ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ ΠΑΡΑΓΩΓΗΣ ΚΑΙ ΔΙΟΙΚΗΣΗΣ

ΜΕΤΑΠΤΥΧΙΑΚΟ ΠΡΟΓΡΑΜΜΑ ΣΠΟΥΔΩΝ
«ΕΠΙΧΕΙΡΗΣΙΑΚΗ ΕΡΕΥΝΑ»

ΜΕΤΑΠΤΥΧΙΑΚΗ ΔΙΑΤΡΙΒΗ

Εργασία που υπεβλήθη για τη μερική ικανοποίηση των απαιτήσεων για την απόκτηση μεταπτυχιακού διπλώματος ειδίκευσης

«ΜΕΘΕΥΡΕΤΙΚΟΣ ΑΛΓΟΡΙΘΜΟΣ ΓΙΑ ΤΗ ΒΕΛΤΙΣΤΟΠΟΙΗΣΗ ΧΡΟΝΙΚΗΣ ΑΚΟΛΟΥΘΙΑΣ ΕΡΓΑΣΙΩΝ»

ΣΠΟΥΔΑΣΤΗΣ: **ΙΩΑΝΝΗΣ ΚΑΡΑΦΥΛΛΙΔΗΣ**
ΑΕΜ: 2014019048

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ: **ΜΑΡΙΝΑΚΗΣ ΙΩΑΝΝΗΣ**

ΕΠΙΤΡΟΠΗ ΑΞΙΟΛΟΓΗΣΗΣ: **ΜΑΡΙΝΑΚΗΣ ΙΩΑΝΝΗΣ**
ΜΑΤΣΑΤΣΙΝΗΣ ΝΙΚΟΛΑΟΣ
ΤΣΑΦΑΡΑΚΗΣ ΣΤΕΛΙΟΣ

ΧΑΝΙΑ: 2018

ΠΕΡΙΛΗΨΗ

Στην παρούσα εργασία πραγματοποιήθηκε η μοντελοποίηση και η επίλυση του προβλήματος χρονοπρογραμματισμού εργασιών συνεχούς ροής αντιμετάθεσης(permutation flowshop scheduling problem) με την βοήθεια μιας καινοτόμου παραλλαγής του αλγορίθμου της πυγολαμπίδας (Hybrid firefly algorithm).

Αρχικά σχεδιάστηκε το πρόβλημα χρονοπρογραμματισμού εργασιών συνεχούς ροής αντιμετάθεσης και επιλύθηκε σε περιβάλλον matlab ώστε να βρεθεί μια αρχική λύση και ο τελικός χρόνος που απαιτείται ώστε να ολοκληρωθεί μια σειρά εργασιών.

Στη συνέχεια εφαρμόστηκαν αλγόριθμοι τοπικής αναζήτησης (1-0 relocate, 1-1 exchange, 2-opt) μεμονωμένα ο καθένας αλλά και σαν συνδυασμός αυτών σε δύο εκδοχές του αλγορίθμου μεταβλητής αναζήτησης γειτονιάς (variable neighborhood search) ή VNS, ώστε να βρεθεί μια νέα λύση του προβλήματος εκτός από την υπάρχουσα αρχική.

Το βασικό μέρος της παρούσας εργασίας αποτελεί η μοντελοποίηση και η επίλυση του μεθευρετικού αλγορίθμου της πυγολαμπίδας. Πραγματοποιήθηκε μια νέα παραλλαγή του αλγορίθμου η οποία διαφοροποιεί την βασική εξίσωση του αλγορίθμου καθώς διαπιστώθηκε ότι η συγκεκριμένη αλλαγή βελτιώνει πολύ τα τελικά αποτελέσματα.

Ο κατάλληλος συνδυασμός όλων των παραπάνω αλγορίθμων-μεθόδων εφαρμόστηκε σε διαθέσιμα παραδείγματα και συνετέλεσε στην εξαγωγή σχετικών αποτελεσμάτων που συγκρίνονται με τα διαθέσιμα της βιβλιογραφίας.

Από την συγκεκριμένη μεταπτυχιακή διατριβή έχουν εκπονηθεί και τα παρακάτω άρθρα:

1. A. Taxidou, I. Karafyllidis, D. Trachanatzi, M. Rigakis, M. Marinaki and Y. Marinakis (2018), “A firefly algorithm for the permutation flowshop scheduling problem”, **Proceedings of the 7th International Symposium & 29th National Conference on Operational Research**, 137-141, 14-16 June 2018, Chania, Greece.
2. A. Taxidou, I. Karafyllidis, M. Marinaki, Y. Marinakis and A. Migdalas (2018), “A hybrid firefly - VNS algorithm for the permutation flowshop scheduling problem”, **6th International Conference on Variable Neighborhood Search**, 4-7 October 2017, Sithonia, Chalkidiki, Greece.

ABSTRACT

In this Thesis, modeling and solution of Permutation Flowshop Scheduling Problem was carried out with the aid of an innovative version of the Firefly Algorithm.

Initially, the Permutation Flowshop Scheduling Problem was designed and solved in a matlab environment in order to find an initial solution and the final time needed to complete a series of tasks.

Then local search algorithm (1-0 relocate, 1-1 exchange, 2-opt) algorithms were applied individually, but also as a combination of these in two versions of the variable neighborhood search algorithm or VNS to find a new solution to the problem in addition to the existing one.

The main part of this thesis is the modeling and solving of the Metaheuristic algorithm of Firefly. A new variation of the algorithm has been made which differentiates the basic equation of the algorithm as it has been found that this change greatly improves the final results.

A suitable combination of the above algorithms-methods was applied to available examples and contributed to the extraction of relative results compared to the available results from the literature.

The following papers have also been published :

3. A. Taxidou, I. Karafyllidis, D. Trachanatzi, M. Rigakis, M. Marinaki and Y. Marinakis (2018), “A firefly algorithm for the permutation flowshop scheduling problem”, **Proceedings of the 7th International Symposium & 29th National Conference on Operational Research**, 137-141, 14-16 June 2018, Chania, Greece.
4. A. Taxidou, I. Karafyllidis, M. Marinaki, Y. Marinakis and A. Migdalas (2018), “A hybrid firefly - VNS algorithm for the permutation flowshop

scheduling problem", **6th International Conference on Variable Neighborhood Search**, 4-7 October 2017, Sithonia, Chalkidiki, Greece.

ΕΥΧΑΡΙΣΤΙΕΣ

Θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή μου Μαρινάκη Ιωάννη για την αμέριστη συμπαράσταση και καθοδήγηση που μου παρείχε καθόλη τη διάρκεια συγγραφής της παρούσας εργασίας.

Θα ήθελα επίσης να ευχαριστήσω τη σύζυγο μου για τη στήριξη και την υπομονή που έδειξε.

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΕΦΑΛΑΙΟ 1^{ο} ΕΙΣΑΓΩΓΗ

1.1. Στόχος – Σκοπός της Εργασίας

Σκοπός της μελέτης αυτής είναι να περιγραφεί, μοντελοποιηθεί και να επιλυθεί το πρόβλημα του χρονοπρογραμματισμού εργασιών συνεχούς ροής μέσω κατάλληλων και αποτελεσματικών αλγορίθμων. Επιπλέον να γίνει εφαρμογή των παραπάνω σε διαθέσιμα παραδείγματα της βιβλιογραφίας. Η παραλλαγή ήδη υπαρχόντων αλγορίθμων αναμένεται να δημιουργήσει νέους αλγορίθμους οι οποίοι δύναται να αποδώσουν καλύτερες εφικτές λύσεις στο δοθέν πρόβλημα.

Αναδύεται πληθώρα ερωτημάτων γύρω από τον τρόπο επιλογής αλγορίθμων που επιλύουν προβλήματα χρονοπρογραμματισμού εργασιών και πως αυτοί εφαρμόζονται στην πράξη. Γίνεται προσπάθεια όλα αυτά τα ερωτήματα να απαντηθούν και επιπλέον να διαπιστωθεί ποιες θα είναι οι μελλοντικές επεκτάσεις που θα προκύψουν μέσω της παρούσας εργασίας.

Μελλοντικός στόχος αποτελεί η επιστημονική ενασχόληση με τέτοιου είδους προβλήματα, η επίλυση των οποίων να αποτελεί καινοτόμο τρόπο αντιμετώπισης του με στόχο να εξάγονται βέλτιστα αποτελέσματα. Πιο συγκεκριμένα η τροποποίηση υπαρχόντων αλγορίθμων και η δημιουργία νέων οι οποίοι θα είναι γρήγοροι στην απόκριση και αποτελεσματικοί αποτελεί έναν μελλοντικό επιστημονικό στόχο.

1.2. Μεθοδολογία

Στην παρούσα εργασία αρχικά μελετήθηκε το πρόβλημα του χρονοπρογραμματισμού εργασιών μέσω της διαθέσιμης βιβλιογραφίας. Έγινε επιλογή του θέματος ώστε να επιλυθεί το πρόβλημα του χρονοπρογραμματισμού εργασιών συνεχούς ροής (flow-shop scheduling problem). Τα δεδομένα του προβλήματος έχουν μελετηθεί στην βιβλιογραφία και επιλέχθηκαν ώστε να δύναται η σύγκρισή αυτών με τα αποτελέσματα που θα εξαχθούν από την παρούσα εργασία. Η μοντελοποίηση του προβλήματος αυτού αποτελεί και τον αντικειμενικό στόχο του συνολικού προβλήματος.

Ακολούθως πραγματοποιήθηκε η επιλογή κατάλληλων αλγορίθμων που να μπορούν να επιλύουν το δεδομένο πρόβλημα. Ο συνδυασμός των επιλεγόμενων αλγορίθμων πραγματοποιήθηκε με κριτήριο την απόδοση τους και της μη επιλογής όλων μαζί σε προηγούμενη εργασία. Επιλέχθηκαν αλγόριθμοι ευρετικοί τοπικής αναζήτησης, ένας κλασικός μεθευρετικός αλγόριθμος για να τους συνδυάσει ο αλγόριθμος μεταβλητής γειτονιάς αναζήτησης και ο βασικός αλγόριθμος αποτελεί έναν αποδοτικό μεθευρετικό αλγόριθμο που ανήκει στην κατηγορία των εξελικτικών αλγορίθμων και πιο συγκεκριμένα στην υποκατηγορία των αλγορίθμων εμπνευσμένων από την φύση, τον αλγόριθμο της πυγολαμπίδας (firefly algorithm). Επιπλέον πραγματοποιήθηκαν συνδυασμοί των ανωτέρω με στόχο να βελτιωθούν τα σχετικά εξαγόμενα αποτελέσματα.

Πραγματοποιήθηκε η μαθηματική περιγραφή τους και ερμηνεία τους, η προσεκτική επιλογή των παραμέτρων που χρησιμοποιούνται και όπου κρίθηκε ωφέλιμο πραγματοποιήθηκαν αλλαγές στις μαθηματικές εξισώσεις των αλγορίθμων δημιουργώντας παρόμοιους υβριδικούς αλγορίθμους.

Τέλος τα εξαγόμενα αποτελέσματα συγκρίνονται με τα διαθέσιμα της βιβλιογραφίας.

Όλα τα ανωτέρω σχεδιάστηκαν και επιλύθηκαν σε περιβάλλον matlab.

1.3. Σύνοψη εργασίας

Η δομή της παρούσας εργασίας είναι η ακόλουθη:

Στο πρώτο κεφάλαιο έγινε μια εισαγωγή παρουσιάζοντας το σκοπό της διπλωματικής εργασίας καθώς και της μεθοδολογίας που ακολουθήσαμε.

Στο δεύτερο κεφάλαιο γίνεται ανάλυση του χρονοπρογραμματισμού εργασιών καθώς και τις κατηγορίες προβλημάτων που περιλαμβάνει.

Στο τρίτο κεφάλαιο γίνεται παρουσίαση των ευρετικών – μεθευρετικών αλγορίθμων .

Στο τέταρτο κεφάλαιο γίνεται μια περαιτέρω ανάλυση του υβριδικού αλγόριθμου που χρησιμοποιήσαμε καθώς και του VNS algorithm.

Στο πέμπτο κεφάλαιο γίνεται παρουσίαση των αποτελεσμάτων του αλγόριθμου καθώς και τα συμπεράσματα που προκύπτουν από αυτά.

ΚΕΦΑΛΑΙΟ 2^{ο} ΧΡΟΝΟΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ΕΡΓΑΣΙΩΝ

2.1. Γενικά

Στη σύγχρονη κοινωνία, οι Βιομηχανικές Μονάδες έρχονται αντιμέτωπες με συνεχώς αυξανόμενα προβλήματα και περιορισμούς που αφορούν τη παραγωγική διαδικασία λόγω της πολυπλοκότητας τους συστήματος παραγωγής αλλά και της δυναμικότητας που παρουσιάζουν. Τέτοια προβλήματα μπορεί να αφορούν στον χρόνο κατεργασίας που χρειάζονται οι μηχανές, στην κατανάλωση ηλεκτρικού ρεύματος, στο ανθρώπινο δυναμικό, στην πρώτη ύλη καθώς και στους περιορισμούς που θέτει κάθε φορά ο πελάτης. Γι' αυτό το λόγο ο χρονοπρογραμματισμός των εργασιών (job-shop scheduling) αποτελεί βασικό κομμάτι της παραγωγικής διαδικασίας ακόμα και από την σχεδίαση. Ο χρονοπρογραμματισμός των εργασιών είναι η κατανομή των εργασιών σε Μηχανές με βασικό σκοπό την αποτελεσματική ολοκλήρωση τους σε εύλογο χρονικό διάστημα [1].

