ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ

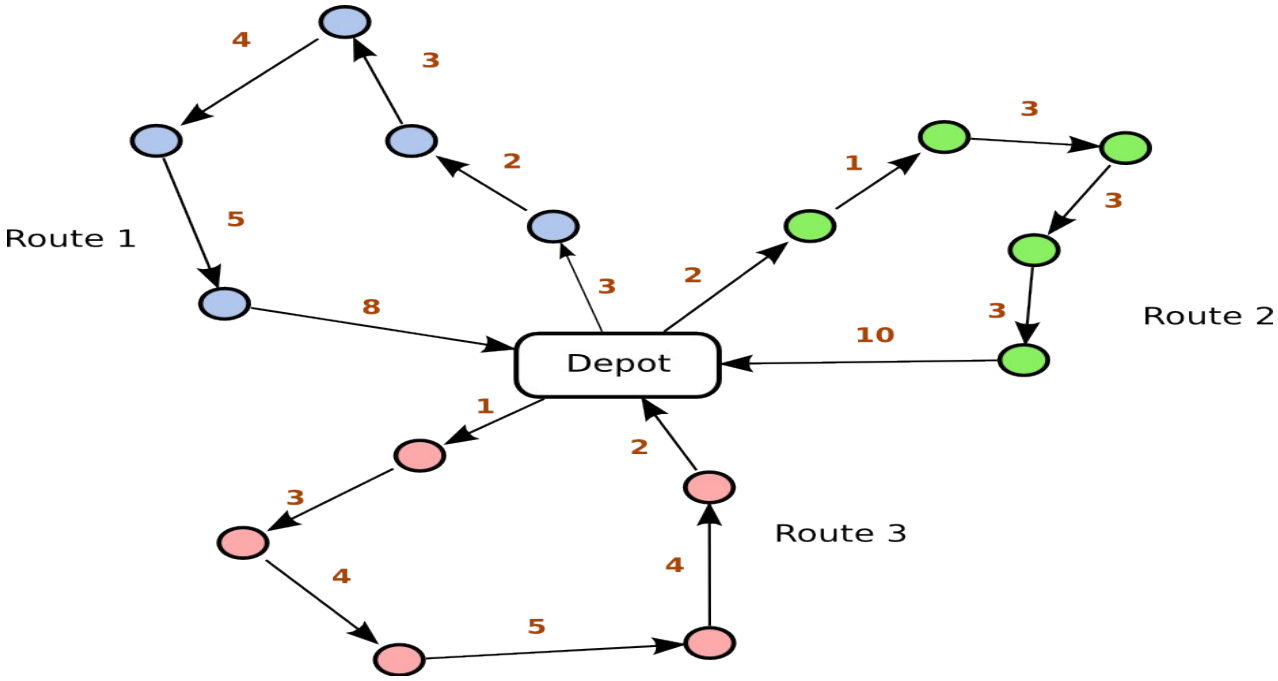
ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ ΠΑΡΑΓΩΓΗΣ ΚΑΙ ΔΙΟΙΚΗΣΗΣ

Διπλωματική Εργασία

Το Συσσωρευτικό Πρόβλημα δρομολόγησης οχημάτων με χρήση αλγορίθμου ACO

Κυριακάκης Νικόλαος-Αντώνιος

Επιβλέπων Καθηγητής
Ιωάννης Μαρινάκης



Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή μου κ. Μαρινάκη Ιωάννη, για όλη τη βοήθεια του, την καθοδήγηση και τις επισημάνσεις του πάνω στην παρούσα διπλωματική εργασία, αλλά κυρίως διότι με ώθησε μέσω της διδασκαλίας του να θέλω να ασχοληθώ με αυτό το επιστημονικό πεδίο.

Θα ήθελα επίσης να ευχαριστήσω όλους τους καθηγητές μου, που σε αυτά τα 5 χρόνια, μέσω των μαθημάτων τους μου έδωσαν τα εφόδια να ατενίζω το μέλλον με αισιοδοξία και προοπτική.

Τέλος, θα ήθελα να ευχαριστήσω την οικογένειά μου για τη στήριξη τους και την εμπιστοσύνη τους. Τους φίλους μου για τον υγιή ανταγωνισμό που είχαμε αυτά τα χρόνια σε σχέση με τη σχολή και λειτούργησε δημιουργικά για την επίτευξη των στόχων μου.

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΕΦΑΛΑΙΟ 1. Εισαγωγή 5
1.1. Η εφοδιαστική αλυσίδα.....6
ΚΕΦΑΛΑΙΟ 2. Συσσωρευτικό Πρόβλημα δρομολόγησης οχημάτων (CCCVRP) 7
2.1. Εισαγωγή.....8
2.2.Μοντελοποίηση.....8
2.3. Μαθηματική απεικόνιση.....9
ΚΕΦΑΛΑΙΟ 3. Βελτιστοποίηση με τον αλγόριθμο της Αποικίας μυρμηγκιών 10
3.1. Αλγόριθμοι.....11
3.1.1. Αλγόριθμοι βελτιστοποίησης.....11
3.1.2. Μεθευρετικοί Αλγόριθμοι.....11
3.1.3. Αλγόριθμοι εμπνευσμένοι από τη φύση.....12
3.2. Τα κοινωνικά έντομα..... 13
3.2.1. Αυτό-οργάνωση.....13
3.2.2. Στιγμεργία.....14
3.2.3. Αναζήτηση τροφής της αποικίας μυρμηγκιών.....14
3.3. Ο Αλγόριθμος της Αποικίας Μυρμηγκιών (ACO)..... 16
3.3.1. Ιστορική αναδρομή.....16
3.3.2. Η "ψηφιοποίηση" της διαδικασίας.....16
3.3.3 Εφαρμογές του ACO.....17
3.3.4.Μαθηματική εφαρμογή του ACO στο πρόβλημα δρομολόγησης.....18
ΚΕΦΑΛΑΙΟ 4. Εφαρμογή και επίλυση.....20
4.1. Εισαγωγή.....21
4.1.1. Επιλογή του Matlab.....21
4.1.2. Δεδομένα.....22
4.1.3. Κατάστρωση αλγορίθμου.....23

4.1.4. Δομή προγράμματος....23
4.2. Συναρτήσεις....24
4.2.1. MAIN.m....24
4.2.2. Data.m....24
4.2.3. MAP.m....24
4.2.4. InitialSolution.m....25
4.2.5. trails.m....25
4.2.6. Results.m....26
4.2.7. DrawLine.m....26
4.2.8. ACO.m....26
4.2.8. rearange.m....26
4.3. Εφαρμογή....27
4.3.1. AS-Simple....27
4.3.2. Elitist Ant System....28
4.3.3. AS-Rank....29
4.3.4. MAX-MIN Ant System....30
4.3.5. Ant Colony System....31
4.4. Επίλυση και αποτελέσματα....33
4.4.1. Πρόβλημα Par1 (50 πόλεις)....33
4.4.2. Προβλήματα Par2, Par3, Par4, Par11, Par12....47
4.4.3. Προβλήματα A-n32k5 μέχρι A-n80k10....50
4.5. Συμπεράσματα και προτάσεις....55
ΒΙΒΛΙΟΓΡΑΦΙΑ....56

Κεφάλαιο 1

Εισαγωγή

1.1 Η εφοδιαστική αλυσίδα

Στον σύγχρονο κόσμο όπου πρώτες ύλες, ενδιάμεσα και τελικά προϊόντα διακινούνται από και προς κάθε γωνιά του κόσμου, η επιστήμη της επιχειρησιακής έρευνας και πιο συγκεκριμένα το κομμάτι που ασχολείται με την εφοδιαστική αλυσίδα τείνει να παίξει καθοριστικό ρόλο στη δημιουργία πλεονεκτήματος έναντι του ανταγωνισμού. Η Διαχείριση εφοδιαστικής αλυσίδας ορίζεται ως η διαχείριση ενός δικτύου εσωτερικά συνδεδεμένων επιχειρήσεων που συμμετέχουν στην απώτερη παροχή πακέτων προϊόντων και υπηρεσιών, τα οποία απευθύνονται στους τελικούς καταναλωτές (Harland, 1996). Η διαχείριση εφοδιαστικής αλυσίδας εκτείνεται σε όλη τη διαδικασία μεταφοράς και αποθήκευσης των πρώτων υλών, απογραφής της εσωτερικής διαδικασίας και παροχής ολοκληρωμένων αγαθών από πλευράς προέλευσης μέχρι και την κατανάλωση τους. Περιλαμβάνει όλα τα ενδιάμεσα στάδια που στόχο έχουν την ικανοποίηση των απαιτήσεων των πελατών. Αποτελείται από προμηθευτές, κατασκευαστές, χώρους αποθήκευσης,αποθέματα, κέντρα διανομής, μεταφορείς, πωλητές, έτοιμα προϊόντα και πελάτες.

Τα ενδιάμεσα στάδια της διανομής και συγκεκριμένα οι αποφάσεις που πρέπει να ληφθούν για αυτά, δημιουργούν επιμέρους προβλήματα που η επιστήμη της εφοδιαστική αλυσίδας καλείται να επιλύσει κατά το βέλτιστο δυνατό τρόπο. Τέτοια προβλήματα αποτελούν, η διαμόρφωση του δικτύου διανομής, δηλαδή ο αριθμός, θέση και το δίκτυο αποστολών των προμηθευτών, των εγκαταστάσεων παραγωγής, των κέντρων διανομής, των αποθηκών, των αποβάθρων και των πελατών. Η στρατηγική διανομής, όπου λαμβάνονται αποφάσεις σχετικά με τον τρόπο ελέγχου των λειτουργιών, τα συστήματα διανομής και τα μέσα. Η "συμφωνία" μεταξύ των επιμέρους δραστηριοτήτων της αλυσίδας και της συμβατότητας μεταξύ αυτών, με στόχο τη συνολική βελτιστοποίηση της. Η διακίνηση των πληροφοριών μέσα στην αλυσίδα, με στοιχεία για την ζήτηση, τις προβλέψεις, το απόθεμα, τη μεταφορά. Η διαχείριση αποθεμάτων, που αφορούν την ποσότητα και τον τόπο πρώτων υλών, ενδιάμεσων και τελικών προϊόντων. Οι χρηματικές ροές, δηλαδή οι αποφάσεις που αφορούν τις πληρωμές και την διακίνηση κεφαλαίων μεταξύ των μερών της αλυσίδας.

Η διαμόρφωση του δικτύου διανομής αποτελείται από επιμέρους υποπροβλήματα, τα οποία χωρίζονται σε τρεις μεγάλες κατηγορίες. Τα προβλήματα δρομολόγησης, τα προβλήματα χρονοπρογραμματισμού και τα προβλήματα χωροθέτησης. Κάθε κατηγορία είναι και μια ομάδα προβλημάτων στα οποία υπάρχουν παραλλαγές. Για παράδειγμα, η ύπαρξη χρονικών παραθύρων, η ζήτηση σε πραγματικό χρόνο, η ταυτόχρονη διανομή και τα πολλαπλά προϊόντα. Σε κάθε εφοδιαστική αλυσίδα ενδέχεται να συνυπάρχουν ταυτόχρονα και τα τρία προβλήματα, των οποίων οι λύσεις θα πρέπει να συνδυάζονται κατάλληλα, ώστε να βελτιστοποιούν το δίκτυο διανομής στο σύνολό του.

Η εργασία αυτή αφορά την πρώτη κατηγορία προβλημάτων, αυτή της δρομολόγησης οχημάτων. Σε αυτό το είδος προβλημάτων αναζητείται η διαδρομή που θα πρέπει να ακολουθήσει κάθε όχημα, ώστε να ελαχιστοποιηθεί το συνολικό κόστος των διαδρομών όλων των οχημάτων. Η δρομολόγηση υπόκειται ταυτόχρονα σε περιορισμούς, με ποιο συχνό αυτόν της ζήτησης, η οποία θα πρέπει να ικανοποιείται και αυτός της χωρητικότητας του οχήματος. Όπως προαναφέρθηκε, οι παραλλαγές είναι πολλές και αφορούν κυρίως στους περιορισμούς που θα πρέπει να ληφθούν υπόψη. Η εργασία μελετά και επιλύει μια τέτοια παραλλαγή, το Συσσωρευτικό Πρόβλημα δρομολόγησης οχημάτων με περιορισμό χωρητικότητας, το οποίο παρουσιάζεται αναλυτικά στο επόμενο κεφάλαιο.

ΚΕΦΑΛΑΙΟ 2

Συσσωρευτικό Πρόβλημα δρομολόγησης οχημάτων (CCVRP)

2.1 Εισαγωγή

Σε πολλές περιπτώσεις στην καθημερινότητα τα προβλήματα που καλούμαστε να επιλύσουμε ξεπερνούν τα κλασικά προβλήματα ελαχιστοποίησης κόστους καθώς σε αντίθεση με αυτά που έχουν σαν στόχο την μεγιστοποίηση του κέρδους παραβλέπουν την ανάγκη για γρήγορη εξυπηρέτηση και παροχή κρίσιμων αγαθών μετά από κάποια καταστροφή. Ενώ οι εμπορικές εφοδιαστικές αλυσίδες εστιάζουν στην ποιότητα και την κερδοφορία, οι ανθρωπιστικές εφοδιαστικές αλυσίδες έχουν ώς πρώτο στόχο την ελαχιστοποίηση των ανθρώπινων απωλειών. Για το λόγο αυτό είναι πιο ανθρωποκεντρικές.

Το Συσσωρευτικό Πρόβλημα δρομολόγησης οχημάτων έχει ως στόχο την ελαχιστοποίηση του χρόνου άφιξης στους πελάτες, αντί για την κλασική ελαχιστοποίηση της απόστασης, έχοντας ως περιορισμό την χωρητικότητα του οχήματος. Το CCVRP είναι ουσιαστικά μια παραλλαγή του Traveling Repairman Problem, με την προσθήκη του περιορισμού χωρητικότητας και την ύπαρξης ομογενούς στόλου οχημάτων.

2.2 Μοντελοποίηση

Θεωρούμε το πρόβλημα CCVRP ορισμένο σε ένα γράφημα $G=(V,E,W)$ όπου $V=\{0,1,2,\ldots,n+1\}$ οι κόμβοι του γραφήματος (οι κόμβοι 0,n+1 είναι η αποθήκη), $V'=V/\{0,n+1\}$ οι κόμβοι των πελατών, Ε είναι το διάνυσμα των τόξων σε κάθε ένα εκ των οποίων αντιστοιχεί ένα βάρος $w_{ij}=w_{ji}\in W$ που αντιστοιχεί στο κόστος ή την χρονική απόσταση ανάμεσα στους κόμβους i και j [2].
Άλλοι παράμετροι του προβλήματος είναι :

- q_i η ζήτηση του κόμβου i
- m ο αριθμός των όμοιων οχημάτων
- Q η χωρητικότητα των οχημάτων

Το αντικείμενο του CCVRP είναι η εύρεση των δρομολογίων κατά τέτοιο τρόπο ώστε κάθε πελάτης να εξυπηρετείται μόνο μια φορά και το άθροισμα των χρόνων άφιξης σε αυτούς να ελαχιστοποιείται. Ένα δρομολόγιο ορίζεται ως ένα κύκλωμα που αρχίζει και τελειώνει στην αποθήκη, με τη συνολική ζήτηση των επισκεπτόμενων κόμβων να μην ξεπερνά την χωρητικότητα Q. Ορίζουμε t_i^k το χρόνο άφιξης του οχήματος k στον κόμβο i και x_{ij} τη δυαδική μεταβλητή ίση με 1 αν το όχημα μεταβαίνει από τον κόμβο i στον j . Παρακάτω γίνεται η μαθηματική απεικόνιση του προβλήματος και παρουσιάζονται οι περιορισμοί του.

2.3 Μαθηματική απεικόνιση

Μαθηματικό μοντέλο :

$$F: Min \sum_{k=1}^{m} \sum_{i \in V'} t_i^k \quad (1)$$

υπό περιορισμούς

$$\sum_{i \in V} x_{ij}^k = \sum_{i \in V} x_{ji}^k, \quad \forall j \in V', \forall k \in [1, \ldots m] \quad (2)$$

$$\sum_{k=1}^{m} \sum_{j \in V} x_{ij}^k = 1, \quad \forall i \in V' \quad (3)$$

$$\sum_{i \in V'} \sum_{j \in V} x_{ij}^k q_i \leq Q, \quad \forall k \in [1, \ldots, m] \quad (4)$$

$$\sum_{j \in V} x_{0\mathrm{j}}^k = 1, \quad \forall k \in [1, \ldots, m] \quad (5)$$

$$\sum_{i \in V} x_{i,n+1}^k = 1, \quad \forall k \in [1, \ldots, m] \quad (6)$$

$$t_i^k + w_{ij} - (1 - x_{ij}^k) T \leq t_j^k, \quad \forall i \in V / \{n+1\}, \forall j \in V, \forall k \in [1, \ldots m] \quad (7)$$

$$t_l \geq 0, \quad \forall i \in V, \forall k \in [1, \ldots, m] \quad (8)$$

$$x_{ij} = 0,1 \quad \forall i \in V, \forall j \in V, i \neq j, \forall k \in [1, \ldots m] \quad (9)$$

$$x_{ij} = 0 \, \acute{\eta} \, 1, \quad (i, j) \in A \quad (10)$$

Αναλυτικά, η αντικειμενική εξίσωση (1) αποτελεί την προς ελαχιστοποίηση συνάρτηση κόστους. Οι περιορισμοί (2) καθορίζουν ότι τα οχήματα που επισκέπτονται τον κόμβο i πρέπει να αποχωρήσουν από αυτόν. Οι περιορισμοί (3) εξασφαλίζουν οτι κάθε πελάτης εξυπηρετείται μία φορά ακριβώς. Οι περιορισμοί (4) είναι περιορισμοί χωρητικότητας. Ο συνολικός φόρτος της διαδρομής δεν μπορεί να υπερβαίνει την χωρητικότητα τους οχήματος. Οι περιορισμοί (5) και (6) εξασφαλίζουν ότι κάθε διαδρομή αρχίζει και τελειώνει στην αποθήκη. Τέλος οι (7) υπολογίζουν τον χρόνο άφιξης σε κάθε κόμβο και εξασφαλίζουν οτι δεν θα υπάρχουν υπο-διαδρομές με τη βοήθεια ενός μεγάλου αριθμού $T \gg t_i^k + w_{ij}$.

ΚΕΦΑΛΑΙΟ 3

Βελτιστοποίηση με τον αλγόριθμο της Αποικίας μυρμηγκιών

3.1 Αλγόριθμοι

3.1.1 Αλγόριθμοι βελτιστοποίησης

Οι περισσότεροι συμβατικοί ή κλασικοί αλγόριθμοι βελτιστοποίησης είναι ντετερμινιστικοί ή αλλιώς αιτιοκρατικοί. Δηλαδή κάθε δράση που λαμβάνει χώρα συνδέεται με αιτιώδη αλυσιδωτή σχέση με τις προηγούμενες κατά απόλυτο τρόπο. Για παράδειγμα η μέθοδος simplex που χρησιμοποιείται στην επίλυση προβλημάτων γραμμικού προγραμματισμού είναι αιτιοκρατική. Μερικοί αιτιοκρατικοί αλγόριθμοι βελτιστοποίησης χρησιμοποιούν την πληροφορία της κλίσης (gradient). Γνωστό παράδειγμα gradient-based αλγόριθμου είναι ο Newton-Raphson, καθώς χρησιμοποιεί τις τιμές της συνάρτησης και τις παραγώγους της. Ένας τέτοιος αλγόριθμος λειτουργεί πολύ αποτελεσματικά για ομαλά μονοτροπικά προβλήματα, ωστόσο σε περιπτωση που υπάρχει ασυνέχεια στην αντικειμενική συνάρτηση δεν είναι αποδοτικός και αντ΄ αυτού προτιμάτε ένας non-gradient αλγόριθμος που δεν χρειάζεται κάποια παράγωγο αλλά μόνο τις τιμές της συνάρτησης. Παραδείγματα gradient-free αλγορίθμων είναι ο Hooke-jeeves pattern search και η μέθοδος Nelder-Mead (downhill simplex method).

Οι στοχαστικοί αλγόριθμοι, δηλαδή οι μη-ντετερμινιστικοί, είναι δυο ειδών, οι ευρετικοί και οι μεθευρετικοι. Οι πρώτοι, ψάχνουν την βέλτιστη λύση μέσω της "δοκιμής και του σφάλματος" (trial & error) και ως εκ τούτου μπορούν να βρουν μια αρκετά καλή λύση σε λογικό χρόνο, χωρίς όμως να μας εγγυούνται οτι είναι η βέλτιστη. Αυτό διότι είναι αδύνατον σε περίπλοκα προβλήματα να εξετάσει όλες τις εφικτές λύσεις.

Εξέλιξη των ευρετικών αλγορίθμων αποτελούν οι μεθευρετικοί οι οποίοι είναι πιο αποτελεσματικοί και η βασική διαφορά τους είναι ότι συνδυάζουν την τοπική αναζήτηση (local search) βέλτιστης λύσης με κάποια γενικότερη στρατηγική αναζήτησης ώστε να ξεφύγουν από το τοπικό βέλτιστο (local optimum) και να προσδιορίσουν το ολικό (global optimum).

3.1.2 Μεθευρετικοί Αλγόριθμοι

Τα δυο κύρια συστατικά κάθε μεθευρετικού αλγορίθμου είναι : η επίταση (intensification) και η διαφοροποίηση(diversification) ή εκμετάλλευση (exploitation) και εξερεύνηση (exploration). Η **επίταση** είναι η εστίαση στην τοπική αναζήτηση εκμεταλλευόμενοι την πληροφορία οτι βρίσκουμε μια καλή λύση σε αυτήν την περιοχή.
Αντιθέτως η **διαφοροποίηση** σημαίνει η εξερεύνηση του χώρου των λύσεων προς αναζήτηση νέας βέλτιστης λύσης. Η διαφοροποίηση εξασφαλίζει ότι ο αλγόριθμος δεν θα παγιδευτεί σε ενα τοπικό ελάχιστο. Με τη σωστή αναλογία των δύο παραπάνω στοιχείων του αλγορίθμου εξασφαλίζεται η εύρεση του ολικού βέλτιστου.

