Πανεπιστήμιο Πειραιώς – Τμήμα Πληροφορικής

Πρόγραμμα Μεταπτυχιακών Σπουδών

«Προηγμένα Συστήματα Πληροφορικής»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	**Πρόβλημα Ένταξης Μονάδων Παραγωγής**
Ονοματεπώνυμο Φοιτητή	**Πατσιούδης Δημήτριος του Γεωργίου**
Αριθμός Μητρώου	**ΜΠΣΠ/07020**
Κατεύθυνση	**Ευφυείς Τεχνολογίες Αλληλεπίδρασης Ανθρώπου-Υπολογιστή**
Επιβλέπων	**Ευάγγελος Φούντας, Καθηγητής**

05/10/2010 **Οκτώβριος 2010**

Τριμελής Εξεταστική Επιτροπή

(υπογραφή)	(υπογραφή)	(υπογραφή)
Ευάγγελος Φούντας Καθηγητής	Π. – Γ. Τσικούρας Καθηγητής	Δημήτριος Αποστόλου Επίκουρος Καθηγητής

Πριν ξεκινήσει η παρουσίαση της μεταπτυχιακή μου διατριβής, θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή μου Δρ. Φούντα Ευάγγελο, καθηγητή & Πρόεδρο του τμήματος Πληροφορικής του Πανεπιστημίου Πειραιώς, έναν ιδιαίτερα προοδευτικό επιστήμονα και εξαίρετο άνθρωπο, ο οποίος υποστήριξε την ερευνητική αυτή προσπάθεια της εργασίας μου με θέρμη και ζήλο.

Εξαίρετη και εξίσου σημαντική ήταν και η συμβολή του Δρ. Βλάχου καθηγητή στο τμήμα Πληροφορικής του Πανεπιστημίου Πειραιώς, ένθερμος υποστηριχτής της προόδου της τεχνολογίας στον συγκεκριμένο τομέα, όπου η πολυετής πείρα του και οι γνώσεις του ήταν πολύ σημαντικές στην ολοκλήρωση της μεταπτυχιακή μου αυτής διατριβής. Θα ήθελα λοιπόν να τον ευχαριστήσω για τις πολύωρες συζητήσεις που είχαμε, οι οποίες εμπλούτισαν και διαφώτισαν τις γνώσεις μου, καθώς και την ευγενή φιλοξενία του για την οποία είμαι βαθύτατα υποχρεωμένος.

Πίνακας Περιεχομένων

Πανεπιστήμιο Πειραιώς – Τμήμα Πληροφορικής 1

Πρόγραμμα Μεταπτυχιακών Σπουδών 1

«Προηγμένα Συστήματα Πληροφορικής» 1

Μεταπτυχιακή Διατριβή 1

Περίληψη 7

Abstract 7

1. Εισαγωγή 8

2. Βασικές Έννοιες 10

2.1 Ένταξη Μονάδων Παραγωγής (Unit Commitment) 10

2.2 Αλγόριθμοι Μυρμηγκιών 11

3. Υπάρχοντα Συστήματα –Μεθοδολογία 14

3.1. Εφαρμογές Αλγορίθμων Μυρμηγκιών 14

3.2. Πρόβλημα Ένταξης Μονάδων Παραγωγής (Unit Commitment Problem) 15

3.3. Μεθοδολογία 18

4. Ένταξη Μονάδων Παραγωγής σε Χρονικό Ορίζοντα 24ώρου 19

4.1 Ορισμός προβλήματος 19

4.1.1 Θεωρία 19

4.1.1.1. Μελέτη Κόστους Μονάδας Παραγωγής 19

4.1.1.2. Συνολική Μελέτη Μονάδων Παραγωγής 20

4.1.1.3. Μελέτη επιβαλλόμενων Περιορισμών / Απαιτήσεων 20

4.1.2 Αντικείμενο Προβλήματος 21

4.2 Αποικίες Μυρμηγκιών σε Προβλήματα Συνεχών Μεταβλητών 21

4.2.1. Υπάρχουσες Προσεγγίσεις 21

4.2.2. Επιλογή Αλγορίθμου 22

4.2.3. Ο Αλγόριθμος Con – ANT 23

5. Υλοποίηση / Εκτέλεση Πειραμάτων 25

5.1 Πρότυπη Υλοποίηση 25
5.1.1 Χρησιμοποιούμενα Εργαλεία Ανάπτυξης 25
5.1.2 Δομή Εφαρμογής / Αλγορίθμου 26
5.1.2.1. Μοντελοποίηση Μονάδων Παραγωγής 26
5.1.2.2. Υλοποίηση Αλγορίθμου Con-ANT 27
5.1.3 Ανάλυση Εφαρμογής 28
5.1.3.1. Περιγραφή Λειτουργικότητας 28
5.1.3.2. Περιγραφή Παραμέτρων 30
5.1.3.3. Περιγραφή Εκτέλεσης 31
5.2 Εκτέλεση Πειραμάτων 32
5.2.1 Περιγραφή Πειραματικού Συστήματος 33
5.2.2 Εκτέλεση Πειραμάτων 34
5.2.3 Εκτέλεση Δοκιμών Μεταβολής Παραμέτρων 40
5.2.4 Προσομοίωση Εγκατάστασης Σε Χρονικό Ορίζοντα 24 Ωρών 42
5.2.4.1 Προσομοίωση Εγκατάστασης 4 Γεννητριών 42
5.2.4.2 Προσομοίωση Εγκατάστασης 10 Γεννητριών 46
5.2.5 Συμπεράσματα 50
6. Σύνοψη - Συμπεράσματα 52
Ακρωνύμια 53
Βιβλιογραφία 54
Παράρτημα Α. Κώδικας Υλοποίησης Con-ANT 62

Πίνακας Εικόνων

Εικόνα 1. Συμπεριφορά Κοινωνίας Μυρμηγκιών 12

Εικόνα 2. Αρχική ομοικατευθυντική αναζήτηση αλγορίθμου Con-ANT 23

Εικόνα 3. Κατευθυνόμενη Αναζήτηση Αλγορίθμου Con-ANT 24

Εικόνα 4. Το περιβάλλον ανάπτυξης του SharpDevelop 26

Εικόνα 5. Διάγραμμα Ροής Αλγορίθμου Con-ANT 28

Εικόνα 6. Αρχική Οθόνη Εισαγωγής Δεδομένων Εφαρμογής 29

Εικόνα 7. Button Εκκίνησης Αλγορίθμου 31

Εικόνα 8. Περιοχή Αποτελεσμάτων Εφαρμογής 32

Εικόνα 9. Σενάριο Α, 5 Ομάδες Μυρμηγκιών 35

Εικόνα 10. Σενάριο Α, 10 Ομάδες Μυρμηγκιών 36

Εικόνα 11. Σενάριο Α, 20 Ομάδες Μυρμηγκιών 37

Εικόνα 12. Σενάριο Β, 5 Ομάδες Μυρμηγκιών 38

Εικόνα 13. Σενάριο Β, 10 Ομάδες Μυρμηγκιών 39

Εικόνα 14. Σενάριο Β, 20 Ομάδες Μυρμηγκιών 40

Εικόνα 15 Χρόνος απόκρισης συναρτήσει της ανθεκτικότητας της φερορμόνης 41

Εικόνα 16 Χρόνος απόκρισης συναρτήσει της πιθανότητας σφάλματος επιλογής ... 42

Εικόνα 17. Εξέλιξη Transaction Cost στις χρονικές περιόδους 24ώρου 46

Εικόνα 18. Εξέλιξη Transaction Cost στις χρονικές περιόδους 24ώρου 50

Περίληψη

Στην εργασία αυτή μελετήθηκε η ένταξη μονάδων παραγωγής σε χρονικό ορίζοντα 24 ωρών. Όπως συμβαίνει σε πολλά συστήματα, έτσι και τα συστήματα παραγωγής ηλεκτρικής ισχύος υπάρχουν συγκεκριμένοι κύκλοι λειτουργίας. Μελετώντας αυτούς του κύκλους λειτουργίας ανακύπτει το πρόβλημα της αποδοτικής ένταξής τους με το ελάχιστο δυνατό κόστος.

Οι βασικές μέθοδοι αντιμετώπισης του προβλήματος μπορούν να ταξινομηθούν σε τρεις κατηγορίες: κλασικές μέθοδοι βελτιστοποίησης, ευριστικές μέθοδοι, και μέθοδοι τεχνητής νοημοσύνης. Μερικά μαθηματικά πρότυπα εμπνευσμένα από τη βιολογία τα οποία βρίσκουν εφαρμογή σε τεχνητές κοινωνίες οντοτήτων, όπως η εξελισσόμενη αποίκιση και τεχνητό σύστημα αποικιών μυρμηγκιών (Ant Colony System - ACS). Μια εξέλιξη του συστήματος αποικιών μυρμηγκιών, ο Con-ANT (Continuous ANT) μελετάται και εφαρμόζεται στο προαναφερθέν πρόβλημα ένταξης μονάδων παραγωγής σε χρονικό ορίζοντα 24ώρου.

Αναπτύχθηκε επίσης στη γλώσσα προγραμματισμού VB.NET της Microsoft η προσομοίωση ενός τέτοιου συστήματος αναζήτησης λύσης για το πρόβλημα αυτό, και διεξήχθησαν πειράματα. Τα αποτελέσματα των πειραμάτων αυτών συγκρίθηκαν με τα αποτελέσματα άλλων αλγορίθμων στην επίλυση του ίδιου προβλήματος με τις ίδιες συνθήκες/εγκαταστάσεις γεννητριών.

Abstract

In this essay we study the unit commitment problem (UCP) of production units in a 24 hour time horizon. As in many research areas, the electrical power generation systems work following specific cycles. Studying those cycles raises the problem of efficient commitment at the lowest possible cost.

The main methods of addressing the problem can be classified into three categories: traditional optimization methods, heuristic methods, and methods of artificial intelligence. Some mathematical models are inspired by biology which can be applied to artificial societies entities, such as colonization and evolving artificial ant colony system (Ant Colony System - ACS). An evolution of ant colony system, the Con-ANT (Continuous ANT) is studied and applied in the above mentioned problem, namely the commitment of electricity generation plants in an horizon of 24 hours.

Also an application is developed using the programming language VB.NET of Microsoft, which simulates such a system and searches for a solution to the aforementioned problem. Moreover we conducted experiments using this application comparing results of the Con-ANT to other existing algorithms addressing the same problem, under the same plant configuration.