Επιπλέον λόγω της ανταγωνιστικότητας που υπάρχει τον βιομηχανικό τομέα, οι ιδιωτικές και οι δημόσιες υπηρεσίες αποσκοπώντας να αποκομίσουν το μέγιστο δυνατό κέρδος, θέτουν πολλαπλούς στόχους ώστε να πετύχουν τον στόχο τους. Κάποιες φορές αυτοί οι στόχοι είναι αντικρουόμενοι μεταξύ τους άρα δύναται να αντιμετωπίζουν προβλήματα στην λήψη αποφάσεων. Το περιβάλλον αυτό γεννά μια αβεβαιότητα ως προς την λήψη των αποφάσεων και οι παράμετροι που πρέπει να συνυπολογιστούν πολλές φορές παραμένουν άγνωστοι.

Για τους παραπάνω λόγους είναι φανερό ότι ο σωστός προγραμματισμός των εργασιών κάθε εταιρίας, όπως και η σωστή

στρατηγική που θα οδηγήσει σε λήψη άρτιων αποφάσεων αποτελεί ένα κρίσιμο ζήτημα. Όλοι οι παράμετροι που επηρεάζουν την αποδοτικότητα και την κερδοφορία των επιχειρήσεων, πρέπει να διαχειριστούν με τέτοιο τρόπο και από άρτια καταρτισμένους επιστήμονες ώστε να επιτυγχάνεται το μέγιστο κέρδος που θα κάνει μια επιχείρηση- εταιρία πετυχημένη.

2.2. Πρόβλημα Χρονοπρογραμματισμού Συνεχούς Ροής - Flow Shop Scheduling Problem

Μια κατηγορία του προβλήματος χρονοπρογραμματισμού εργασιών είναι το πρόβλημα χρονοπρογραμματισμού εργασιών συνεχούς ροής (flow-shop scheduling problem) που προτάθηκε από τον Johnson [5]. Συγκεκριμένα υφίσταται όταν όλες οι *n* εργασίες επεξεργάζονται σε όλες τις *m* μηχανές με την ίδια σειρά. Ωστόσο υπάρχουν ορισμένα πρακτικά προβλήματα τα οποία εστιάζουν σε δύο σημαντικές αποφάσεις

- Η διαδοχική σειρά των εργασιών που θα επεξεργαστούν σειριακά από δύο ή περισσότερες μηχανές
- Ο προγραμματισμός φόρτωσης της μηχανής, ο οποίος προσδιορίζει τη διαδοχική διάταξη των χρόνων εκκίνησης και τερματισμού σε κάθε μηχανή για διάφορες εργασίες.

Ωστόσο προτιμάται συνηθέστερα η σειρά εργασίας και τα σχετικά χρονοδιαγράμματα φόρτωσης μηχανών που επιτρέπουν την ελαχιστοποίηση του συνολικού χρόνου επεξεργασίας, του μέσου χρόνου ροής, της μέσης απόκλισης και της μέσης καθυστέρησης. Για να μοντελοποιηθεί το flow-shop πρέπει να υπάρχουν *m* μηχανές διατεταγμένες σε σειρά στις οποίες επεξεργάζονται *n* εργασίες. Κάθε μία από τις *n* εργασίες απαιτεί *m* διαφορετικές λειτουργίες που εκτελούνται η κάθε μία σε διαφορετική

μηχανή. Οι μηχανές αριθμούνται από 1,2,3….,*m* και οι λειτουργίες των εργασιών j θα αριθμούνται σε *(1,j), (2,j), (3,j),.... (m,j)* [2].

Κάθε εργασία πρέπει να υποστεί επεξεργασία από κάθε μηχανή και σε μια προκαθορισμένη σειρά καθιστώντας το διάγραμμα ροής εργασιών αμφίδρομο. Θεωρούμε ότι μία μηχανή δεν μπορεί να επεξεργαστεί ταυτόχρονα πάνω από μία εργασία και ότι κάθε εργασία εκτελείται μόνο σε μία μηχανή. Θεωρούμε επίσης ότι κάθε εργασία δεν επισκέπτεται την ίδια μηχανή πάνω από μία φορά και ότι οι χρόνοι εκκίνησης και προετοιμασίας των μηχανών περιλαμβάνονται στους χρόνους επεξεργασίας κάθε εργασίας από μία μηχανή. Τέλος, ότι οι εργασίες που υφίστανται επεξεργασία από κάθε μηχανή δεν μπορούν να διακοπούν [4].

Επομένως, σε ένα σύστημα χρονοπρογραμματισμού εργασιών συνεχούς ροής (flow-shop scheduling problem) για να αντιστοιχήσουμε τις λειτουργίες σε κάθε μηχανή, μπορούμε να αριθμήσουμε τις μηχανές από 1,2, ..., *m* και τις λειτουργίες κάθε εργασίας *n* ως εξής: (1,n), (2,n), ... , (m,n) . Επιπλέον ο χρόνος επεξεργασίας κάθε εργασίας *n* στη μηχανή *m* μπορεί να συμβολιστεί ως $\mathrm{p_{mn}}$. Για παράδειγμα, το $\mathrm{p_{23}}$ συμβολίζει το χρόνο που χρειάστηκε η μηχανή 2 να επεξεργαστεί την εργασία 3 [2].

Ο αντικειμενικός σκοπός του προβλήματος είναι η εύρεση μιας σειράς εργασιών που θα ελαχιστοποιεί τον συνολικό χρόνο επεξεργασίας ή αλλιώς makespan [5]. Ως makespan Cmax ορίζεται ο συνολικός χρόνος που απαιτείται για να επεξεργαστούν όλες οι εργασίες από όλες τις μηχανές. Δηλαδή, το makespan ισούται με τον χρόνο περάτωσης των εργασιών στην τελευταία μηχανή:

Cmax = max($\mathrm{R_{iM}}$)

όπου $\mathrm{R_{iM}}$ ο χρόνος επεξεργασίας κάθε εργασίας *i* στην τελευταία μηχανή *M*.

Όπως είναι φυσικό δεν μπορούν να υπολογιστούν όλες οι πιθανές λύσεις ώστε να ευρεθεί αυτή που ελαχιστοποιεί το makespan, δεδομένου ότι ο αριθμός όλων των δυνατών λύσεων του προβλήματος $(n!)^m$ είναι διαφορετικοί χρονοπρογραμματισμοί εργασιών, ένα μέγεθος που οδηγεί σε υπολογιστικό αδιέξοδο.

2.3. Πρόβλημα Χρονοπρογραμματισμού Συνεχούς Ροής Αντιμετάθεσης-The Permutation Flow Shop Scheduling Problem

Αν στην περίπτωση που στο σύστημα χρονοπρογραμματισμού συνεχούς ροής προσθέσουμε την προϋπόθεση ότι όλες οι εργασίες πρέπει να εκτελεστούν με την ίδια ακριβώς σειρά από κάθε μία από τις k μηχανές καταλήγουμε στο Πρόβλημα Χρονοπρογραμματισμού Συνεχούς Ροής Αντιμετάθεσης (Permutation Flow Shop Problem)

Το συγκεκριμένο πρόβλημα ακολουθεί τους κάτωθι ισχυρισμούς:

- Όλες οι εργασίες είναι έτοιμες για επεξεργασία τη χρονική στιγμή 0.
- Οι μηχανές είναι συνεχώς διαθέσιμες από τη στιγμή 0 και έπειτα (δεν υπάρχουν βλάβες).
- Κάθε χρονική στιγμή, μία μηχανή μπορεί να επεξεργαστεί το πολύ μία εργασία και κάθε εργασία μπορεί να υφίσταται επεξεργασία το πολύ από μία μηχανή.
- Δεν επιτρέπονται προτεραιότητες. Αυτό σημαίνει ότι από τη στιγμή που μία μηχανή επεξεργάζεται μία εργασία, αυτή δε μπορεί να διακοπεί για να επεξεργαστεί άλλη εργασία με υψηλότερη προτεραιότητα.
- Μόνο προγραμματισμοί αντιμετάθεσης επιτρέπονται. Δηλαδή όλες οι εργασίες έχουν την ίδια σειρά σε όλες τις μηχανές [4].

Ο καθορισμός μιας σειράς εργασιών (job permutation) μπορεί να συμβολιστεί ως εξής $\pi^* = \pi_1^* + \pi_2^*, \ldots, \pi_n^*$όπου οι *n* εργασίες θα πρέπει να επεξεργαστούν με την ίδια σειρά από τις *m* μηχανές. Ορίζουμε με $C(\pi_j, k)$τον χρόνο ολοκλήρωσης της εργασίας π_j στη μηχανή *k* και υπολογίζεται ως ακολούθως :

$C(\pi_1, 1) = p_{\pi_1,1}$

$C(\pi_j, 1) = C(\pi_{j-1}, 1) + p_{\pi_j,1}$, j=2,…,n

$C(\pi_1, k) = C(\pi_1, k-1) + p_{\pi_1,k}$, k=2,…,m

$C(\pi_j, k) = \max \{ C(\pi_{j-1}, k), C(\pi_j, k-1) + p_{\pi_j,k} \}$, j=2,…,n, k=2,…,m

Το makespan ορίζεται ως ο χρόνος περάτωσης της τελευταίας εργασίας π_n^* στη τελευταία μηχανή m:

$C_{max}(\pi) = C(\pi_n, m)$

Ο στόχος είναι να βρεθεί η βέλτιστη αντιμετάθεση π^* στο σύνολο όλων των αντιμεταθέσεων Π ώστε να ισχύει:

$C_{max}(\pi^*) \leq C(\pi_n, m)$ για κάθε αντιμετάθεση π που ανήκει στο Π [5].

Η υπολογιστική πολυπλοκότητα του Προβλήματος Χρονοπρογραμματισμού Συνεχούς Ροής Αντιμετάθεσης έχει αποδειχτεί ότι ανήκει στην κατηγορία των μη πολυωνυμικών δύσκολων **(NP-Hard)** προβλημάτων , που σημαίνει ότι όχι μόνο δεν είναι γνωστός κάποιος εφικτός αλγόριθμος για την επίλυση του προβλήματος αλλά είναι απίθανο να υπάρχει ένας τέτοιος αλγόριθμος. Γι αυτό το λόγο ένας αριθμός ευρετικών και μεθευρετικών αλγορίθμων έχει αναπτυχθεί στο παρελθόν για την επίλυση του :

- Επαναληπτικός Αλγόριθμος Απληστίας (Iterated Greedy Algorithm) [14]
- Υβριδικός Γενετικός Αλγόριθμος (Hybrid Genetic Algorithms) [3][13][15][16][19]
- Αλγόριθμος Περιορισμένης Αναζήτησης (Tabu Search) [20][21]
- Αλγόριθμος διαφορικής εξέλιξης (Differential Evolution) [17]
- Υβριδικός αλγόριθμος βελτιστοποίησης σμήνους σωματιδίων (Hybrid Particle Swarm Optimization) [23][24]
- Αλγόριθμος βελτιστοποίησης αποικίας μυρμηγκιών (Ant Colony Optimization Algorithms) [25][26]
- Υβριδικός αλγόριθμος τεχνητής αποικίας μελισσών (Hybrid Artificial Bee Colony Algorithm)[27]

ΚΕΦΑΛΑΙΟ 3^{ο} ΕΥΡΕΤΙΚΟΙ-ΜΕΘΕΥΡΕΤΙΚΟΙ ΑΛΓΟΡΙΘΜΟΙ

3.1. Ευρετικοί Αλγόριθμοι

Ένας τεράστιος αριθμός πιθανών λύσεων που παρουσιάζεται στα προβλήματα επιχειρησιακής έρευνας καλείται να αξιολογηθεί σε ένα εύλογο χρονικό διάστημα. Αυτό γίνεται ολοένα και δυσκολότερο καθώς όσο αυξάνει το μέγεθος του προβλήματος, αυξάνεται και ο χρόνος που απαιτείται για να βρεθεί η λύση του, καθιστώντας το πρακτικά αδύνατο να βρεθεί σε λογικό χρόνο. Το πρόβλημα αυτό ανήκει στην κατηγορία των μη πολυωνυμικων δύσκολων (NP-Hard) προβλημάτων [9], που σημαίνει ότι όχι μόνο δεν είναι γνωστός κάποιος εφικτός αλγόριθμος για την επίλυση του προβλήματος αλλά είναι απίθανο να υπάρχει ένας τέτοιος αλγόριθμος.

Αυτά τα προβλήματα χρειάζονται συγκεκριμένους προσεγγιστικούς αλγόριθμους οι οποίοι επιχειρούν μέσα από συγκεκριμένα βήματα και τεχνικές να εντοπίσουν το ολικό βέλτιστο ή να το πλησιάσουν. Όσο μικρότερη είναι η απόκλιση από το ολικό βέλτιστο τόσο αυξάνει η ποιότητα της λύσης.

Μια λύση ενός ευρετικού αλγορίθμου γίνεται αποδεκτή αν ικανοποιεί ορισμένα κριτήρια όπως η ποιότητα της λύσης, δηλαδή η απόκλισή της από τη βέλτιστη, η ευκολία απόκτησης μιας λύσης, η λογική πάνω στην οποία στηρίζονται οι κανόνες του ευρετικού αλγορίθμου που χρησιμοποιήθηκαν για να οδηγηθούμε στη λύση. Για κάθε πρόβλημα βελτιστοποίησης δεν υπάρχει μονάχα ένας ευρετικός αλγόριθμος που να δίνει τη βέλτιστη λύση, αλλά έχουν αναπτυχθεί αρκετοί αλγόριθμοι οι οποίοι συγκρινόμενοι μεταξύ τους, οδηγούν ολοένα και σε καλύτερες λύσεις.