Οι μεθευρετικοί αλγόριθμοι μπορούν να κατηγοριοποιηθούν με πολλούς τρόπους. Ένας από αυτούς είναι οι αλγόριθμοι που χρησιμοποιούν πληθυσμούς (population-based) και εκείνοι που χρησιμοποιούν την τροχιά(trajectory based). Παράδειγμα του πρώτου είδους ειναι μεθοδος βελτιστοποίησης σμήνους (particle swarm optimization) χρησιμοποιώντας έναν πληθυσμό υποψηφίων λύσεων που τον μετακινεί στο χώρο αναζήτησης. Αντίθετα ένας αλγόριθμος βασισμένος στην τροχιά, είναι η προσημειωμένη ανόπτηση (simulated annealing), που χρησιμοποιεί ένα στοιχείο –υποψήφια λύση – την οποία μετακινεί τμηματικά στον χώρο των λύσεων. Μια μετακίνηση σε καλύτερη λύση γίνεται πάντα αποδεκτή, ενώ μια μέτρια μετακίνηση γίνεται αποδεκτή με κάποια πιθανότητα.

3.1.3 Αλγόριθμοι εμπνευσμένοι από τη φύση

Στη βιβλιογραφία [8] υπάρχει πληθώρα αλγορίθμων που βασίζονται σε διαδικασίες που συμβαίνουν στη φύση και εμπνέονται από επιστήμες όπως η χημεία, η φυσική και κυρίως η βιολογία. Τέτοιοι είναι :

- Οι **Εξελικτικές Στρατηγικές** (Evolutionary Strategies) και οι Γενετικοί Αλγόριθμοι (Genetic Algorithms), που προέρχονται απο την μίμηση της διαδικασίας της εξέλιξης των ειδών στη φύση. Κάτ' αναλογία με τη θεωρία του Δαρβίνου της φυσικής διαλογής, στην οποία επιβιώνει ο ισχυρότερος, έτσι και στην επίλυση του προβλήματος επιδιώκουμε την επικράτηση της βέλτιστης λύσης. Οι δύο αυτοί αλγόριθμοι βασίζονται στο ίδιο σκεπτικό και διαφοροποιούνται στον τρόπο με τον οποίο γίνεται η διαλογή των ατόμων-λύσεων, η διασταύρωση και η μετάλλαξη τους. Η διαλογή αποτελεί το μηχανισμό επίτασης (intensification) του αλγορίθμου, ενώ η διασταύρωση και η μετάλλαξη καθορίζουν τη διαφοροποίηση (diversification).

- Τα **Νευρωνικά Δίκτυα** (Neural Nets) προέρχονται από τον τρόπο λειτουργίας του ανθρωπίνου εγκεφάλου. Ο εγκέφαλος αποτελείται από νευρώνες (10.000.000.000 κατά προσέγγιση), οι οποίοι είναι μαζικά διασυνδεδεμένοι με ένα μέσο όρο από διάφορες χιλιάδες διασυνδέσεις ανά νευρώνα. Κάθε νευρώνας είναι ένα εξειδικευμένο κύτταρο το οποίο έχει τη δυνατότητα μετάδοσης ενός ηλεκτροχημικού σήματος. Ένας νευρώνας πυροδοτεί ένα ηλεκτροχημικό σήμα όταν το συνολικό σήμα το οποίο λήφθηκε στο ίδιο, υπερβεί ένα συγκεκριμένο επίπεδο, δηλαδή, το κατώτατο όριο βολής (firing threshold).

- Η **Προσομοιωμενη Ανόπτηση** (Simulated Annealing) που πρώτος χρησιμοποίησε ο kirkpatrick, βασίζεται στην διαδικασία ανόπτησης των μετάλλων, δηλαδή στην θέρμανση τους σε υψηλές θερμοκρασίες και στη συνέχεια η αργή ψύξη τους σε θερμοκρασία δωματίου. Έτσι στην βελτιστοποίηση κάθε λύση αποτελεί μια ενεργειακή στάθμη και ο αλγόριθμος επιδιώκει να βρεθεί στην χαμηλότερη ενεργειακή στάθμη δηλαδή τη βέλτιστη λύση.

- Ο **Αλγόριθμος βελτιστοποίησης Σμήνους Σωματιδίων** (particle swarm optimization) προσομοιώνει την κοινωνική συμπεριφορά ορισμένων οργανισμών στη φύση, όπως τα σμηνοι πουλιών και τα κοπάδια ψαριών. Χαρακτηριστικό του αλγορίθμου είναι η ύπαρξη μνήμης που κληρονομείτε από γενιά σε γενιά. Αυτό έχει σαν πλεονέκτημα οτι λύσεις που έχουν απορριφθεί δεν επαναλαμβάνονται.

- Ο **Αλγόριθμος βελτιστοποίησης Ζευγαρώματος Μελισσών** (Honey Bees mating optimization) είναι ένας σύγχρονος αλγόριθμος που αντιγράφει τη διαδικασία με την οποία η βασίλισσα κάνει την τυχαία πτήση ζευγαρώματος, ακολουθούμενη απο το σμήνος των κηφήνων και επιλέγει με βάση την ταχύτητα της, τον κηφήνα με τον οποίο θα ζευγαρώσει.

- Ο **Αλγόριθμος της Αποικίας Μυρμηγκιών** (ant colony optimization), όπως διαφαίνεται και από το όνομα του, είναι εμπνευσμένος από την συμπεριφορά των μυρμηγκιών στη φύση και συγκεκριμένα από τον τρόπο με τον οποίο αυτά αναζητούν το βέλτιστο μονοπάτι μεταξύ της αποικίας και της πηγής της τροφής.

3.2 Τα κοινωνικά έντομα

3.2.1 Αυτό-οργάνωση

Αυτό-οργάνωση είναι η διαδικασία όπου κάποιας μορφής τάξη ανακύπτει στο "γενικό επίπεδο", μέσω των αλληλεπιδράσεων ανάμεσα στα στοιχεία που βρίσκονται στο "τοπικό επίπεδο", ενός αρχικά ανοργάνωτου συστήματος[8].

Αυτή η διαδικασία είναι αυθόρμητη, δηλαδή δεν ελέγχεται, ούτε καθοδηγείται από οποιοδήποτε στοιχείο εντός του συστήματος αλλά ούτε και εκτός αυτού. Παρόλα αυτά οι κανόνες που διέπουν το σύστημα και οι αρχικές του συνθήκες μπορεί να έχουν επιλεγεί η προκληθεί από κάποιον εντός ή εκτός του συστήματος.

Η αυτό-οργάνωση ως έννοια, συναντάται σε πληθώρα θετικών επιστημών όπως η φυσική, η χημεία, τα μαθηματικά, η οικονομία, η πληροφορική αλλά και σε πολλές κοινωνικές επιστήμες. Στην εργασία αυτή, θα αναλύσουμε την αυτο-οργάνωση από τη σκοπιά της βιολογίας και συγκεκριμένα, πως αυτή λαμβάνει χώρα σε μια αποικία μυρμηγκιών.

Ένα αυτο-οργανωμένο σύστημα, περιλαμβάνει τα εξής χαρακτηριστικά.

- Την **Θετική Ανατροφοδότηση**. (Την ενίσχυση). Δηλαδή την συμπεριφορά που ενισχύει τη δημιουργία μιας δομής. Στην περίπτωση των μυρμηγκιών και συγκεκριμένα στην αναζήτηση της τροφής τους, ένα παράδειγμα ενίσχυσης είναι η προτροπή με κάποιον τρόπο και άλλων μυρμηγκιών να ακολουθήσουν το ίδιο μονοπάτι όπου ένα μυρμήγκι βρήκε τροφή.

- Την **Αρνητική Ανατροφοδότηση.** (απόσβεση). Ουσιαστικά το αντίβαρο της ενίσχυσης, την αποθάρρυνση από τη δημιουργία της δομής. Παράδειγμα απόσβεσης στον πληθυσμό των μυρμηγκιών είναι η περίπτωση όπου μυρμήγκι ειδοποιεί τα υπόλοιπα ότι τα αποθέματα της τροφής στην συγκεκριμένη πηγή εξαντλούνται, μειώνονται ή υπάρχει συνωστισμός.

- Τις **Διακυμάνσεις**. Με άλλα λόγια ο βαθμός τυχαιότητας με την οποία δημιουργούνται οι δομές. Συγκεκριμένα στην αναζήτηση τροφής των μυρμηγκιών, το κατά πόσο θα ακολουθήσει ένα μυργμήγκι την προτροπή ενός άλλου να ακολουθήσει συγκεκριμένο μονοπάτι, παίζει τεράστιο ρόλο στην αποτελεσματικότητα της αναζήτησης, καθότι χωρίς την ύπαρξη οποιασδήποτε τυχαιότητας θα περιοριζόντουσαν μόνο στην πηγή τροφής που θα έβρισκε το πρώτο μυρμήγκι. Επομένως τα "λάθη" και οι αποκλίσεις στις διαδρομές δεν αποτελούν απαραίτητα κάτι αρνητικό.

- Τις **Αλληλεπιδράσεις**. Το κάθε άτομο από μόνο του, μπορεί να δημιουργήσει μια δομή, ωστόσο σε ένα αυτοοργανομενο σύστημα οφείλει να ανταλλάσσει πληροφορίες με τον υπόλοιπο πληθυσμό. Δηλαδή να επηρεάζει με το έργο του και να επηρεάζεται από το έργο των άλλων.

3.2.2 Στιγμεργία

Στην προηγούμενη παράγραφο αναφερθήκαμε στην ανάγκη ύπαρξης αλληλεπίδρασης μεταξύ των μυρμηγκιών,δηλαδή κάποιο τρόπο επικοινωνίας. Αυτή μπορεί να είναι άμεση ή έμμεση. Η άμεση επικοινωνία συνίσταται στην ανταλλαγή της πληροφορίας μέσω την οπτικής επαφής των εντόμων, ή με την ανταλλαγή χημικών ουσιών μεταξύ τους. Η έμμεση επικοινωνία συνίσταται στην πληροφορία που μεταβιβάζουν μεταβάλλοντας το περιβάλλων τους και κατ επέκταση συνδιαμορφώνουν με τα άλλα μέλη του πληθυσμού.

Πρωτοπόρος στην μελέτη της κοινωνικής συμπεριφοράς των εντόμων ήταν τη δεκαετία του 1940 ο Γάλλος εντομολόγος Pierre-Paul Grasse[1]. Το 1946 ανακάλυψε ότι τα έντομα ήταν ικανά να αντιδρούν κατά ένα συγκεκριμένο τρόπο, γενετικά καθορισμένο, σε ερεθίσματα στο περιβάλλον τους. Αργότερα, το 1959, διαπίστωσε ότι τα αποτελέσματα αυτών των αντιδράσεων λειτουργούν ως νέα ερεθίσματα τόσο για τα έντομα που τα παρήγαγαν όσο και για τα υπόλοιπα.

Ο όρος που χρησιμοποίησε για να περιγράψει "το ερέθισμα που προκαλούν οι εργάτες (κοινωνική τάξη των τερμιτών) στο περιβάλλον τους, ως αποτέλεσμα της εργασίας τους" είναι η στιγμεργία. Αυτό που διαφοροποιεί την στηγμεργία από άλλους τρόπους επικοινωνίας είναι :

(a) Η απτή, μη συμβολική φύση της πληροφορίας όπου αντιστοιχεί στην αλλαγή που επιφέρει το έντομο στο περιβάλλων που επισκέπτεται.
(b) Η τοπική φύση της πληροφορίας, όπου πρόσβαση έχουν μόνο τα έντομα που θα βρεθούν στον τόπο που εκείνη έχει εναποθετηθεί.

Η στιγμεργία παρατηρείται και στις αποικίες των μυρμηγκιών. Σε πολλά είδη μυρμηγκιών, καθώς αυτά περπατάνε από και προς την πηγή τροφής, εναποθέτουν στο έδαφος την ουσία φερομόνη. Τα υπόλοιπα μυρμήγκια, οσμίζονται την φερομόνη και η παρουσία της επηρεάζει την απόφαση τους για τον ποιο δρόμο θα ακολουθήσουν.

3.2.3 Αναζήτηση τροφής της αποικίας μυρμηγκιών

Τα μυρμήγκια δημιουργούν μονοπάτια φερομονης κατά τη διάρκεια αναζήτησης της πηγής της τροφής και επιστρέφουν στη φωλιά αφήνοντας και πάλι το στίγμα τους στη διαδρομή που ακολουθούν. Τα υπόλοιπα μυρμήγκια εντοπίζουν αυτά τα μονοπάτια και ανάλογα την ένταση της φερομόνης σε αυτά, αποφασίζουν με κάποια πιθανότητα αν θα τα ακολουθήσουν ή όχι.

Το 1990 ο deneubourg έκανε ένα πείραμα ονόματι "το πείραμα της δυαδικής γέφυρας" (binary bridge experiment), προκειμένου να μελετήσει την συμπεριφορά των μυρμηγκιών και να ποσοτικοποιήσει την αλληλεπίδραση τους με τη φερομόνη. Χρησιμοποιώντας μυρμήγκια Linepithema humile, συνέδεσε την φωλιά των μυρμηγκιών με την πηγή της τροφής, μέσω 2 γεφυρών ίσου μήκους. Τα μυρμήγκια ελεύθερα μπορούσαν να επιλέξουν από ποια γέφυρα θα έψαχναν για την τροφή.

Αρχικά οι γέφυρες δεν είχαν φερομόνη. Τα μυρμήγκια εξερευνούν τον χώρο και τελικά περνάνε μία από τις γέφυρες. Καθώς κινούνται από και προς την τροφή, εναποθέτουν φερομόνη στη γέφυρα που χρησιμοποιούν. Με το πέρας του χρόνου,λόγο της διακύμανσης, τα επίπεδα φερομόνης στη μία γέφυρα θα είναι ισχυρότερα. Τα μυργμήγκια τείνουν να ακολουθήσουν το μονοπάτι με την εντονότερη φερομένη, επομένως η αποικία μυρμηγκιών συγκλίνει στην χρησιμοποίηση μιας μόνο γέφυρας.

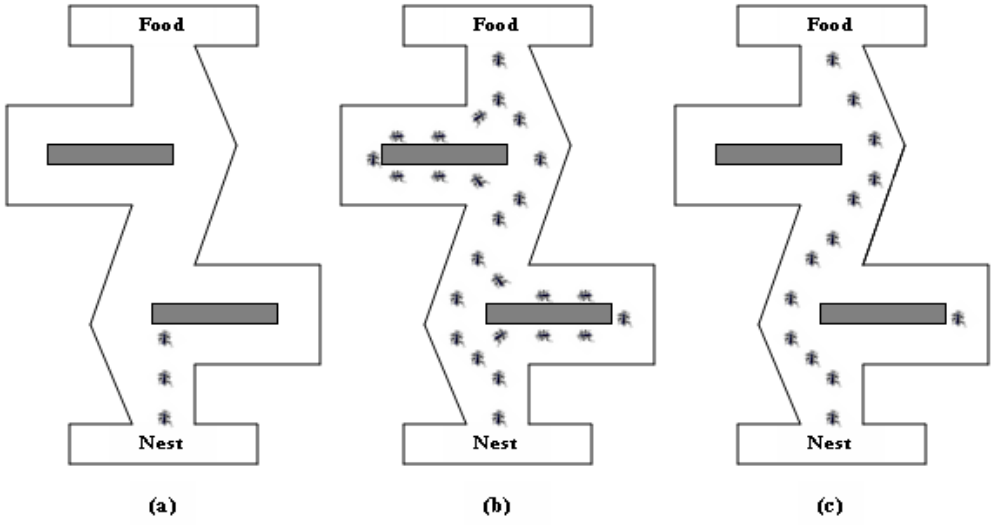
Σε ένα άλλο πείραμα το 1989, ο Goss, χρησιμοποίησε δυο γέφυρες διαφορετικού μήκους. Παρατήρησε οτι η τυχαία διακύμανση στην αρχική επιλογή γέφυρας πέρασε σε δεύτερη μοίρα, καθώς, τα μυρμήγκια που κατά τύχη διάλεγαν την συντομότερη γέφυρα για να φτάσουν στην τροφή, γυρνούσαν στη φωλιά από την ίδια με μεγαλύτερη πιθανότητα. Έτσι χάρις την φερομόνη σύντομα όλα τα μυρμήγκια συνέκλιναν στην χρήση της συντομότερης διαδρομής.

Αναλυτικότερα η διαδικασία εύρεσης τροφής των μυρμηγκιών είναι η εξής:

1. Τα μυρμήγκια ξεκινούν την αναζήτηση τροφής γύρο από τη περιοχή της πηγής της τροφής κατά τυχαίο τρόπο, επιδιώκοντας της εύρεση της συντομότερης διαδρομής.
2. Κατά τη διάρκεια της περιπλάνησης τους εκκρίνουν τη φερομόνη στο μονοπάτι που ακολουθούν. Η ποσότητα της φερομόνης εξαρτάται από την απόσταση, την ποιότητα και την ποσότητα της τροφής που υπάρχει στην πηγή.
3. Το επόμενο μυρμήγκι που θα ξεκινήσει την αναζητη τροφης, επιλέγει ή οχι με κάποια πιθανότητα, το μονοπάτι που είδη έχει φερομόνη.
4. Το μυρμήγκι θα αφήσει και αυτό με τη σειρά του το αποτύπωμα φερομόνης στο μονοπάτι που θα ακολουθήσει. Αν επιλέξει ένα ήδη μαρκαρισμένο μονοπάτι, τότε αυτό θα ενισχυθεί. Καθώς η ώρα περνάει, η φερομόνη εξατμίζεται και επομένως τα μονοπάτια που δεν είναι δελεαστικά για τα μυρμήγκια εξαλείφονται.
5. Εν τελη, απο όλα τα αρχικά μονοπάτια, υπερισχύει ένα μόνο, αυτό της συντομότερης διαδρομής.

Ορισμένα είδη μυρμηγκιών καθώς πηγαίνουν προς την πηγή της τροφής από ένα νέο μονοπάτι (χωρίς φερομόνη), εναποθέτουν τη φερομόνη οχι συνεχόμενα, αλλά σε συγκεκριμένες αποστάσεις. Κατά την επιστροφή τους στη φωλιά, αν ακολουθήσουν το μονοπάτι από το οποίο ήρθαν, θα εναποθέσουν φερομόνη συνεχόμενα και σύμφωνα με τα κριτήρια που προαναφέρθηκαν. Επίσης όταν το πρώτο μυρμήγκι εντοπίσει τροφή, κρατάει μια ποσότητα στο στόμα του και όταν επιστρέφει στην φωλιά ταΐζει μερικά από τον υπόλοιπο πληθυσμό ώστε να τα "ειδοποιήσει" ότι το μονοπάτι που δημιούργησε, οδηγεί σε τροφή.

Από αυτήν την διαδικασία και κατά αναλογία με τον τρόπο που τα μυρμήγκια βρίσκουν την συντομότερη διαδρομή, δημιουργήθηκαν αλγόριθμοι που στόχο έχουν την εύρεση της βέλτιστης λύσης σε πολλές κατηγορίες προβλημάτων.



3.3 Ο Αλγόριθμος της Αποικίας Μυρμηγκιών (ACO)

3.3.1 Ιστορικη αναδρομή

Ο πρώτος αλγόριθμος βασισμένος στη διαδικασία εύρεσης τροφής των μυρμηγκιών παρουσιάστηκε το 1992 στην διδακτορική διατριβή του M.Dorigo το 1992 στο Πολυτεχνείο του Μιλάνου και ονομαζόταν Ant System. Ο αλγόριθμος αυτός, εμπνευσμένος από τις παρατηρήσεις του Deneubourg και του Gross πάνω στα κοινωνικά έντομα και την έμμεση επικοινωνία των μυρμηγκιών με την μέθοδο της στιγμεργιας για την αναζήτηση τροφής, στα επόμενα χρόνια εξελίχθηκε σε αυτό που σήμερα είναι γνωστό ως Ant Colony Optimization.

Ο αλγόριθμος Ant System επιλύει διακριτά προβλήματα, όπως για παράδειγμα το πρόβλημα του πλανόδιου πωλητή (TSP), με πάρα πολύ καλά αποτελέσματα. Αργότερα υπήρξαν πολλές παραλλαγές του, όπως ο Elitist Ant System όπου η καλύτερη λύση ενισχύεται σε κάθε επανάληψη, με αποτέλεσμα την ταχύτερη σύγκλιση. Μια άλλη παραλλαγή είναι ο Max-Min Ant System, όπου όπως προδίδει και η ονομασία του, τα επίπεδα φερομόνης κινούνται μεταξύ 2 τιμών. Άλλες τροποποιήσεις είναι οι αλγόριθμοι Rank-based Ant System, Continuous Orthogonal Ant Colony και ACO with Fuzzy Logic με τον τελευταίο να συνδυάζει τον αλγόριθμο των μυρμηγκιών με την ασαφή λογική προκειμένου τα μυρμήγκια να είναι αποτελεσματικότερα στην αναζήτηση των μονοπατιών που θα ακολουθήσουν.

3.3.2 Η “ψηφιοποίηση” της διαδικασίας

Η μετατροπή της φυσικής διαδικασίας εύρεσης της βέλτιστης διαδρομής σε αλγόριθμο υπολογισμού της σε κάποιο ψηφιακό μοντέλο, προϋποθέτει ορισμένες τροποποιήσεις. Τα μυρμηγκιά που χρησιμοποιεί ο αλγόριθμος θα είναι απλουστευμένα σε σχέση με τα πραγματικά ωστόσο έχουν επιπλέον ιδιότητες, ανάλογα το πρόβλημα που επιλύουν.