1. Εισαγωγή

Στη λειτουργία ηλεκτρικών συστημάτων, οι λύσεις των διαφόρων προβλημάτων προσδιορίζονται βάσει της εμπειρίας και την κρίσης του ανθρώπινου παράγοντα, ή με άλλα λόγια των χειριστών των συστημάτων αυτών. Γενικά, η εμπειρία και οι κρίσεις των χειριστών μπορούν να διατυπωθούν ως ένα σύνολο κανόνων. Επαγωγικά, μια μηχανή που θα ήταν σε θέση να εφαρμόσει αυτούς τους κανόνες, θα έλυνε τα προαναφερθέντα προβλήματα σύμφωνα με το σύνολο των κανόνων που δομείται από τους χειριστές. Η υλοποίηση μιας τέτοιας διαδικασίας αυτοματοποίησης για την αντικατάσταση του ανθρώπινου παράγοντα στις κρίσιμες καταστάσεις καλείται προσέγγιση «έμπειρων συστημάτων» (*expert system*). Το πλεονέκτημα που έχει ένα «έμπειρο σύστημα» δεν περιορίζεται στη μείωση του ανθρώπινου φόρτου εργασίας στη διαδικασία λήψης απόφασης, αλλά επιλύει και τις διάφορες δυσκολίες σε πολύ συντομότερο διάστημα, και αποτρέπει τα ανθρώπινα λάθη.

Πιο συγκεκριμένα, τα προβλήματα των οποίων η λύση θα μπορούσε να διατυπωθεί σαν διακριτά σύνολα κανόνων έχουν μελετηθεί εκτενώς από την ερευνητική κοινότητα. Οι μελέτες αυτές περιλαμβάνουν την αποκατάσταση ηλεκτρικών συστημάτων, την ανάλυση ασφάλειας, την επεξεργασία συναγερμών, τη διάγνωση ελαττωμάτων, και άλλους τομείς, και τα αποτελέσματα είναι ελπιδοφόρα [1.2].

Εντούτοις, υπάρχουν διάφορα προβλήματα που συνδέονται με τη λειτουργία ηλεκτρικών συστημάτων που δεν μπορούν να αντιμετωπιστούν άμεσα από ένα σύνολο κανόνων. Τα περισσότερα από αυτά τα προβλήματα συσχετίζονται με την οικονομική λειτουργία, όπου στη βελτιστοποίηση των αντικειμενικών συναρτήσεων υπεισέρχονται συνήθως σύνθετοι περιορισμοί, και είναι δύσκολο να δομηθεί ένα κατάλληλο σύνολο κανόνων για την εύρεση της βέλτιστης λύσης για κάθε ένα από αυτά τα προβλήματα.

Ένα από τα χαρακτηριστικά παραδείγματα των προαναφερθέντων προβλημάτων είναι το πρόβλημα ένταξης μονάδων παραγωγής ενέργειας. Πράγματι λόγω της πολυπλοκότητας και του μεγάλου αριθμού περιορισμών δεν έχει καταγραφεί μέχρι στιγμής εργασία ανάλογη του μεγέθους του προβλήματος για την εύρεση «έμπειρων συστημάτων» που θα διαχειρίζονταν τέτοια συστήματα.

Το πρόβλημα ένταξης μονάδων παραγωγής (*Unit Commitment Problem - UCP*) είναι ένα μη γραμμικό, συνδυαστικό πρόβλημα βελτιστοποίησης ακέραιων αριθμών. Ορίζεται ως το πρόβλημα σχεδιασμού λειτουργίας γεννητριών σε ένα ηλεκτρικό σύστημα προκειμένου να καλυφθούν οι απαιτήσεις φορτίου με τον οικονομικότερο δυνατό τρόπο. Συνήθως αυτό το πρόβλημα εξετάζεται στα πλαίσια μιας συγκεκριμένης χρονικής περιόδου, από 24 ώρες μέχρι και λίγες εβδομάδες.

Στη συνέχεια η εργασία δομείται ως εξής:

Στο δεύτερο κεφάλαιο γίνεται μια μελέτη των βασικών εννοιών στις οποίες στηρίζεται η παρούσα εργασία, όπως οι κοινωνίες μυρμηγκιών και οι αλγόριθμοι που προκύπτουν από την παρατήρηση των κοινωνιών αυτών. Ακόμη γίνεται μια σύντομη παρουσίαση του προβλήματος ένταξης μονάδων παραγωγής. Στο τρίτο κεφάλαιο παρουσιάζονται οι υπάρχουσες προσεγγίσεις στα συναφή πεδία που πραγματεύεται η παρούσα εργασία. Μελετώνται εφαρμογές των αλγορίθμων μυρμηγκιών, παρουσιάζεται διεξοδικότερα το πρόβλημα ένταξης μονάδων παραγωγής, καθώς και προτεινόμενες λύσεις στο πρόβλημα αυτό. Κλείνοντας το κεφάλαιο αναλύεται η μεθοδολογία που ακολουθήθηκε τόσο σε θεωρητικό επίπεδο, όσο και σε επίπεδο σχεδιασμού και υλοποίησης του προγραμματιστικού μοντέλου.

Στο τέταρτο κεφάλαιο αναπτύσσεται το θεωρητικό μέρος της παρούσας εργασίας, το πρόβλημα ένταξης δηλαδή μονάδων παραγωγής σε χρονικό ορίζοντα εικοσιτετραώρου. Στο πέμπτο κεφάλαιο παρουσιάζεται ένα μοντέλο επίλυσης του προαναφερθέντος προβλήματος με τη χρήση συνεχούς αλγορίθμου μυρμηγκιών (Continuous ANT). Στο έκτο κεφάλαιο περιγράφεται η διαδικασία σχεδιασμού και υλοποίησης του μοντέλου, καθώς και τα αποτελέσματα που προέκυψαν από τα πειράματα που διεξήχθησαν βασισμένα στην υλοποίηση αυτή.

Τέλος, η εργασία κλείνει με μια σύνοψη όσων αναπτύχθηκαν, την εξαγωγή συμπερασμάτων, καθώς και τα περαιτέρω βήματα που μπορούν να ακολουθηθούν στις κατευθυντήριες που ακολουθήθηκαν στα πλαίσια της εργασίας αυτής.

2. Βασικές Έννοιες

Όπως συμβαίνει σε πολλά συστήματα, έτσι και τα συστήματα παραγωγής ηλεκτρικής ισχύος υπάρχουν συγκεκριμένοι κύκλοι λειτουργίας. Έτσι η απαίτηση για ηλεκτρική ισχύ είναι υψηλότερη την ημέρα, ενώ μειώνεται τις νυχτερινές ώρες. Αυτή η κυκλική λειτουργία καθιστά απαραίτητο το σχεδιασμό της παραγωγής ενέργειας σαν συνάρτηση της ώρας. Έτσι προκύπτει το πρόβλημα ένταξης μονάδων παραγωγής (Unit Commitment Problem - UCP) το οποίο αναλύεται στη συνέχεια.

Μια πρόταση για την επίλυση του προαναφερθέντος προβλήματος, η οποία θα εξεταστεί και στα πλαίσια της παρούσας διπλωματικής, αποτελεί η χρήση αλγορίθμων μυρμηγκιών, μια τεχνική που βρίσκει εφαρμογή σε πολλούς ερευνητικούς τομείς. Για το λόγο αυτό στο δεύτερο μέρος αυτής της παραγράφου γίνεται μια συνοπτική αναφορά στους αλγορίθμους μυρμηγκιών (Ant Algorithms).

2.1 Ένταξη Μονάδων Παραγωγής (Unit Commitment)

Το πρόβλημα της ένταξης μονάδων παραγωγής (Unit Commitment Problem - UCP) στην παραγωγική διαδικασία πηγάζει από την ανάγκη για προσδιορισμό ενός προγράμματος ενεργοποίησης και απενεργοποίησης μονάδων παραγωγής ηλεκτρικής ενέργειας με τέτοιο τρόπο ώστε να ελαχιστοποιηθεί το κόστος παραγωγής ενώ παράλληλα να καλύπτονται οι απαιτήσεις σε παραγόμενο φορτίο και μια σειρά λειτουργικών περιορισμών. Το κόστος παραγωγής περιλαμβάνει τα κόστη καυσίμου, εκκίνησης, τερματισμού καθώς και το κόστος λειτουργίας χωρίς φορτίο. Οι προαναφερθέντες περιορισμοί συνίστανται στην απαίτηση για ύπαρξη πλεονάσματος προς διάθεση ισχύος, στους ελάχιστους χρόνους εκκίνησης τερματισμού, στη μέγιστη δυνατή μετάδοση ισχύος μέσω των υπαρχουσών γραμμών και άλλοι λειτουργικοί περιορισμοί. Το UCP αποτελεί συνδυαστικό και συνεχές πρόβλημα βελτιστοποίησης, του οποίου η επίλυση είναι εξαιρετικά πολύπλοκη εξαιτίας των τεραστίων διαστάσεων, των μη γραμμικών αντικειμενικών συναρτήσεων αλλά και του μεγάλου αριθμού περιορισμών.

Το UCP είναι μια εξαιρετικά σημαντική ερευνητική περιοχή, η οποία μάλιστα κεντρίζει το ενδιαφέρον της ερευνητικής κοινότητας ολοένα και περισσότερο, καθώς ακόμα και η μικρότερη μείωση λειτουργικού κόστους ανά ώρα μπορεί να οδηγήσει σε πολύ σημαντικά οικονομικά οφέλη.

Με στόχο να μειωθούν οι απαιτήσεις αποθήκευσης και υπολογισμού λύσεων του UCP σε πραγματικά ηλεκτρικά συστήματα, διάφορες μη βέλτιστες μέθοδοι λύσης έχουν προταθεί. Οι βασικές μέθοδοι μπορούν να ταξινομηθούν σε τρεις κατηγορίες: κλασικές μέθοδοι βελτιστοποίησης, ευριστικές μέθοδοι, και μέθοδοι τεχνητής νοημοσύνης (Artificial Intelligence - AI). Οι κλασικές μέθοδοι βελτιστοποίησης όπως ο δυναμικός προγραμματισμός (dynamic programming - DP) και η Lagrangian Relaxation χρησιμοποιήθηκαν πρωτογενώς για την επίλυση του συγκεκριμένου προβλήματος. Οι ευριστικές μέθοδοι όπως ο κατάλογος προτεραιότητας (Priority List - PL) παρέχουν μια μη βέλτιστη λύση λόγω της ελλιπούς αναζήτησης του χώρου των λύσεων. Οι μέθοδοι τεχνητής νοημοσύνης, όπως τα νευρωνικά δίκτυα, η μιμούμενη ανόπτηση (Simulated Annealing), η αναζήτηση ταμπού και οι γενετικοί αλγόριθμοι (Genetic Algorithms - GA) [1] φαίνονται να είναι πολύ ελπιδοφόρες, επιτυγχάνοντας καλά αποτελέσματα και παρουσιάζοντας περαιτέρω περιθώρια βελτίωσης. Επιπλέον, έχουν μελετηθεί και υβριδικά πρότυπα που συνδυάζουν δύο ή περισσότερες από τις προαναφερθείσες μεθόδους [2]. Εκτενέστερη ανάλυση σχετικά με τις προτεινόμενες μεθόδους επίλυσης του προβλήματος UCP γίνεται σε ακόλουθη παράγραφο.