Η επίλυση ενός προβλήματος βελτιστοποίησης γίνεται δυσκολότερη όσο αυξάνει το μέγεθος του προβλήματος, καθιστώντας πρακτικά αδύνατο να βρεθεί μια ολικά βέλτιστη λύση

Υπάρχουν οι εξής κατηγορίες ευρετικών αλγορίθμων

- **Αλγόριθμοι απληστίας**

  Σε αυτή την κατηγορία αλγορίθμων πραγματοποιείται η προσπάθεια να βρεθεί μια εφικτή λύση του προβλήματος, αλλά τις περισσότερες φορές απαιτούν μεγάλο χρονικό διάστημα, διότι είναι μυωπικοί αλγόριθμοι.
- **Προσεγγιστικοί αλγόριθμοι**

  Σε αυτό του είδους τους αλγορίθμους γίνεται προσπάθεια να επιλυθεί ένα δοθέν πρόβλημα χρησιμοποιώντας επιπλέον πληροφορία
- **Αλγόριθμοι τοπικής αναζήτησης**

  Σε αυτό το είδος των αλγορίθμων οι αλγόριθμοι ξεκινάνε από μια αρχική εφικτή λύση και προσπαθούν με κάποια μέθοδο αναζήτησης στην γειτονιά της αρχικής λύσης να βελτιώσουν την αρχική αυτή λύση.

  Η βασική διαφορά των τριών ανωτέρω κατηγοριών είναι ότι οι δυο πρώτες κατηγορίες χρησιμοποιούνται για την παραγωγή μιας αρχικής λύσης, ενώ η τρίτη κατηγορία αλγορίθμων προσπαθεί να βελτιώσει μια ήδη υπάρχουσα λύση [6].

Η διαδικασία εύρεσης μιας αρχικής λύσης είναι μια πολύ σημαντική διαδικασία για την αποτελεσματικότητα του αλγορίθμου. Αν η αρχική λύση είναι καλή τότε είναι πολύ πιθανόν ο αλγόριθμος να οδηγηθεί σε σύγκλιση. Αν υπάρξει κακή αρχική λύση τότε θεωρητικά θα αργήσει να συγκλίνει σε κάποια καλή λύση. Για να δημιουργηθεί μια καλή λύση υπάρχουν δυο τρόποι. Ο πρώτος είναι να πραγματοποιηθεί η χρήση μιας τυχαιοποιημένης διαδικασίας κατάλληλης και προσαρμοσμένης στο πρόβλημα που επιλύεται

κάθε φορά. Ο δεύτερος τρόπος είναι με τη χρήση ενός αλγορίθμου απληστίας. Ποιον από τους δύο τρόπους θα επιλέξουμε εξαρτάται από το είδους του προβλήματος που επιλύεται κάθε φορά. Για παράδειγμα αν το πρόβλημα είναι συνεχές τότε καλύτερα θα ήταν να δημιουργήσουμε αρχικές λύσεις με τυχαίο τρόπο. Σε περίπτωση προβλήματος συνδυαστικής βελτιστοποίησης δίνουν καλύτερα αποτελέσματα οι αρχικές λύσεις που προέρχονται από αλγορίθμους απληστίας. Εάν στην επίλυση του εκάστοτε προβλήματος ο αλγόριθμος χρησιμοποιεί έναν πληθυσμό από λύσεις για να βελτιώσει τις λύσεις του κατά την διάρκεια των προκαθορισμένων επαναλήψεων τότε οι αρχικές λύσεις θα δημιουργηθούν είτε με τυχαίο τρόπο είτε με χρήση διαφορετικών αλγορίθμων απληστίας είτε συνδυασμό των παραπάνω μεθόδων.

Στην παρούσα εργασία θα χρησιμοποιήσουμε αλγορίθμους τοπικής αναζήτησης. Η τοπική αναζήτηση έχει αποδειχθεί πολύ επιτυχημένη μέθοδος όταν εφαρμόζεται σε πολλά προβλήματα συνδυαστικής βελτιστοποίησης. Η φιλοσοφία των αλγορίθμων αυτών είναι η εξής:

Αρχικά δίνεται ένα πρόβλημα βελτιστοποίησης και ένα κόστος του προβλήματος. Το κόστος μπορεί να είναι διάφορα μεγέθη ανάλογα το είδους του προβλήματος (χρόνος, χρηματικό κόστος, κτλ) . Στην γειτονιά της αρχικής λύσης εφαρμόζεται αναζήτηση ώστε να βρεθεί μια καλύτερη λύση από την αρχική αν υπάρχει, δηλαδή να δίνει μικρότερο κόστος. Για όσες επαναλήψεις υπάρχει καλύτερη λύση αντικαθιστούμε την υπάρχουσα με την καλύτερη και συνεχίζεται η αναζήτηση μέχρι το σημείο που θα συναντήσουμε κάποιο τοπικό ελάχιστο και η λύση που θα έχουμε δεν θα βελτιώνεται παραπάνω.

Επιγραμματικά η διαδικασία

Τοπική αναζήτηση

Δοθέντος μιας αρχικής λύσης του προβλήματος s

Do while να βρεθεί μια βελτιωμένη λύση

s=improve(s)

return s

end

Το πλεονέκτημα της μεθόδου αυτής είναι ότι μπορεί να εκτελείται από πολλά διαφορετικά αρχικά σημεία και να επιλέγεται το καλύτερο ως βέλτιστη λύση. Για να διαπιστωθεί ότι η τοπική αναζήτηση είναι επιτυχής θα πρέπει να γίνει σωστή επιλογή γειτονιάς που θα εφαρμοστεί η αναζήτηση [6].

3.2. Μεθευρετικοί αλγόριθμοι βασισμένοι στη γειτονιά αναζήτησης

Οι μεθευρετικοί αλγόριθμοι, είναι αλγόριθμοι ειδικά σχεδιασμένοι ώστε να βοηθούν την αναζήτηση να ξεφύγει από κάποιο τοπικό ελάχιστο. Ο διαχωρισμός των μεθευρετικών αλγορίθμων σχετίζεται με τον αριθμό των λύσεων που χρησιμοποιούν. Κάποιοι αλγόριθμοι χρησιμοποιούν μία λύση και πραγματοποιούν αναζήτηση στην γειτονιά αυτής της λύσης ενώ υπάρχουν άλλοι που χρησιμοποιούν έναν πληθυσμό από λύσεις και πραγματοποιούν αναζήτηση σε όλο τον χώρο των λύσεων. Υπάρχουν τέσσερις κατηγορίες τέτοιων αλγορίθμων και η διαφοροποίηση τους σχετίζεται με τον τρόπο που χρησιμοποιούν για να αποφύγουν το τοπικό ελάχιστο [8].

α) Επαναληπτικές διαδικασίες που αρχίζουν από διαφορετικές αρχικές λύσεις

β) Αλγόριθμοι που δέχονται γειτονικές κινήσεις που δεν βελτιώνουν την λύση, που υπό προϋποθέσεις μπορεί να γίνει δεκτή

γ) Αλγόριθμοι που αλλάζουν την γειτονιά αναζήτησης. Όταν κολλήσουν σε κάποιο τοπικό ελάχιστο αλλάζουν τον αλγόριθμο που χρησιμοποιούν

δ) Αλγόριθμοι που αλλάζουν την αντικειμενική συνάρτηση ή κάποια από τα δεδομένα του προβλήματος [7].

Στην παρούσα εργασία για την επίλυση του προβλήματος του χρονοπρογραμματισμού εργασιών συνεχούς ροής χρησιμοποιήθηκε μια παραλλαγή του αλγορίθμου της πυγολαμπίδας.

3.3. Αλγόριθμος Πυγολαμπίδας (Firefly Algorithm)

ΓΕΝΙΚΑ

Ο Αλγόριθμός της Πυγολαμπίδας είναι ένας μεθευρετικός αλγόριθμός εμπνευσμένος από τη φύση και αναπτύχθηκε από τον Xin-She Yang το 2007 [1]. Ακολουθεί την ίδια αρχή με τους αλγόριθμους που στηρίζονται στην ευφυΐα του σμήνους όπως για παράδειγμά τον Αλγόριθμο Βελτιστοποίησης Σμήνους Σωματιδίων. Στη φύση υπάρχουν περίπου 2000 είδη πυγολαμπίδων και τα περισσότερα από αυτά παράγουν μια σύντομη και ρυθμική λάμψη. Αυτή η λάμψη παράγεται έπειτα από μια διαδικασία βιοφωταύγειας και χρησιμεύει ως μέσο επικοινωνίας μεταξύ των πυγολαμπίδων για το ζευγάρωμα όσο και στη προειδοποίηση για κάποιο επικείμενο κίνδυνο. Βιοφωταύγεια ή βιοφωτισμός χαρακτηρίζεται η δημιουργία φωτός σε διάφορα μήκη κύματος που εκπέμπεται από διάφορους ζώντες οργανισμούς, που συχνά καλείται εσφαλμένα φωσφορισμός [10].

Για την ανάπτυξη και περιγραφή του αλγορίθμου ισχύουν τρία χαρακτηριστικά:

1. Οι πυγολαμπίδες έλκονται μεταξύ τους ανεξάρτητα από το φύλο τους, που σημαίνει ότι δεν απαιτείται να εφαρμοστεί κάποιος τελεστής μετάλλαξης
2. Το μοίρασμα των πληροφοριών ανάμεσα στις πυγολαμπίδες είναι ανάλογο με την ελκυστικότητα τους, δηλαδή είναι ανάλογο με την απόσταση που βρίσκονται, έτσι όσο πιο κοντά είναι δύο πυγολαμπίδες τόσο περισσότερες πιθανότητες είναι να υπάρξει έλξη μεταξύ τους. Αν δεν υπάρχει πυγολαμπίδα που είναι πιο

λαμπερή από την πυγολαμπίδα που εξετάζουμε τότε η συγκεκριμένη πυγολαμπίδα κινείται τυχαία στο χώρο.

3. Η λάμψη μιας πυγολαμπίδας εξαρτάται από την τιμή της αντικειμενικής συνάρτησης. Για προβλήματα μεγιστοποίησης όσο μεγαλύτερη είναι η τιμή της αντικειμενικής συνάρτησης τόσο πιο λαμπερή είναι η πυγολαμπίδα, αντίθετα σε προβλήματα ελαχιστοποίησης όσο πιο μικρή είναι η τιμή της αντικειμενικής συνάρτησης τόσο πιο λαμπερή είναι η πυγολαμπίδα [4][18].

Ο ψευδοκώδικάς όπως παρουσιάστηκε από τον Xin-She Yang φαίνεται στην παρακάτω εικόνα [1].

Firefly Algorithm

Objective function f ($\boldsymbol{x}$), $\boldsymbol{x} = (x_1, \ldots, x_d)^T$
Generate initial population of fireflies $\boldsymbol{x_i}$ (i $= 1, 2, \ldots.., n$)
Light intensity I_i at $\boldsymbol{x_i}$ is determined by $f(\boldsymbol{x_i})$
Define light absorption coefficient γ
while (t < MaxGeneration)
for I = 1 : n all n fireflies
  for j = 1 : n all n fireflies (inner loop)
  if ($I_i < I_j$), Move firefly i towards j ; **end if**
    Vary attractiveness with distance r via exp [-γr]
    Evaluate new solutions and update light intensity
  end for j
end for i
Rank the fireflies and find the current global best g_*
end while
Postprocess results and visualization

Πίνακας 1 : Ψευδοκώδικας όπως παρουσιάστηκε από τον Yang

3.4. Μαθηματική διατύπωση του αλγορίθμου

Στον αλγόριθμο πυγολαμπίδας υπάρχουν δύο σημαντικές παράμετροι: Η μεταβολή της λάμψης της πυγολαμπίδας και η διατύπωση της ελκυστικότητας. Ωστόσο για λόγους ευκολίας μπορούμε να υποθέσουμε ότι η ελκυστικότητα μιας πυγολαμπίδας είναι ανάλογη με την ένταση του φωτός που εκπέμπεται προς τις άλλες πυγολαμπίδες οπότε εξαρτάται από την τιμή της αντικειμενικής συνάρτησης.

Η λάμψη I είναι ανάλογη της αντικειμενικής συνάρτησης f στο σημείο x με I(x) ∝ f(x). Η λάμψη ποικίλει και είναι αντιστρόφως ανάλογη με την απόσταση μεταξύ των δύο πυγολαμπίδων και είναι ίση με:

$I = Is / d^2$

Όπου, Is είναι η λάμψη στη πηγή.

Η απόσταση μεταξύ δύο πυγολαμπίδων συμβολίζεται με $r_j = d(x_i, x_j)$ έτσι η λάμψη δίνεται από την εξίσωση (η οποία στην απλή της μορφή είναι μονότονη και εκθετική

$$I = I_0 e^{-\gamma r^2}$$

όπου το I_0 είναι η αρχική λάμψη και το γ είναι η σταθερά που δείχνει την απορροφητικότητα της λάμψης. Το r_{ij} είναι η απόσταση μεταξύ δύο πυγολαμπίδων και είναι ίση με:

$$r_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$$

Η ελκυστικότητα είναι ανάλογη της λάμψης (I) και είναι ίση με:

$$B = \beta_0 e^{-\gamma r^2}$$

όπου β_0 η ελκυστικότητα στο r = 0

Η κίνηση μιας πυγολαμπίδας i που βρίσκεται στην θέση x_i και έλκεται από μια άλλη πυγολαμπίδα j που λάμπει περισσότερο από την i και βρίσκεται στην θέση x_j καθορίζεται από τη σχέση:

$$X_i (t + 1) = X_i (t) + \beta_0 e^{-\gamma r_{ij}^2} (X_j (t) – X_i (t)) \quad + \alpha \epsilon_t$$

όπου το πρώτο μέλος είναι η τρέχουσα θέση, το δεύτερο μέλος δείχνει πόσο θα δει μια πυγολαμπίδα μια πιο λαμπερή πυγολαμπίδα που βρίσκεται στην περιοχή της και το τρίτο μέλος είναι μια τυχαία κίνηση που κάνει μια πυγολαμπίδα όταν δεν υπάρχει καμιά πιο λαμπερή από αυτή στην περιοχή της. Η σταθερά α είναι ένας τυχαίος αριθμός.