Οι αντιστοιχίες φυσικών ιδιοτήτων με τις ψηφιακές είναι οι εξής :

- Ο αλγόριθμος χρησιμοποιεί ένα πληθυσμό από **μυρμήγκια** όπου κάθε ένα από αυτά θα ακολουθήσει ένα μονοπάτι το οποίο είναι μια λύση του προβλήματος. Ο πληθυσμός των μυρμηγκιών είναι μια σημαντική παράμετρος του αλγορίθμου που επηρεάζει το χρόνο προσέγγισης της βέλτιστης λύσης.

- Η **φερομόνη** στον αλγόριθμο βελτιστοποίησης της Αποικίας Μυρμηγκιών αντιστοιχεί σε μια αριθμητική πληροφορία που θα εναποθέτουν τα μυρμήγκια και θα είναι προσβάσιμη από όλα τα υπόλοιπα, σχετικά με τη διαδρομή που ακολούθησαν. Αυτή η πληροφορία θα πρέπει να φθίνει στο χρόνο με τρόπο αντίστοιχο της εξάτμισης της φερομόνης στη φύση.

- Η **Επιλογή** της διαδρομής που θα ακολουθήσουν δεν είναι καθαρά αιτιοκρατικός, αλλά είναι σε ένα βαθμό τυχαίος. Τα ψηφιακά μυρμήγκια δεν έχουν μνήμη και επομένως η επιλογή της διαδρομής που θα ακολουθήσουν (στο αιτιοκρατικό της μέρος), εξαρτάται μόνο από την πληροφορία που λαμβάνει από στο συγκεκριμένο χώρο, τον συγκεκριμένο χρόνο και δεν γνωρίζει τι συνέβαινε σε παρελθόντα χρόνο ούτε τη πληροφορία υπάρχει στον υπόλοιπο χώρο.

Τα ψηφιακά μυρμήγκια εκτός από τα βασικά χαρακτηριστικά τους, μπορούν ανάλογα με την εφαρμογή να εφοδιαστούν με επιπλέον ιδιότητες, που θα επηρεάζουν την ταχύτητα σύγκλισης στη βέλτιστη λύση και γενικά την απόδοση του αλγορίθμου.

Τα μυρμήγκια του αλγορίθμου μπορούν να αποκτήσουν μνήμη των προηγούμενων πράξεων τους και μονοπατιών που έχουν διασχίσει. Επίσης η ύπαρξη όρασης στο χώρο πέραν του σημείου που βρίσκονται, και ο υπολογισμός των αποστάσεων προτού διασχίσουν μια διαδρομή. Τα ψηφιακά μυρμήγκια έχουν την ικανότητα να εναποθέτουν τη φερομόνη κατά τη διάρκεια της περιπλάνησης τους ή μετά το πέρας αυτής. Ένα επιπλέον προσόν είναι η ικανότητα να χρησιμοποιούν αλγορίθμους ώστε να βελτιστοποιούν τοπικά τη διαδρομή.

3.3.3 Εφαρμογές του ACO

Αλγόριθμοι ACO εφαρμόζονται σε πληθώρα προβλημάτων συνδυαστικής βελτιστοποίησης και έχουν επεκταθεί στην επίλυση δυναμικών προβλημάτων, στοχαστικών προβλημάτων και προβλημάτων όπου χρησιμοποιούνται συμπληρωματικά. Έχουν σημαντικό πλεονέκτημα έναντι άλλων αλγορίθμων όπως για παράδειγμα της Προσομοιωμένης Ανόπτησης και των γενετικών αλγορίθμων όσο αφορά προβλήματα που αλλάζουν δυναμικά, διότι η αποικία μυρμηγκιών προσαρμόζεται σε αλλαγές σε πραγματικό χρόνο. Για το λόγο αυτό βρίσκει πολλές εφαρμογές σε δίκτυα αστικών συγκοινωνιών.

Ο πρώτος αλγόριθμος βελτιστοποίησης που χρησιμοποιεί την αποικία μυρμηγκιών είναι ο Ant System. Ο στόχος του αρχικά ήταν η επίλυση του προβλήματος του πλανόδιου πωλητή (TSP). Ωστόσο η εξέλιξη του, οι αλγόριθμοι ACO επιλύουν μια τεράστια γκάμα προβλημάτων πολλών κατηγοριών. Ενδεικτικά αναφέρονται :

<u>Προβληματα Χρονοπρογραμματισμου -Scheduling Problems</u>

- Πρόβλημα προγραμματισμού & ελέγχου παραγωγής σε σταθμό εργασίας,Job-shop scheduling problem (JSP)
- Ανοιχτό πρόβλημα προγραμματισμού & ελέγχου παραγωγής,Open-shop scheduling problem (OSP)
- Μεταθετικό πρόβλημα προγραμματισμού γραμμής παραγωγής,flow shop problem (PFSP)
- Πρόβλημα ολικής βραδύτητας μιας μηχανής,Single machine total tardiness problem (SMTTP)
- Προγραμματισμός εργασιών με περιορισμούς πόρων,Resource-constrained project scheduling problem (RCPSP)
- Πρόβλημα προγραμματισμού παραγωγής σε πολλαπλούς σταθμούς εργασίας,Group-shop scheduling problem (GSP)

Προβλήματα Δρομολογησης Οχημάτων(ΠΔΟ)-Vehicle Routing Problems

- ΠΔΟ με περιορισμό χωρητικότητας, Capacitated vehicle routing problem (CVRP)
- ΠΔΟ, με πολλαπλές αποθήκες Multi-depot vehicle routing problem (MDVRP)
- Περιοδικό ΠΔΟ,Period vehicle routing problem (PVRP)
- Split delivery vehicle routing problem (SDVRP)
- Στοχαστικό ΠΔΟ,Stochastic vehicle routing problem (SVRP)
- ΠΔΟ με παραλαβή και διανομή Vehicle routing problem with pick-up and delivery (VRPPD)
- ΠΔΟ με χρονικά παράθυρα,Vehicle routing problem with time windows (VRPTW)
- Χρονικά εξαρτώμενο ΠΔΟ με χρονικά παράθυρα, Time Dependent Vehicle Routing Problem with Time Windows (TDVRPTW)

Προβλήματα Αναθεσης -Assignment Problem

- Πρόβλημα τετραγωνικής ανάθεσης,Quadratic assignment problem(QAP)
- Γενικό πρόβλημα ανάθεσης,Generalized assignment problem(GAP)
- Πρόβλημα ανάθεσης με συχνότητες,Frequency assignment problem(FAP)
- Πρόβλημα κατανομής εφεδρείας, Redundancy allocation problem(RAP)

Επίσης χρησιμοποιούνται σε εφαρμογές Classification, Data minning, Image processing και πολλές άλλες.

3.3.4 Μαθηματική εφαρμογή του ACO στο πρόβλημα δρομολόγησης

Στα προβλήματα δρομολόγησης οχημάτων (VRP), το ζητούμενο είναι η εύρεση της βέλτιστης οικονομικά,χρονικά ή χωρικά διαδρομής που πρέπει να ακολουθήσουν οχήματα με συγκεκριμένη αφετηρία (και κατάληξη) προκειμένου να ικανοποιήσουν μια δεδομένη ζήτηση, τηρώντας ορισμένους περιορισμούς.

Σε αυτού του είδους τα προβλήματα η μαθηματική εφαρμογή του ACO γίνεται ως εξής [9]:

Το μυρμήγκι-λυση έχοντας ως αφετηρία μια πόλη/αποθήκη i αποφασίζει να κινηθεί προς κάποια πόλη j ώστε να διανείμει μια ποσότητα προϊόντος. Προκειμένου να διαλέξει ποια θα είναι αυτή, χρησιμοποιεί 2 πληροφορίες. Η μια πληροφορία είναι η ευρετική πληροφορία $n_{ij}=1/c_{ij}$ όπου εκφράζει το πόσο θελκτική είναι η επιλογή αυτής διαδρομής ως συνάρτηση αντιστρόφως ανάλογης του κόστους (ή του χρόνου ή της απόστασης) c_{ij} μεταξύ αφετηρίας και προορισμού. Η δεύτερη πληροφορία είναι τα επίπεδα φερομόνης τ_{ij} που υπάρχουν στην συγκεκριμένη διαδρομή. Η ποσότητα τ_{ij} που εντοπίζει το μυρμήγκι προέρχεται από την φερομόνη που που έχουν αφήσει τα μυρμήγκια που περάσαν από εκεί σε προηγούμενες χρονικές στιγμές. Όσο περισσότερη, τόσο πιο συμφέρουσα την θεώρησαν τα υπόλοιπα μυρμήγκια.

Έχοντας αυτά τα δυο δεδομένα το μυρμήγκι-λύση ακολουθεί τη διαδρομή (i, j) με πιθανότητα ίση με $p_{ij}=\frac{[\tau_{ij}]^a[n_{ij}]^b}{\sum_{l=1}^{M}[\tau_{lj}]^a[n_{lj}]^b}$ όπου M το σύνολο των πόλεων που μπορεί να μεταβεί το μυρμήγκι μετά τον κόμβο i και a,b παράμετροι που καθορίζουν το βαθμό τυχαιότητα στην επιλογή της διαδρομής. Αν το $\alpha=0$ τότε η απόφαση δεν λαμβάνει υπόψιν τη φερομόνη που υπάρχει στο μονοπάτι αυτό. Αντιθέτως αν $b=0$ τότε η απόφαση επαφίεται καθαρά στην πληροφορία για το πόσο καλή είναι η διαδρομή αυτή με βάση τα προηγούμενα μυρμήγκια.

Όπως συμβαίνει και στη φύση, η φερομόνη εξατμίζεται με το πέρασμα του χρόνου. Στον αλγόριθμο αυτό συμβαίνει σε κάθε επανάληψη, δηλαδή όταν όλα τα μυρμήγκια έχουν ολοκληρώσει μια λύση του προβλήματος. Ο συντελεστής εξάτμισης συμβολίζεται με ρ και εφαρμόζεται σε όλες ή μερικές διαδρομές (ανάλογα την εφαρμογή) σύμφωνα με τον τύπο $\tau_{ij}^{new}=(1-\rho)\tau_{ij}^{old}$ με $0<\rho<1$. Έτσι οι κακές λύσεις χρησιμοποιούνται όλο και λιγότερο και τείνουν να εξαλείφονται. Η φερομόνη που εναποθέτει κάθε μυρμήγκι δίνεται από τον τύπο $\Delta\tau_{ij}=\frac{1}{c}$ όπου c το κόστος της λύσης που δημιούργησε και αφορά μόνο τα τόξα από όπου πέρασε. Συνολικά, μετά από κάθε επανάληψη η ανανέωση της φερομόνης γίνεται με την μαθηματική έκφραση $\tau_{ij}^{new}=\tau_{ij}^{old}+\sum_{k=1}^{m}\Delta\tau_{ij}^{k}$,όπου m ο πληθυσμός των μυρμηγκιών.

Γίνεται αντιληπτό ότι η εφαρμογή του αλγορίθμου δεν είναι απόλυτη αλλά εξαρτάται από το είδος του προβλήματος δρομολόγησης καθώς και τις ιδιομορφίες που κάθε ένα παρουσιάζει. Πάρα ταύτα, το γενικό πλαίσιο στο οποίο όλοι κινούνται είναι αυτό που παρουσιάστηκε παραπάνω. Πέραν των συντελεστών α , b και του αριθμού των μυρμηγκιών m που επαφίενται στην επιλογή του χρήστη υπάρχουν και διαφορές που αφορούν τον τρόπο ανανέωσης της φερομόνης. Για παράδειγμα ο αλγόριθμος Elitist Ant System όπου το βέλτιστο μυρμήγκι εναποθέτει φερομόνη σε κάθε επανάληψη, ο αλγόριθμος Max-Min Ant System όπου η ποσότητα φερομόνης κινείται σε συγκεκριμένο εύρος και ο Rank-based Ant System όπου οι λύσεις ταξινομούνται με βάση το κόστος τους και η φερομόνη που αφήνουν τα μυρμήγκια είναι ανάλογη της θέσης τους σε αυτήν την κατάταξη.

ΚΕΦΑΛΑΙΟ 4

Εφαρμογή και Επίλυση

4.1 Εισαγωγή

4.1.1 Επιλογή του MATLAB®

Το περιβάλλων Matlab είναι ενα διαδραστικό περιβάλλων για αριθμιτικους υπολογισμούς, απεικόνιση δεδομένων και ταυτόχρονα μια γλώσσα προγραμματισμού υψηλού επιπέδου. Χρησιμοποιείται για την ανάλυση δεδομένων, την ανάπτυξη αλγορίθμων, τη δημιουργία μοντέλων και εφαρμογών. Η γλώσσα σε συνδυασμό με τα εργαλεία και τις έτοιμες συναρτήσεις επιτρέπουν την επίτευξη της λύσης ευκολότερα σε σχέση με άλλες κλασσικές γλώσσες προγραμματισμού. Χρησιμοποιείται ευρέως στον επιστημονικό κοινότητα κυρίως στη μηχανική, τις φυσικές επιστήμες και τα οικονομικά.

Ο λόγος που επελέγη το περιβάλλων Matlab έναντι κάποιας άλλης γλώσσας έγκυται στα παρακάτω πλεονεκτήματα:

- Το βασικό δομικό στοιχείο δεδομένων είναι ο πίνακας. Ένας απλός ακέραιος θεωρείται πινακας 1X1. Αρκετές μαθηματικές πράξεις σε πίνακες περιλαμβάνονται στο Matlab, όπως το εξωτερικό γινόμενο, η ορίζουσα και η αντιστροφή πινάκων.
- Περιλαμβάνει πράξεις διανυσμάτων, για παράδειγμα η ένωση δυο διανυσμάτων γίνεται με μια εντολή αντί για μια δομή επανάληψης.
- Η ευκολία στην γραφική απεικόνιση της λύσης
- Η ευκολία εισαγωγής και εξαγωγής δεδομένων από διαφόρους τύπους δεδομένων όπως το αρχεία excel.

Ωστόσο παρ όλες τις ευκολίες για τον προγραμματιστή/χρήστη, εμπεριέχει δύο κύρια μειονεκτήματα:

- Καταλαμβάνει πολύ χώρο στη μνήμη και σε αργούς υπολογιστές αργεί να βγάλει αποτελέσματα.
- Αποτελεί εφαρμογή του λειτουργικού και επομένως χρησιμοποιεί όσο χρόνο CPU το λειτουργικό του επιτρέπει. Αυτό κάνει εφαρμογές πραγματικού χρόνου πολύ δύσκολες.

Η ευκολία κατά τον προγραμματισμό ήταν η αιτία που χρησιμοποιήθηκε η Matlab έναντι κάποιας άλλης γλώσσας η οποία θα ήταν πιο γρήγορη κατά την εκτέλεση.

4.1.2 Δεδομένα

Το πρόγραμμα διαβάζει δεδομένα από αρχείο excel (.xls) τα οποία έχουν μια συγκεκριμένη δομή, ή αλλιώς ένα πρότυπο που παρουσιάζει τις πληροφορίες για την επίλυση του προβλήματος. Παρακάτω χρησιμοποιείται το αρχείο δεδομένων par1.xls για την απεικόνιση της δομής των αρχείων.

51			NN=αριθμός Κόμβων		
1			Αριθμός για ακεραίους/φυσικούς		
160			CAP=Χωρητικότητα φορτηγών		
999999			Μέγιστος χρόνος παραμονής στη διαδρομή		
0			Χρόνος εξυπηρέτησης		
1	30	40	Αύξων αριθμός κόμβου i	Συντεταγμένη	Συντεταγμένη
2	37	52		Χ-άξονα	Υ-άξονα
3	49	49			
.	.	.			
.	.	.			
.	.	.			
51	56	37			
1	0		Αύξων αριθμός κόμβου i	Ζήτηση κόμβου i	
2	7				
3	30				
.	.				
.	.				
.	.				
50	18				
51	10				

Στη 1η γραμμή εμφανίζεται ο αριθμός των κόμβων NN. Συνάρτηση του αριθμού των κόμβων είναι εύκολο να υπολογιστούν οι συντεταγμένες των κελιών που βρίσκονται τα δεδομένα των συντεταγμένων και της ζήτησης τους. Για τις συντεταγμένες οι γραμμές είναι από 6 μέχρι NN+6 και για την ζήτηση από NN+7 μέχρι 2*NN+7.

Με το πρόγραμμα που παρουσιάστηκε στην προηγούμενη παράγραφο, θα επιχειρήσουμε να επιλύσουμε αρχεία δεδομένων για διάφορους αριθμούς κόμβων.

4.1.3 Κατάστρωση του αλγορίθμου

Η στρατηγική του αλγορίθμου επίλυσης περιλαμβάνει την εξής ακολουθία :

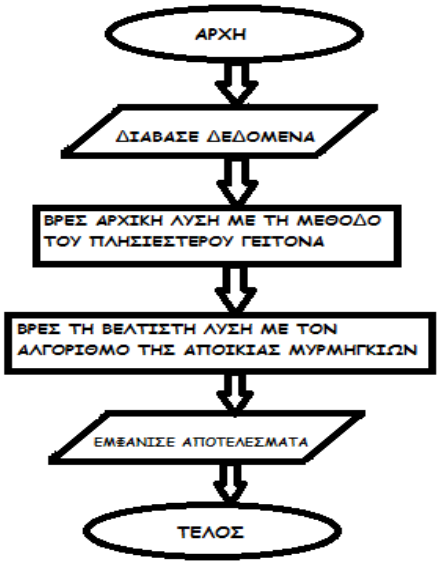
Αρχικά κατασκευάζεται μια αρχική λύση με τον αλγόριθμο του πλησιέστερου γείτονα. Η λύση αυτή είναι μια εφικτή λύση, δηλαδή υπακούει σε όλους του περιορισμούς ωστόσο κατά πάσα πιθανότητα δεν είναι ή βέλτιστη. Επειδή όμως η λύση δεν είναι τυχαία, αλλά αιτιοκρατική και συγκεκριμένα βασίζεται στην ένωση κόμβων που είναι πλησιέστερα οικονομικά, δίνει μια πρώτη εικόνα για το ποιες λύσεις είναι καλύτερες έναντι άλλων.

Στη συνέχεια χρησιμοποιώντας την αρχική λύση αρχικοποιούνται τα επίπεδα φερομόνης τ στις διαδρομές του γραφήματος. Επίσης αρχικοποιούνται οι μεταβλητές α , β και η ευρετική πληροφορία n βάση των οποίων τα μυρμήγκια αποφασίζουν τον επόμενο κόμβο που θα επισκεφτούν. Έπειτα επιλύεται το πρόβλημα με τον αλγόριθμο της αποικίας μυρμηγκιών. Κάθε μυρμήγκι είναι και μια λύση, η οποία περιλαμβάνει ένα δρομολόγιο για κάθε όχημα. Καθότι όμως, το πρόβλημα είναι συσσωρευτικό, η διαδρομή Α-Β-Γ-Α ενδέχεται να έχει διαφορετικό κόστος από τη διαδρομή Α-Γ-Β-Α και επομένως θα πρέπει να ελεγχθεί για κάθε όχημα ποια από τις δυο αλληλουχίες είναι η πλέον συμφέρουσα. Ο έλεγχος αυτός γίνεται με έναν αλγόριθμο τοπικής αναζήτησης, συγκεκριμένα τον 2-opt. Με αυτόν τον βελτιστοποιείται η διαδρομή κάθε οχήματος, σε κάθε λύση που υπολογίζει ο ACO και ύστερα επιλέγετε το καλύτερο μυρμήγκι που θα εναποθέσει φερομόνη στο γράφημα.

Καθολη τη διάρκεια που το πρόγραμμα "τρέχει" ο χρήστης πληροφορείται για την πορεία της λύσης. Αρχικά με ενα γράφημα όπου εμφανίζονται τα αποτελέσματα της αρχικής λύσης, κατά τη διάρκεια εκτέλεσης του ACO με γράφημα που ανανεώνεται σε κάθε επανάληψη παρουσιάζοντας τα επίπεδα φερομονης και τον ειδοποιεί για το μεχρι στιγμής βέλτιστο κόστος καθώς και τον αριθμό της επανάληψης. Τέλος εμφανίζει ένα τελικό γράφημα της βέλτιστης δρομολόγησης.

4.1.4 Δομή προγράμματος

Το πρόγραμμα αποτελείται από τη Main.m και 8 υπορουτίνες οι οποίες εκτελούν τις διάφορες εργασίες που χρειάζονται. Στην επόμενη παράγραφο εξηγείται επακριβώς η λειτουργία κάθε μιας συνάρτησης.



4.2 Συναρτήσεις

4.2.1 Main.m

Η Main.m είναι η βασική συνάρτηση που υλοποιεί τον αλγόριθμο. Σε αυτή συγκεντρώνονται όλα τα αποτελέσματα από τις υπορουτίνες προκειμένου να εξαχθεί η τελική λύση του προβλήματος. Αρχικά, καλεί την DATA.m προκειμένου να διαβάσει τα δεδομένα του προβλήματος. Έπειτα καλεί την MAP.m η οποία θα σχεδιάσει το γράφημα που θα είναι ορατό στο χρήστη. Στη συνέχεια με την InitialSolution.m δημιουργεί μια αρχική λυση και υπολογίζει το κόστος της. Έχοντας τα δεδομένα για να κάνει την αρχικοποίηση του γραφήματος φερομόνης, καλεί την ACO.m, την συνάρτηση που θα εκτελέσει τον αλγόριθμο ACO. Τέλος, αφού εκτελεστεί ο αλγόριθμος της αποικίας των μυρμηγκιών εμφανίζει τα αποτελέσματα στο χρήστη με τη συνάρτηση Results.m.