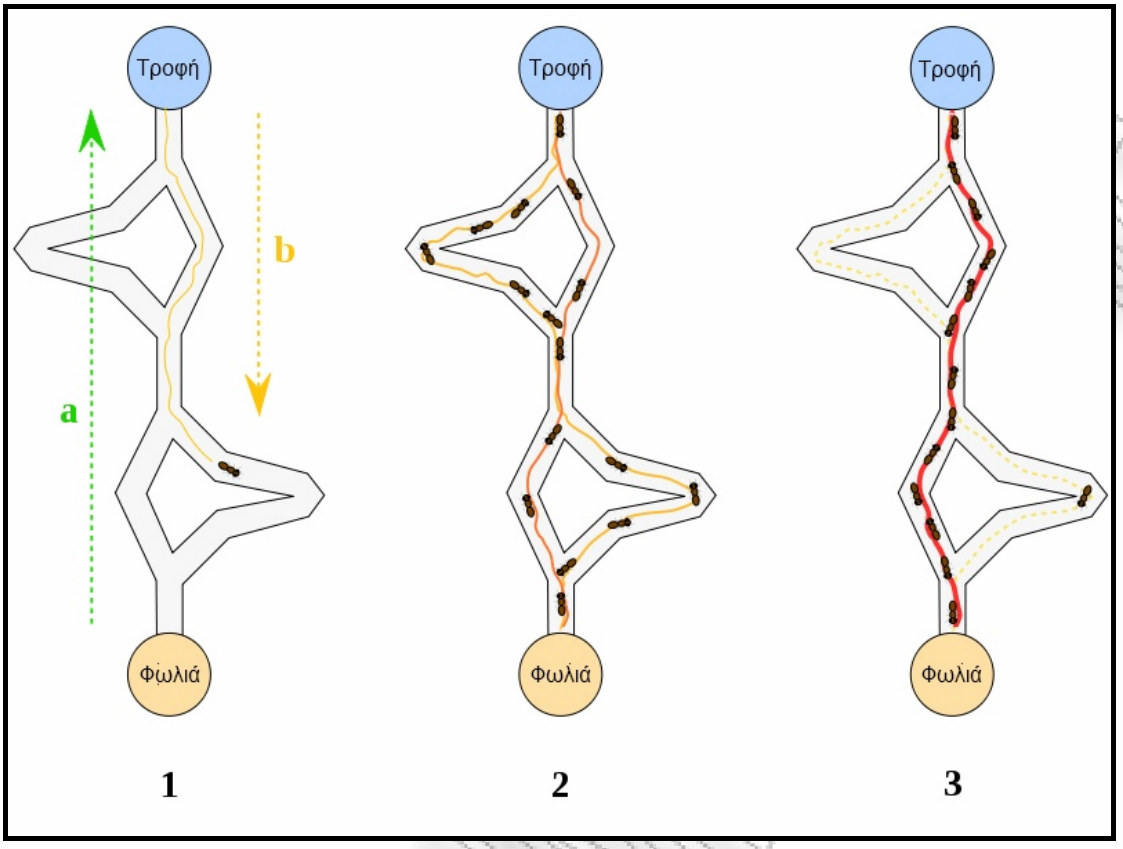
2.2 Αλγόριθμοι Μυρμηγκιών

Ο αλγόριθμος βελτιστοποίησης αποικιών μυρμηγκιών (Ant Colony Optimization - ACO) είναι μια πιθανολογική τεχνική για την επίλυση υπολογιστικών προβλημάτων που μπορούν να περιοριστούν στην εύρεση των καλών «μονοπατιών – λύσεων» με τη χρήση γράφων. Ο αλγόριθμος αυτός είναι μέλος της οικογένειας αλγορίθμων αποικιών μυρμηγκιών και παρέχει κάποιες μετα-ευριστικές βελτιστοποιήσεις. Προτάθηκε το 1992 από τον Marco Dorigo στα πλαίσια της διδακτορικής του διατριβής [3] [4], και ο πρώτος αλγόριθμος στόχευε να ψάξει για μια βέλτιστη πορεία σε ένα γράφο, με βάση τη μοντελοποίηση της συμπεριφοράς των μυρμηγκιών όταν αυτά αναζητούν μια σύντομη πορεία μεταξύ της αποικίας τους και μιας πηγής τροφίμων. Η αρχική ιδέα έχει διαφοροποιηθεί από τότε για την αντιμετώπιση μιας ευρύτερης κατηγορίας αριθμητικών προβλημάτων, και κατά συνέπεια, διάφορα προβλήματα έχουν προκύψει, επικεντρώνοντας την προσοχή των ερευνητών στις διάφορες πτυχές της συμπεριφοράς των μυρμηγκιών.

Στην πραγματικότητα τα μυρμήγκια αρχικά περιπλανιούνται τυχαία, και αφού βρουν τροφή επιστρέφουν στην αποικία τους αφήνοντας ίχνη φερορμόνης καθ' όλη τη διαδρομή. Εάν κάποια μυρμήγκια βρουν ένα μονοπάτι όπου ανιχνεύσουν συσσωρευμένη φερορμόνη, είναι πιθανό να μην συνεχίσουν τυχαία, αλλά να ακολουθήσουν το ίχνος, και επιστρέφοντας από αυτό να ενισχύσουν την συγκέντρωση φερορμόνης εάν βρουν τελικά τα τρόφιμα.

Παρόλα αυτά, με την πάροδο του χρόνου το ίχνος φερορμόνης αρχίζει να εξατμίζεται, μειώνοντας κατά συνέπεια την δύναμη να ελκύει τα μυρμήγκια. Όσο περισσότερο χρόνο κάνει για ένα μυρμήγκι για να διανύσει το μονοπάτι και να γυρίσει, τόσο περισσότερο χρόνο εξατμίζονται οι φερορμόνες. Ένα σύντομο συγκριτικά μονοπάτι, καλύπτεται γρηγορότερα από τα μυρμήγκια, και έτσι η συγκέντρωση φερορμόνης παραμένει υψηλή καθώς εναποτίθεται στο μονοπάτι πιο γρήγορα από ότι μπορεί να εξατμιστεί. Η εξάτμιση φερορμονών έχει επίσης το πλεονέκτημα ότι συνεισφέρει στη σύγκλιση σε μια τοπικά βέλτιστη λύση. Εάν δεν υπήρχε εξάτμιση, τα μονοπάτια που επιλέγονται από τα πρώτα μυρμήγκια θα έτειναν να είναι υπερβολικά ελκυστικές στα ακόλουθα. Σε αυτήν την περίπτωση, η δοκιμή πιθανών μονοπατιών θα περιοριζόταν.

Κατά συνέπεια, όταν ένα μυρμήγκι βρίσκει ένα καλό (δηλαδή σύντομο) μονοπάτι από την αποικία προς μια πηγή τροφίμων, τα υπόλοιπα μυρμήγκια είναι πιθανότερο να ακολουθήσουν αυτό το μονοπάτι, και η θετική ανατροφοδότηση τελικά οδηγεί όλα τα μυρμήγκια στο σύντομο αυτό μονοπάτι. Ο αλγόριθμος αποικιών μυρμηγκιών αποσκοπεί στην μίμηση αυτής της συμπεριφοράς με «προσομοιωμένα μυρμήγκια» που «περπατάνε» γύρω από το γράφο που αντιπροσωπεύει το προς επίλυση πρόβλημα.



Εικόνα 1. Συμπεριφορά Κοινωνίας Μυρμηγκιών

Η αρχική ιδέα πηγάζει από την παρατήρηση της εκμετάλλευσης των πηγών τροφίμων από τα μυρμήγκια, τα οποία λόγω των περιορισμένων μεμονωμένων γνώσεων είναι σε θέση να βρουν την συντομότερη διαδρομή μεταξύ μιας πηγής τροφίμων και της φωλιάς τους μόνο συλλογικά.

Όπως φαίνεται στην Εικόνα 1, το πρώτο μυρμήγκι βρίσκει την πηγή τροφίμων (Τροφή), μέσω οποιουδήποτε τρόπου (α), κατόπιν επιστρέφει στη φωλιά (Φωλιά), αφήνοντας στο μονοπάτι που επέλεξε ένα ίχνος φερορμόνης (β). Τα μυρμήγκια στο συγκεκριμένο παράδειγμα ακολουθούν αδιακρίτως τέσσερις πιθανές διαδρομές, αλλά η ενίσχυση της φερορμόνης σε κάποιο μονοπάτι το καθιστά ελκυστικότερο ως συντομότερο. Τα μυρμήγκια επιλέγουν τη συντομότερη διαδρομή, καθώς τα ίχνη φερορμόνης στις μεγαλύτερες διαδρομές εξατμίζονται. Σε μία σειρά των πειραμάτων σε μια αποικία των μυρμηγκιών με μια επιλογή μεταξύ δύο άνισων σε μήκος μονοπατιών που οδηγούν σε μια πηγή τροφίμων, οι βιολόγοι έχουν παρατηρήσει ότι τα μυρμήγκια έτειναν να χρησιμοποιήσουν το συντομότερο μονοπάτι. [5] [6]

Ένας αλγόριθμος που μοντελοποιεί αυτήν την συμπεριφορά είναι ο ακόλουθος:

Ένα μυρμήγκι διατρέχει τυχαία από την αποικία ψάχνοντας την πηγή τροφής.

Εάν ανακαλύψει την πηγή τροφίμων, επιστρέφει στη φωλιά, αφήνοντας στην πορεία του ένα ίχνος της φερορμόνης.

Αυτό το ίχνος φερορμόνης είναι ελκυστικό, κάνοντας τα πλησιέστερα μυρμήγκια να τείνουν να το ακολουθήσουν.