Όσο το γ πλησιάζει στο μηδέν η λάμψη $\beta = \beta_0$ είναι σταθερή και δεν μεταβάλλεται από την απόσταση που σημαίνει ότι η λάμψη μιας πυγολαμπίδας μπορεί να είναι ορατή από όλες τις άλλες ενώ όσο το γ μεγαλώνει τότε η ελκυστικότητα της κάθε πυγολαμπίδας μειώνεται που σημαίνει ότι καμία πυγολαμπίδα δεν μπορεί να δει την άλλη και όλες οι πυγολαμπίδες κινούνται τυχαία στο χώρο [10][16].

3.5. Επίλυση του προβλήματος με χρήση υβριδικού αλγορίθμου της πυγολαμπίδας

Η επίλυση του σχετικού προβλήματος πραγματοποιήθηκε με 5 διαφορετικές παραλλαγές, ώστε να διαπιστωθεί πότε προκύπτουν τα καλύτερα αποτελέσματα σε διαθέσιμα παραδείγματα της βιβλιογραφίας.

Χρησιμοποιήθηκαν τρείς αλγόριθμοι τοπικής αναζήτηση, 1-0 relocate, 1-1 exchange, 2-opt οι οποίοι εφαρμόστηκαν ο καθένας ανεξάρτητα από τον

άλλον και επιπλέον δυο παραλλαγές του αλγορίθμου μεταβλητής γειτονιάς αναζήτησης (Variable Neighborhood Search, VNS).

Ολόκληρη η επίλυση του προγράμματος επιλύθηκε σε περιβάλλον matlab.

Αρχικά πραγματοποιήθηκε η εύρεση της αρχικής λύσης μέσω κατάλληλου αλγορίθμου και εξισώσεων του προβλήματος χρονοπρογραμματισμού εργασιών συνεχούς ροής. Δηλαδή, σαν αποτέλεσμα εξάγεται και η σειρά των εργασιών που πρέπει να πραγματοποιηθούν αλλά και ο συνολικός χρόνος που απαιτείται για αυτήν τη διαδικασία. Το πρόβλημα του χρονοπρογραμματισμού των εργασιών συνεχούς ροής αποτελεί την αντικειμενική συνάρτηση του προβλήματος μας δηλαδή την f(x).

Δεύτερο βήμα στην επίλυση του προβλήματος είναι η εύρεση μιας ακόμα λύσης στην γειτονιά της αρχικής λύσης. Αυτή η δυνητικά καλύτερη λύση προέρχεται κάθε φορά από τους αλγορίθμους τοπικής αναζήτησης όπως έχει προαναφερθεί. Αρχικά θεωρούμε σαν καλύτερη λύση την αρχική. Εάν η νέα λύση είναι καλύτερη, τότε είναι αυτή που θεωρούμε σαν καλύτερη λύση κοκ. Η διαδικασία συνεχίζεται μέχρι να ικανοποιηθεί κάποιο κριτήριο τερματισμού. Στην παρούσα εργασία έχουμε θέσει σαν κριτήριο τερματισμού έναν προκαθορισμένο αριθμό επαναλήψεων. Για να καθοριστεί αυτός ο αριθμός, πραγματοποιήθηκαν αρκετές δοκιμές ώστε να διαπιστωθεί ποιος είναι ο κατάλληλος συνδυασμός αριθμού επαναλήψεων, δηλαδή πότε θα προκύψει η καλύτερη δυνατή λύση.

Η σειρά που πραγματοποιήθηκε η τοπική αναζήτηση είναι η εξής:

α) 1-0 relocate -1-0 επανατοποθέτηση

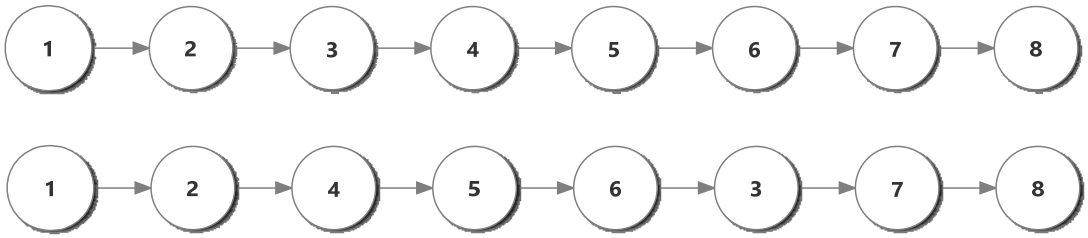
Σε αυτήν τη μέθοδο, επιλέγουμε τυχαία τη θέση - σειρά που πραγματοποιείται μια εργασία, την αφαιρούμε από τη θέση που βρίσκεται και την τοποθετούμε ανάμεσα σε δύο άλλες εργασίες. Ο αλγόριθμος αυτός βασίζεται στην ιδέα της διαγραφής ενός πελάτη από τη θέση του και την

επανατοποθέτηση του σε μια άλλη θέση η οποία ενδεχομένως θα επιφέρει μικρότερο κόστος.

Μετά από δοκιμές που έγιναν στη σχετική επανατοποθέτηση της επιλεγόμενης εργασίας το μικρότερο δυνατό κόστος προήλθε από την εξής επανατοποθέτηση. Επιλέγουμε τυχαία τη σειρά δυο διαφορετικών εργασιών. Η εργασία με τον μικρότερο αύξοντα αριθμό στη σειρά, διαγράφεται από τη θέση που βρίσκεται και τοποθετείται ακριβώς μετά τη δεύτερη επιλεγόμενη εργασία με τον μεγαλύτερο αύξοντα αριθμό.

Γίνεται επιλογή αριθμού επαναλήψεων της παραπάνω διαδικασίας και κάθε φορά επιλέγουμε σαν καλύτερη λύση, αυτή που μας δίνει τον μικρότερο συνολικό χρόνο διεκπεραίωσης της συνολικής διαδικασίας.

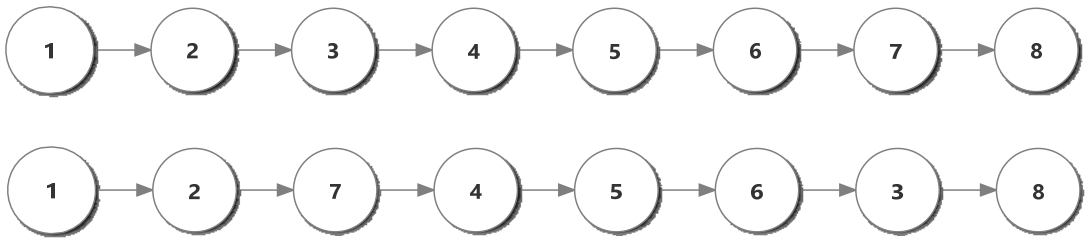
Παρακάτω απεικονίζεται σχηματικά πως επιλέξαμε να εφαρμόσουμε τον αλγόριθμο 1-0 relocate. Οι αριθμοί απεικονίζουν τον αύξοντα αριθμό των εργασιών και η σειρά που τοποθετούνται είναι η σειρά πραγματοποίησης των εργασιών. Αρχικά απεικονίζεται η αρχική σειρά που από αυτήν προκύπτει η αρχική λύση του προβλήματος και η δεύτερη σειρά απεικονίζει κάθε φορά πως επιλέγεται μια νέα λύση ώστε να διαπιστωθεί εάν είναι καλύτερη από την αρχική η όχι.



Εικόνα 1 : 1-0 relocate

β) 1-1 exchange –Τοπική αναζήτηση 1-1 ανταλλαγή

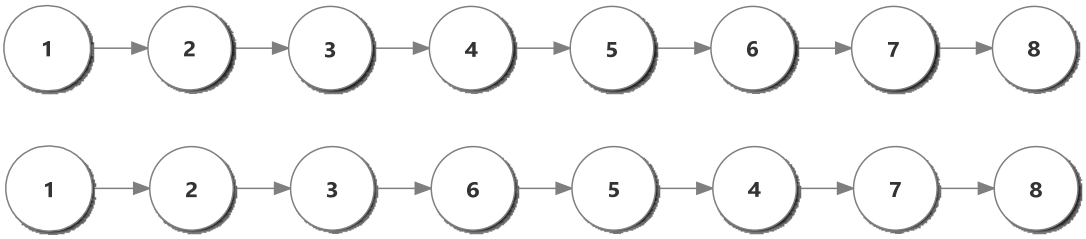
Σε αυτή τη μέθοδο, επιλέγουμε τυχαία δυο εργασίες, και τις αντιμεταθέτουμε ώστε η μία να αντικαταστήσει την σειρά (θέση) της άλλης. Προκύπτει μια νέα σειρά εργασιών από την οποία υπολογίζουμε τον συνολικό χρόνο διεκπεραίωσης της διαδικασίας. Όμοια με την παραπάνω μέθοδο όταν τελειώσει ο αριθμός των επαναλήψεων ελέγχουμε αν η λύση που προκύπτει είναι καλύτερη από την αρχική διαθέσιμη. Παρακάτω φαίνεται και σχηματικά η διαδικασία.



Εικόνα 2 : 1-1 exchange

γ)2-opt

Η διαδικασία που πραγματοποιείται κατά την τοπική αναζήτηση 2-opt είναι η ακόλουθη. Επιλέγονται τυχαία δυο εργασίες και οι ενδιάμεσες από αυτές εργασίες αντιστρέφονται αλλάζοντας την κατεύθυνση τους. Πάλι προκύπτει μια νέα λύση όταν τελειώσουν οι επαναλήψεις που έχουν οριστεί και πραγματοποιείται σχετικός έλεγχος αν είναι καλύτερη από την αρχική. Τα παραπάνω απεικονίζονται στο παρακάτω διάγραμμα.



Εικόνα 3 : 2-opt

Και στις 3 παραπάνω περιπτώσεις εάν η προκύπτουσα λύση είναι καλύτερη από την αρχική θα περιγραφεί παρακάτω πως αξιοποιείται αυτή η πληροφορία μέσα στον κυρίως αλγόριθμο.

Συνοπτικά οι αλγόριθμοι τοπικής αναζήτησης λειτουργούν ως εξής:

Αρχικοποίηση Επίλεξε μια αρχική λύση xo

Επανάλαβε την διαδικασία για συγκεκριμένο αριθμό επαναλήψεων

Δημιούργησε μια νέα γειτονιά x που ανήκει στην περιοχή της αρχικής λύσης

Σύγκρινε την αντικειμενική συνάρτηση με την παρούσα λύση και με την αρχική λύση, δηλαδή

If $f(x)<f(x_0)$ τότε

$x_0=x$

End if

Επανάλαβε την διαδικασία μέχρι να ολοκληρωθεί ο αριθμός των επαναλήψεων

Επέστρεψε την βέλτιστη λύση

Μετά και το πέρας της τοπικής αναζήτησης στην παρούσα εργασία εφαρμόζεται ο κυρίως αλγόριθμος. Σκοπός μας είναι να βρούμε την καλύτερη δυνατή λύση προσομοιώνοντας το πρόβλημα μας με την διαδικασία επιλογής συντρόφου των πυγολαμπίδων.

ΚΕΦΑΛΑΙΟ 4^{ο} HYBRID FIREFLY ALGORITHM

4.1. Παραλλαγή Αλγόριθμου

Στην παρούσα εργασία, πραγματοποιήθηκε η λύση του προβλήματος μέσω μιας παραλλαγής του κλασικού αλγορίθμου της πυγολαμπίδας. Θα γίνει ανάλυση βήμα προς βήμα πως πραγματοποιήθηκε η επίλυση αυτή.

Πριν προχωρήσουμε στον κυρίως αλγόριθμο θα πρέπει να πραγματοποιηθεί ένα μεταβατικό βήμα ώστε να επιλυθεί ένα πρακτικό πρόβλημα που προκύπτει. Μέχρι τώρα, δηλαδή μετά τον υπολογισμό της αντικειμενικής συνάρτησης και της τοπικής αναζήτησης, οι τιμές στα διανύσματα των σχετικών λύσεων είναι διακριτές. Στον αλγόριθμο που έχουμε επιλέξει, για την χρήση της βασικής συνάρτησης του, θα πρέπει να εισαχθούν συνεχείς τιμές, αλλά οι τιμές που έχουμε διαθέσιμες είναι διακριτές. Για να ξεπεραστεί αυτός ο σκόπελος θα πρέπει να γίνει η μετατροπή των διακριτών τιμών σε συνεχείς.

Η διαδικασία μετατροπής των διακριτών τιμών του προβλήματος σε συνεχείς είναι η εξής:

Επιλέγουμε κάθε λύση που αποτελεί σειρά του αρχικού μας πίνακα, δηλαδή ένα διάνυσμα, και επιλέγουμε ποιος είναι ο μεγαλύτερος αριθμός κατ' απόλυτη τιμή. Στη συνέχεια διαιρούμε όλους τους αριθμούς της λύσης με αυτόν τον αριθμό προκύπτοντας μια νέα λύση ισοδύναμη αλλά συνεχής. Δηλαδή μετατρέπουμε την λύση σε μια μορφή κατάλληλη για να εφαρμοστούν οι εξισώσεις του αλγορίθμου.