Συνοπτικά :

ΔΙΑΒΑΣΕ δεδομένα από αρχείο excel (DATA.m)
ΣΧΕΔΙΑΣΕ το γράφημα των κόμβων (MAP.m)
ΥΠΟΛΟΓΙΣΕ αρχική λύση με τη μέθοδο του πλησιέστερου γείτονα (InitialSolution.m)
ΥΠΟΛΟΓΙΣΕ τη βέλτιστη λύση με τη μέθοδο της αποικίας μυρμηγκιών (ACO.m)
ΕΜΦΑΝΙΣΕ αποτελέσματα (Results.m)

4.2.2 Data.m

Η συνάρτηση data.m χρησιμοποιείται για τη εισαγωγή δεδομένων από αρχείο excel (.xls) και την επεξεργασία τους με κατάλληλο τρόπο ώστε να είναι χρησιμοποιήσιμα από το πρόγραμμα. Τα αρχεία excel στα οποία υπάρχει η απαραίτητη πληροφορία έχουν συγκεκριμένη δομή. Η συνάρτηση διαβάζει το αρχείο, υπολογίζει τις αποστάσεις μεταξύ των κόμβων και επιστρέφει στη main.m τον αριθμό των κόμβων, τη χωρητικότητα των οχημάτων, τη ζήτηση κάθε κόμβου, τις συντεταγμένες τους καθώς και τις αποστάσεις μεταξύ τους.

4.2.3 MAP.m

Η επόμενη κατά σειρά συνάρτηση που καλεί το πρόγραμμα είναι η συνάρτηση που δημιουργεί το αρχικό γράφημα. Εμφανίζει με κόκκινες κουκκίδες τους κόμβους και με τετράγωνο την αποθήκη στην οποία γράφει δίπλα “warehouse”. Επίσης ορίζει το μέγεθος των αξόνων, τον τίτλο του γραφήματος και σχεδιάζει όλες τις πιθανές συνδέσεις μεταξύ των κόμβων.

4.2.4 InitialSolution.m

Όπως προδίδει το όνομα σε αυτή τη συνάρτηση υπολογίζεται η αρχική λύση. Η συνάρτηση δέχεται από τη Main.m τους πίνακες της ζήτησης, της απόστασης, τον αριθμού των κόμβων και τις συντεταγμένες τους. Μέσα σε αυτή τη συνάρτηση υλοποιείται ο αλγόριθμος του πλησιέστερου γείτονα ως εξής:

Αρχικοποίηση
ΟΣΟ το άθροισμα των κόμβων που τα οχήματα έχουν επισκεπτεί < αριθμού των κόμβων
-Βρίσκω την κοντινότερη οικονομικά πόλη
ΑΝ το απόθεμα του οχήματος φτάνει για την ικανοποίηση της ζήτησης ΤΟΤΕ
-Θεώρησε την πόλη οτι την έχουν επισκέφτηκε
-Σημείωσε τη μετάβαση και πιο φορτηγό την έκανε
-Σχεδίασε στο γράφημα την μετάβαση αυτή. (DrawLine.m)
-Υπολόγισε τη ροή στη μετάβαση αυτή
-Καταχώρησε την πόλη στην διαδρομή
-Πρόσθεσε στη συνολική απόσταση του οχήματος την απόσταση της μετάβασης
-Θέσε ως επόμενη πόλη αφετηρίας την πόλη προορισμού.
ΤΕΛΟΣ_ΑΝ
ΑΝ το απόθεμα του οχήματος δεν φτάνει Ή όλες τις έχουμε επισκεφτεί ΤΟΤΕ
-Επέστρεψε στην αποθήκη
-Σημείωσε τη μετάβαση και πιο φορτηγό την έκανε
-Σχεδίασε στο γράφημα την μετάβαση αυτή. (DrawLine.m)
-Υπολόγισε τη ροή στη μετάβαση αυτή
-Πρόσθεσε στη συνολική απόσταση του οχήματος την απόσταση της μετάβασης
-Θέσε ως επόμενη πόλη αφετηρίας την αποθήκη.
-Επόμενο όχημα
ΤΕΛΟΣ_ΑΝ
ΤΕΛΟΣ_ΟΣΟ
ΤΥΠΩΣΕ την διαδρομή κάθε φορτηγού

Η συνάρτηση επιστρέφει τον αριθμό των οχημάτων, τη ροή του γραφήματος, τη διαδρομή των οχημάτων, τις αποστάσεις που διένυσαν και τους πίνακες που δείχνουν την αλληλουχία των μεταβάσεων και από ποια οχήματα αυτές έγιναν.

4.2.5 trails.m

Μεσώ της συνάρτησης αυτής, γίνεται η απεικόνιση του επιπέδου φερομόνης σε κάθε τόξο του γραφήματος. Δέχεται ως είσοδο τις συντεταγμένες, τον αριθμό της επανάληψης, το φθηνότερο κόστος και τον πίνακα φερομόνης, καλεί την MAP.m και πάνω στο γράφημα, εμφανίζει το κόστος, την επανάληψη και αφού κανονικοποιήσει τη φερομόνη εμφανίζει τα τόξα με διαφορετικά χαρακτηριστικά ανάλογα με τα επίπεδά της.

4.2.6 Results.m

Για κάθε όχημα που που θα χρησιμοποιηθεί, υπολογίζει τη συνολική απόσταση που θα διανύσει και σχεδιάζει σε γράφημα το τελικό δρομολόγιο του μέσω της συνάρτησης DrawLine.m. Τέλος, εμφανίζει στο χρήστη τις αποστάσεις που θα διανύσει κάθε όχημα, τη διαδρομή του και την αρχική του ποσότητα.

4.2.7 DrawLine.m

Η συνάρτηση παίρνει τον αριθμό του οχήματος, τους κόμβους αφετηρίας, προορισμού και τις συντεταγμένες, και ανάλογα τον αριθμό δημιουργεί γραμμές που ενώνουν τους κόμβους αυτούς με διαφορετικό χρώμα.

4.2.8 ACO.m

Η συνάρτηση ACO είναι εκείνη που επιλύει κατά το βέλτιστο τρόπο το πρόβλημα του CCVRP. Λαμβάνει τα δεδομένα της απόστασης, της ζήτησης, της χωρητικότητας, των κόμβων και της αρχικής λύσης από τη Main.m και δίνει ως αποτέλεσμα τα δρομολόγια των οχημάτων, τον αριθμό τους, το συνολικό κόστος της δρομολόγησης καθώς και τις ποσότητες που θα πρέπει να είναι εφοδιασμένα ξεκινώντας από την αποθήκη.

Παρακάτω θα εξηγηθεί αναλυτικά η συνάρτηση ACO.m και θα γίνει συγκριτική παρουσίαση όλων των παραλλαγών ACO αλγορίθμων που χρησιμοποιήθηκαν.

4.2.9 rearange.m

Αυτή η συνάρτηση εξυπηρετεί δύο στόχους. Αρχικά διαβάζει μια λύση από τη ACO.m, ελέγχει αν χρησιμοποιεί τα ελάχιστα φορτηγά και σε περίπτωση που χρησιμοποιεί παραπάνω, διαμορφώνει τις διαδρομές κατάλληλα ώστε τα οχήματα να ελαχιστοποιηθούν. Αυτό όμως δεν είναι πάντα εφικτό. Έτσι επιστρέφει και μια λογική μεταβλητή και όταν αυτή είναι αρνητική η ACO.m επαναλαμβάνει την λύση.

4.3 Εφαρμογή

Στην παρούσα εργασία το πρόβλημα επιλύθηκε με διάφορους τύπους αλγορίθμων, όπου όλοι ανήκουν στην οικογένεια των ACO. Ουσιαστικά η δομή του προγράμματος παραμένει ίδια, αλλά αλλάζει ο τρόπος αρχικοποίησης της φερομόνης, ο τρόπος η ανανέωση της και η στρατηγική με την οποία αναζητείται η βέλτιστη λύση.

4.3.1 AS-simple

Ο αλγόριθμος AntSystem είναι ο πρώτος που χρησιμοποιήθηκε και ο απλούστερος στην εφαρμογή του. Η αρχικοποίηση της φερομόνης στα τόξα (i, j) γίνεται από τον τύπο $\tau_{ij} = \tau_0 = m / C^{IS}$ όπου m το πλήθος των μυρμηγκιών και C^{IS} το κόστος της αρχικής λύσης Εφόσον όλα τα μυρμήγκια έχουν κατασκευάσει μια λύση η φερομόνη ανανεώνεται. Πρώτα τα επίπεδα φερομόνης μειώνονται σε όλα τα τόξα κατά ένα σταθερό ποσοστό και έπειτα το καλύτερο μυρμήγκι της επανάληψης εναποθέτει φερομόνη στα τόξα που περιλαμβάνονται στη διαδρομή του, όπως παρουσιάστηκε στο 3.3.4.

ACO.m

ΑΡΧΙΚΟΠΟΙΗΣΕ φερομόνη,ευρετική πληροφορία
ΓΙΑ έναν αριθμό επαναλήψεων
ΓΙΑ κάθε μυρμήγκι
-Αρχικοποιησε πίνακα μεταβάσεων, κόμβους που δεν έχει επισκεφτεί και απόθεμα
-Θέσε την αποθήκη ως κόμβο που έχει επισκεφθεί και ξεκίνα από αυτό
ΟΣΟ δεν τις έχεις επισκεπτεί όλες τις πόλεις -κόμβους
ΥΠΟΛΟΓΙΣΕ πιθανότητες μετάβασης
-Βρες επόμενο κόμβο
ΑΝ το απόθεμα αρκεί
-Καταχώρησε την μετάβαση στον πίνακα μεταβάσεων
- Αφαίρεσε τον κόμβο από τη λίστα
-Αφαίρεσε τη ζήτηση του κόμβου από το απόθεμα
-Θέσε επόμενο κόμβο αφετηρίας τον κόμβο που πήγες
ΤΕΛΟΣ_ΑΝ
ΑΝ έχεις επισκεφθεί όλους τους κόμβους Ή δεν αρκεί το απόθεμα
-Καταχώρησε τη μετάβαση απο τον κόμβο που είσαι στην αποθήκη
-Θέσε επόμενη αφετηρία την αποθήκη
-Αρχικοποίησε το απόθεμα
ΤΕΛΟΣ_ΑΝ
ΤΕΛΟΣ_ΟΣΟ
-Ανέλυσε τη λύση σε δρομολόγια /Αν είναι αδύνατη επανέλαβε το μυρμηγκι
-Βελτιστοποίησε τη λύση με αλγόριθμο τοπικής αναζήτησης 2opt
-Αποθήκευσε την καλύτερη λύση ως εκείνο το μυρμήγκι
ΤΕΛΟΣ_ΓΙΑ
-Ανανέωσε τη φερομόνη στο γράφημα
-Αποθήκευσε την καλύτερη λύση ως αυτή την επανάληψη
ΤΕΛΟΣ_ΓΙΑ

4.3.2 Elitist Ant System

Η πρώτη βελτίωση που έγινε στον AS ήταν να ακολουθηθεί μια ελιτίστικη στρατηγική κατά την ανανέωση της φερομόνης, δηλαδή πέραν του καλύτερου μυρμηγκιού της επανάληψης και το καλύτερο μυρμήγκι ως τώρα, να εναποθέτει φερομόνη μετά το πέρας κάθε επανάληψης. Ο τύπος με τον εναποτίθεται φερομόνη είναι $\tau_{ij}^{new} = \tau_{ij}^{old} + \sum_{k=1}^{m} \Delta\tau_{ij}^{k} + e\Delta\tau_{ij}^{BS}$ όπου e ένας συντελεστής βαρύτητας.

ACO.m

ΑΡΧΙΚΟΠΟΙΗΣΕ φερομόνη,ευρετική πληροφορία
ΓΙΑ έναν αριθμό επαναλήψεων
ΓΙΑ κάθε μυρμήγκι
-Αρχικοποίησε πίνακα μεταβάσεων, κόμβους που δεν έχει επισκεφτεί και απόθεμα
-Θέσε την αποθήκη ως κόμβο που έχει επισκεφθεί και ξεκίνα από αυτό
ΟΣΟ δεν τις έχεις επισκεπτεί όλες τις πόλεις -κόμβους
ΥΠΟΛΟΓΙΣΕ πιθανότητες μετάβασης
-Βρες επόμενο κόμβο
ΑΝ το απόθεμα αρκεί
-Καταχώρησε την μετάβαση στον πίνακα μεταβάσεων
- Αφαίρεσε τον κόμβο από τη λίστα
-Αφαίρεσε τη ζήτηση του κόμβου από το απόθεμα
-Θέσε επόμενο κόμβο αφετηρίας τον κόμβο που πήγες
ΤΕΛΟΣ_ΑΝ
ΑΝ έχεις επισκεφθεί όλους τους κόμβους Ή δεν αρκεί το απόθεμα
-Καταχώρησε τη μετάβαση από τον κόμβο που είσαι στην αποθήκη
-Θέσε επόμενη αφετηρία την αποθήκη
-Αρχικοποίησε το απόθεμα
ΤΕΛΟΣ_ΑΝ
ΤΕΛΟΣ_ΟΣΟ
-Ανέλυσε τη λύση σε δρομολόγια /Αν είναι αδύνατη επανέλαβε το μυρμήγκι
-Βελτιστοποίησε τη λύση με αλγόριθμο τοπικής αναζήτησης 2opt
-Αποθήκευσε την καλύτερη λύση ως εκείνο το μυρμήγκι
ΤΕΛΟΣ_ΓΙΑ
-Ανανέωσε τη φερομόνη στο γράφημα με τα μυρμήγκια της επανάληψης
-Αποθήκευσε την καλύτερη λύση ως αυτή την επανάληψη
-Ανανέωσε τη φερομόνη στο γράφημα με το καλύτερο μυρμήγκι ως τώρα.
ΤΕΛΟΣ_ΓΙΑ

4.3.3 AS-Rank

Μια άλλη βελτίωση του AS είναι ο AS-Rank. Σε αυτόν οι λύσεις ταξινομούνται με βάση το κόστος τους και η ποσότητα φερομόνης που εναποθέτουν μειώνεται καθώς μειώνεται η κατάταξης τους. Επίσης σε κάθε επανάληψη το καλύτερο ως τώρα μυρμήγκι εναποθέτει πάντα την μεγαλύτερη ποσότητα φερομόνης.
Σε κάθε επανάληψη τα (w-1) καλύτερα μυρμήγκια και το καλύτερο ως τώρα, εναποθέτουν φερομόνη με τον ακόλουθο τύπο : $\tau_{ij}^{new} = \tau_{ij}^{old} + \sum_{r=1}^{w-1} (w-r)\,\Delta\tau_{ij}^{r} + w\Delta\tau_{ij}^{BS}$

ACO.m

ΑΡΧΙΚΟΠΟΙΗΣΕ φερομόνη,ευρετική πληροφορία
ΓΙΑ έναν αριθμό επαναλήψεων
ΓΙΑ κάθε μυρμήγκι
-Αρχικοποίησε πίνακα μεταβάσεων, κόμβους που δεν έχει επισκεφτεί και απόθεμα
-Θέσε την αποθήκη ως κομβο που εχει επισκεφθεί και ξεκίνα από αυτό
ΟΣΟ δεν τις έχεις επισκεπτεί όλες τις πόλεις -κόμβους
ΥΠΟΛΟΓΙΣΕ πιθανότητες μετάβασης
-Βρες επόμενο κόμβο
ΑΝ το απόθεμα αρκεί
-Καταχώρησε την μετάβαση στον πίνακα μεταβάσεων
- Αφαίρεσε τον κόμβο από τη λίστα
-Αφαίρεσε τη ζήτηση του κόμβου από το απόθεμα
-Θέσε επόμενο κόμβο αφετηρίας τον κόμβο που πήγες
ΤΕΛΟΣ_ΑΝ
ΑΝ έχεις επισκεφθεί όλους τους κόμβους Ή δεν αρκεί το απόθεμα
-Καταχώρησε τη μετάβαση από τον κόμβο που είσαι στην αποθήκη
-Θέσε επόμενη αφετηρία την αποθήκη
-Αρχικοποίησε το απόθεμα
ΤΕΛΟΣ_ΑΝ
ΤΕΛΟΣ_ΟΣΟ
-Ανέλυσε τη λύση σε δρομολόγια /Αν είναι αδύνατη επανέλαβε το μυρμήγκι
-Βελτιστοποίησε τη λύση με αλγόριθμο τοπικής αναζήτησης 2opt
ΤΕΛΟΣ_ΓΙΑ
-Κατέταξε τα δρομολόγια από το καλύτερο στο χειρότερο
-Ανανέωσε τη φερομονη στο γράφημα με σύμφωνα με την κατάταξη τους
-Ανανέωσε τη φερομονη στο γράφημα με το καλύτερο μυρμήγκι ως τώρα.
ΤΕΛΟΣ_ΓΙΑ

4.3.4 MAX-MIN Ant System

Η μεγάλη στρατηγική διαφορά του MAX-MIN Ant System με τους προηγούμενους αλγορίθμους είναι η εκτενής εκμετάλλευση της καλύτερης λύσης. Σε αυτή την εφαρμογή μόνο το καλύτερο μυρμήγκι της επανάληψης ή το ως τώρα καλύτερο (εμείς επιλέξαμε της επανάληψης) αφήνει φερομόνη. Αυτό μπορεί να οδηγήσει σε σημείο στασιμότητας λόγο της εναπόθεσης φερομόνης στο ίδιο μονοπάτι σε κάθε επανάληψη. Για να το αντιμετωπίσουμε αυτό ο MMAS εισάγει όριο στα επίπεδα φερομόνης κάθε τόξου. Έτσι η φερομόνη κινείται σε ένα εύρος $[\tau_{min}, \tau_{max}]$. Μια ακόμα διαφορά είναι ότι η αρχικοποίηση της φερομόνης γίνεται στο άνω όριο, η οποία σε συνδυασμό με το μικρό βαθμό εξάτμισης ενισχύει την εξερεύνηση στην αρχή της αναζήτησης.

Ορίζουμε $\tau_{max}=1/\rho C^{IS}$ και $\tau_{min}=\tau_{max}/a$ όπου α μία παράμετρος. Ο αλγόριθμος αυτός είναι από τους πιο εκτενώς μελετημένους και υπάρχουν πάρα πολλές παραλλαγές του, όπως για παράδειγμα η επαναρχικοποίηση της φερομόνης όταν πλησιάζει σε σημείο στασιμότητας ή η εναλλαγή μεταξύ καλύτερου ως τώρα και καλύτερου επανάληψης μυρμηγκιού. Ο αλγόριθμος που υλοποιήσαμε είναι η απλούστερη μορφή του.

ACO.m

ΑΡΧΙΚΟΠΟΙΗΣΕ φερομόνη,ευρετική πληροφορία
ΓΙΑ έναν αριθμό επαναλήψεων
ΓΙΑ κάθε μυρμήγκι
-Αρχικοποίησε πίνακα μεταβάσεων, κόμβους που δεν έχει επισκεφτεί και απόθεμα
-Θέσε την αποθήκη ως κομβο που εχει επισκεφθεί και ξεκίνα από αυτό
ΟΣΟ δεν τις έχεις επισκεπτεί όλες τις πόλεις -κόμβους
ΥΠΟΛΟΓΙΣΕ πιθανότητες μεταβάσης
-Βρες επόμενο κόμβο
ΑΝ το απόθεμα αρκεί
-Καταχώρησε την μετάβαση στον πίνακα μεταβάσεων
- Αφαίρεσε τον κόμβο από τη λίστα
-Αφαίρεσε τη ζήτηση του κόμβου από το απόθεμα
-Θέσε επόμενο κόμβο αφετηρίας τον κόμβο που πήγες
ΤΕΛΟΣ_ΑΝ
ΑΝ έχεις επισκεφθεί όλους τους κόμβους Ή δεν αρκεί το απόθεμα
-Καταχώρησε τη μετάβαση απο τον κόμβο που είσαι στην αποθήκη
-Θέσε επόμενη αφετηρία την αποθήκη
-Αρχικοποίησε το απόθεμα
ΤΕΛΟΣ_ΑΝ
ΤΕΛΟΣ_ΟΣΟ
-Ανέλυσε τη λύση σε δρομολόγια /Αν είναι αδύνατη επανέλαβε το μυρμηγκι
-Βελτιστοποίησε τη λύση με αλγόριθμο τοπικής αναζήτησης 2opt
ΤΕΛΟΣ_ΓΙΑ
-Αποθήκευσε την καλύτερη λύση ως εκείνο το μυρμήγκι
-Ανανέωσε τη φερομόνη στο γράφημα με το καλύτερο μυρμήγκι της επανάληψης.
-Διόρθωσε τα επίπεδα φερομόνης ώστε να βρίσκονται εντός του επιτρεπτού.
ΤΕΛΟΣ_ΓΙΑ

4.3.5 Ant Colony System

Ο αλγόριθμος ACS εφαρμόζει μια διαφορετική στρατηγική προσέγγιση της βέλτιστης λύσης. Πρώτων ωθεί την εξερεύνηση με ένα επιθετικό κριτήριο επιλογής της επόμενης πόλης και κατά δεύτερων η εξάτμιση και η εναποθέτηση φερομόνης γίνονται μόνο στην ως τώρα καλύτερη διαδρομή. Μια τρίτη διαφορά είναι το γεγονός οτι κάθε μυρμήγκι αμέσως μόλις διασχίσει ένα τόξο, αφαιρεί μια ποσότητα φερομόνης από αυτό ώστε το επόμενο μυρμήγκι να αναζητήσει άλλη διαδρομή.
Η διαδικασία επιλογής της πόλης γίνεται ως εξής :
-Παράγουμε έναν τυχαίο αριθμό q με ομοιόμορφη κατανομή στο [0,1]
- $j = argmax_{l \in N_i^k}\{\tau_{il}[n_{il}^{\beta}]\}\ if\ q \le q_0$ αλλιώς j=J σύμφωνα με τις πιθανότητες που ορίστηκαν και υπολογίστηκαν στην παρουσίαση του αλγορίθμου στη γενική του μορφή.
Η διαδικασία ανανέωσης της φερομόνης γίνεται σε δύο φάσεις.
Γενική ανανέωση: $\tau_{ij}^{new} = (1-\rho)\tau_{ij}^{old} + \rho\Delta\tau_{ij}^{BS}\ \forall (i,j) \in T^{BS}$
Τοπική ανανέωση: $\tau_{ij}^{new} = (1-\xi)\tau_{ij}^{old} + \xi\tau_0$ όπου $\tau_0 = 1/nC^{IS}$

ACO.m

ΑΡΧΙΚΟΠΟΙΗΣΕ φερομόνη,ευρετική πληροφορία
ΓΙΑ έναν αριθμό επαναλήψεων
ΓΙΑ κάθε μυρμήγκι
-Αρχικοποίησε πίνακα μεταβάσεων, κόμβους που δεν έχει επισκεφτεί και απόθεμα
-Θέσε την αποθήκη ως κόμβο που έχει επισκεφθεί και ξεκίνα από αυτό
ΟΣΟ δεν τις έχεις επισκεπτεί όλες τις πόλεις -κόμβους
ΥΠΟΛΟΓΙΣΕ πιθανότητες μεταβάσης
-Βρες επόμενο κόμβο
ΑΝ το απόθεμα αρκεί
-Καταχώρησε την μετάβαση στον πίνακα μεταβάσεων
- Αφαίρεσε τον κόμβο από τη λίστα
-Αφαίρεσε τη ζήτηση του κόμβου από το απόθεμα
-Θέσε επόμενο κόμβο αφετηρίας τον κόμβο που πήγες
ΤΕΛΟΣ_ΑΝ
ΑΝ έχεις επισκεφθεί όλους τους κόμβους Ή δεν αρκεί το απόθεμα
-Καταχώρησε τη μετάβαση από τον κόμβο που είσαι στην αποθήκη
-Θέσε επόμενη αφετηρία την αποθήκη
-Αρχικοποίησε το απόθεμα
ΤΕΛΟΣ_ΑΝ
ΤΕΛΟΣ_ΟΣΟ
-Ανέλυσε τη λύση σε δρομολόγια /Αν είναι αδύνατη επανέλαβε το μυρμήγκι
-Βελτιστοποίησε τη λύση με αλγόριθμο τοπικής αναζήτησης 2opt
-Μείωσε τη φερομόνη τοπικά
ΤΕΛΟΣ_ΓΙΑ
-Αποθήκευσε την καλύτερη λύση ως εκείνο το μυρμήγκι
-Ανανέωσε τη φερομόνη στο γράφημα με το καλύτερο μυρμήγκι ως τώρα
ΤΕΛΟΣ_ΓΙΑ

4.3.6 2-opt

Σε κάθε αλγόριθμο χρησιμοποιήθηκε αλγόριθμος τοπικής βελτιστοποίησης 2-opt. Η εφαρμογή του έγινε σε κάθε δρομολόγιο της κάθε λύσης, ώστε να βρεθεί η οικονομικότερη σειρά με τις οποίες θα τις επισκεφθεί.