Επιστρέφοντας στην αποικία, αυτά τα μυρμήγκια θα ενισχύσουν τη ίχνος φερορμόνης, καθώς με την επιστροφή τους θα εναποθέσουν κι αυτά φερορμόνη.

Εάν υπάρχουν δύο δυνατά μονοπάτια από τη Φωλιά στην Τροφή, η πιο σύντομη θα επιλεγεί από περισσότερα μυρμήγκια από τη μακρύτερη, με αποτέλεσμα η συντομότερη διαδρομή να ενισχύεται σε φερορμόνη όλο και περισσότερο, και επομένως να γίνει ελκυστικότερη.

Η συντομότερη διαδρομή ενισχύεται σε φερορμόνη όλο και περισσότερο, και επομένως να γίνει ελκυστικότερη.

Η μακρύτερη διαδρομή θα πάψει να επιλέγεται τελικά, καθώς οι φερορμόνες είναι πτητικές και θα εξατμιστούν.

Τελικά, όλα τα μυρμήγκια έχουν επιλέξει την κοντύτερη διαδρομή.

Τα μυρμήγκια χρησιμοποιούν το περιβάλλον τους ως μέσο επικοινωνίας. Ανταλλάσσουν τις πληροφορίες έμμεσα με την εναπόθεση των φερορμονών, οι οποίες «προδίδουν» τη θέση τους και την επιλογή μονοπατιού τους. Οι πληροφορίες που ανταλλάσσονται έχουν ένα τοπικό χαρακτήρα, καθώς μόνο ένα μυρμήγκι που βρίσκεται στο χώρο όπου έχουν εναποτεθεί φερορμόνες αντιλαμβάνονται την εν λόγω πληροφορία. Αυτό το σύστημα καλείται «Στιγμεργία» και εμφανίζεται σε πολλές ζωικές κοινωνίες (έχει μελετηθεί και στην περίπτωση της κατασκευής των φωλιών των τερμιτών). Ο μηχανισμός αυτός είναι ένα πολύ καλό παράδειγμα ενός αυτοοργανούμενου συστήματος, πράγμα που μπορεί να καταστήσει τη συστεμική συμπεριφορά των μυρμήγκων έναν τρόπο επίλυσης σύνθετων προβλημάτων. Αυτό το σύστημα είναι βασισμένο τόσο στη θετική ανατροφοδότηση (η εναπόθεση της φερορμόνης προσελκύει άλλα μυρμήγκια που θα την ενισχύσουν με τη σειρά τους) όσο και στην αρνητική (οι διαφορετικές διαδρομές βοηθούν την εξάτμιση της φερορμόνης αποτρέποντας έτσι την συγκέντρωση μυρμηγκιών σε μεγάλες διαδρομές). Θεωρητικά, εάν η ποσότητα φερορμόνης παρέμενε η ίδια με την πάροδο του χρόνου σε όλες τις άκρες, καμία διαδρομή δεν θα επιλεγόταν. Εντούτοις, λόγω ανατροφοδότησης, μια μικρή διαφοροποίηση θα υπάρχει σε ένα μονοπάτι και θα επιτρέψει έτσι την επιλογή αυτού του μονοπατιού. Ο αλγόριθμος θα κινηθεί από μια ασταθή κατάσταση στην οποία κανένα μονοπάτι δεν είναι προτιμότερο από το άλλο, προς μια κατάσταση ισορροπίας όπου η προτιμητέα διαδρομή είναι ευκρινώς διαχωρισμένη.

3. Υπάρχοντα Συστήματα –Μεθοδολογία

3.1. Εφαρμογές Αλγορίθμων Μυρμηγκιών

Μερικά μαθηματικά πρότυπα εμπνευσμένα από τη βιολογία τα οποία βρίσκουν εφαρμογή σε τεχνητές κοινωνίες οντοτήτων, όπως η εξελισσόμενη αποίκιση και τεχνητό σύστημα αποικιών μυρμηγκιών (Ant Colony System - ACS), έχουν συνεισφέρει στην παραγωγή κατανεμημένων αλγορίθμων που έχουν εφαρμοστεί επιτυχώς στην παράλληλη βελτιστοποίηση και σχεδιασμό σε συστήματα όπως για παράδειγμα η λειτουργία δεξαμενών. Η βασική ιδέα είναι ότι μερικά σύνθετα προβλήματα μπορούν να είναι αναλυθούν σε περισσότερα απλούστερα, τα οποία θα επεξεργαστούν από κατανεμημένες διεργασίες, μειώνοντας έτσι τις απαιτήσεις από παράλληλα συστήματα υψηλών προδιαγραφών σε απλά υπολογιστικά στοιχεία [7].

Οι αλγόριθμοι μυρμηγκιών εμπνεύστηκαν από την παρατήρηση των πραγματικών αποικιών μυρμηγκιών. Η ιδιαίτερη συμπεριφορά των αποικιών μυρμηγκιών που αναλύθηκε σε προηγούμενο κεφάλαιο, αποτέλεσε πηγή έμπνευσης για τον αλγόριθμο βελτιστοποίησης αποικιών μυρμηγκιών (Ant Colony Optimization - ACO), στον οποίο ένα σύνολο «τεχνητών» μυρμηγκιών συνεργάζεται για την εύρεση λύσεων σε ένα δεδομένο πρόβλημα βελτιστοποίησης με την κατάθεση των ιχνών φερορμονών σε όλο το διάστημα αναζήτησης [8, 9]. Τελικά η συμπεριφορά αυτή έχει τυποποιηθεί στο πλαίσιο της βελτιστοποίησης αποικιών μυρμηγκιών (ACO) [9]. Η τεχνική αυτή βελτιστοποίησης έχει αποδειχθεί ένα αποδοτικό και ευπροσάρμοστο εργαλείο για διάφορα συνδυαστικά προβλήματα. Διάφορες εκδόσεις της τεχνικής ACO έχουν προταθεί, αλλά όλες ακολουθούν τις ίδιες βασικές ιδέες:

Αναζήτηση που εκτελείται από έναν πληθυσμό οντοτήτων («τεχνητών» μυρμηγκιών), δηλαδή απλοί ανεξάρτητοι πράκτορες,

Σταδιακή κατασκευή των λύσεων,

Πιθανολογική επιλογή των τμημάτων λύσης με βάση τις πληροφορίες για την στιγμεργία του συστήματος,

Καμία άμεση επικοινωνία μεταξύ των πρακτόρων. [10]

Οι περιπτώσεις του ACO έχουν εφαρμοστεί εκτενώς σε ποικίλα συνδυαστικά προβλήματα βελτιστοποίησης όπως το πρόβλημα του περιφερόμενου πωλητή (Travelling Salesman Problem - TSP) [11, 12], το τετραγωνικό πρόβλημα ανάθεσης (QAP) [8, 9], το πρόβλημα δρομολόγησης δικτύων επικοινωνιών [13], το πρόβλημα δρομολόγησης οχημάτων [14], οργάνωση εργασιών και καταστημάτων (job – shop scheduling), διαδοχική ταξινόμηση [15, 16], χρωματισμός γραφικών παραστάσεων [15], βελτιστοποίηση σχημάτων, και ούτω καθεξής [17, 18]. Από την εμφάνιση των αλγορίθμων μυρμηγκιών ως εργαλείο βελτιστοποίησης, μερικές προσπάθειες έγιναν επίσης για τη χρησιμοποίησή τους για την αντιμετώπιση των συνεχών προβλημάτων βελτιστοποίησης, ειδικά στην εφαρμοσμένη μηχανική. Παρόλα αυτά η εφαρμογή των μετα-ευριστικών τεχνικών του ACO σε προβλήματα συνεχούς χώρου δεν είναι απλή. Ως εκ τούτου, οι προτεινόμενες τεχνικές συχνά βασίζονταν στη βελτιστοποίηση ACO, αλλά δεν ακολούθησαν ακριβώς την ίδια μεθοδολογία. Ο Jalali και η ομάδα του [18] πρότειναν τη διακριτή βελτιστοποίηση κοινωνιών μυρμηγκιών (discrete ant colony optimization - DACO) για τη λειτουργία δεξαμενών. Μέσω ενός συνόλου συνεργαζόμενων πρακτόρων αποκαλούμενων «μυρμήγκια», μια λύση αρκετά κοντά στη βέλτιστη της λειτουργίας των δεξαμενών μπορεί να επιτευχθεί αποτελεσματικά. Για την εφαρμογή των αλγορίθμων ACO, το πρόβλημα προσεγγίζεται με την εξέταση ενός πεπερασμένου χρονικού ορίζοντα με μια χρονική σειρά εισόδου, ταξινομώντας και ορίζοντας τον όγκο του περιεχομένου των δεξαμενών για συγκεκριμένα χρονικά διαστήματα, και την απόφαση για αυξομειώσεις του περιεχομένου σε τακτά χρονικά διαστήματα σύμφωνα με ένα προκαθορισμένο βελτιστοποιημένο πλάνο. Ο Maier και η ομάδα του [19] ανέπτυξε μια μέθοδο που επιτρέπει στους αλγορίθμους ACO για να χρησιμοποιηθούν για τη βελτιστοποίηση του συστήματος διανομής νερού. Αυτή η μέθοδος εφαρμόστηκε σε δύο προβλήματα βελτιστοποίησης συστημάτων διανομής νερού, έγιναν οι συγκριτικές μετρήσεις επιδόσεων και τα αντίστοιχα αποτελέσματα γενετικών αλγορίθμων. Η βελτιστοποίηση ACO

έχει χρησιμοποιηθεί επίσης για την επίλυση περιπτώσεων του προβλήματος τετραγωνικής ανάθεσης (Quadratic Assignment Problem - QAP), καθώς επίσης και τα προβλήματα δικτύων τηλεπικοινωνιών που περιλαμβάνουν δρομολόγηση. Όλα αυτά τα παραδείγματα περιλαμβάνουν τη διαχείριση ενός αριθμού στοιχείων v, όπου το μέγεθος του διαστήματος αναζήτησης περιορίζεται στο v! μεταπτώσεις. Άλλα παρόμοια παραδείγματα εφαρμογής περιλαμβάνουν το πρόβλημα σειριακής ταξινόμησης και το πρόβλημα δρομολόγησης οχημάτων.

Τα συμπεράσματα έδειξαν ότι οι αλγόριθμοι ACO είναι μια ελκυστική εναλλακτική λύση έναντι των γενετικών αλγορίθμων τόσο από άποψη επίτευξης μιας λύσης αρκετά κοντά στη βέλτιστη, όσο και αναφορικά με την αποδοτικότητα σε χρήση υπολογιστικών πόρων.