Αφού η λύση περιέχει συνεχείς τιμές, μπορεί να αρχίσει το κυρίως μέρος της λύσης του προβλήματος. Πρέπει να οριστούν παράμετροι όπως ο

αριθμός των πυγολαμπίδων, η λάμψη της κάθε πυγολαμπίδας. Τα δεδομένα που πρέπει να οριστούν είναι τα εξής:

- $f(x_i)$ η τιμή της αντικειμενικής συνάρτησης
- σμήνος από πυγολαμπίδες
- η θέση x_i της κάθε πυγολαμπίδας
- αρχική λάμψη της πυγολαμπίδας

Ο αλγόριθμος ορίζει ότι κάθε πυγολαμπίδα έλκεται από την πιο φωτεινή που υπάρχει στην γειτονιά της καθώς και ότι αυτή η λάμψη μειώνεται καθώς η απόσταση των δυο πυγολαμπίδων αυξάνεται.

Αρχικά πρέπει να ορίσουμε την αρχική λάμψη της πυγολαμπίδας I_0 , και η μεταβλητή γ που είναι μια σταθερά που δείχνει την απορροφητικότητα της λάμψης.

Στη συνέχεια θα πρέπει να διαπιστωθεί αν η κάθε πυγολαμπίδα θα κινηθεί προς μια ελκυστικότερη πυγολαμπίδα μέσω των εξισώσεων που έχουν αναφερθεί σε παραπάνω παράγραφο.

Συνοπτικά τα βήματα του αλγορίθμου είναι τα ακόλουθα:

- Αρχικά πραγματοποιείται η δημιουργία του αρχικού πληθυσμού από πυγολαμπίδες και τοποθετούνται σε τυχαίες θέσεις στον χώρο. Αυτή είναι και η αρχική λύση του προβλήματος
- Αμέσως μετά πρέπει να γίνει η αρχικοποίηση των παραμέτρων όπως η αρχική τους λάμψη, ο αριθμός των πυγολαμπίδων, οι σταθερές γ, α και ε
- Έπειτα πρέπει να υπολογιστούν οι τιμές της έντασης του φωτός και μέσω αυτής να υπολογιστεί και η αρχική τιμή της συνάρτησης κόστους
- Για να διαπιστωθεί αν η πυγολαμπίδα που μελετάμε θα κινηθεί προς μία άλλη θα πρέπει να συγκρίνουμε την ένταση της λάμψης της με την

ένταση της λάμψης κάθε άλλης πυγολαμπίδας. Η τρέχουσα πυγολαμπίδα έχει θέση x_i

- Εάν η ένταση της λάμψης της τρέχουσας πυγολαμπίδας είναι μικρότερη μιας άλλης, δηλαδή $I_i<I_j$ τότε η τρέχουσα πυγολαμπίδα δηλαδή αυτή που βρίσκεται στην θέση x_i μετακινείται προς την πυγολαμπίδα που βρίσκεται στην θέση x_j.
- Επιλέγουμε ποια είναι η καλύτερη πυγολαμπίδα κάθε φορά, και κρατάμε ποια είναι αυτή η κίνηση που μας δίνει την μικρότερη τιμή για την αντικειμενική μας συνάρτηση (permutation flowshop scheduling)
- Υπολογίζουμε τις νέες τιμές για την συνάρτηση κόστους για κάθε πυγολαμπίδα και πραγματοποιείται ενημέρωση των νέων τιμών για την ένταση της λάμψης
- Πραγματοποιείται ταξινόμηση των πυγολαμπίδων και βρίσκουμε ποια είναι η καλύτερη, δηλαδή ποια έχει την μεγαλύτερη λάμψη
- Πραγματοποιείται η διαδικασία από το σημείο του καθορισμού της έντασης του φωτός για την επόμενη πυγολαμπίδα μέχρι να ικανοποιηθεί ένα κριτήριο τερματισμού, συγκεκριμένα στην παρούσα εργασία έχουμε θέσει έναν μέγιστο αριθμό επαναλήψεων.

Κατά την εξέλιξη της παραπάνω διαδικασίας και στο σημείο όπου η κίνηση της πυγολαμπίδας δίνεται από τον τύπο όπως έχει αναφερθεί στον κλασικό αλγόριθμο της πυγολαμπίδας

$$X_i (t + 1) = X_i (t) + \beta_0 e^{-\gamma r_{ij}^2} (X_j (t) – X_i (t))+ \alpha \epsilon_t$$

Διαπιστώσαμε μετά από αρκετές παραλλαγές ότι οι καλύτερες λύσεις προκύπτουν εάν μετατραπεί ο τύπος στην παρακάτω μορφή.

$$X_i (t + 1) = \text{rand} (X_i (t) + \beta_0 e^{-\gamma r_{ij}^2} (X_j (t) – X_i (t))+ \alpha \epsilon_t)$$

Άρα στην παρούσα εργασία χρησιμοποιήσαμε τον παραπάνω τύπο για την επίλυση του προβλήματος.

Σε κάθε επανάληψη πραγματοποιείται πάλι τοπική αναζήτηση με τον ίδιο αλγόριθμο τοπικής αναζήτησης που είχε εφαρμοστεί πριν την έναρξη του αλγορίθμου της πυγολαμπίδας ώστε να είναι ανταγωνιστικές οι νέες λύσεις σε σχέση με τις αρχικές.

Αφού ολοκληρωθεί η διαδικασία του αλγορίθμου θα πρέπει οι λύσεις να βρίσκονται σε διακριτή μορφή. Για να έχουμε τη λύση μας με διακριτές τιμές πρέπει τις συνεχείς τιμές να τις μετατρέψουμε σε διακριτές, δηλαδή να ακολουθήσουμε την αντίστροφη διαδικασία από την αρχική. Αφού καταλήξουμε στη λύση μας, η οποία είναι με την μορφή διανύσματος, στη μικρότερη τιμή του διανύσματος των συνεχών τιμών αναθέτουμε τον αριθμό 1, στη δεύτερη μικρότερη τον αριθμό 2 κτλ. Αφού πραγματοποιηθεί η μετατροπή αυτή υπολογίζουμε το κόστος της τρέχουσας λύσης με τις διακριτές τιμές.

Αρχικά όλη η παραπάνω διαδικασία πραγματοποιήθηκε για κάθε έναν αλγόριθμο τοπικής αναζήτησης ξεχωριστά. Η σειρά που πραγματοποιήθηκαν είναι 1-0 relocate, 1-1 exchange, 2-opt. Τα αποτελέσματα θα αναλυθούν σε επόμενο κεφάλαιο.

Ο ψευδοκώδικας του υβριδικού αλγορίθμου της πυγολαμπίδας παρουσιάζεται παρακάτω.

Αρχικοποίηση

Αντικειμενική συνάρτηση f(x) όπου $x=(x_1,...x_n)$

Υπολογισμός του αρχικού κόστους της συνάρτησης (makespsan)

Ορισμός αριθμού πυγολαμπίδων

Εύρεση νέου αρχικού πληθυσμού πυγολαμπίδων – τοπική αναζήτηση x_i, i=1,2,…,m

Ορισμός έντασης λάμψης I_i στην x_i , σταθεράς α ,ε και γ

While μέγιστος αριθμός επαναλήψεων

for i=1: m (m=αριθμός πυγολαμπίδων)

for J=1:m

if (I_j>I_i)

η τρέχουσα πυγολαμπίδα i μετακινείται προς την πυγολαμπίδα j.

Endif

Εύρεση νέας λύσης και αξιολόγησης αυτής

Ανανέωση έντασης λάμψης

End for

End for

Ταξινόμηση πυγολαμπίδων και εύρεση της καλύτερης μέχρι τώρα τιμής (μικρότερο κόστος)

End while

Επέστρεψε την βέλτιστη λύση

4.2. VNS (Variable neighborhood search) - Αλγόριθμος μεταβλητής Γειτονιάς αναζήτησης

Η βασική ιδέα της μεθόδου αυτής είναι να πραγματοποιηθεί αναζήτηση σε γειτονιές κοντά στην αρχική λύση ώστε να βρεθεί μία καλύτερη λύση. Με την έννοια γειτονιά εννοούμε τους αλγορίθμους τοπικής αναζήτησης που έχουν αναλυθεί ήδη στην παρούσα εργασία. Η αναζήτηση μπορεί να πραγματοποιηθεί με τυχαίο ή συστηματικό τρόπο. Στην παρούσα εργασία έχει πραγματοποιηθεί με τυχαίο τρόπο.

Το πλεονέκτημα της μεθόδου αυτής είναι ότι εκμεταλλεύεται το γεγονός ότι διαφορετικές μέθοδοι τοπικής αναζήτησης δύναται να οδηγήσουν σε διαφορετικά τοπικά ελάχιστα. Καθώς εξελίσσεται ο αλγόριθμος αν βρεθεί μια καλύτερη λύση από την τρέχουσα η αναζήτηση ξεκινάει από την αρχή [12].

Υπάρχουν πολλές εφαρμογές της διαδικασίας αυτής σε συνδυασμό με άλλους μεθευρετικούς αλγορίθμους.

- ✓ Εφαρμογές στην βιομηχανία (βιομηχανία πετρελαίου)
- ✓ Σχεδίαση προβλημάτων στον τομέα των επικοινωνιών (ραντάρ, τοπολογικό σχέδιο)
- ✓ Χωροταξικά προβλήματα (προβλήματα p-median and p-centre)

- Προβλήματα data mining (cluster analysis, classification)
- Προβλήματα γράφων (k-subgraph problem, cycle basic πρόβλημα)
- Μεικτά ακέραια προβλήματα (προβλήματα θορύβου, ανάλυση χαρτοφυλακίου)
- Προβλήματα χρονοδιαγράμματος (προβλήματα προσανατολισμού, πρόβλημα του πλανώδιου πωλητή)
- Προβλήματα χρονοδρομολόγησης (χρονοπρογραμματισμός εργασιών, χρονοπρογραμματισμός εργασιών συνεχούς ροής)[11].

Για να αρχίσει η διαδικασία εφαρμογής του VNS πρέπει να οριστεί ο αριθμός Ν από επιλεγόμενες γειτονιές. Σε κάθε επανάληψη του αλγορίθμου υπάρχουν 3 βασικά βήματα

- Ανακίνηση
- Τοπική αναζήτηση
- Κίνηση

Ο αλγόριθμος μεταβλητής γειτονιάς αναζήτησης ξεκινάει με μία μέθοδο τοπικής αναζήτησης και μόλις αυτή η μέθοδος βρει κάποιο τοπικό ελάχιστο δοκιμάζει κάποιες άλλες μεθόδους για να δει αν μπορεί να βελτιώσει τη λύση.

Περιληπτικά ο αλγόριθμος λειτουργεί ως εξής:

Έχοντας την αρχική λύση s_0 σε κάθε επανάληψη δημιουργείται μια νέα λύση s_1. Μέσω της αντικειμενικής συνάρτησης, στην παρούσα εργασία μέσω του makespan του permutation flowshop scheduling problem, γίνεται έλεγχος αν η λύση αυτή οδηγεί σε καλύτερη λύση, δηλαδή στην παρούσα εργασία προκύπτει μικρότερο Makespan. Στη συνέχεια η διαδικασία αυτή αρχίζει με την χρήση της νέας γειτονιάς και την νέα λύση που βρέθηκε, s_1. Αν δεν βρεθεί μια καλύτερη λύση, τότε ο αλγόριθμος προχωράει στην επόμενη γειτονιά αναζήτησης.

Αλγόριθμος: Μεταβλητή γειτονιά Αναζήτησης

Αρχικοποίηση

Επίλεξε έναν σύνολο γειτονιών (N_i, i=1,2,…,i_{max})

Επίλεξε μια αρχική λύση s_0

Δημιούργησε μια νέα λύση s_1 στην γειτονιά του N_i

Δημιούργησε το s_2 όπου προέρχεται από την τοπική αναζήτηση της s, $s_2=LS(s_0)$

Εφάρμοσε διαδικασία τοπικής αναζήτηση στην s_1

If $f(s_2)<f(s_1)$ τότε

$S=s_2$

i=1

αλλιώς i=i+1

end if

επανάλαβε μέχρι $i<= i_{max}$

Επίστρεψε την βέλτιστη λύση

Στην παρούσα εργασία ο VNS εφαρμόστηκε στις 2 βασικές εκδοχές του. Αρχικά εφαρμόστηκε στην βασική του μορφή και έπειτα στην παράλληλη.

α)Basic VNS

Στην απλή μορφή του αλγορίθμου εφαρμόζονται και οι 3 αλγόριθμοι τοπικής αναζήτησης, 1-0 relocate, 1-1 exchange, 2-opt. Αρχικά εφαρμόζεται ο 1-0 relocate όσες επαναλήψεις έχουν οριστεί. Αμέσως μετά ο 1-1 exchange όσες επαναλήψεις έχουν οριστεί και τέλος ο 2-opt πάλι όσες φορές έχει οριστεί. Η διαδικασία που ακολουθήθηκε περιγράφεται παρακάτω. Το τμήμα του συνολικού αλγορίθμου πριν από την τοπική αναζήτηση και μετά από αυτήν παραμένει το ίδιο.