Η μέθοδος αυτή στη γενική του μορφή περιλαμβάνει την διαγραφή 2 ακμών και την επανασύνδεση των μονοπατιών έτσι ώστε να δημιουργηθεί μια καινούρια διαδρομή. Παρακάτω παρουσιάζουμε την υλοποίηση του σε ψευδογλώσσα.

ΑΠΟΘΗΚΕΥΣΕ την αρχική διαδρομή
ΜΕΧΡΙ να καταλήξεις στην ίδια διαδρομή
ΓΙΑ κάθε κόμβο i από 2 μέχρι N-2 της διαδρομής
ΓΙΑ κάθε κόμβο από j από i+1 μέχρι N-1
-Δημιούργησε νέα διαδρομή κατατάσσοντας τους κόμβους i εώς j της παλαιάς διαδρομής αντίθετα. στην νέα
-Υπολόγισε το κόστος της νέας διαδρομής
ΑΝ τη βελτιστοποιεί
-Αποθήκευσε τη ως βέλτιστη
ΤΕΛΟΣ_ΑΝ
ΤΕΛΟΣ_ΓΙΑ
ΤΕΛΟΣ_ΓΙΑ
ΤΕΛΟΣ_ΜΕΧΡΙ

Παράδειγμα της διαδικασίας που ακολουθεί ο αλγόριθμος :
Έστω διάνυσμα δρομολογίου Di=[1 2 3 4 1]

```
Di =
     1     3     2     4     1
Di =
     1     4     2     3     1
Di =
     1     4     3     2     1
Di =
     1     3     4     2     1
Di =
     1     2     4     3     1
Di =
     1     2     3     4     1
```

4.4 Επίλυση και αποτελέσματα

Τα προβλήματα τα οποία τρέξαμε είναι τα par1, par2, par3, par3, par4, par5, par 11, par 12 καθώς και τα προβλήματα A-n32κ5 μέχρι και A-n80κ10. Τα δεδομένα αυτά αρχικά δημιουργήθηκαν για το πρόβλημα VRP,ωστόσο στη βιβλιογραφία χρησιμοποιούνται και για το CCVRP.

Για κάθε αλγόριθμο και για κάθε πρόβλημα δοκιμάστηκαν συνδυασμοί διαφόρων τιμών για τις παραμέτρους. Στην βιβλιογραφία υπάρχουν προτεινόμενες τιμές αλλά τα αποτελέσματα δεν είναι πάντοτε τα επιθυμητά. Ο λόγος είναι ότι συνήθως αναφέρονται στο Πρόβλημα του πλανόδιου πωλητή (TSP). Το CCVRP είναι πιο περίπλοκο και επομένως οι παράμετροι σε ορισμένες περιπτώσεις να θέλουν προσαρμογή. Εκτός από την περιπλοκότητα, η βέλτιστη ρύθμιση των παραμέτρων να εξαρτάται και από το εκάστοτε πρόβλημα. Για παράδειγμα, στο πρόβλημα n32 η λύση δεν επηρεάζεται σε μεγάλο βαθμό από κάποια μικρή αλλαγή στις παραμέτρους. Όμως για το Par1 παρατηρούμαι μεγάλη διαφορά στο αποτέλεσμα.

4.4.1 Πρόβλημα Par1 (50 πόλεις)

Τα αποτελέσματα του αλγορίθμου **AS-simple**

Παράμετροι

AS-Simple
a=1 b=4
m=51
r=0,5

Λύση

Μέσος Όρος	χειρότερη λύση	καλύτερη λύση	1	2	3	4	5
2431,7339	2456,9691	2401,4945	2456,969	2449,518	2401,495	2417,554	2433,134

Παράμετροι

AS-Simple
a=1 b=3
m=51
r=0,1

Λύση

Μέσος Όρος	χειρότερη λύση	καλύτερη λύση	1	2	3	4	5
2391,00116	2418,5296	2373,3	2418,53	2373,303	2373,3	2386,247	2403,626

Παράμετροι

AS-Simple
a=1 b=3
m=51
r=0,01
Best

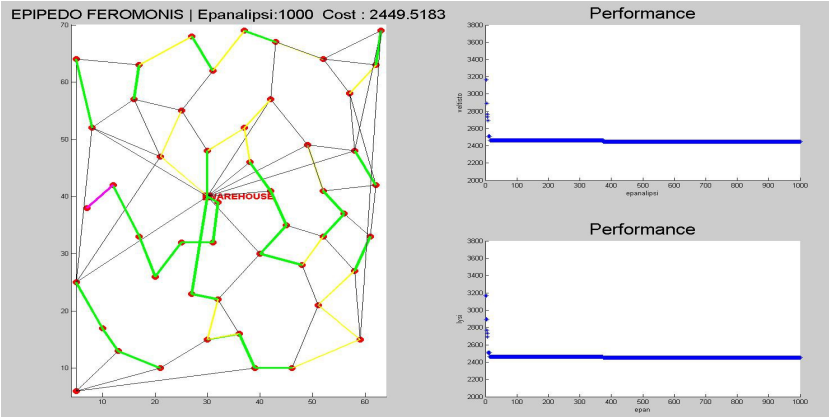
Λύση

Μέσος Όρος	χειρότερη λύση	καλύτερη λύση	1	2	3	4	5
2358,48542	2367,4	**2346,8271**	2346,827	2365,1	2362,1	2351	2367,4

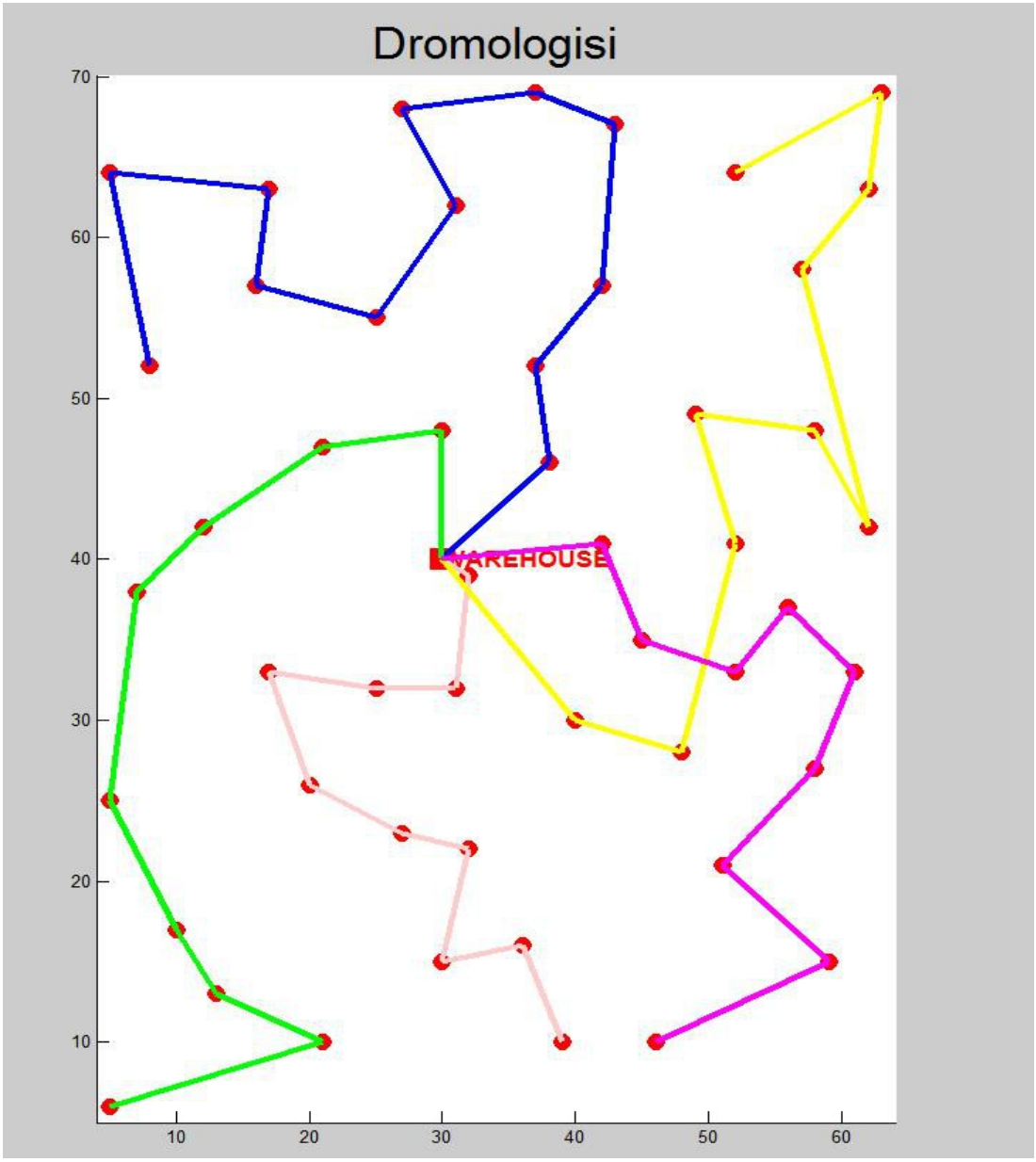
Σχολιασμός

Όπως είναι γνωστό από τη βιβλιογραφία ο απλός AS αλγόριθμος δεν έχει πολύ καλά αποτελέσματα ακόμα για το απλό πρόβλημα TSP. Πόσο μάλλον για το CCVRP όπου το πρόβλημα είναι ακόμα πιο δυσεπίλυτο. Για τον αλγόριθμο δοκιμάστηκαν πάρα πολλές ρυθμίσεις στις παραμέτρους χωρίς καμία να μην δίνει "καλό" αποτέλεσμα. Σε όλες τις περιπτώσεις τα μυρμήγκια εγκλωβίζονται σε τοπικό ελάχιστο.

Όπως αναφέραμε σε προηγούμενο κεφάλαιο, ο κώδικας που δημιουργήθηκε μας εμφανίζει σε πραγματικό χρόνο τα επίπεδα φερομόνης σε γράφημα και δίπλα γραφική παράσταση με τον αριθμό των επαναλήψεων στον άξονα Χ και τη έως τώρα βέλτιστη στον Υ. Παρακάτω φαίνεται ενδεικτικά.



Για το πρόβλημα των 50 πόλεων η γραφική αναπαράστασταση της καλύτερης λύσης που βρέθηκε με τον AS- simple είναι η εξής :



Τα αποτελέσματα του αλγορίθμου **Elitist AS** για τις καλύτερες ρυθμίσεις που βρέθηκαν.

Παράμετροι

Elitist Ant System
a=1 b=3
epsilon=NN
m=51
ro=0.5

Λύση

Μέσος Όρος	χειρότερη λύση	καλύτερη λύση	1	2	3	4	5
2435,4521	2456,332	2388,9884	2456,332	2388,988	2453,1512	2425,8891	2452,9

Παράμετροι

Elitist Ant System
a=1 b=3
epsilon=NN
m=51
ro=0.1

Λύση

Μέσος Όρος	χειρότερη λύση	καλύτερη λύση	1	2	3	4	5
2419,8138	2430,0693	2406,9014	2420,045	2417,877	2430,0693	2406,9014	2424,176

Παράμετροι

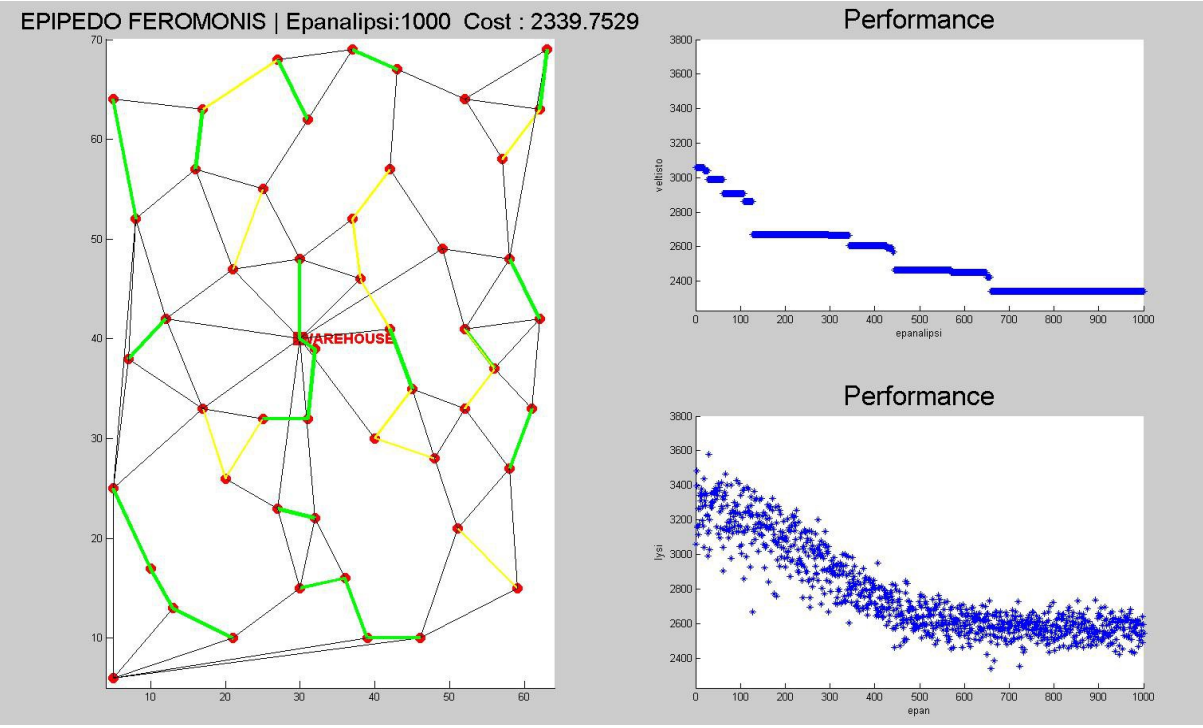
Elitist Ant System
a=1 b=3
epsilon=NN
m=51
ro=0.01
best

Λύση

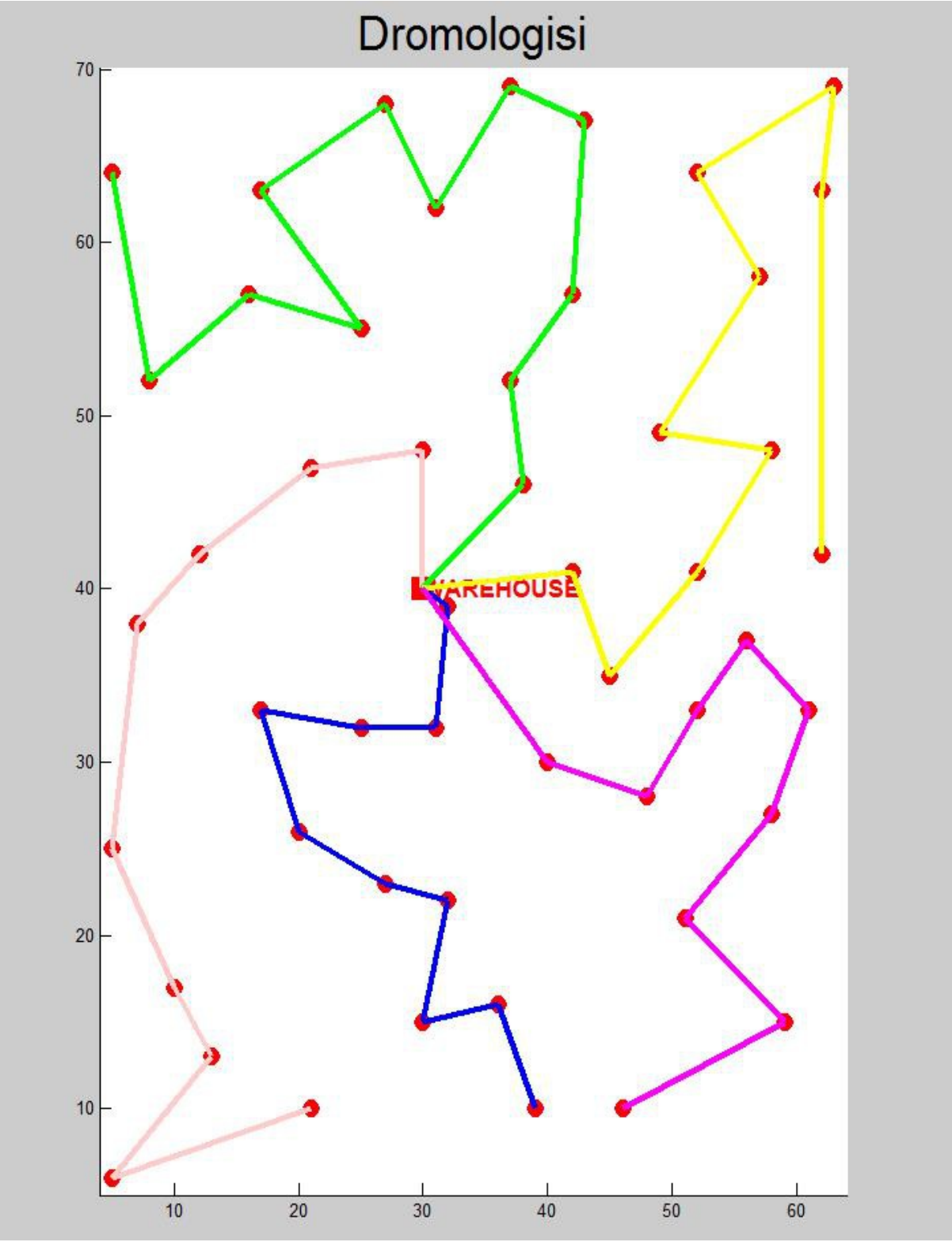
Μέσος Όρος	χειρότερη λύση	καλύτερη λύση	1	2	3	4	5
2379,056	2415,1552	**2339,7529**	2374,881	2376,595	2388,8958	2415,1552	2339,753

Σχολιασμός

Ο αλγόριθμος elitist-AS δίνει καλύτερο αποτέλεσμα από τον απλό Ant System. Όπως φαίνεται και από το παρακάτω διάγραμμα, η επιπλέον φερομόνη που εναποθέτει το καλύτερο έως τώρα μυρμήγκι αυξάνει την αναζήτηση λύσης κοντά σε αυτή.



Για το πρόβλημα των 50 πόλεων η γραφική αναπαράστασταση της καλύτερης λύσης που βρέθηκε με τον elitist-AS είναι η εξής :



Τα αποτελέσματα του αλγορίθμου **AS-Rank** για τις καλύτερες ρυθμίσεις που βρέθηκαν.