Μέχρι σήμερα, σχεδόν όλη η ερευνητική προσπάθεια στον τομέα ACO αφορά τα διακριτά προβλήματα βελτιστοποίησης, και λίγη σημασία έχει δοθεί στη δυνατότητα μεταφοράς της ιδέας των αποικιών μυρμηγκιών στη βελτιστοποίηση προβλημάτων του συνεχούς χώρου.

3.2. Πρόβλημα Ένταξης Μονάδων Παραγωγής (Unit Commitment Problem)

Το UCP αποτελείται από δύο υποπροβλήματα που λύνονται συνήθως χωριστά. Πρώτα τίθεται η δημιουργία ενός προγράμματος ενεργοποίησης και απενεργοποίησης των μονάδων παραγωγής. Στη συνέχεια, βασισμένοι σε αυτό το πρόγραμμα, τις απαιτήσεις και τους περιορισμούς, υπολογίζονται τα ποσά ενέργειας που παράγονται από κάθε γεννήτρια κάθε στιγμή ώστε να επιτυγχάνεται ο αρχικός στόχος βάσει των προδιαγεγραμμένων περιορισμών. Διάφορες προσεγγίσεις υπάρχουν στη βιβλιογραφία για την αντιμετώπιση του UCP, όπως η δυναμικός προγραμματισμός (Dynamic Programming - DP) [20, 21], η χαλάρωση Lagrange, ευριστικές μέθοδοι με υβρίδια [22-25], η μέθοδος Branch and bound (BB), αποσύνθεση του Benders (Benders' decomposition) [27], η μιμούμενη ανόπτηση (simulated annealing) [28], την ταμπού-αναζήτηση [29], διακριτός προγραμματισμός, προγραμματισμός ροής δικτύων, τους εξελικτικούς αλγορίθμους [30 - 36] και πολλά υβρίδια. Μια λεπτομερής έρευνα μπορεί να βρεθεί στα [37 - 39].

Οι εξελικτικοί αλγόριθμοι (Evolutionary Algorithms - EAs) [40] είναι τεχνικές βελτιστοποίησης με χρήση πληθυσμών οντοτήτων και βασίζονται στους κλασικούς νόμους του Mendel, της κληρονομιάς και της θεωρίας της εξέλιξης Δαρβίνου. Οι εξελικτικοί αλγόριθμοι αποτελούν έναν ενιαίο όρο που καλύπτει διάφορες προσεγγίσεις, δηλαδή γενετικούς αλγορίθμους (genetic algorithms - GA) [41], εξελικτικές στρατηγικές (evolutionary strategies - ES) [42], γενετικό προγραμματισμό (genetic programming - GP) [43] και η διαφορική εξέλιξη (differential evolution - DE) [44] που είναι βασισμένες στις ίδιες αρχές αλλά διαφέρουν στην εφαρμογή των αρχών αυτών. Οι εξελικτικοί αλγόριθμοι έχουν χρησιμοποιηθεί επιτυχώς σε πολλές περιοχές προβλημάτων.

Οι γενετικοί αλγόριθμοι, η εξελικτική στρατηγική και η διαφορική εξέλιξη είναι κυρίως για την αναζήτηση και τη βελτιστοποίηση ενώ ο γενετικός προγραμματισμός χρησιμοποιείται περισσότερο για την παραγωγή αυτόματου προγράμματος, για πρόβλεψη διαδικασίες εκμάθησης μηχανών. Η λύση στο UCP δίνεται ως ένα σύνολο δυαδικών μεταβλητών που καταδεικνύουν ποιες μονάδες γεννητριών είναι σε λειτουργία και ποιες δεν λειτουργούν για κάθε δεδομένη χρονική στιγμή. Αυτή η λύση επιτυγχάνεται μέσω της ελαχιστοποίησης μιας συνάρτησης κόστους, διατηρώντας παράλληλα το σύστημα σε τέτοια κατάσταση ώστε να πληρούνται οι όποιοι περιορισμοί έχουν τεθεί. Επομένως αυτό το πρόβλημα μπορεί να αντιμετωπιστεί ως αναζήτηση των εφικτών λύσεων που βελτιστοποιούν μια συνάρτηση.

Στην μελέτη που παρουσιάζεται στην εργασία [45] αρχικές δοκιμές που χρησιμοποιούν μόνο την προσέγγιση DE εκτελέσθηκαν από τους ερευνητές και τα αναμενόμενα αποτελέσματα επαληθεύθηκαν. Αυτή η μελέτη προάγει τη προηγούμενη δουλειά με να ερευνήσει τη συμπεριφορά τριών σημαντικών εξελικτικών αλγορίθμων ως μεθόδους επίλυσης του UCP. Αναφορικά με αυτές τις τρεις εξελικτικές παραλλαγές γίνονται συγκριτικές μετρήσεις των επιδόσεών τους για το δεδομένα ενός UCP προβλήματος καθώς και σε πραγματικά στοιχεία ενός μέρους πραγματικού δικτύου ηλεκτρικής ενέργειας. Τα

αποτελέσματα συγκρίνονται επίσης με τα επιτυχή αποτελέσματα που αναφέρονται στη βιβλιογραφία για τα ίδια σύνολα δεδομένων συγκριτικής μέτρησης επιδόσεων.

Αρκετές ακόμη μέθοδοι βελτιστοποίησης έχουν υιοθετηθεί για την επίλυση του προβλήματος UCP. Τα πρότυπα που χρησιμοποιούνται για το πρόβλημα UCP διαφέρουν σε λεπτομέρειες, αλλά ο κύριος στόχος όλων είναι η εύρεση ενός βέλτιστου ή σχεδόν βέλτιστου προγράμματος για την επίλυση του προβλήματος, το οποίο περιλαμβάνει και διακριτές και συνεχείς μεταβλητές απόφασης. Μερικές από τις παραδοσιακές μεθόδους βελτιστοποίησης που χρησιμοποιούνται είναι η εξαντλητική απαρίθμηση (exhaustive enumeration), ο δυναμικός προγραμματισμός (Dynamic Programming - DP), Branch and bound (BB), αποσύνθεση του Benders (Benders' decomposition), διακριτός προγραμματισμός, προγραμματισμός ροής δικτύων, Lagrange Relaxation (LR), αυξημένη Lagrangian Relaxation (ALR), ανάλυση κινδύνου, και ανάλυση απόφασης. Οι μετα-ευριστικές μέθοδοι όπως οι γενετικοί αλγόριθμοι, η μιμούμενη ανόπτηση (simulated annealing) ή τα έμπειρα συστήματα (expert systems) έχουν υιοθετηθεί επίσης. Ο στοχαστικός προγραμματισμός έχει εξεταστεί επίσης και έχουν διεξαχθεί δοκιμές για την απόδειξη της αξίας αυτής της μεθόδου. [53]

Η LR είναι μια ευρέως διαδεδομένη μέθοδος, η οποία είναι ιδιαίτερα χρήσιμη όταν υπάρχει ένα σύνολο περιορισμών, καθώς καθιστά τη λύση του γενικότερου προβλήματος απλούστερη. Από όλες τις διαφορετικές μεθόδους που υιοθετούνται για να λύσουν το πρόβλημα Ένταξης Μονάδων Παραγωγής, η LR είναι η ευρύτερα χρησιμοποιούμενη μέθοδος λόγω της επιτυχίας της στην επίλυση των μεγάλης κλίμακας προβλημάτων. Επομένως, η εστίαση στην έρευνα UC ήταν να βρεθούν οι μέθοδοι που μπορούν να καταστήσουν αυτήν την μέθοδο αποδοτικότερη.

Ιστορικά, δύο μέθοδοι έχουν επικρατήσει για την αντιμετώπιση των προβλημάτων που προκύπτουν από την χρήση της τεχνικής LR: δημιουργία στηλών και κλιμακωτή βελτιστοποίηση. Η παραγωγή στηλών ρυθμίζει τους πολλαπλασιαστές Lagrange με το να λύσει ένα υποπρόβλημα πρώτα και το συνολικό πρόβλημα σειριακά. Η κλιμακωτή βελτιστοποίηση τροποποιεί τους πολλαπλασιαστές Lagrange με τον καθορισμό μιας κλιμακωτής κατεύθυνσης, η οποία μεγιστοποιεί την αντικειμενική συνάρτηση του δυαδικού προβλήματος. Στην επίλυση του προβλήματος Ένταξης Μονάδων Παραγωγής (UCP), η τελευταία μέθοδος προτιμάται επειδή οι υπολογισμοί μιας δεδομένης επανάληψης είναι απλοί και γρήγοροι. [55] Η πρώτη εφαρμογή της LR στο πρόβλημα UCP αναλύεται από τους Muckstadt και Koenig. [56] Στην ανάλυση αυτή επιχειρείται κατάτμηση του συνολικού προβλήματος σε υποπροβλήματα που αφορούν απλούστερα υποσυστήματα μονάδων ξεχωριστά και χρησιμοποιούν τη μέθοδο «branch and bound» για την επίλυση των απλούστερων αυτών δικτύων. Οι Merlin και Sandrin [57] χρησιμοποιούν την LR και χρησιμοποιούν μια τροποποιημένη έκδοση της κλιμακωτής βελτιστοποίησης για να ενημερώσουν τους πολλαπλασιαστές Lagrange και να λύσουν ένα πρόβλημα UCP για 172 μονάδες σε λιγότερο από 2 λεπτά. Στη μελέτη αυτή προτείνεται μια διαφορετική προσέγγιση για τον προσδιορισμό των επιπέδων SR αλλά δεν την εφαρμόζουν αυτό ώστε να οδηγηθούν σε οποιαδήποτε αποτελέσματα.

Ο Bard χρησιμοποιεί μια παρόμοια προσέγγιση αλλά περιλαμβάνει τους περιορισμούς κλίσης (ramping constraints) στο πρόβλημα και αντί της μεθόδου branch and bound χρησιμοποιούν τον συνδεδεμένο δυναμικό προγραμματισμό [58]. Οι Zhuang και Galiana [59] υιοθετούν μια παρόμοια προσέγγιση και λύνουν το πρόβλημα UCP χρησιμοποιώντας έναν νέο ευρετικό αλγόριθμο για να βρουν μια λύση.