Basic VNS

Εφαρμογή 1-0 relocate

If makespan(αρχικής λύσης)<makespan(1-0 relocate)

Κρατάμε σαν καλύτερη λύση την αρχική

Else

Κρατάμε σαν καλύτερη λύση αυτή που προήλθε από το 1-0 relocate

End

Επέστρεψε βέλτιστη λύση μέχρι τώρα

Εφαρμογή 1-1 exchange

If makespan (βέλτιστης λύσης μέχρι τώρα)<makespan (1-1 exchange)

Κρατάμε σαν καλύτερη λύση την βέλτιστη μέχρι τώρα

Else

Κρατάμε σαν καλύτερη λύση αυτή που προήλθε από το 1-1 exchange

End

Επέστρεψε βέλτιστη λύση μέχρι τώρα

Εφαρμογή 2-opt

If makespan (βέλτιστης λύσης μέχρι τώρα)<makespan (2-opt)

Κρατάμε σαν καλύτερη λύση την βέλτιστη μέχρι τώρα

Else

Κρατάμε σαν καλύτερη λύση αυτή που προήλθε από το 2-opt

End

Επέστρεψε βέλτιστη λύση μέχρι τώρα

β) Parallel VNS

Σε αυτή τη μορφή του αλγορίθμου οι 3 αλγόριθμοι τοπικής αναζήτησης εφαρμόζονται ταυτόχρονα, σε κάθε περίπτωση υπολογίζεται το makespan και η σύγκριση για την βέλτιστη λύση γίνεται αφού έχουν τελειώσει οι προκαθορισμένες επαναλήψεις

Parallel VNS

Ορισμός συνολικών επαναλήψεων

Εφαρμογή 1-0 relocate

Υπολογισμός makespan (1-0 relocate)

Εφαρμογή 1-1 exchangre

Υπολογισμός makespan (1-1 exchange)

Εφαρμογή 2-opt

Υπολογισμός makespan (2-opt)

Επιλογή του μικρότερου makespan

Επέστρεψε την καλύτερη λύση μέχρι τώρα

ΚΕΦΑΛΑΙΟ 5ο ΑΠΟΤΕΛΕΣΜΑΤΑ

Ο αλγόριθμός εφαρμόστηκε σε 120 παραδείγματα της βιβλιογραφίας [22]. Σε αυτά τα παραδείγματα υπάρχουν διαφορετικοί αριθμοί από σύνολο προβλημάτων και συγκεκριμένα προβλήματα με 20, 50, 100, 200 και 500 εργασίες και με 5, 10, 20 μηχανές. Υπάρχουν 10 προβλήματα μέσα σε κάθε αριθμού σετ. Στο σύνολο υπάρχουν 12 σύνολα τα οποία είναι 20x5(20 εργασίες και 5 μηχανές), 20x10, 20x20, 50x5, 50x10, 50x20, 100x5, 100x10, 100x20, 200x10, 200x20 και 500x20.

Problem	Firefly + 1-0 relocate	Firefly + 1-1 exchange	Firefly + 2opt	Firefly + SVNS	Firefly + PVNS (HFVNS)
20 X 5	1,49	1,49	1,49	1,49	0,47
20 X 5	0,00	0,07	0,59	0,07	0,07
20 X 5	3,33	4,72	5,64	3,98	1,57
20 X 5	1,93	5,34	5,34	3,40	1,24
20 X 5	0,73	0,89	3,89	1,21	0,65
20 X 5	2,09	3,01	2,43	2,43	0,42
20 X 5	0,97	0,97	1,29	0,97	0,81
20 X 5	3,15	3,65	5,06	3,07	0,58
20 X 5	2,93	3,82	3,58	2,28	2,03
20 X 5	2,35	2,62	4,33	2,35	1,17
20 X 10	4,93	6,13	6,45	5,56	0,25
20 X 10	4,28	4,04	6,45	5,18	2,11
20 X 10	5,07	6,80	4,73	4,47	1,67
20 X 10	6,25	6,39	5,52	5,23	1,74
20 X 10	2,75	5,21	8,17	7,05	2,54
20 X 10	4,58	4,72	5,08	4,44	1,93
20 X 10	3,84	2,63	5,93	3,44	1,62
20 X 10	4,53	5,38	7,51	4,66	1,62
20 X 10	2,70	4,71	4,77	2,82	1,95
20 X 10	5,09	4,02	5,72	5,22	1,51
20 X 20	4,18	4,00	5,48	3,70	1,22
20 X 20	4,19	5,33	6,04	4,28	0,86
20 X 20	4,12	3,48	5,11	3,18	1,37
20 X 20	3,10	4,81	5,03	4,27	1,48
20 X 20	4,36	3,79	4,88	4,23	1,26
20 X 20	3,90	3,81	5,56	2,20	1,62
20 X 20	3,48	4,93	5,06	4,00	2,33

20 X 20	3,22	4,00	5,68	3,13	1,27
20 X 20	4,87	4,73	5,94	4,96	1,16
20 X 20	5,19	5,79	7,02	4,68	1,42
50 X 5	0,92	1,03	1,03	1,03	0,18
50 X 5	3,52	3,52	3,81	2,26	1,16
50 X 5	2,82	2,75	2,79	2,52	1,03
50 X 5	3,34	3,34	4,00	3,34	1,13
50 X 5	0,98	1,54	2,06	1,15	0,87
50 X 5	3,08	2,40	2,65	2,55	0,64
50 X 5	3,23	3,12	3,78	2,61	1,21
50 X 5	3,35	3,06	3,84	2,05	1,19
50 X 5	3,64	3,10	3,88	2,35	0,98
50 X 5	0,90	1,94	2,37	0,58	0,07
50 X 10	10,16	11,43	11,84	8,83	6,39
50 X 10	10,22	10,46	13,01	11,44	6,70
50 X 10	11,52	11,73	14,30	10,78	7,08
50 X 10	7,97	8,88	10,06	8,52	5,19
50 X 10	8,80	11,63	11,49	10,89	6,35
50 X 10	10,15	9,45	11,24	9,12	5,19
50 X 10	8,37	9,12	9,05	8,63	5,30
50 X 10	8,69	8,40	9,58	7,57	4,68
50 X 10	11,15	10,87	11,63	9,15	5,94
50 X 10	9,69	10,31	10,57	8,87	5,84
50 X 20	12,03	12,39	12,60	11,71	6,26
50 X 20	12,10	11,45	13,42	11,88	7,91
50 X 20	12,61	13,65	14,12	12,99	7,28
50 X 20	11,91	13,09	13,76	12,20	7,66
50 X 20	14,60	14,38	15,73	14,32	8,14
50 X 20	11,68	13,07	14,24	11,08	7,39
50 X 20	13,39	13,28	13,12	12,37	8,07
50 X 20	11,41	10,89	13,28	12,73	8,26
50 X 20	11,01	11,97	12,96	11,30	7,67
50 X 20	11,47	13,53	13,98	11,85	7,43
100 X 5	1,31	0,69	1,80	1,64	0,51
100 X 5	2,35	1,88	2,07	0,91	0,30
100 X 5	2,84	2,80	3,07	1,82	1,18
100 X 5	1,54	2,23	2,09	2,03	0,60
100 X 5	2,46	2,70	2,38	2,29	1,09
100 X 5	1,97	1,77	2,03	1,97	0,51
100 X 5	2,33	2,27	2,44	2,25	0,86
100 X 5	3,04	2,96	3,00	2,10	1,00
100 X 5	1,89	2,29	2,31	2,31	1,14
100 X 5	2,25	2,63	3,10	2,22	0,94
100 X 10	8,34	8,46	8,82	7,64	4,42
100 X 10	9,01	8,62	9,61	8,94	4,69

100 X 10	7,10	5,90	6,52	6,57	4,07
100 X 10	8,41	9,34	9,95	8,77	5,50
100 X 10	7,03	8,81	9,28	8,37	3,71
100 X 10	6,49	5,95	7,44	5,86	3,38
100 X 10	8,15	7,23	8,67	7,80	4,09
100 X 10	6,86	7,09	7,31	5,81	3,49
100 X 10	6,24	5,87	6,24	6,12	3,66
100 X 20	13,00	13,95	15,45	14,25	9,63
100 X 20	13,83	14,09	15,22	14,77	9,56
100 X 20	13,46	13,79	14,88	13,84	9,23
100 X 20	12,70	13,88	14,82	11,95	8,34
100 X 20	13,10	13,26	13,08	13,03	8,98
100 X 20	13,06	13,28	13,42	12,99	9,19
100 X 20	15,08	14,42	15,03	14,96	10,23
100 X 20	13,79	13,86	15,25	12,87	9,94
100 X 20	14,41	13,40	14,74	13,18	9,18
100 X 20	12,43	12,23	14,08	11,59	7,32
200 X 10	5,43	5,08	5,25	5,25	2,76
200 X 10	8,34	8,20	8,17	8,54	4,89
200 X 10	5,56	6,13	6,30	5,88	3,34
200 X 10	4,79	4,83	4,04	4,07	2,33
200 X 10	7,93	7,35	7,21	8,00	4,11
200 X 10	7,70	7,76	7,75	7,34	4,58
200 X 10	6,97	6,88	7,03	6,67	3,64
200 X 10	7,52	6,96	6,84	6,99	4,16
200 X 10	7,99	7,11	7,19	6,66	4,48
200 X 10	6,32	6,98	6,62	6,41	4,22
200 X 20	13,01	13,05	13,85	13,26	9,36
200 X 20	14,84	15,11	15,31	13,61	10,57
200 X 20	14,32	13,87	15,10	13,37	9,82
200 X 20	12,97	13,37	13,45	12,98	9,67
200 X 20	13,14	13,48	13,80	12,64	9,06
200 X 20	14,34	15,09	14,80	14,41	9,88
200 X 20	13,39	14,32	13,94	13,93	9,76
200 X 20	13,14	12,61	13,23	13,02	9,40
200 X 20	13,80	13,83	14,86	13,37	10,19
200 X 20	13,11	13,70	14,48	13,59	9,52
500 X 20	10,87	11,14	11,18	10,56	8,12
500 X 20	10,63	10,85	10,94	10,69	8,02
500 X 20	10,14	10,34	10,34	9,96	7,87
500 X 20	10,09	9,93	9,90	9,51	10,09
500 X 20	10,84	10,09	9,95	10,49	7,90
500 X 20	10,87	10,62	10,86	10,50	7,72
500 X 20	9,59	9,52	10,13	9,80	6,90
500 X 20	10,57	10,07	10,33	9,95	7,74

500 X 20	11,61	11,12	11,51	10,66	8,60
500 X 20	10,43	10,53	10,41	10,10	7,28

Πίνακας 2 : Αποτελέσματα Αλγόριθμου

Οι παράμετροι του προτεινόμενου αλγόριθμου επιλέχθηκαν έπειτα από διεξοδικές δοκιμές. Οι τελικές τιμές που επιλέχθηκαν για τις σχετικές παραμέτρους είναι αυτές οι οποίες έδωσαν τα καλυτέρα αποτελέσματα όσο αφορά τη ποιότητα της λύσης αλλά και τον υπολογιστικό χρόνο που χρειάζεται. Η αποδοτικότητα του νέου υβριδικού αλγόριθμού υπολογίζεται από τη ποιότητα της παραγόμενης λύσης. Η ποιότητα δίδεται ως προς τη σχετική απόκλιση από την καλύτερη γνωστή λύση, η οποία υπολογίζεται με το τύπο $\omega = \frac{(Τρέχουσα\ Λύση - Βέλτιστη\ Λύση)}{Βέλτιστη\ Λύση}\%$.

Problem	Firefly + 1-0 relocate	Firefly + 1-1 exchange	Firefly + 2opt	Firefly + SVNS	Firefly + PVNS (HFVNS)
20 X 5	1,90	2,66	3,36	2,12	0,90
20 X 10	4,40	5,00	6,03	4,81	1,69
20 X 20	4,06	4,47	5,58	3,86	1,40
50 X 5	2,58	2,58	3,02	2,04	0,85
50 X 10	9,67	1,23	11,28	9,38	5,86
50 X 20	12,22	12,77	13,72	12,24	7,61
100 X 5	2,20	2,22	2,43	1,95	0,81
100 X 10	7,69	7,66	8,42	7,49	4,26
100 X 20	13,48	13,62	14,60	13,35	9,16
200 X 10	6,86	6,73	6,64	6,58	3,85
200 X 20	13,61	13,84	14,28	13,42	9,72
500 X 20	10,56	10,42	10,55	10,22	8,03

Πίνακας 3 : Μέσος όρος Ποιότητας Λύσεων

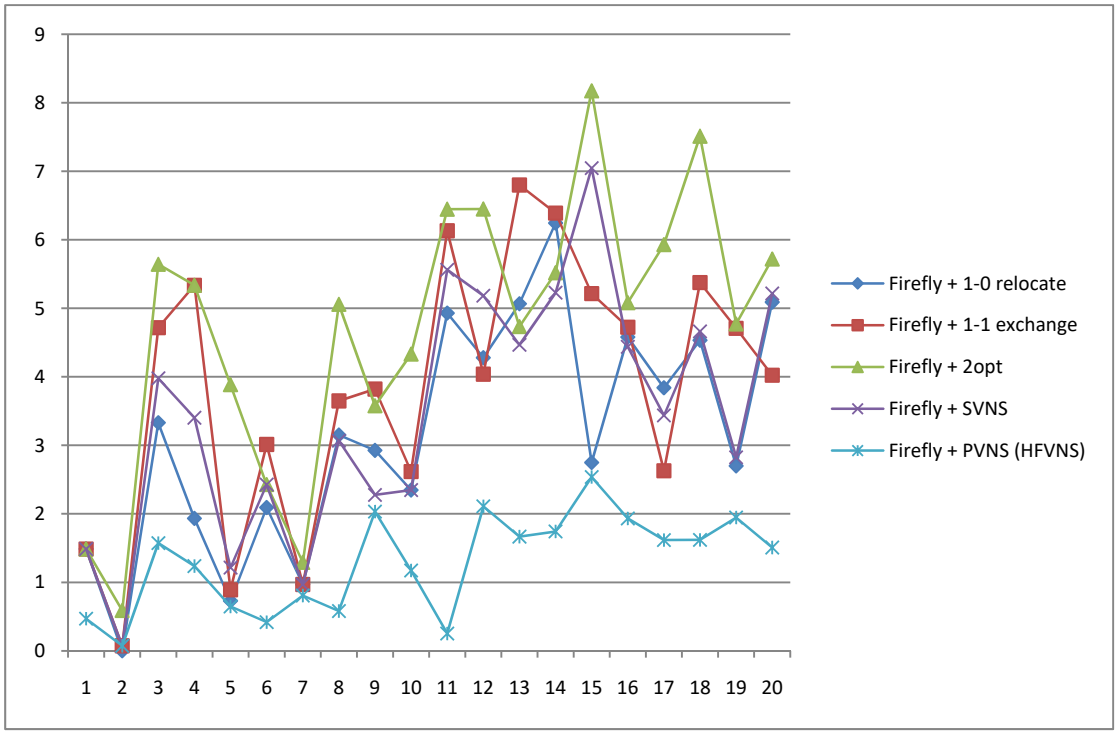
Στον Πίνακα 2 παρουσιάζονται τα αποτελέσματα από τα 120 Παραδείγματα αναφοράς της βιβλιογραφίας (Taillard). Συγκεκριμένα σε αυτό το πίνακα παρουσιάζεται το ω (ποιότητα λύσεων) για κάθε περίπτωση που μελετήσαμε στο παρών πρόβλημα.