Παράμετροι

AS-Rank
sigma=6
a=1 b=4
m=51
r=0,1

Λύση

Μέσος Όρος	χειρότερη λύση	καλύτερη λύση	1	2	3	4	5
2360,292	2430,5987	2317,5715	2430,5987	2328,0619	2346,442	2378,7859	2317,572

Παράμετροι

AS-Rank
sigma=8
a=1 b=4
m=51
r=0,1
Best

Λύση

Μέσος Όρος	χειρότερη λύση	καλύτερη λύση	1	2	3	4	5
2317,18578	2374,4215	**2289,1452**	2306,0647	2374,4215	2324,492	2289,1452	2291,805

Παράμετροι

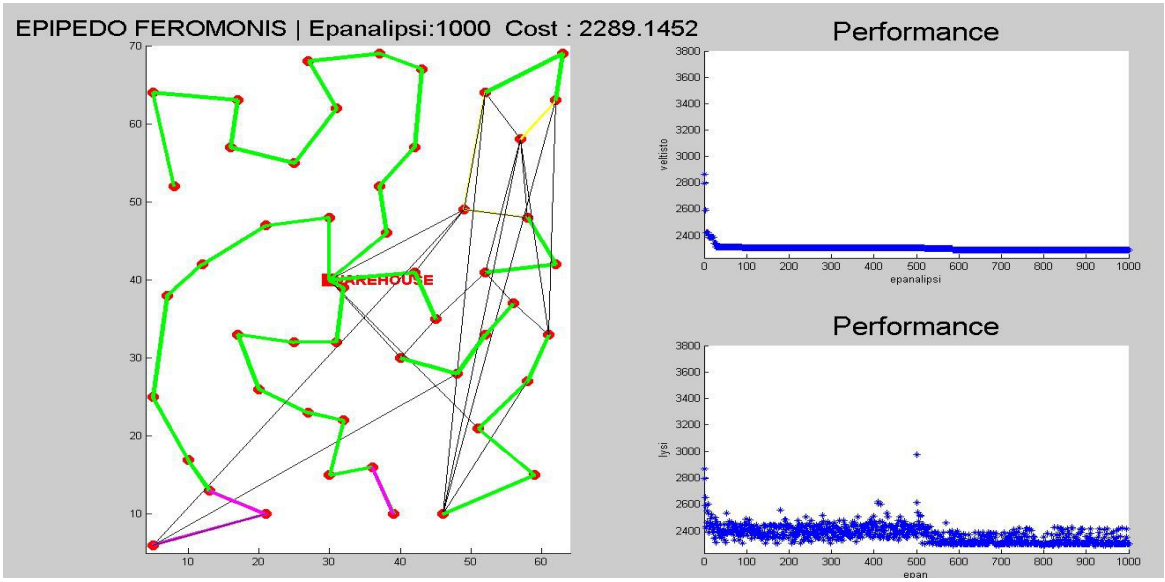
AS-Rank
sigma=10
a=1 b=4
m=51
r=0,1

Λύση

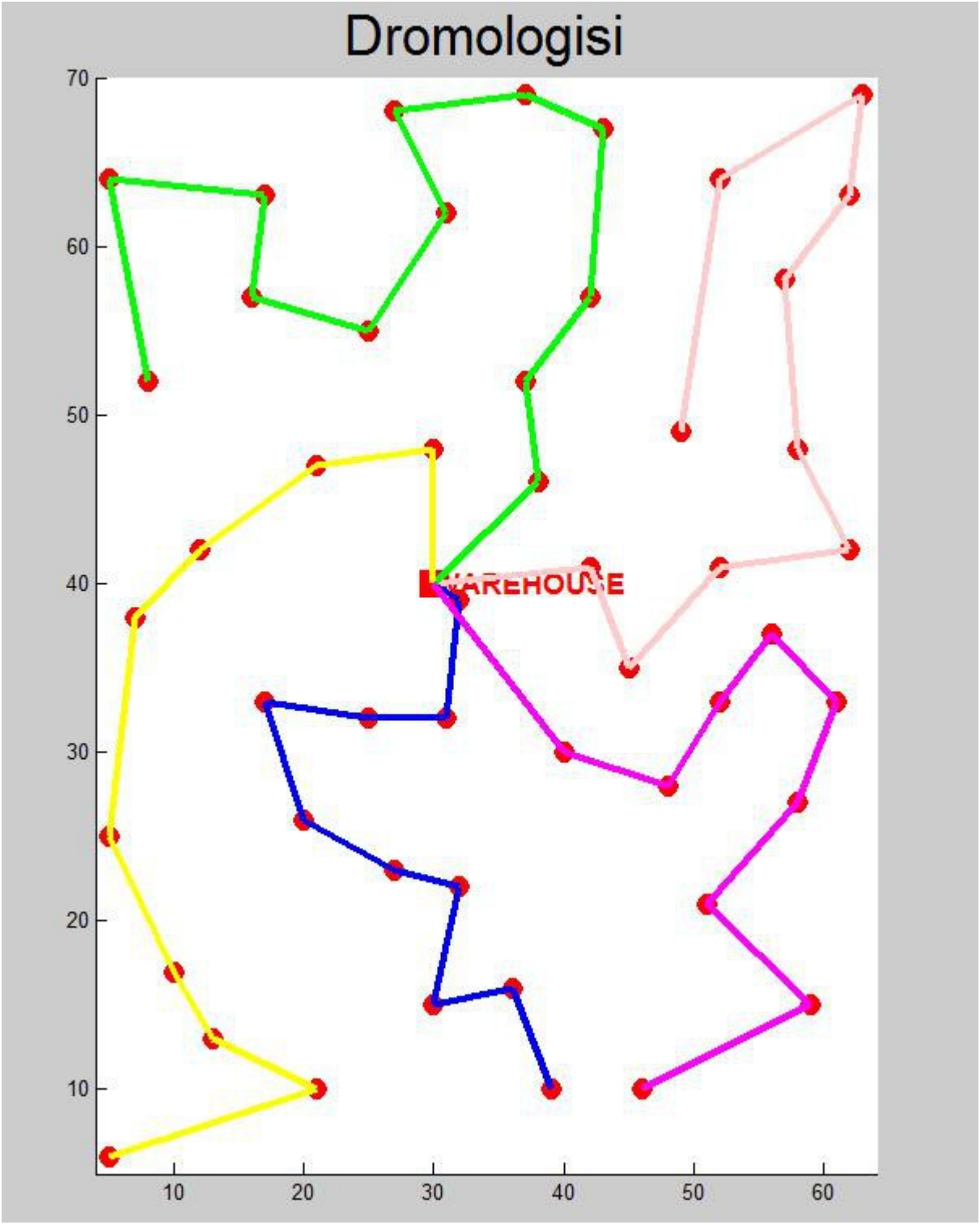
Μέσος Όρος	χειρότερη λύση	καλύτερη λύση	1	2	3	4	5
2346,03996	2404,4765	2315,4137	2404,4765	2348,26	2324,492	2337,5573	2315,414

Σχολιασμός

Τα αποτελέσματα του AS-Rank είναι αισθητά καλύτερα. Αρκεί να σημειώσουμε ότι οι μέσοι όροι όλων των δοκιμών είναι καλύτεροι από τον ""καλύτερο μ.ο" του AS-Elitist! Η βέλτιστή λύση που δίνει ο AS-Rank είναι σαφώς καλύτερη των δύο προηγουμένων. Ο λόγος είναι ότι τα σίγμα(s) καλύτερα μυρμήγκια που εναποθέτουν φερομόνη, αυξάνουν την αναζήτηση στο χώρο αυτών τον λύσεων και δεν επικεντρώνονται σε μία όπως στο Elisti-AS. Έτσι καταλήγουμε σε καλύτερα αποτελέσματα όπως αποδεικνύεται και παρακάτω στα υπόλοιπα προβλήματα.



Για το πρόβλημα των 50 πόλεων η γραφική αναπαράσταση της καλύτερης λύσης που βρέθηκε με τον AS-Rank είναι η εξής :



Τα αποτελέσματα του αλγορίθμου **AntColonySystem** για τις καλύτερες ρυθμίσεις που βρέθηκαν.

Παράμετροι

ACS
a=1 b=4
r=0.1
ksi=0.1
qo=0.9
M=10 best

Λύση

Μέσος Όρος	χειρότερη λύση	καλύτερη λύση	1	2	3	4	5
2348,73962	2409,4044	**2305,761**	2305,761	2358,362	2356,6206	2313,55	2409,404

Παράμετροι

ACS
a=1 b=4
r=0.1
ksi=0.1
qo=0.9
m=5

Λύση

Μέσος Όρος	χειρότερη λύση	καλύτερη λύση	1	2	3	4	5
2357,1	2404,2	2306	2377,1	2404,2	2306	2341,6	2356,6

Παράμετροι

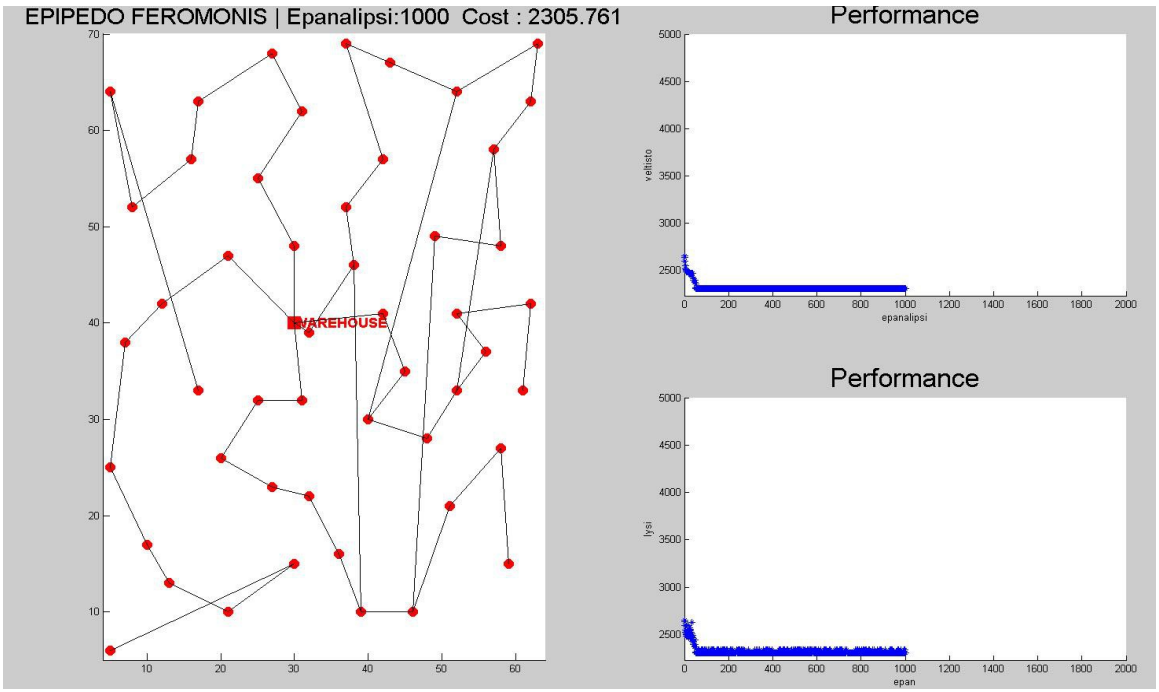
ACS
a=1 b=5
r=0.1
ksi=0.1
qo=0.9
m=10

Λύση

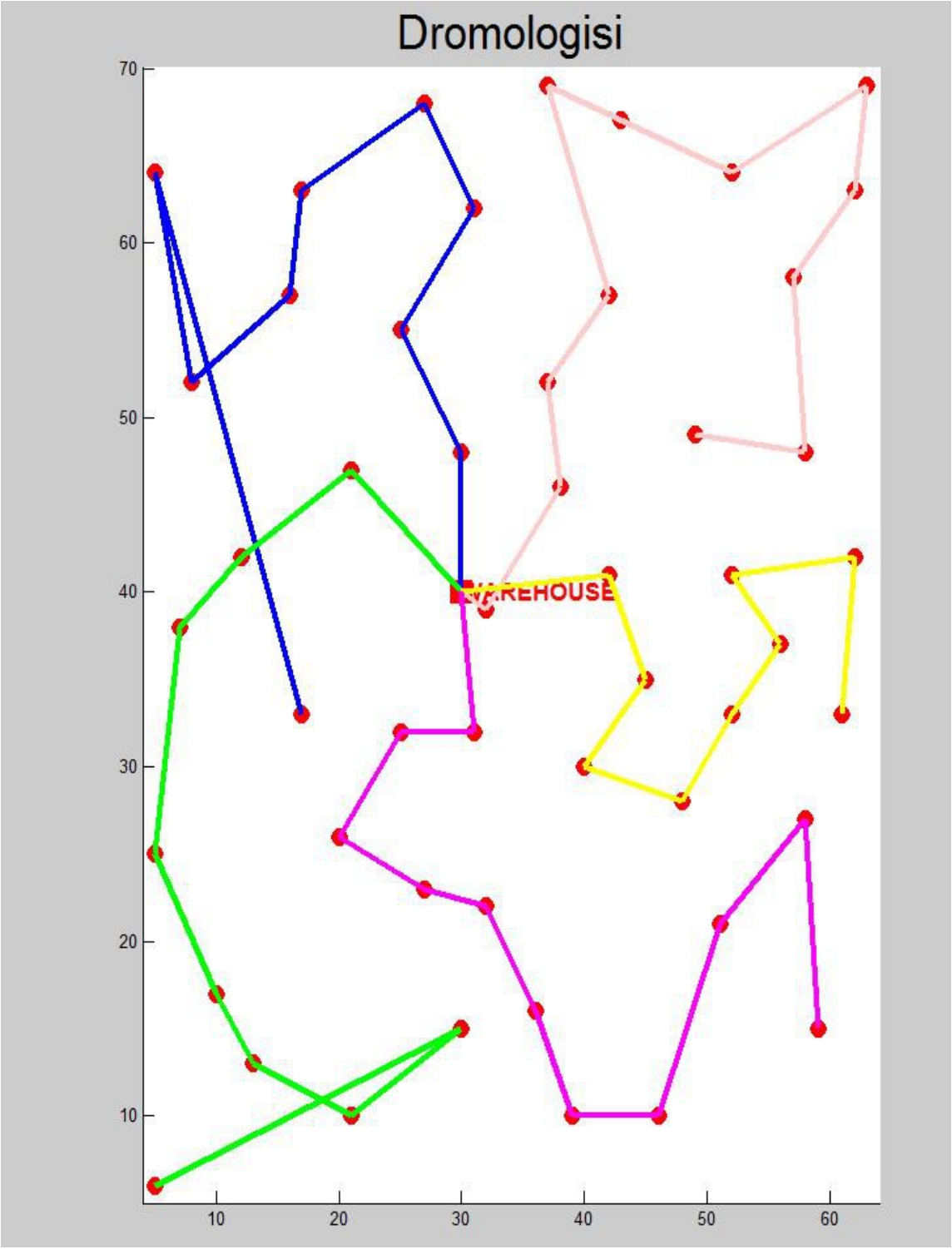
Μέσος Όρος	χειρότερη λύση	καλύτερη λύση	1	2	3	4	5
2339,4	2362,3	2308,5	2362,3	2308,5	2325,9	2357,2	2311,8

Σχολιασμός

Αν και ο Ant Colony System θεωρείται από τους καλύτερους αλγορίθμους, είναι ιδιαίτερα δύσκολος στην ρύθμιση των παραμέτρων του. Η στρατηγική με την οποία ψάχνει για λύσεις διαφέρει από τους υπολοίπους καθώς αφαιρεί φερομόνη από τις διαδρομές κάθε επανάληψης και προσθέτει στις καλύτερης λύσης. Έτσι ενισχύεται η αναζήτηση άλλων λύσεων. Για να είναι αποτελεσματική αυτή η αναζήτηση και να μην αποκλίνει ο αλγόριθμος απαιτείται να υπάρχει μια ισορροπία μεταξύ της φερομόνης που εξατμίζεται, του αριθμού των λύσεων και της φερομόνης που τοποθετεί το βέλτιστο μυρμήγκι. Δοκιμάσαμε τις προτεινόμενες από τη βιβλιογραφία ρυθμίσεις όμως τα αποτελέσματα δεν ήταν τα αναμενόμενα.



Για το πρόβλημα των 50 πόλεων η γραφική αναπαράσταση της καλύτερης λύσης που βρέθηκε με τον ACS είναι η εξής :



Τα αποτελέσματα του αλγορίθμου **Min-Max** AS για τις καλύτερες ρυθμίσεις που βρέθηκαν.

Παράμετροι

MINMAX
a=1
b=5
m=NN
r=0,02
tmin=tmax/300 best

Λύση

Μέσος Όρος	χειρότερη λύση	καλύτερη λύση	1	2	3	4	5
2290,1399	2299,5097	**2276,2**	2294,7	2299,5097	2287,7637	2292,5261	2276,2

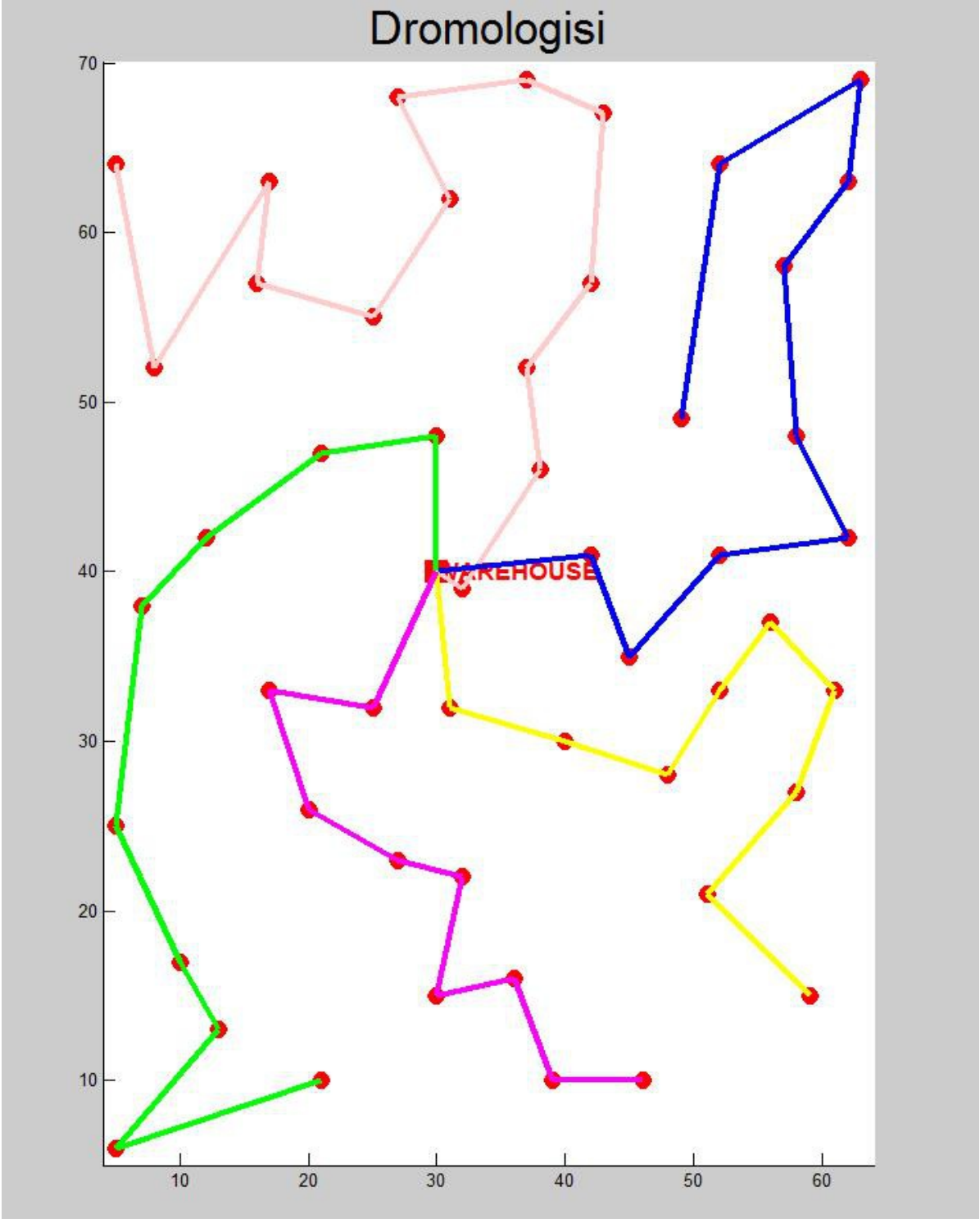
Παράμετροι

MINMAX
a=1
b=5
m=10
r=0,1
tmin=tmax/300

Λύση

Μέσος Όρος	χειρότερη λύση	καλύτερη λύση	1	2	3	4	5
2308,54	2321,8	2284,9	2303,1	2315,4	2317,5	2321,8	2284,9

Για το πρόβλημα των 50 πόλεων η γραφική αναπαράσταση της καλύτερης λύσης που βρέθηκε με τον Min-Max AS (Η καλύτερη από όλους τους αλγορίθμους) είναι η εξής



4.4.2 Προβλήματα Par2, Par3, Par4, Par11, Par12

Για τα προβλήματα αυτά παρουσιάζεται συγκεντρωτικός πίνακας αποτελεσμάτων. Για αυτά τα προβλήματα χρησιμοποιήθηκαν οι αποτελεσματικότεροι από τους αλγορίθμους, δηλαδή ο Min-Max AS και ο AS-Rank Στα προβλήματα αυτά χρησιμοποιήθηκαν οι καλύτερες ρυθμίσεις που βρέθηκαν στο πρόβλημα par1. Στις παρενθέσεις είναι τα καλύτερα αποτελέσματα απο την πηγή [5]