Ένα πρόβλημα που αντιμετωπίζουν οι μέθοδοι που βασίζονται στην LR είναι ότι η λύση στην οποία καταλήγουν είναι σπάνια εφικτή, κάτι το οποίο επιβάλλει την ύπαρξη μιας επιπρόσθετης διαδικασίας για την εξεύρεση υλοποιήσιμης λύσης. Επιπλέον, όταν πρέπει το UCP πρόβλημα έχει χρονικό ορίζοντα που μπορεί να φτάνει και τον έναν μήνα, αυτή η μέθοδος απαιτεί μεγάλο χρόνο ώστε να επιτευχθεί σύγκλιση στη βέλτιστη λύση, ενώ σε μερικές περιπτώσεις ο αλγόριθμος συνεχίζει γύρω από το βέλτιστο σημείο χωρίς να μπορεί να επιτύχει την προαναφερθείσα σύγκλιση. Ο Aoki με τους συνεργάτες του [60] παρουσιάστε έναν αλγόριθμο που χρησιμοποιεί μια μεταβλητή μετρική (variable metric) μέθοδο που αντιμετωπίζει αυτό το πρόβλημα. Η μεταβλητή μετρική μέθοδος είναι γενικά μια ειδική συζευγμένη μέθοδος κλίσης και μπορεί να ερμηνευθεί ως επέκταση της μεθόδου Newton, η οποία προσδιορίζει το μέγιστο μιας συνάρτησης μελετώντας τους μηδενισμούς της

παραγώγου. Σε κάθε βήμα της μεθόδου Newton, η αντικειμενική συνάρτηση προσεγγίζεται από ένα πολυώνυμο Taylor δεύτερης τάξης. Εντούτοις, η μεταβλητή μετρική μέθοδος δεν απαιτεί τα μερικά παράγωγα δεύτερης τάξης που απαιτούνται για το πολυώνυμο του Taylor. Αντ' αυτού, προσεγγίζει αυτές τις τιμές χρησιμοποιώντας τις πληροφορίες από τις προηγούμενες επαναλήψεις. Εντούτοις, όταν η αντικειμενική συνάρτηση δεν είναι έντονα κυρτή, αυτή η μέθοδος δεν είναι σταθερή και δεν μπορεί να φθάσει σε μια καλή λύση.

Μια πρόσφατη σύγκριση μεταξύ των τεχνικών ALR και της LR για το πρόβλημα UC γίνεται από τους Beltran και Heredia. [62] οι οποίοι δείχνουν ότι η LR παράγει τις ανέφικτες πρώτες λύσεις και παρότι δεν διασφαλίζεται η διαφοροποιησιμότητα αλλά η αντικειμενική συνάρτηση είναι προσδιορίσιμη. Αντίθετα, η ALR παράγει εφικτές λύσεις και ενώ η διαφοροποιησιμότητα εξασφαλίζεται η αντικειμενική συνάρτηση δεν μπορεί να προσδιοριστεί. Εντούτοις, η χρήση APP με ALR εγγυάται τη σύγκλιση, η οποία την καθιστά προτιμητέα.

Η εφαρμοσιμότητα των μετα-ευριστικών μεθόδων έχει αποδειχτεί και στην επίλυση του προβλήματος ένταξης μονάδων παραγωγής. Οι Dasgupta και McGregor. [64] μελετούν δέκα θερμικές γεννήτριες και επιτυγχάνουν «αρκετά καλά» αποτελέσματα. Άλλες μέθοδοι όπως οι γενετικοί αλγόριθμοι έχουν χρησιμοποιηθεί επίσης ως βοηθητική μέθοδος στη LR. Οι Orero και Irving [65] υιοθετούν αυτήν την μέθοδο και λύνουν ένα σύστημα δέκα μονάδων αποδεικνύοντας την εφαρμοσιμότητα αυτής της μεθόδου. Η βασική δυσκολία αυτής της μεθόδου είναι η έλλειψη μιας θεωρητικής βάσης για την επιλογή των παραμέτρου ελέγχου του γενετικού αλγορίθμου σε ένα ευρύ φάσμα των περιοχών προβλήματος. [65], [66] Ομοίως η εφαρμοσιμότητα των εξελικτικών προγραμμάτων ως μέθοδο εναλλακτικής βελτίωσης αποδεικνύεται και από την ερευνητική ομάδα του Duo. [67] Υπάρχουν κι άλλες δυνατές βελτιώσεις, όπως η προσθήκη πρόσθετων περιορισμών στην τάση και τη μετάδοση στο πρόβλημα για εύρεση ρεαλιστικών – εφαρμόσιμων αποτελεσμάτων. Τέτοια παραδείγματα εφάρμοσαν οι Shaw, Xia και ο Wang με τις ομάδες τους αντίστοιχα [68, 69, 70].

Αυτή τη στιγμή, όλοι οι αλγόριθμοι ένταξης μονάδων βασίζονται στις μαθηματικές τεχνικές προγραμματισμού. Οι χαρακτηριστικές προσεγγίσεις περιλαμβάνουν το δυναμικό προγραμματισμό, το μικτό προγραμματισμό ακέραιων αριθμών, την αποσύνθεση Benders, και τη Lagrangian relaxation [71, 72, 73].

Ακόμα κι αν οι τελευταίες δύο προσεγγίσεις έχουν παρουσιάσει σημαντικές δυνατότητές στο χειρισμό των προβλημάτων μεγάλης κλίμακας χωρίς εξάντληση των διαθέσιμων υπολογιστών πόρων, οι απαραίτητοι χρόνοι υπολογισμού θεωρούνται ακόμα υπερβολικοί για εφαρμογές πραγματικού χρόνου (real time applications). Ο βασικός λόγος για αυτήν την μακροχρόνια διαδικασία υπολογισμού αποδίδεται στην αναζήτηση «στα τυφλά» μιας λύσης μέσα σε ένα μεγάλο σύνολο πιθανών λύσεων.

Παραδείγματος χάριν, η επιλογή του παραθύρου στον περιορισμένο δυναμικό προγραμματισμό (truncated dynamic programming) δεν είναι αποδοτική για τον περιορισμό του διαστήματος λύσης. Επαγωγικά, και ο μαθηματικός προγραμματισμός δεν θεωρείται αποδοτικός για το πρόβλημα ένταξης μονάδων παραγωγής [74, 75].

Μελετώντας τα μειονεκτήματα του μαθηματικού προγραμματισμού, θα ήταν ενδιαφέρον να εφαρμοστεί ένα έμπειρο σύστημα συμπληρωματικά στους υπάρχοντες αλγορίθμους μαθηματικού προγραμματισμού για την αντιμετώπιση του σχεδιασμού των ηλεκτρικών συστημάτων. Στην εργασία [χ] προτείνεται ένα μη συνδεδεμένο έμπειρο σύστημα βασισμένο σε ένα σύνολο ευριστικών κανόνων για την υποστήριξη της διαδικασίας ένταξης μονάδων. Άλλες δημοσιεύσεις σχετικά με έμπειρα συστήματα ερευνούν περαιτέρω μεθόδους σχετικές με τον σχεδιασμού την ένταξη των μονάδων παραγωγής [77]. Εντούτοις, οι προτεινόμενες μέθοδοι δεν ενδιαφέρθηκαν για τη βελτιστοποίηση του προγράμματος ένταξης. Η εφαρμογή έμπειρων συστημάτων ήταν πρώτιστα για την προετοιμασία των δεδομένων εισόδου που θα χρησιμοποιούνταν στη διαδικασία του μαθηματικού προγραμματισμού. Εν προκειμένω, στην εργασία [77] χρησιμοποιείται το έμπειρο σύστημα για την παρουσίαση της συνολικής καθημερινής κατανάλωσης νερού για κάθε μονάδα πριν από την εφαρμογή των αλγορίθμων μαθηματικού προγραμματισμού. Συμπερασματικά, η εφαρμογή του έμπειρου συστήματος στο σχεδιασμό παραγωγής ενέργειας από κάποια εγκατάσταση μονάδων παραγωγής δεν έχει άμεση σχέση με τη διαδικασία σχεδιασμού.

3.3. Μεθοδολογία

Για την εκπόνηση της παρούσας εργασίας, ακολουθήθηκε η ακόλουθη μεθοδολογία. Μετά την αρχική έρευνα για την κατανόηση των βασικών εννοιών που σχετίζονται με τον ευρύτερο ερευνητικό χώρο στον οποίο δραστηριοποιείται η παρούσα εργασία, έγινε μια διεξοδική μελέτη των υπαρχουσών προσεγγίσεων, οποίες αναλύθηκαν στο παρόν κεφάλαιο.

Στη συνέχεια, επικεντρωθήκαμε στην μελέτη του συγκεκριμένου προβλήματος που τίθεται προς λύση, της ένταξης δηλαδή μονάδων παραγωγής. Πιο συγκεκριμένα δόθηκε ο ορισμός του προβλήματος, καταγράφηκαν οι πιθανοί περιορισμοί και απαιτήσεις, για να γίνει δυνατή η τελική μοντελοποίηση του προβλήματος. Έχοντας, πλέον, μια σαφή εικόνα του προβλήματος, προχωρήσαμε σε αναλυτικότερη μελέτη της οικογένειας αλγορίθμων που επρόκειτο να χρησιμοποιηθούν στην επίλυση του δεδομένου προβλήματος. Έτσι, μελετήθηκαν οι αλγόριθμοι που βασίζονται στη λειτουργία των αποικιών μυρμηγκιών, καθώς και οι διάφορες εφαρμογές που μπορούν να βρουν στα διάφορα σύνθετα προβλήματα, είτε διακριτών, είτε συνεχών μεταβλητών.

Στο τελευταίο μέρος του παρόντος πονήματος σχεδιάστηκε και αναπτύχθηκε μια πρότυπη εφαρμογή, η οποία αποσκοπεί στην επαλήθευση όσων θεωρητικά μελετήθηκαν μέχρι αυτό το σημείο. Η επαλήθευση αυτή έρχεται με τον ορισμό σεναρίων και την εκτέλεση των σχετικών πειραμάτων.

4. Ένταξη Μονάδων Παραγωγής σε Χρονικό Ορίζοντα 24ώρου

Στο κεφάλαιο αυτό γίνεται μια προσέγγιση του προβλήματος ένταξης μονάδων παραγωγής ξεκινώντας από μελέτη του μαθηματικού μοντέλου υπολογισμού κόστους ανά μονάδα και εν συνόλω. Στη συνέχεια καταγράφονται οι πιθανές απαιτήσεις και περιορισμοί και περιγράφεται η αρχιτεκτονική του μοντέλου του προβλήματος προς επίλυση. Στη συνέχεια γίνεται αναφορά στους αλγορίθμους αποικιών μυρμηγκιών και την εφαρμογή τους σε προβλήματα συνεχών μεταβλητών –όπως και το πρόβλημα ένταξης μονάδων παραγωγής – με ιδιαίτερη έμφαση στον αλγόριθμο Con-ANT που είναι και ο αλγόριθμος που επιλέγεται για την αντιμετώπιση του εν λόγω προβλήματος.