Όπως φαίνεται και στο Πίνακα ο προτεινόμενος αλγόριθμος βρίσκει πολύ καλύτερα αποτελέσματα συγκριτικά με τις άλλες μεθόδους. Επίσης αξίζει να τονισθούν τα ικανοποιητικά αποτελέσματα καθώς προκύπτουν λύσεις οι οποίες έχουν απόκλιση από τις βέλτιστες λιγότερο από 1% σε 19 παραδείγματα. Επίσης προκύπτουν λύσεις με απόκλιση μεταξύ 1% και 5% σε 50 παραδείγματα. Τα παραδείγματα μπορούν να διαιρεθούν σε 5 διαφορετικές κατηγορίες ανάλογα με τον αριθμό των εργασίων. Η πρώτη κατηγορία περιλαμβάνει τα παραδείγματα για τα οποία οι εργασίες είναι 20 και ο αριθμός των μηχανών μεταξύ 5 και 20. Σε αυτά τα παραδείγματα η απόκλιση των λύσεων από το βέλτιστο είναι μεταξύ 0,07 και 2,53. Στη δεύτερη κατηγορία ο αριθμός των εργασιών είναι 50 και ο αριθμός των μηχανών μεταξύ 5 και 20. Σε αυτά τα παραδείγματα η απόκλιση των λύσεων από το βέλτιστο κυμαίνεται μεταξύ 0,07 και 8,26. Στη τρίτη κατηγορία ο αριθμός των εργασιών είναι 100 και ο αριθμός των μηχανών κυμαίνεται μεταξύ 5 και 20. Σε αυτά τα παραδείγματα η απόκλιση των λύσεων κυμαίνεται μεταξύ 0,3 και 10,22. Στη τέταρτη κατηγορία ο αριθμός των εργασιών ισούται με 200 και ο αριθμός των μηχανών μεταξύ 10 και 20. Σε αυτά τα παραδείγματα η απόκλιση των λύσεων από τη βέλτιστη κυμαίνεται μεταξύ 2,33 και 10,56. Τέλος στη πέμπτη κατηγορία ο αριθμός των εργασιών είναι 500 και ο αριθμός των μηχανών είναι 20. Σε αυτή τη περίπτωση η απόκλιση από το βέλτιστο κυμαίνεται από 6,9 έως 10,09.

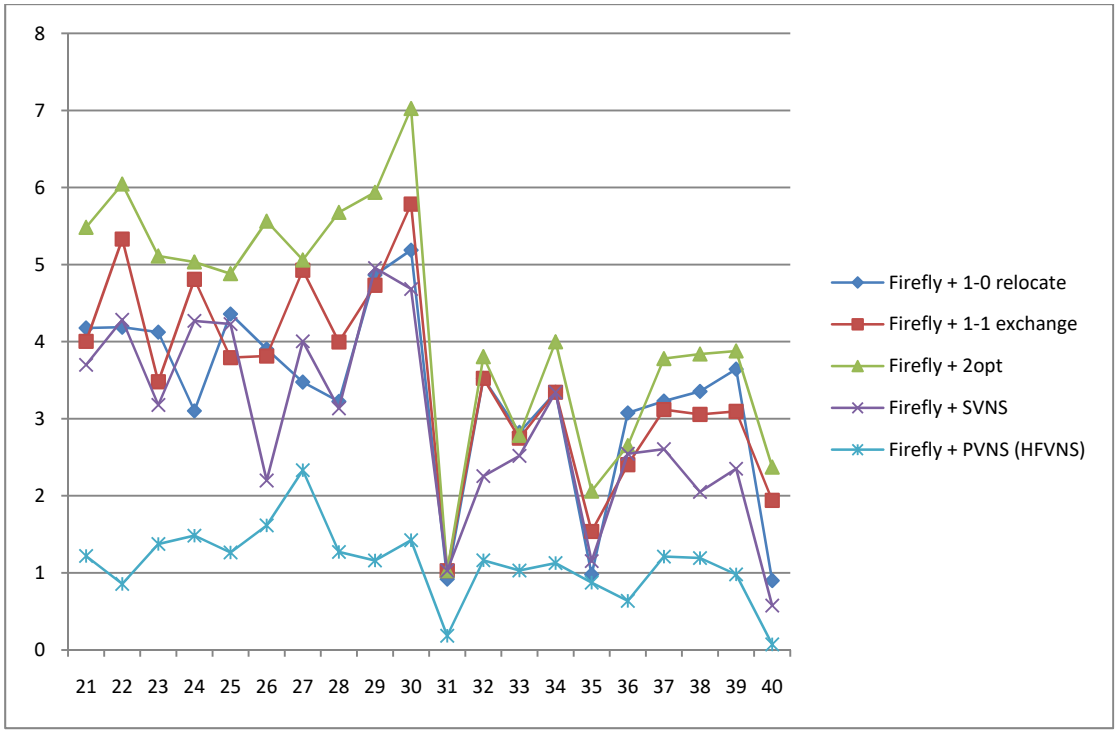
Όπως έχει ήδη αναφερθεί οι αποστάσεις μπορούν να διαιρεθούν σε 12 σύνολα βασισμένα στο αριθμό των εργασιών και στον αριθμό των μηχανών. Στον Πίνακα 3 κάθε ένα από τα δώδεκα σύνολα έχουμε υπολογίσει κατά μέσο όρο την ποιότητα των 10 αντίστοιχων παραδειγμάτων που ανήκουν σε κάθε ένα από τα 12 σύνολα. Στον πίνακα 2 εκτός από τα αποτελέσματα του προτεινόμενου αλγόριθμου παρουσιάζονται και τα αποτελέσματα άλλων τεσσάρων αλγορίθμων. Στο συγκεκριμένο πίνακα στις στήλες 2 με 4

παρουσιάζονται τα αποτελέσματα του αλγόριθμου πυγολαμπίδας χρησιμοποιώντας τοπική αναζήτηση. Στη στήλη 5 παρουσιάζονται τα αποτελέσματα του βασικού VNS αλγορίθμου πυγολαμπίδας και στη τελευταία στήλη τα αποτελέσματα του προτεινόμενου αλγόριθμου.

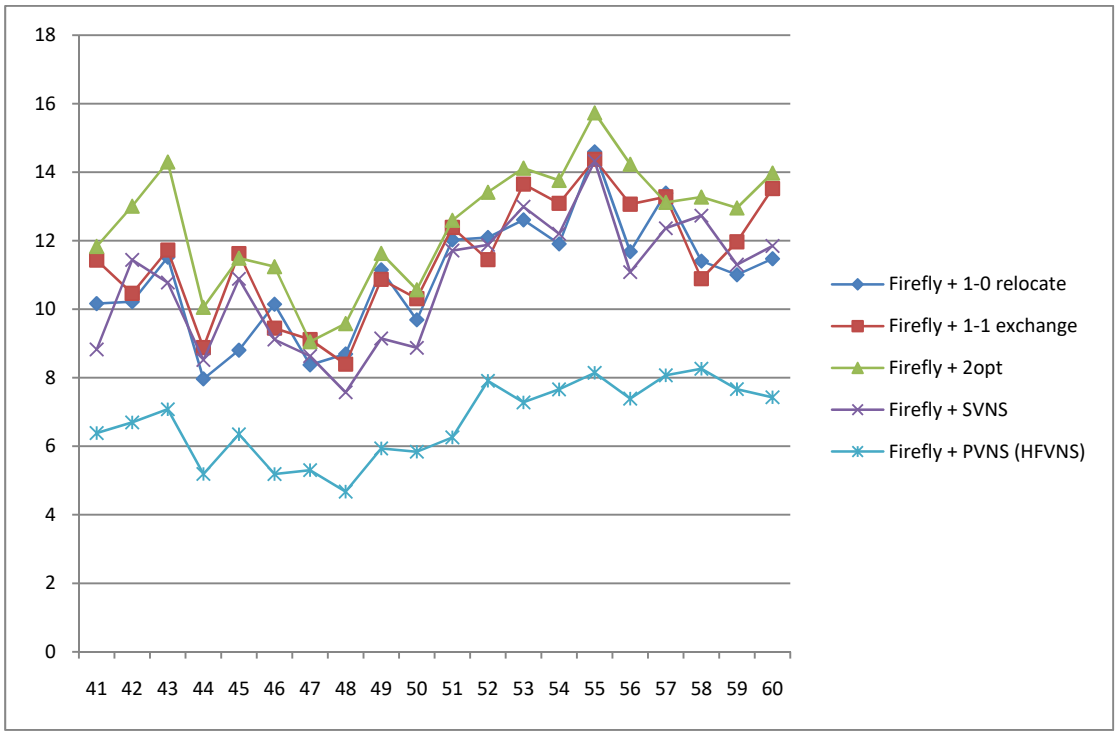
Παρατηρούμε από τα αποτελέσματα ότι ο προτεινόμενος αλγόριθμος αποδίδει καλύτερα αποτελέσματα σε σχέση με τους άλλους που χρησιμοποιούν αλγόριθμο τοπικής αναζήτησης. Είναι ενδιαφέρον επίσης η μεγάλη διαφορά όσο αφορά την ποιότητα των λύσεων μεταξύ του βασικού VNS αλγορίθμου πυγολαμπίδας και του παράλληλου VNS αλγόριθμου.



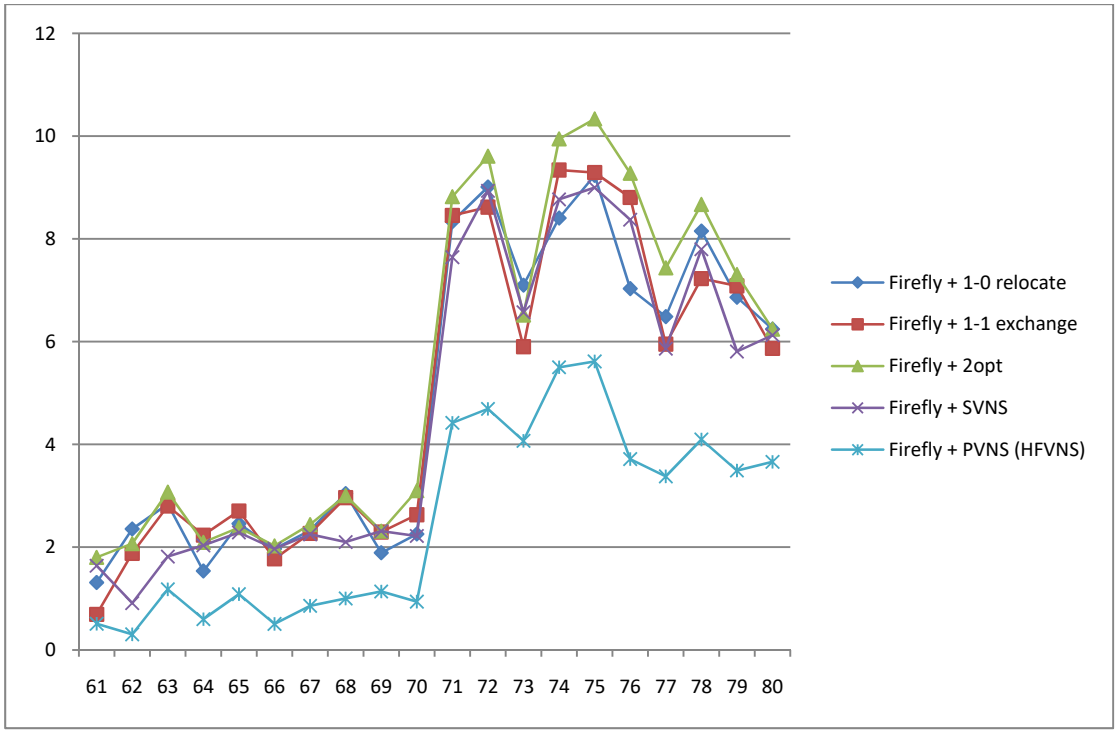
Πίνακας 4 : Αποτελέσματα παραδειγμάτων 1-20