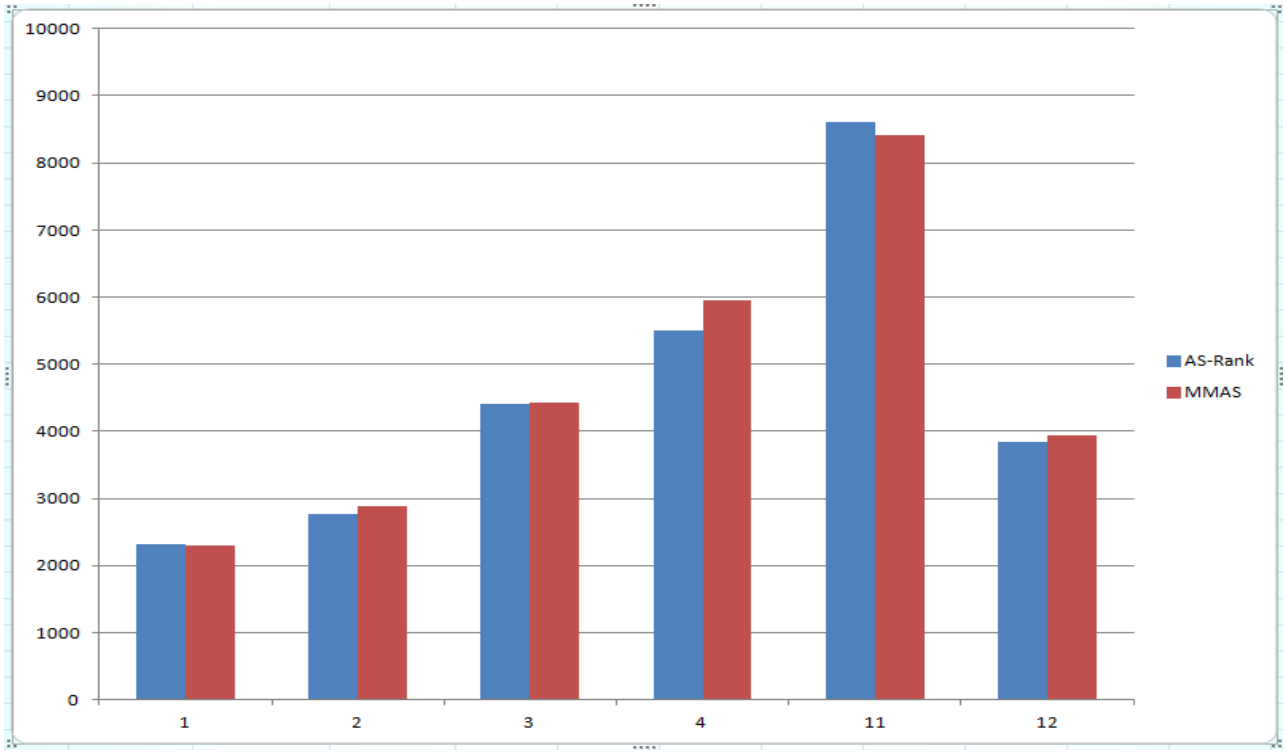
AS-Rank	Πρόβλημα	Πόλεις	Μέσος Όρος	χειρότερη λύση	καλύτερη λύση	1	2	3	4	5
sigma=8	*1(2230,35)*	50	2317,18578	2374,4215	**2289,1452**	2306,0647	2374,4215	2324,492	**2289,1452**	2291,805
a=1 b=4	*2(2391,63)*	75	2771,48	2865,9	**2579,1**	**2579,1**	2788,7	2782,2	2865,9	2841,5
m=51	*3(4045,42)*	100	4406,02	4435,5	**4368,1**	4435,5	4418,8	4400,7	4407	**4368,1**
r=0,1	*4(4987,52)*	150	5503,26	5534,8	**5491,5**	5534,8	5493,5	5500,3	**5491,5**	5496,2
Best	*11(7315,87)*	120	8615,08	8696,5	**8536**	8647,6	8696,5	8634,8	8560,5	**8536**
	12(3558,92)	100	3850,56	3883,9	**3825,5**	3840,3	3883,9	3867,6	**3825,5**	3835,5

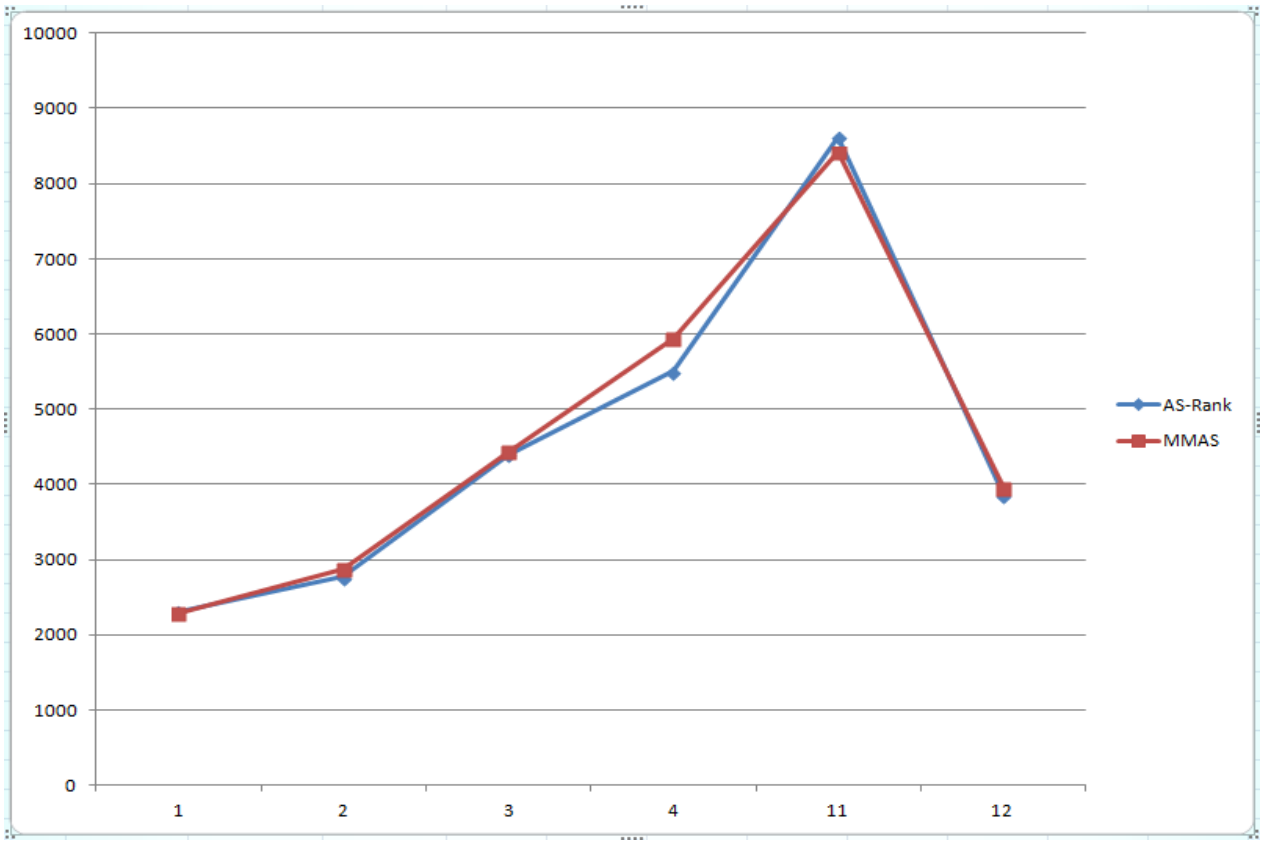
MINMAX	Πρόβλημα	Πόλεις	Μέσος Όρος	χειρότερη λύση	καλύτερη λύση	1	2	3	4	5
a=1	*1(2230,35)*	50	**2290,1399**	2299,5097	**2276,2**	2294,7	2299,5097	2287,7637	2292,5261	**2276,2**
b=5	*2(2391,63)*	75	2880,63504	3019,2	**2799,7**	**2799,7**	2890,787	2837,7882	3019,2	2855,7
m=NN	*3(4045,42)*	100	4432,84	4471,2	**4328,4**	4453	4440,4	**4328,4**	4471,2	4471,2
r=0,02	*4(4987,52)*	150	5946,7	6169,6	**5816**	6169,6	5842,4	5903,2	**5816**	6002,3
tmin=tmax/300	*11(7315,87)*	120	8416,78	8496,9	**8354,1**	**8354,1**	8354,1	8484,6	8496,9	8394,2
best	*12(3558,92)*	100	3947,48	3966,7	**3910,4**	3958,2	**3910,4**	3966,7	3966,4	3935,7

Σχολιασμός

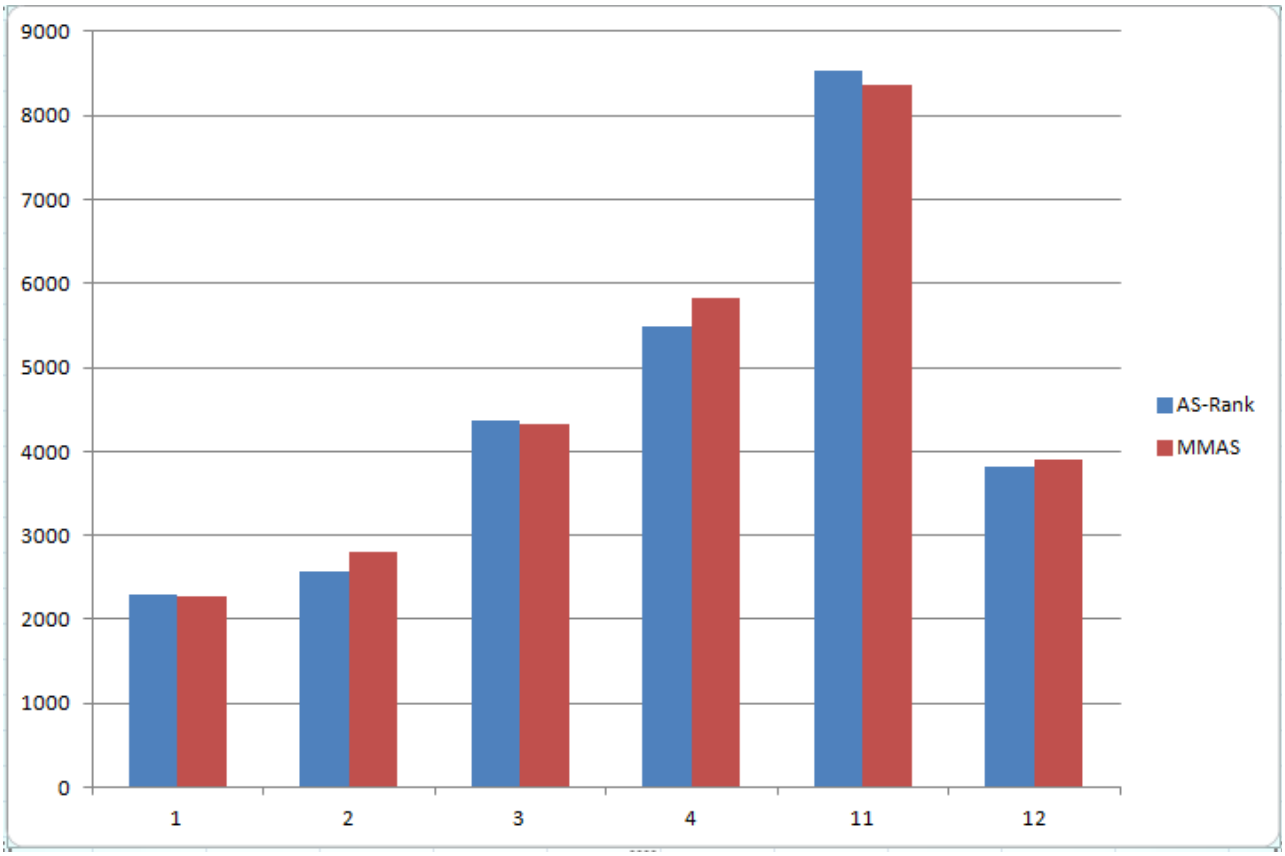
Όπως παρατηρούμε, δεν υπάρχει ένας αλγόριθμος που να δίνει καλύτερο αποτέλεσμα σε όλα τα προβλήματα. Διαπιστώνουμε ότι δεν παίζει ρόλο μόνο το πλήθος των πόλεων σε αυτά, αλλά και η γεωγραφία του γραφήματος. Για παράδειγμα ο Min-Max δίνει καλύτερα αποτελέσματα στο Par1, Par11 αλλά χειρότερα στα Par2, Par12. Έτσι δεν μπορούμε να εξάγουμε ασφαλές συμπέρασμα για το ποιος αλγόριθμος είναι πιο αποδοτικός για τα συγκεκριμένα προβλήματα. Πιθανώς με διαφορετικές ρυθμίσεις των παραμέτρων και οι δύο αλγόριθμοι να έδιναν καλύτερα αποτελέσματα.

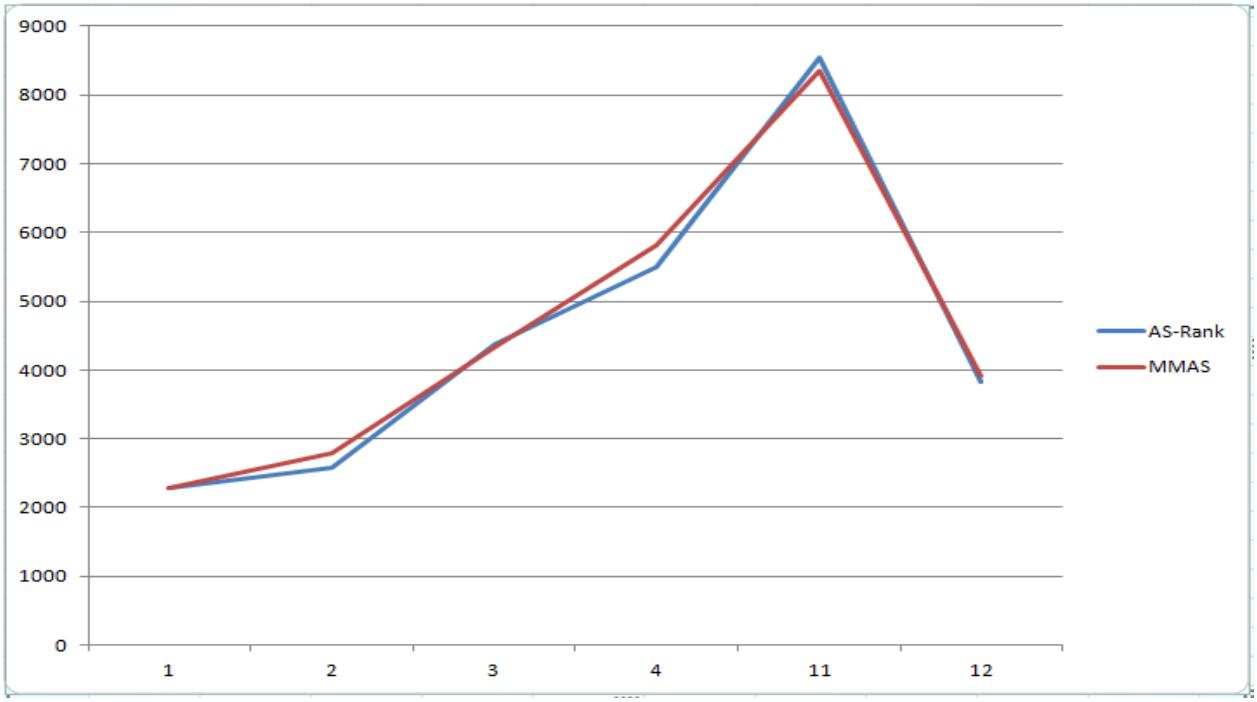
Παρακάτω φαίνεται το συγκριτικό διάγραμμα των μ.ο των αποτελεσμάτων των δύο αλγορίθμων για τα προβλήματα του πακέτου Par.





Αντίστοιχα για τα καλύτερα αποτελέσματα μεταξύ των δύο τα διαγράμματα είναι τα εξής:





4.4.3 Προβλήματα A-n32k5 μέχρι A-n80k10

Παρουσιάζουμε συγκεντρωτικό πίνακα αποτελεσμάτων για τους αλγορίθμους Min-Max AS και AS-Rank. Τα αποτελέσματα ήταν ιδιαίτερα καλά και παρακάτω παρουσιάζονται αναλυτικότερα τα βέλτιστα που βρέθηκαν σε ορισμένες περιπτώσεις.

Τα αποτελέσματα εκτός παρενθέσεως είναι από την εργασία

Heuristic Solution Approaches for the Cumulative Capacitated Vehicle Routing Problem

Journal:	*Optimization*
Manuscript ID:	GOPT-2012-0259
Manuscript Type:	Original Article
Date Submitted by the Author:	27-Sep-2012
Complete List of Authors:	Ozsoydan, Fehmi; Dokuz Eylul University, Industrial Engineering Department Sipahioglu, Aydin; Osmangazi University,
Keywords:	Cumulative capacitated vehicle routing problem, genetic algorithm, tabu search algorithm, particle swarm optimization

Για το πρόβλημα n32k5 το αποτέλεσμα εντός παρενθέσεως είναι από την πηγή

An effective memetic algorithm for the cumulative capacitated vehicle routing problem

Sandra Ulrich Ngueveu[a,*], Christian Prins[a], Roberto Wolfler Calvo[b]

[a]Institut Charles Delaunay - LOSI, Université de Technologie de Troyes (UTT), 12, rue Marie Curie, 10000 Troyes, France
[b]Laboratoire d'Informatique de l'Université Paris-Nord (LIPN) UMR 7030, 99, av. Jean-Baptiste Clément, 93430 Villetaneuse, France

και συγκεκριμένα το εμφανίζει σε παράδειγμα όπως φαίνεται στις παρακάτω εικόνες

Fig. 1 illustrates optimal CVRP and OVRP solutions as well as our best known CCVRP solution on a classical instance A-n32-k5 from the vehicle routing literature. The CVRP solution depicted in Fig. 1(a) has been converted into a valid CCVRP solution by removing from each route one of the two edges connected to the depot. As shown in Fig. 1, optimal solutions of the three problems can differ significantly.

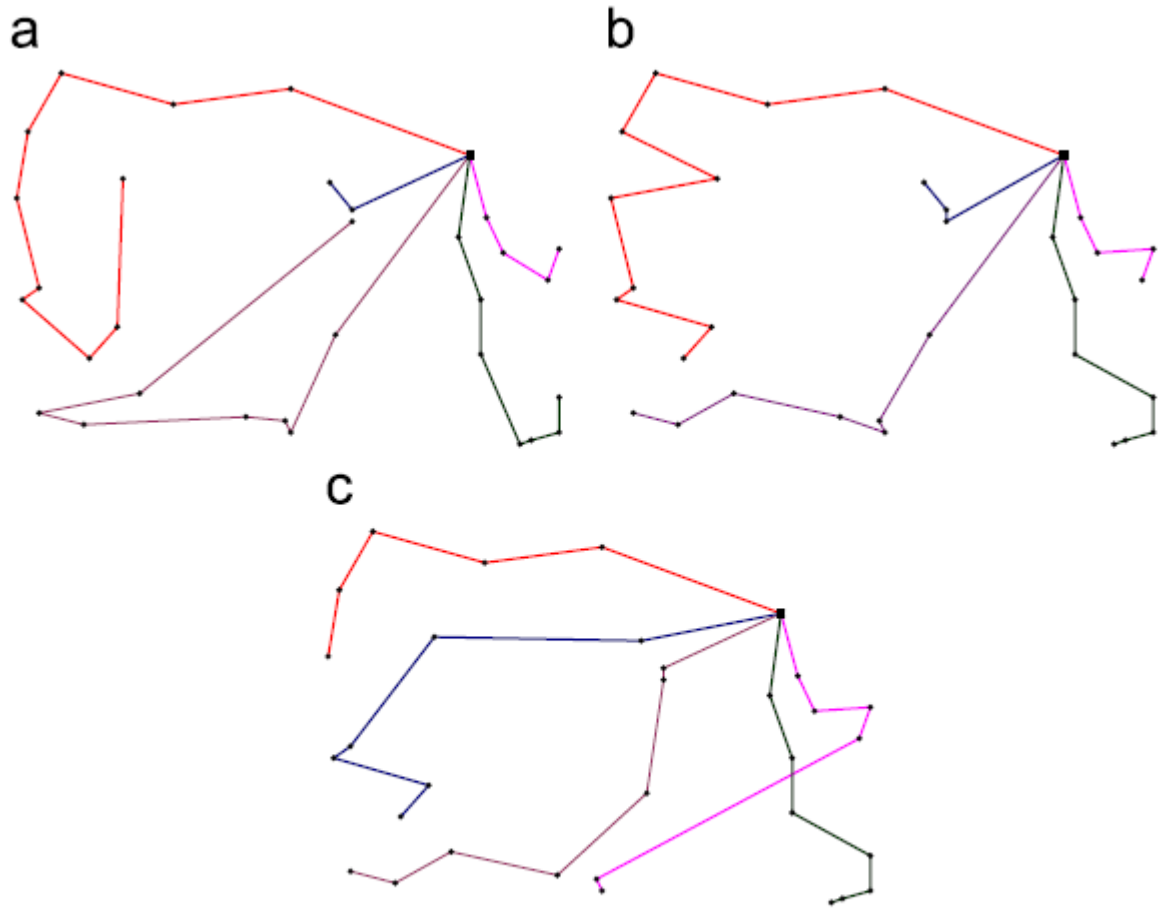


Fig. 1. Optimal solutions from CVRP and OVRP compared to CCVRP for A-n32-k5. (a) Optimal CVRP solution: recomputed CCVRP cost = 3470. (b) Optimal OVRP solution: recomputed CCVRP cost = 2512. (c) Best known CCVRP solution: cost = 2192.

Επομένως μπορούμε να υποθέσουμε οτι μερικά από τα υπόλοιπα βέλτιστα ίσως να μην είναι τα πραγματικά βέλτιστα που έχουν βρεθεί μέχρι σήμερα. Παρ'ολα αυτά, θα τα χρησιμοποιήσουμε ως μέτρο σύγκρισης των αποτελεσμάτων μας.

Στους πίνακες αποτελεσμάτων με κόκκινο φαίνονται τα αποτελέσματα τα οποία είναι καλύτερα από αυτά της εργασίας "Heuristic solution approaches for the cumulative capacitated vehicle routing problem" που αναφέραμε παραπάνω. Με Bold είναι τα καλύτερα αποτελέσματα.