4.1 Ορισμός προβλήματος

4.1.1 Θεωρία

Το πρώτο στάδιο στην προσπάθεια επίλυσης του οποιουδήποτε προβλήματος, είναι η θεωρητική προσέγγισή του, με σκοπό τον σαφή μαθηματικό προσδιορισμό. Στην περίπτωσή μας καλούμαστε να ορίσουμε την αντικειμενική συνάρτηση που περιγράφει το πρόβλημα ένταξης μονάδων παραγωγής. Για τον σκοπό αυτό, θα χωρίσουμε τη μελέτη σε δύο στάδια: στο πρώτο θα μελετήσουμε μεμονωμένα τη γενική περίπτωση μιας μονάδας παραγωγής / γεννήτριας, και θα εξάγουμε την αντικειμενική της συνάρτηση όσον αφορά το κόστος.

Σε δεύτερο στάδιο θα ασχοληθούμε με μια εγκατάσταση πολλών τέτοιων μονάδων παραγωγής και θα προσδιορίσουμε τόσο το συνολικό κόστος συναρτήσει των απαιτήσεων και των περιορισμών, αλλά και θα γίνει ειδική μελέτη στο σχεδιασμό λειτουργίας μιας τέτοιας εγκατάστασης σε χρονικό ορίζοντα 24ώρου.

4.1.1.1. Μελέτη Κόστους Μονάδας Παραγωγής

Το κόστος λειτουργίας μιας μονάδας παραγωγής απαρτίζεται από τρεις συνιστώσες. Οι συνιστώσες αυτές είναι οι ακόλουθες:

- **Κόστος λειτουργίας / καυσίμου**: Είναι το κόστος που επιβάλλεται από το χρησιμοποιούμενο καύσιμο για τη λειτουργία της μονάδας και είναι συνάρτηση της απαιτούμενης παραγόμενης ισχύος από τη μονάδα τη δεδομένη χρονική περίοδο. Στα πλαίσια της παρούσας εργασίας και για τις ακόλουθες εξισώσεις η προαναφερθείσα συνάρτηση κόστους παριστάνεται ως FC(*p(t)*) (Functional Cost σαν συνάρτηση της ισχύος σε μια δεδομένη χρονική στιγμή t, p(t)). Σημειώνεται ότι στο κόστος λειτουργίας / καυσίμου περιλαμβάνεται και το κόστος συντήρησης της μονάδας.
- **Κόστος εκκίνησης μονάδας**: Είναι το κόστος που απαιτείται για την εκκίνηση μιας μονάδας. Το κόστος αυτό είναι συνάρτηση του χρόνου στον οποίο εκκινείται η δεδομένη μονάδα, και φυσικά των χαρακτηριστικών αυτής καθαυτής της μονάδας. Στη συνέχεια γίνεται αναφορά στην εν λόγω παράμετρο ως OnC(t) (On Cost σαν συνάρτηση του χρόνου t).
- **Κόστος τερματισμού μονάδας**: Είναι το κόστος που απαιτείται για τον τερματισμό μιας μονάδας. Το κόστος αυτό είναι συνάρτηση του χρόνου στον οποίο τερματίζεται η δεδομένη μονάδα, και φυσικά των χαρακτηριστικών αυτής καθαυτής της μονάδας. Στη συνέχεια γίνεται αναφορά στην εν λόγω παράμετρο ως OffC(t) (Off Cost σαν συνάρτηση του χρόνου t).

Οι παραπάνω συνιστώσες αθροιζόμενες δίνουν το συνολικό κόστος TC (*Total Cost*) για την συνεισφορά μιας μονάδας σαν μέρος μιας εγκατάστασης σαν συνάρτηση του χρόνου αναφοράς t δίνονται από τον παρακάτω τύπο:

$$TC(t) = FC(p(t)) + OnC(t) + OffC(t) \quad (5.1)$$

4.1.1.2. Συνολική Μελέτη Μονάδων Παραγωγής

Κάνοντας αναφορά σε ένα συνολικό σύστημα περισσότερων της μίας μονάδων παραγωγής που αποτελεί και το περιβάλλον μελέτης της παρούσας εργασίας, καλούμαστε να υπολογίσουμε το επαγόμενο κόστος από τη λειτουργία και τις εκκινήσεις και τερματισμούς όλων των συμμετεχουσών μονάδων.

Λαμβάνοντας λοιπόν υπόψη την ύπαρξη πολλαπλών μονάδων προς ένταξη και λειτουργία αλλά και χρονικό ορίζοντα 24 ωρών για τη συγκεκριμένη μελέτη ο συνολικός τύπος εξαγωγής του κόστους ένταξης μονάδων παραγωγής σε χρονικό ορίζοντα 24 ωρών UCC(t,i) (Unit Commitment Cost σαν συνάρτηση της χρονικής στιγμής t και του αύξοντα αριθμού της σειριακά προστιθέμενης μονάδας i) είναι ο παρακάτω:

$$UCC = \sum_{t=1}^{T}\sum_{i=1}^{N} TC(t,i) = \sum_{t=1}^{T}\sum_{i=1}^{N}\left\{FC\left(p(t,i)\right) + OnC(t,i) + OffC(t,i)\right\} \quad (5.2)$$

Σημειώνεται πως στην εξίσωση 5.2 η αναφορά στην συνάρτηση TC όπως αυτή ορίζεται στην εξίσωση 5.1 γίνεται προσθήκη της μεταβλητής του αύξοντα αριθμού της υπό μελέτη μονάδας (i), καθώς εδώ γίνεται η άθροιση του κόστους όλων των συμμετεχουσών μονάδων. Τα αθροίσματα αφορούν το σύνολο των μονάδων (μεταβλητή i από 1 έως N, όπου N το πλήθος των μονάδων) και το σύνολο του χρόνου μελέτης (μεταβλητή t από 1 έως T, όπου T η χρονική περίοδος αναφοράς, ήτοι 24 ώρες στη συγκεκριμένη μελέτη).

4.1.1.3. Μελέτη επιβαλλόμενων Περιορισμών / Απαιτήσεων

Το πρόβλημα ένταξης μονάδων παραγωγής μπορεί να έχει πολλούς περιορισμούς ανάλογα με τους διαφορετικούς κανόνες που επιβάλλονται κατά το σχεδιασμό από το ίδιο το ηλεκτρικό σύστημα, την απαιτούμενη και τη διαθέσιμη ισχύ καθώς και την ενδεχόμενη δεξαμενή ενέργειας (power pool) ή τις απαιτήσεις αξιοπιστίας.

Το **απόθεμα λειτουργίας** (spinning reserve) είναι το συνολικό ποσό ενέργειας που διαθέσιμο από όλες τις μονάδες που λειτουργούν στο σύστημα, μείον την υπάρχουσα ισχύ και τις απώλειες. Όταν μια γεννήτρια δυσλειτουργεί, πρέπει να υπάρξει αρκετό απόθεμα από τις υπόλοιπες για να αντισταθμιστεί η επικείμενη απώλεια σε ένα καθορισμένο χρονικό διάστημα. Το απόθεμα υπολογίζεται ως ποσοστό της προβλεπόμενης μέγιστης ζήτησης, και πρέπει να είναι σε θέση να αναπληρώσει πιθανή απώλεια της πιο υπερφορτωμένης μονάδας σε μια δεδομένη χρονική περίοδο. Επίσης μερικές φορές υπολογίζεται ως συνάρτηση της πιθανότητας της αδυναμίας κάλυψης του απαιτούμενου φορτίου από τις υπάρχουσες μονάδες. Το επιθυμητό απόθεμα πρέπει να προσδιοριστεί συνυπολογίζοντας εάν το σύστημα είναι γρήγορη ή αργή απόκριση σε νέα δεδομένα. Το πρόβλημα ένταξης μονάδων παραγωγής περιλαμβάνει τα «σχεδιασμένα αποθέματα» ή τα offline αποθέματα που μπορούν να ενεργοποιηθούν, να συγχρονιστούν, και να τεθούν σε λειτουργία γρήγορα.

Τα αποθέματα πρέπει επίσης να κατανέμονται στο ηλεκτρικό σύστημα με τέτοιο τρόπο ώστε να μπορούν να χρησιμοποιηθούν ακόμη και στην περίπτωση που το σύστημα παρουσιάζει προβλήματα στις συνδέσεις μεταξύ των μονάδων λόγω κάποιου προβλήματος.

Οι **θερμικές μονάδες** απαιτούν μια ομάδα προσωπικού για την ενεργοποίησή τους, ειδικά όταν τίθενται εντός ή εκτός λειτουργίας. Δεδομένου ότι οι αλλαγές θερμοκρασίας είναι εξορισμού κλιμακωτές, απαιτούνται μερικές ώρες μέχρι την πλήρη ενεργοποίηση της μονάδας. Οι περιορισμοί αυτοί στη λειτουργία θερμικών εγκαταστάσεων, επιφέρουν μια σειρά άλλων πιθανών περιορισμών, όπως:

- **Ελάχιστο χρόνο ενεργοποίησης**:

Όταν ενεργοποιηθεί η μονάδα, δεν μπορεί να απενεργοποιηθεί αμέσως. Ο χρόνος που απαιτείται να μεσολαβήσει αναφέρεται ως «*ελάχιστος χρόνος ενεργοποίησης*»

- **Ελάχιστο χρόνο απενεργοποίησης**:

Μόλις αποδεσμευθεί η μονάδα, υπάρχει ένας ελάχιστος χρόνος προτού να μπορέσει να επανα-ενεργοποιηθεί. Ο χρόνος που απαιτείται να μεσολαβήσει αναφέρεται ως «*ελάχιστος χρόνος απενεργοποίησης*»

- **Περιορισμοί προσωπικού**:

Σε εγκαταστάσεις με περισσότερες από μια μονάδες και δη σε μεγάλες σχετικά αποστάσεις μεταξύ τους, μπορεί να μην υπάρχει αρκετό προσωπικό για ταυτόχρονη παρουσία και στις δύο ή περισσότερες μονάδες με σκοπό ταυτόχρονη ενεργοποίηση/απενεργοποίησή τους.