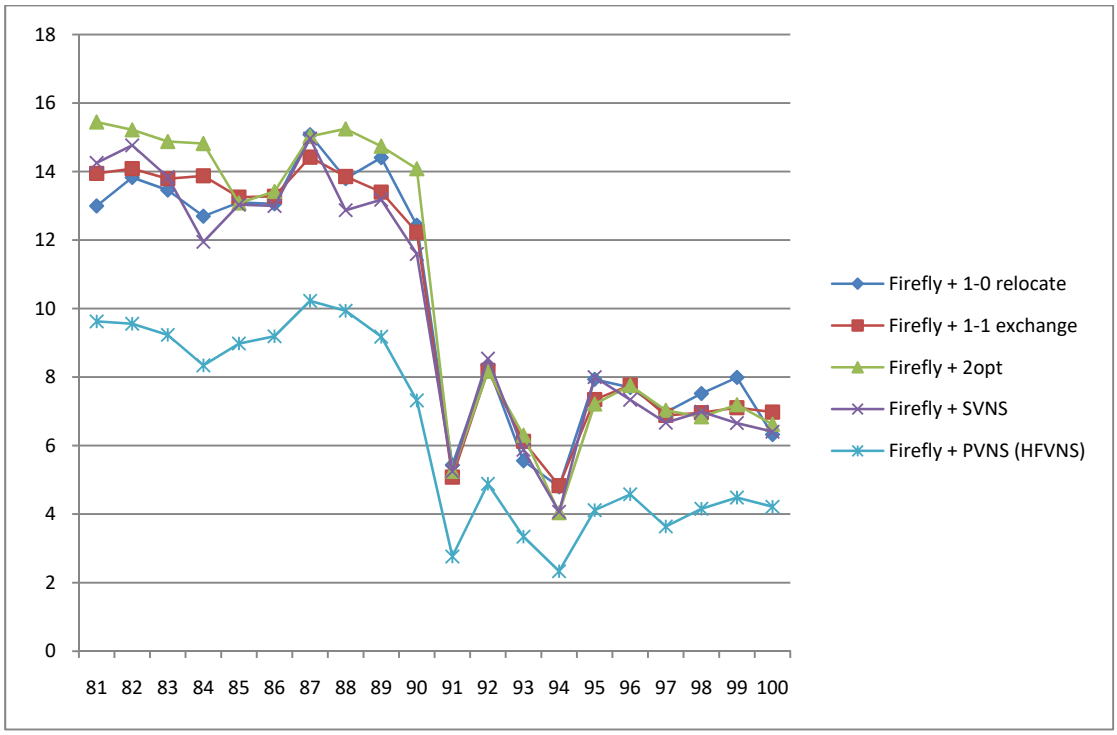
Πίνακας 5 : Αποτελέσματα παραδειγμάτων 21-40



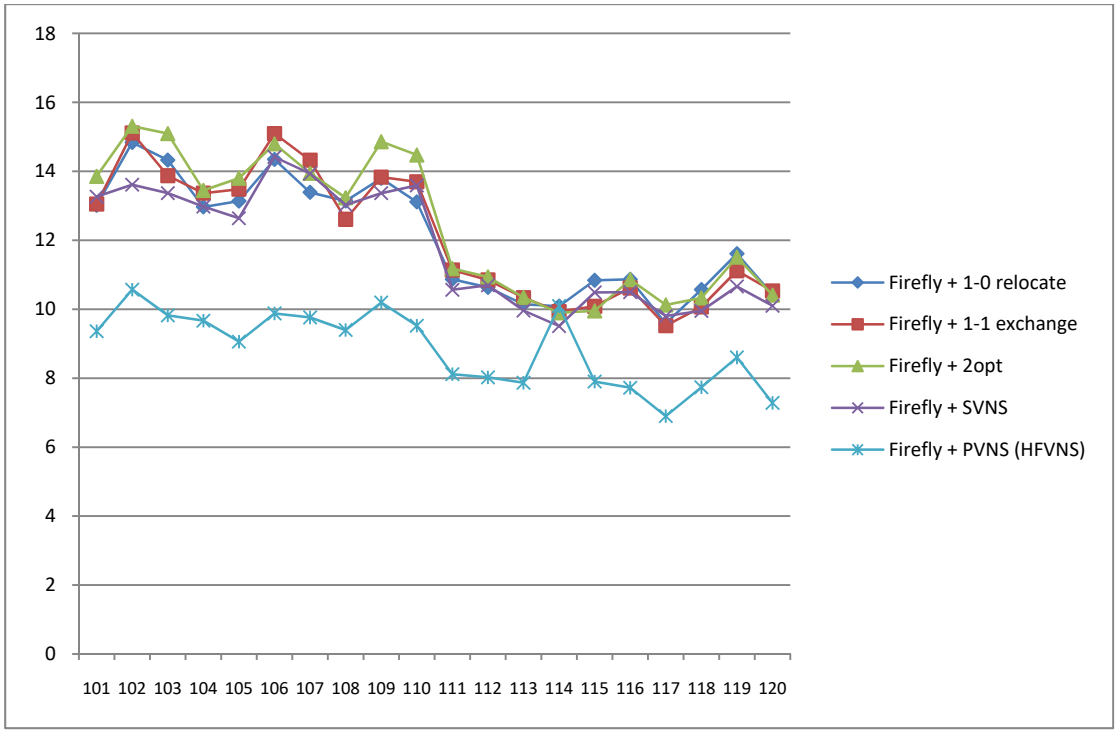
Πίνακας 6 : Αποτελέσματα παραδειγμάτων 41-60



Πίνακας 7 : Αποτελέσματα παραδειγμάτων 61-80



Πίνακας 8 : Αποτελέσματα παραδειγμάτων 81-100



Πίνακας 9 : Αποτελέσματα παραδειγμάτων 101-120

Στους Πίνακες 3-8 παρουσιάζεται η απόκλιση όλων των αλγορίθμων στα 120 παραδείγματα της βιβλιογραφίας. Η επιλογή για παρουσίαση των αποτελεσμάτων σε 6 διαφορετικούς πίνακες έγινε λόγω του αριθμού των παραδειγμάτων όσο και των αλγορίθμων καθώς και λόγω της ευκρίνειας των αποτελεσμάτων. Σε κάθε Πίνακα παρουσιάζονται τα αποτελέσματα για 20 παραδείγματα. Πιο συγκεκριμένα στο τρίτο Πίνακα παρουσιάζονται τα αποτελέσματα των πρώτων 20 παραδειγμάτων ,στον τέταρτο τα παραδείγματα των επόμενων 20 και ούτω καθεξής. Όπως φαίνεται από τους Πίνακες τα αποτελέσματα του προτεινόμενου αλγόριθμου είναι πολύ καλύτερα από τους υπόλοιπους αλγόριθμους.

ΒΙΒΛΙΟΓΡΑΦΙΑ

1. Yang, X. S., & Press, L. (2010). Nature-Inspired Metaheuristic Algorithms Second Edition.
2. Ibrahim M. Alharkan (2005). Algorithms for Sequencing and Scheduling, Riyadh, Saudi Arabia: Industrial Engineering Department College of Engineering King Saud University
3. Chen, S.-H., Chang, P.-C., Cheng, T.C.E., Zhang, Q. (2012). A Self-guided Genetic Algorithm for Permutation Flowshop Scheduling Problems, Computers and Operations Research, 39, 1450-1457.
4. Schmid, J., Kieser, L., Hanne, T., & Dornberger, R. (2017, November). Optimizing different parameters of a discrete firefly algorithm for solving the permutation flow shop problem. In Computational Intelligence (SSCI), 2017 IEEE Symposium Series on (pp. 1-6). IEEE.
5. Marinakis, Y., & Marinaki, M. (2013). Particle swarm optimization with expanding neighborhood topology for the permutation flowshop scheduling problem. Soft Computing, 17(7), 1159-1173.
6. Johnson, S. (1954). Optimal Two-and-three Stage Production Schedules with Setup Times Included, Naval Research Logistics Quarterly, 1, 61-68.
7. Ιωάννης Μαρινάκης, Μαγδαληνή Μαρινάκη (2010). Σημειώσεις Μεταπτυχιακού Μαθήματος: Εξελικτικοί Αλγόριθμοι και Βελτιστοποίηση Συστημάτων Μεγάλης Κλίμακας, Χανιά: Πολυτεχνείο Κρήτης
8. Talbi, E.-G. (2009). Metaheuristics : From Design to Implementation, John Wiley and Sons, USA.
9. Papadimitriou, C. H., Steiglitz, K. (1982). Combinatorial Optimization -
10. Algorithms and Complexity, Prentice-Hall, Englewood Cliffs, N. J.
11. Srikakulapu, R., Vinatha, U. (2017). Combined approach of firefly algorithm with travelling salesmen problem for optimal design of offshore wind farm. 2017 IEEE Power and Energy Society General Meeting, 1-5.
12. Hansen, P., Mladenović, N., & Pérez, J. A. M. (2010). Variable neighbourhood search: methods and applications. Annals of Operations Research, 175(1), 367-407.
13. Ruiz, R., Maroto, C., Alcaraz, J. (2006) Two New Robust Genetic Algorithms for the Flowshop Scheduling Problem, Omega, 34, 461-476.

14. Ruiz, R., Stutzle, T. (2007) A Simple and Effective Iterated Greedy Algorithm for the Permutation Flowshop Scheduling Problem, European Journal of Operational Research, 177, 2033-2049.
15. Tseng, L.-Y., Lin, Y.-T. (2009). A Hybrid Genetic Local Search Algorithm for the Permutation Flowshop Scheduling Problem, European Journal of Operational Research, 198, 84-92.
16. Tseng, L.-Y., Lin, Y.-T. (2010). A Genetic Local Search Algorithm for Minimizing Total Flowtime in the Permutation Flowshop Scheduling Problem, International Journal of Production Economics, 127, 121-128.
17. Pan, Q.-K., Tasgetiren, M.F., Liang, Y.-C. (2008). A Discrete Differential Evolution Algorithm for the Permutation Flowshop Scheduling Problem, Computers and Industrial Engineering, 55, 795-816.
18. Osaba, E., Yang, X. S., Diaz, F., Onieva, E., Masegosa, A. D., & Perallos, A. (2017). A discrete firefly algorithm to solve a rich vehicle routing problem modelling a newspaper distribution system with recycling policy. Soft Computing, 21(18), 5295-5308.
19. Zobolas, G.I., Tarantilis, C.D., Ioannou, G. (2009). Minimizing Makespan in Permutation Flow Shop Scheduling Problems using a Hybrid Metaheuristic Algorithm, Computers and Operations Research, 36, 1249-1267.
20. Grabowski, J., Wodecki, M. (2004) A Very Fast Tabu Search Algorithm for the Permutation Flow Shop Problem with Makespan Criterion, Computers and Operations Research, 31, 1891-1909.
21. Nowicki, E., Smutnicki, C. (1996). A Fast Tabu Search Algorithm for the Permutation Flow-shop Problem, European Journal of Operational Research, 91, 160-175.
22. Taillard, E. (1993). Benchmarks for Basic Scheduling Problems, European Journal of Operational Research, 64, 278-285
23. Liao, C.-J., Tseng, C.-T., Luarn P. (2007). A Discrete Version of Particle Swarm Optimization for Flowshop Scheduling Problems, Computers and Operations Research, 34, 3099-3111.
24. Liu, B., Wang, L., Jin, Y.-H. (2007). An Effective PSO-based Memetic Algorithm for Flow Shop Scheduling, IEEE Transactions on Systems, Man, and Cybernetics Part B: Cybernetics, 37(1), 18-27.
25. Rajendran, C., Ziegler, H. (2004). Ant-colony Algorithms for Permutation Flow-shop Scheduling to Minimize Makespan/Total Flowtime of Jobs, European Journal of Operational Research, 155(2), 426-438.

26. Rajendran, C., Ziegler, H. (2005). Two Ant-colony Algorithms for Minimizing Total Flowtime in Permutation Flowshops, Computers and Industrial Engineering, 48(4), 789-797.
27. Liu, Y.-F., Liu, S.-Y. (2013). A Hybrid Discrete Artificial Bee Colony Algorithm for Permutation Flowshop Scheduling Problem, Applied Soft Computing, 13, 1459- 1463.

Όνομα αρχείου: ΔΙΠΛΩΜΑΤΙΚΗ ΚΑΡΑΦΥΛΛΙΔΗΣ ΤΕΛΙΚΟ
Κατάλογος: C:\Users\user\Desktop
Πρότυπο:
C:\Users\user\AppData\Roaming\Microsoft\Πρότυπα\Normal.dotm
Τίτλος:
Θέμα:
Συντάκτης: FIDIS
Λέξεις - κλειδιά:
Σχόλια:
Ημερομηνία δημιουργίας: 8/3/2018 10:17:00 μμ
Αριθμός αλλαγής: 120
Τελευταία αποθήκευση: 4/7/2018 1:40:00 μμ
Τελευταία αποθήκευση από: Yannis
Συνολικός χρόνος επεξεργασίας: 16.747 Λεπτά
Τελευταία εκτύπωση: 4/7/2018 1:55:00 μμ
Στοιχεία εγγράφου όπως καταγράφηκαν την τελευταία φορά που εκτυπώθηκε πλήρως
Αριθμός σελίδων: 53
Αριθμός λέξεων: 9.384 (περίπου)
Αριθμός χαρακτήρων: 50.678 (περίπου)