Τα αποτελέσματα για τον αλγόριθμο **AS-Rank**

AS-Rank	Πρόβλημα	known best	Μέσος Όρος	χειρότερη λύση	καλύτερη λύση	1	2	3	4	5
sigma=8	n32k5	*2281,73(2192)*	2332,18802	2389,0254	**2257,6066**	2297,444	2389,025	2375,01	**2257,61**	2341,9
a=1 b=4	n33k5	*1771,17*	1724,22	1727,9	1723,3	1723,3	1723,3	1723,3	1727,9	1723,3
m=NN	n37k5	*2078,6*	2139,26	2183,8	2076,1	2076,1	2148,3	2183,8	2125,7	2162,4
r=0,1	n39k6	*2313,71*	2403,32	2442,2	**2369,2**	2442,2	2409,4	2398,3	**2369,2**	2397,5
Best	n48k7	*3239,95*	3255,88	3274,2	3236,5	3236,5	3258	3274,2	3241,8	3268,9
	n53k7	*3461,53*	3351,36	3437,8	3231	3437,8	3309,4	3231	3417,7	3360,9
	n62k8	*4162,68*	4311,04	4340,6	**4264,9**	4323,9	4307,6	**4264,9**	4320,1	4338,7
	n63k9	*4854,53*	5099,06	5126,1	**5036,9**	5075,9	5119,6	5136,8	**5036,9**	5126,1
	n69k9	*3781,63*	3802,1	3913,1	3728,4	3762,4	3913,1	3728,4	3749,2	3857,4
	n80k10	*6441,31*	6722,04	6798,9	**6659**	6798,9	6673,5	**6659**	6763,7	6715,1

Σχολιασμός

Σε αυτό το πακέτο προβλημάτων ο αλγόριθμος τρέχει πολύ ικανοποιητικά. Είναι προβλήματα μικρότερης δυσκολίας που ταιριάζουν στις ρυθμίσεις των παραμέτρων που επιλέξαμε από το Par1.Ο χρόνος στον οποίο επιτυγχάνονται τα αποτελέσματα αυτά είναι ελάχιστος, της τάξεως του λεπτού στα προβλήματα με λίγες πόλεις, έως και μερικά λεπτά για τα μεγαλύτερα προβλήματα.

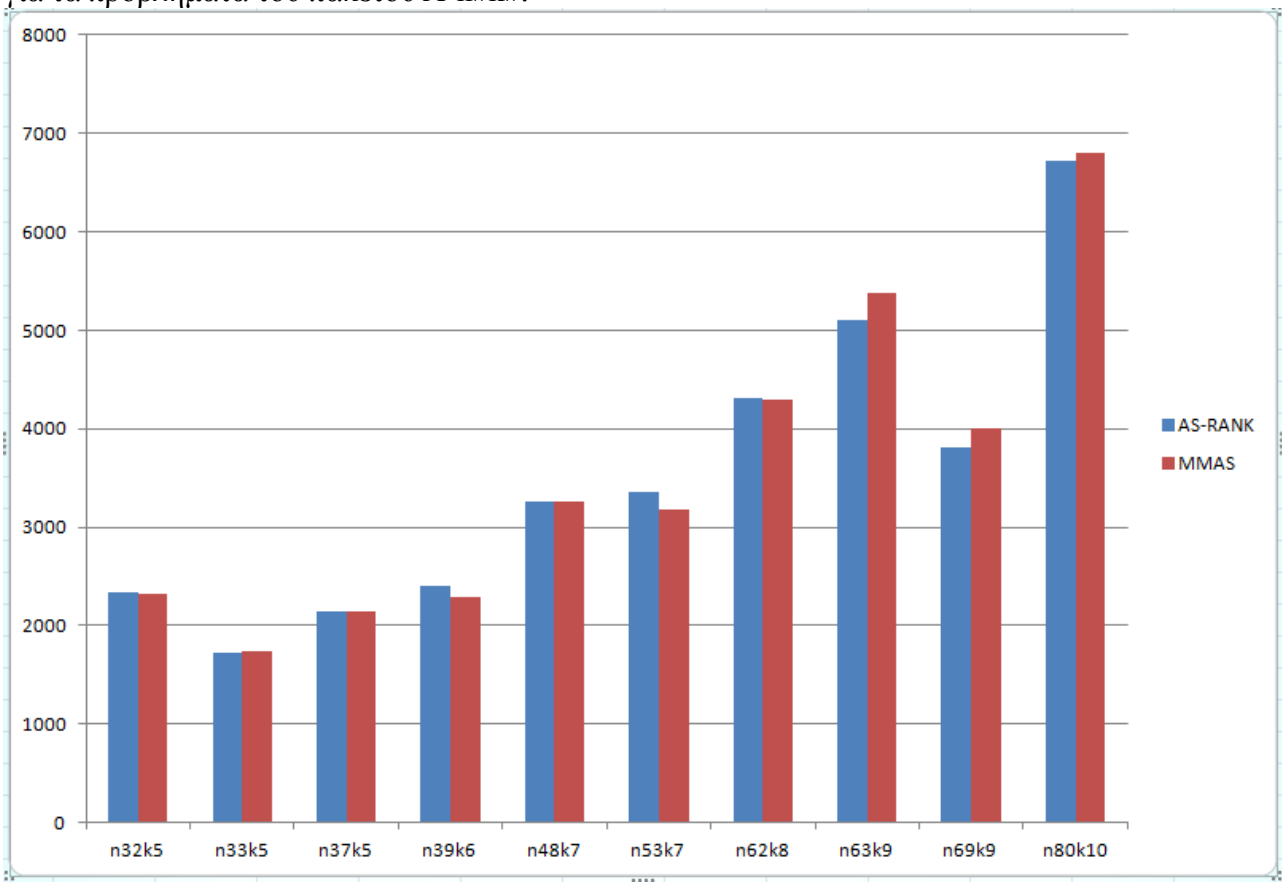
Τα αποτελέσματα για τον αλγόριθμο **Min-Max AS**

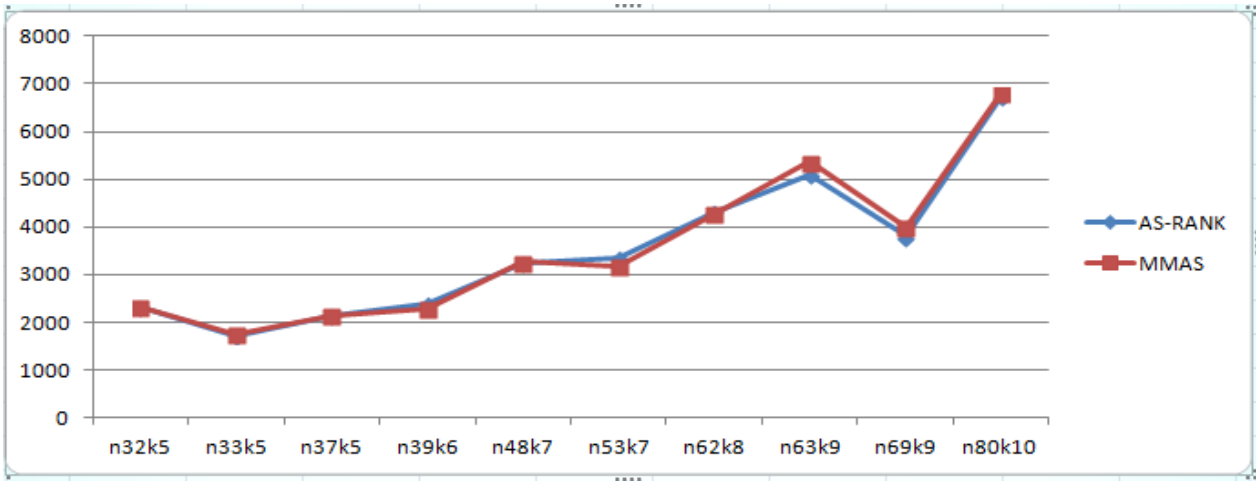
MINMAX	Πρόβλημα	known best(else best)	Μέσος Όρος	χειρότερη λύση	καλύτερη λύση	1	2	3	4	5
a=1	n32k5	*2281,73(2192)*	2325,32	2339,5	**2300,5**	2300,5	2323,7	2339,8	2323,1	2339,5
b=5	n33k5	*1771,17*	1745,7	1765	1736,9	1744,6	1740,4	1741,6	1736,9	1765
m=NN	n37k5	*2078,6*	2138,54	2153,6	**2125,1**	2136,7	2141	**2125,1**	2136,3	2153,6
r=0,02	n39k6	*2313,71*	2285,38	2311,9	2254,8	2285,5	2311,9	2272,3	2254,8	2302,4
tmin=tmax/300	n48k7	*3239,95*	3266,92	3280,2	**3249,5**	3280,2	3271,2	3270,5	**3249,5**	3263,2
best	n53k7	*3461,53*	3185,7	3280,2	3146,5	3146,5	3164	3280,2	3146,5	3191,3
	n62k8	*4162,68*	4287,36	4339,2	**4270,5**	4339,2	4252,9	4290,4	**4270,5**	4283,8
	n63k9	*4854,53*	5370,7	5451,8	**5189,1**	**5189,1**	5412,2	5359,2	5441,2	5451,8
	n69k9	*3781,63*	3996,92	4069,4	**3917,1**	4069,4	4040,5	**3917,1**	3917,1	4040,5
	n80k10	*6441,31*	6797,5	6841,2	**6716,5**	**6716,5**	6827,8	6863	6739	6841,2

Σχολιασμός

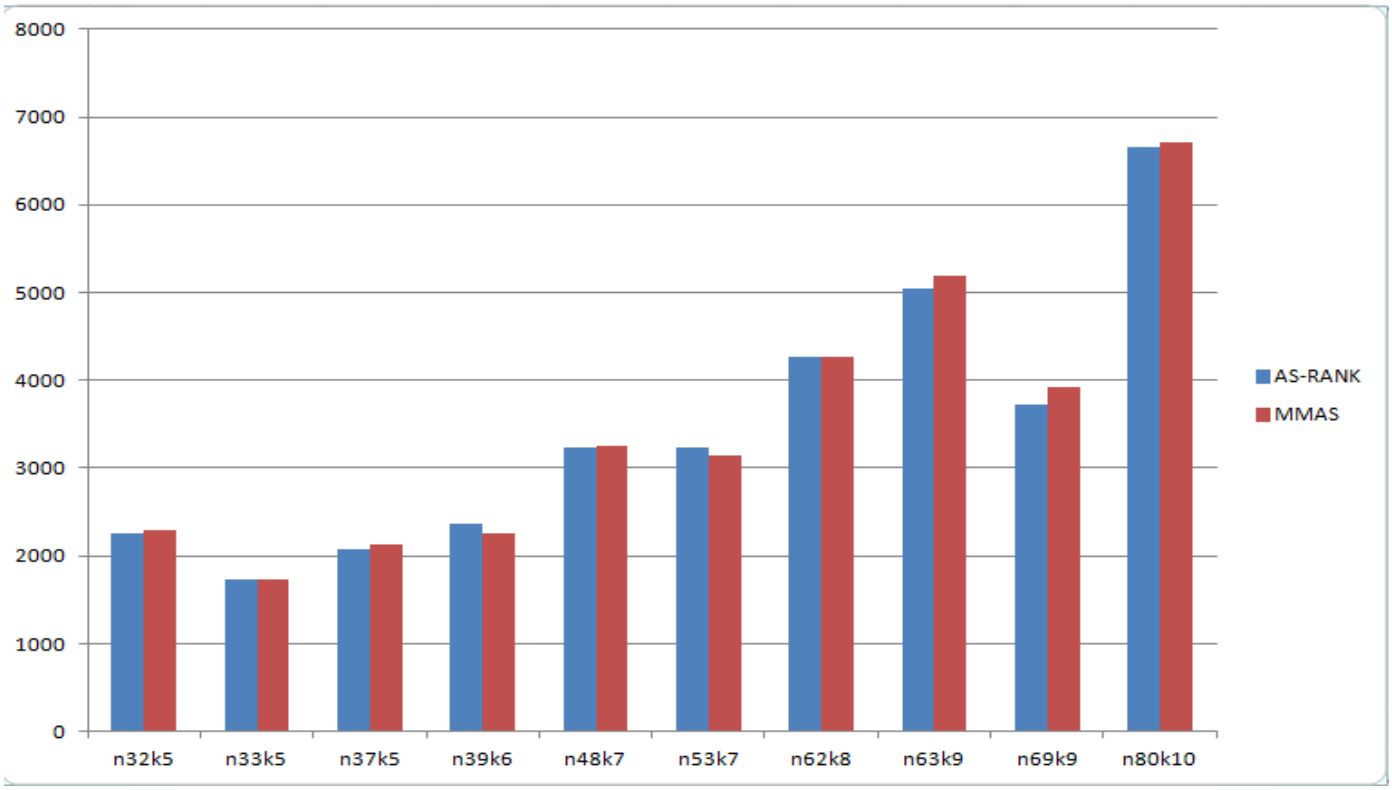
Ο αλγόριθμος Min-Max AS, λειτουργεί εξίσου ικανοποιητικά παρόλο που ο AS Rank δίνει καλύτερους μέσους όρους στην πλειοψηφία των προβλημάτων. Γενικά και σε αυτό το πακέτο προβλημάτων δεν μπορεί να εξαχθεί απόλυτο συμπέρασμα μεταξύ των δύο αλγορίθμων, ωστόσο πλέον θα μπορούσαμε να δώσουμε ένα ελαφρύ προβάδισμα στον AS Rank.

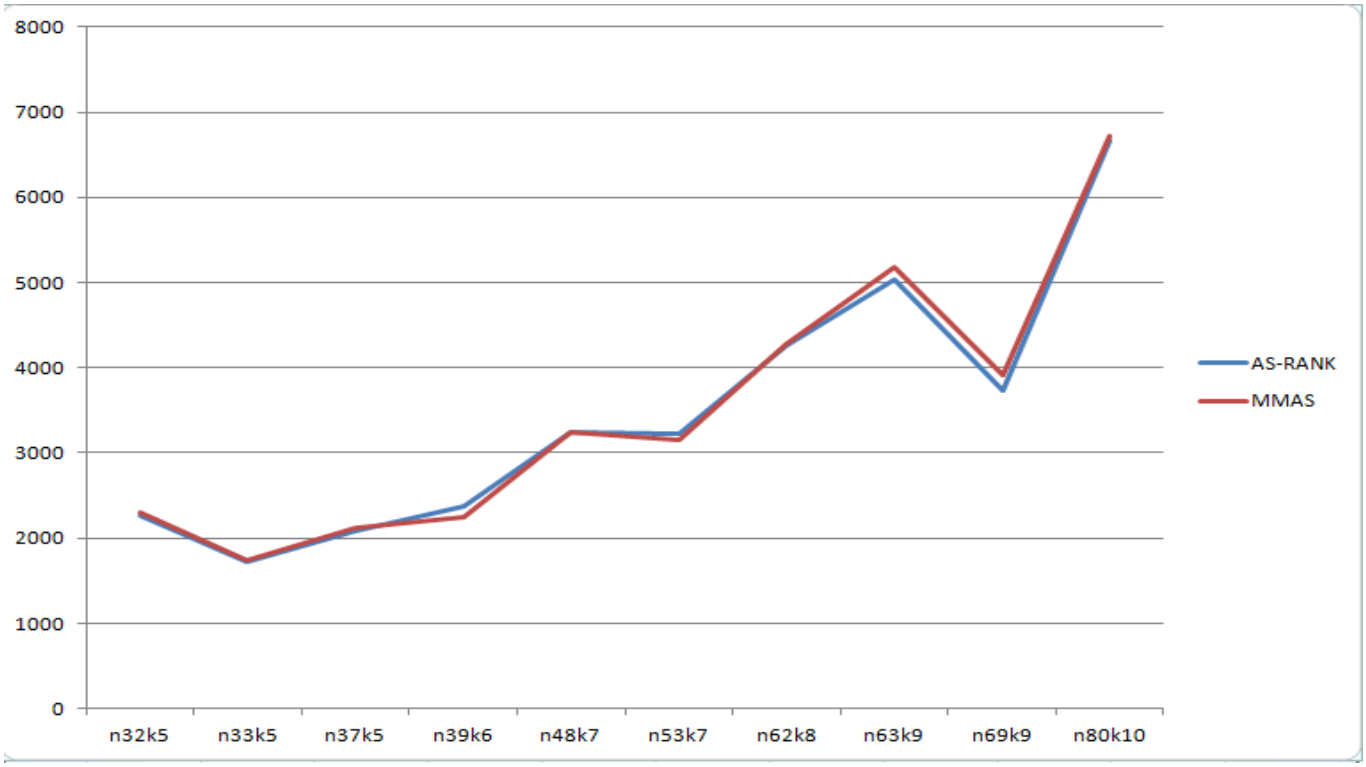
Παρακάτω φαίνεται το συγκριτικό διάγραμμα των μ.ο των αποτελεσμάτων των δύο αλγορίθμων για τα προβλήματα του πακέτου A-n#k#.





Παρακάτω φαίνεται το συγκριτικό διάγραμμα των καλύτερων αποτελεσμάτων των δύο αλγορίθμων για τα προβλήματα του πακέτου A-n#k#.





4.5 Συμπεράσματα και προτάσεις

Ο αλγόριθμος των μυρμηγκιών είναι εν γένει ένας πολύ αποδοτικός αλγόριθμος. Παρόλα αυτά, τα αποτελέσματα του από εκδοχή σε εκδοχή διαφέρει. Επίσης διαφοροποίηση μπορεί να υπάρξει και μεταξύ του ίδιου αλγορίθμου, ανάλογα τις ρυθμίσεις των παραμέτρων που αφορούν την αρχικοποίηση και ανανέωση της φερομόνης, τον αριθμό των μυρμηγκιών και τις πιθανότητες επιλογής της επόμενης πόλης.

Κάθε πρόβλημα προς επίλυση επισύρει προσαρμογή του αλγορίθμου ώστε τα αποτελέσματα που θα βγάλει να είναι βέλτιστα. Στην παραπάνω εφαρμογή, μελετήσαμε τους αλγορίθμους για ρυθμίσεις παραμέτρων κοντά στις προτεινόμενες από τη βιβλιογραφία τιμές για το πρόβλημα των 50 πόλεων και ύστερα εφαρμόσαμε τις ρυθμίσεις που υπερίσχυσαν και στα υπόλοιπα προβλήματα. Ο τρόπος που είναι δομημένα τα προγράμματα ευνοεί τα προβλήματα μικρού μεγέθους, ενώ σε μεγάλα προβλήματα παρουσιάζει καθυστερήσεις. Η χρήση της Matlab καθιστά ακόμα πιο αργό τον κώδικα, σε σχέση με το χρόνο που θα έπαιρνε σε γλώσσες όπως η C ή Fortran.

Στον κώδικα κατά τη δημιουργία μιας λύσης, ελέγχεται αν αυτή χρησιμοποιεί τον ελάχιστο απαιτούμενο αριθμό οχημάτων και σε περίπτωση που μπορεί να τα τοποθετήσει σε άλλες διαδρομές την κρατάει αλλιώς επαναλαμβάνει την αναζήτηση λύσης. Αυτό έχει ως αποτέλεσμα, προβλήματα που χωρίς μεγάλη ελαστικότητα ως προς τον αριθμό των οχημάτων, να καθυστερούν να βρουν εφικτές λύσεις. Το φαινόμενο αυτό παρατηρήθηκε στο πρόβλημα par2 που αν και έχει 75 πόλεις άργησε πολύ να δώσει λύσεις.

Αυτή η κωλυσιεργία, θα μπορούσε να επιλυθεί εάν αντί για την επαναληπτική μέθοδο, στην οποία, βάσει των πιθανοτήτων αναμένουμε κάποια στιγμή να καταλήξει σε εφικτή λύση, χρησιμοποιούνταν ένα πέναλτι ή ενός είδους μνήμη για τα μυρμήγκια ώστε οι επόμενες λύσεις να αποφεύγουν αυτόν τον συνδυασμό δρομολογίων.

Όσον αφορά τον αλγόριθμο αυτό καθ αυτό θα μπορούσε να χρησιμοποιηθεί ένας συνδυασμός αυτών, όπως για παράδειγμα ο AS-Rank με την ύπαρξη ελάχιστου και μέγιστη ορίου φερομόνης κατά τα πρότυπα του Min Max AS. Μια τέτοια εφαρμογή θα αύξανε την εξερεύνηση (exploration) και ταυτόχρονα θα εστίαζε γύρω από ένα χώρο καλών λύσεων. Έτσι δεν θα εγκλωβιζόταν σε τοπικά ελάχιστα καθώς η φερομόνη δεν θα έτεινε σε μηδενικές τιμές σε κανένα τόξο.

Συνοψίζοντας, οι κώδικες που δημιουργήσαμε ικανοποιούν το στόχο της εργασίας, δηλαδή τη μελέτη ως προς την αποδοτικότητα τους μεταξύ τους αλλά και όσο το δυνατόν καλύτερη προσέγγιση των βέλτιστων που έχουν βρεθεί για αυτά. Οι προσαρμογές είναι απαραίτητες για την βέλτιστη επίλυση κάθε ενός προβλήματος ξεχωριστά.

ΒΙΒΙΟΓΡΑΦΙΑ

Ξένη

1. .Marco Dorigo, Thomas Stützle,2004. "Ant Colony Optimization", Milan

2. Sandra Ulrich Ngueveua, Christian Prinsa, Roberto WolflerCalvob, "An effective memetical gorithm for the cumulative capacitated vehicle routing problem"

3. Amir Salehipour, Kenneth Sörensen, Peter Goos & Olli Bräysy, " An efficient GRASP+VND metaheuristic for the traveling repairman problem"

4. Ping Chen, Xingye Dong, and Yanchao Niu,"An Iterated Local Search Algorithm for the Cumulative Capacitated Vehicle Routing Problem"

5. Glaydston Mattos Ribeiro,Gilbert Laporte"An Adaptive Large Neighborhood Search Heuristic for the Cumulative Capacitated Vehicle Routing Problem"

6. Sandra Ulrich NGUEVEU, Christian PRINS, Roberto WOLFLER-CALVOY "A Memetic Algorithm for the Cumulative Capacitated Vehicle Routing Problem"

7. James Kennedy, Rusell C Eberhart "Swarm intelligence"

8. Eric Bonabeau, Marco Dorigo,Guy Theraulaz,"Swarm intelligence from natural to artificial systems "1999,Oxford University Press

Ελληνική

9. Ιωάννης Μαρινάκης, Μυγδάλας Αθανάσιος, 2008, "Σχεδιασμός και Βελτιστοποίηση της εφοδιαστικής αλυσίδας", Σοφία

10. Μποζάνης Δ. Παναγιώτης,2005, "Αλγόριθμοι", Τζιόλα