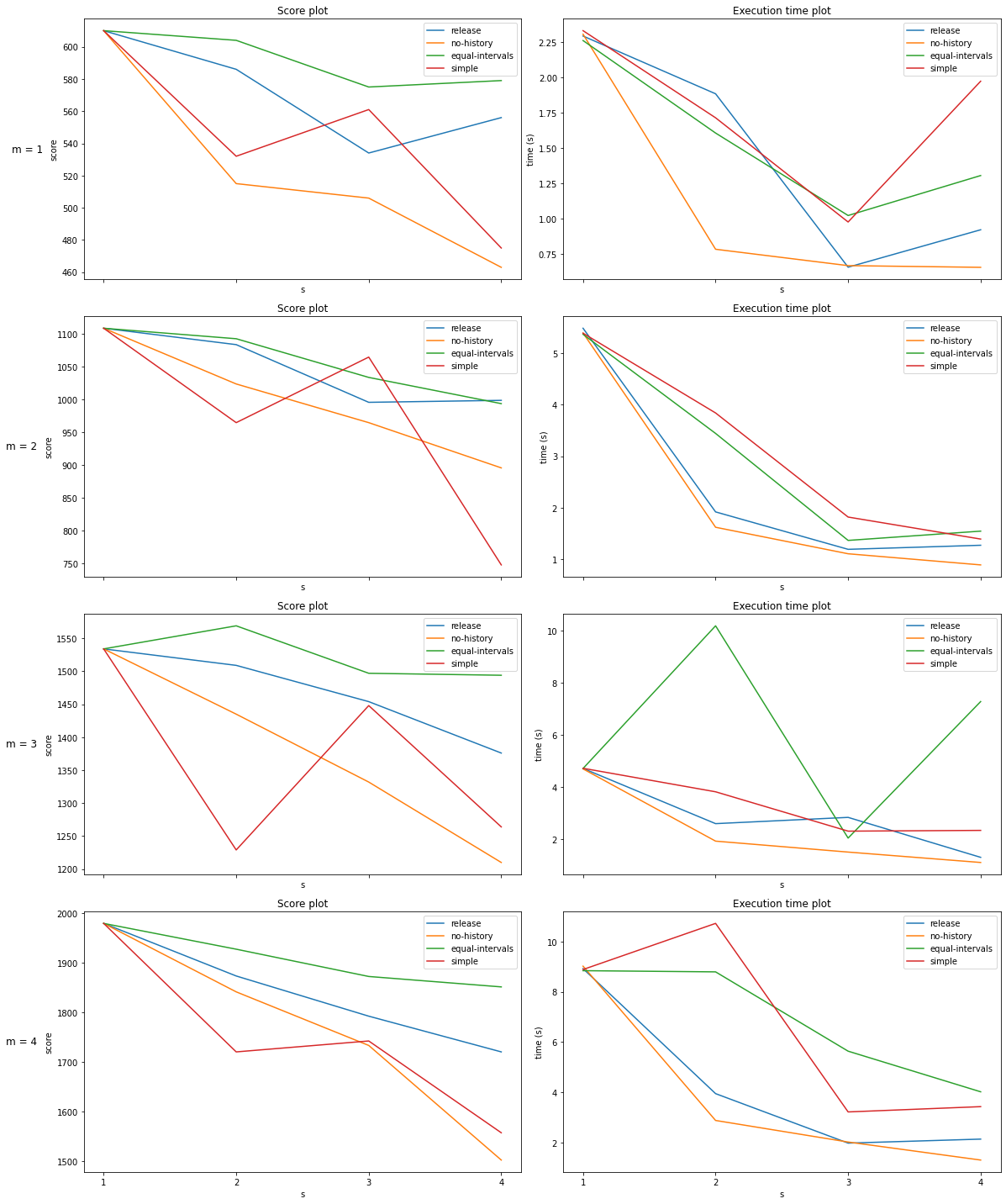


Figure 20: pr20







# Βιβλιογραφία

Abdel Monaem F.M. AbdAllah, D. L. (2017). On solving periodic re-optimization dynamic vehicle routing problems. *Applied Soft Computing*.

Andrea Attanasio, J. F. (2004). Parallel tabu search heuristics for the dynamic multi-vehicle dial-a-ride problem. Στο *Parallel Computing* (σσ. 377-387).