- **Κόστος ενεργοποίησης μονάδων**:

Ένα συγκεκριμένο ποσό της ενέργειας χρησιμοποιείται κατά την ενεργοποίηση της μονάδας. Αυτό το ποσό ενέργειας δεν παράγεται και συμπεριλαμβάνεται στους υπολογισμούς του προβλήματος ένταξης μονάδων παραγωγής ως «*κόστος ενεργοποίησης μονάδων*».

- **Περιορισμοί σε υδρο-ηλεκτρικές μονάδες**

Η ένταξη μονάδων εάν διαχωριστεί από το σχεδιασμό των υδρο-ηλεκτρικών μονάδων σαν ένα ξεχωριστό πρόβλημα συντονισμού ή υδροθερμικού σχεδιασμού μπορεί να μην οδηγήσει σε μια βέλτιστη λύση.

- **Υποχρεωτική λειτουργία**

Κάποιες μονάδες είναι απαραίτητο να είναι ενεργοποιημένες κατά τη διάρκεια ορισμένων χρονικών περιόδων για την υποστήριξη τάσης στο δίκτυο μετάδοσης ή για τον παροχή για χρήσεις εκτός της εγκατάστασης των μονάδων.

- **Περιορισμοί καυσίμων**

Σε κάποιες περιπτώσεις μπορεί να υπάρχει συγκεκριμένος περιορισμός στα καύσιμα, με αποτέλεσμα να είναι απαραίτητο κάθε μονάδα να καταναλώνει συγκεκριμένη ποσότητα καυσίμου ανά δεδομένα χρονικά διαστήματα. Αυτός ο περιορισμός αποτελεί μια σημαντική και πολύπλοκη πρόκληση στη διαδικασία ένταξης μονάδων παραγωγής.

4.1.2 Αντικείμενο Προβλήματος

Σκοπός του προβλήματος που καλούμαστε να λύσουμε είναι η ελαχιστοποίηση της συνάρτησης κόστους που ορίζεται στην παράγραφο *4.1.1.2*, πληρώντας παράλληλα όλες τις απαιτήσεις και περιορισμούς, όπως αυτοί ορίζονται στην παράγραφο *4.1.1.3*.

$$\text{Min}(UCC) \quad (5.3)$$

4.2 Αποικίες Μυρμηγκιών σε Προβλήματα Συνεχών Μεταβλητών

4.2.1. Υπάρχουσες Προσεγγίσεις

Η μοντελοποίηση της λειτουργίας αποικιών μυρμηγκιών αποδείχθηκε επιτυχής προσέγγιση για να λύσει κάποια δύσκολα προβλήματα βελτιστοποίησης. Αρχικά, υπάρχει η ιδέα ότι ένα κατανεμημένο σύστημα που παρουσιάζει ένα είδος νοημοσύνης σμηνών (*swarm*

intelligence) [78] μπορεί να είναι αποδοτικό στη λύση προβλημάτων, και ιδιαίτερα προβλήματα βελτιστοποίησης. Σήμερα υπάρχουν αρκετοί αλγόριθμοι που έχουν κοινά στοιχεία με αυτή την ιδέα, ειδικά όσοι είναι βασισμένοι σε κάποιου είδους «εικονικές» κοινωνίες, όπως τη βελτιστοποίηση «σμηνών μορίων» (particle swarm) [84], οι γενετικοί αλγόριθμοι [88], η διαφορική εξέλιξη [89], ο εξελικτικός προγραμματισμός [85] και ούτω καθεξής. Παρόλα αυτά, ένα από τα καλύτερα παραδείγματα τέτοιων συστημάτων αποτελεί η βελτιστοποίηση με τη χρήση αποικιών μυρμηγκιών. Ο πρώτος αλγόριθμος που εμπνέεται από τις αποικίες μυρμηγκιών είναι το «σύστημα μυρμηγκιών» [82], το οποίο έχει εφαρμοστεί σε πολλά συνδυαστικά προβλήματα.

Μέχρι τώρα, λίγες μόνο προσεγγίσεις για τη βελτιστοποίηση προβλημάτων συνεχών χώρων με τη χρήση αλγορίθμων μυρμηγκιών έχουν προταθεί στη βιβλιογραφία. Η πρώτη μέθοδος που αναπτύχθηκε αποκαλείται συνεχής βελτιστοποίηση αποικιών μυρμηγκιών (*Continuous Ant Colony Optimization - CACO*) και προτάθηκε από τους Bilchev και Parmee [91], ενώ χρησιμοποιήθηκε και σε μερικές άλλες ερευνητικές προσπάθειες [90, 92]. Η μέθοδος CACO χρησιμοποιεί τις αποικίες μυρμηγκιών για να εκτελέσει τις τοπικές αναζητήσεις σε έναν υποσύνολο του χώρου των πιθανών λύσεων, ενώ λαμβάνει χώρα και μια σφαιρική αναζήτηση η οποία διεξάγεται από έναν γενετικό αλγόριθμο. Πράγματι, τα μυρμήγκια που αναλαμβάνουν την σφαιρική αναζήτηση εκτελούν μια απλή αξιολόγηση μερικών περιοχών που επιλέγεται στο διάστημα αναζήτησης, προκειμένου να ενημερωθεί η καταλληλότητα των περιοχών για εύρεση πιθανής λύσης. Ο ορισμός νέων περιοχών γίνεται με μια παρόμοια διαδικασία αυτής των γενετικών αλγορίθμων (GA), χρησιμοποιούνται δηλαδή κοινοί πράκτορες που προσομοιώνονται από τους ερευνητές με την πραγματική συμπεριφορά αποικιών μυρμηγκιών όπως τον τυχαίο περίπατο (μιμούμενοι το ρόλο των διασταυρώσεων και των μεταλλαγών). Το τοπικό πρόβλημα αντιμετωπίζεται από τα μυρμήγκια με συστηματικότερη εξερεύνηση μιας περιοχής, προκειμένου να κινηθούν οι περιοχές πλησιέστερα στο βέλτιστο. Ο αλγόριθμος στέλνει μερικά μυρμήγκια στις «τοπικές» περιοχές και αυτά τα μυρμήγκια «επισημαίνουν» μερικά σημεία αφήνοντας φερορμόνη όταν βρίσκουν μια βελτίωση της αντικειμενικής συνάρτησης του συστήματος, κάνοντας τα σημεία αυτά πιο ελκυστικά για όλα τα μυρμήγκια της αποικίας.

Αυτή η διαδικασία είναι κοντά στην πραγματική συμπεριφορά των μυρμηγκιών, αλλά δυστυχώς, η χρήση δύο διαφορετικών διαδικασιών από τον αλγόριθμο CACO οδηγεί σε απαίτηση για έναν λεπτομερή καθορισμό των παραμέτρων [93, 94, και 95].

Άλλες μέθοδοι περιλαμβάνουν τον αλγόριθμο ασύγχρονης παράλληλης υλοποίησης (*Asynchronous Parallel Implementation - API*) (ο αλγόριθμος API εμπνέεται από την πρωτόγονη συμπεριφορά μυρμηγκιών και τη χρησιμοποίηση ενός διαδοχικού τρεξίματος που περιλαμβάνει δύο μυρμήγκια και καταλήγει στη συγκέντρωση όλων των πρακτόρων-μυρμηγκιών στην ίδια περιοχή) όπως εισάγεται από τον Monmarche [96], και τηη αποικία μυρμηγκιών συνεχούς αλληλεπίδρασης (Continuous Interacting Ant Colony - CIAC) , όπως εισάγεται από τους Dreo και Siarry [93]. Αν και ο CACO και ο CIAC υποστηρίζουν ότι βασίζονται σε ένα μετα-ευρετικό μοντέλου του ACO, δεν παρουσιάζουν και τόσες ομοιότητες. Όλοι οι αλγόριθμοι προσθέτουν μερικούς πρόσθετους μηχανισμούς (π.χ. άμεση επικοινωνία - CIAC και API - ή φωλιά - CACO) που δεν υπάρχουν στον κανονικό αλγόριθμο ACO. Δεν λαμβάνουν επίσης υπόψη μερικούς άλλους μηχανισμούς που είναι χαρακτηριστικοί του ACO (π.χ. απουσία της στιγμεργίας από τον αλγόριθμο API - ή η απουσία σταδιακής κατασκευής των λύσεων από όλους τους προαναφερθέντες αλγορίθμους). Οι αλγόριθμοι CACO και CIAC αφιερώνονται αυστηρά στη συνεχή βελτιστοποίηση, ενώ το API μπορεί επίσης να χρησιμοποιηθεί για διακριτά προβλήματα.

4.2.2.Επιλογή Αλγορίθμου

Για την επιλογή μιας μεθόδου αντιμετώπισης ενός δεδομένου συνδυαστικού προβλήματος βελτιστοποίησης πρέπει να ληφθούν υπόψη μια σειρά από παραμέτρους. Μερικές από τις βασικότερες παραμέτρους που πρέπει να μελετηθούν πριν την επιλογή μεθόδου/αλγορίθμου είναι τα παρακάτω:

- Πόσο γρήγορα δίνει αποτελέσματα,

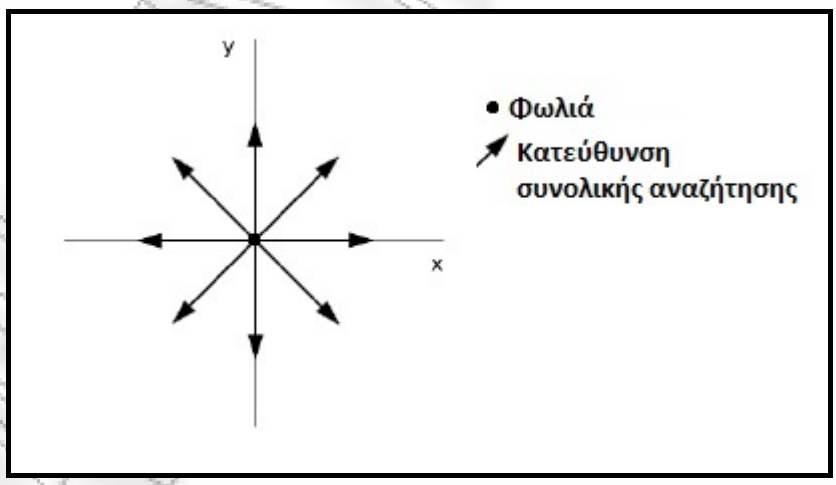
- Η ποιότητα των αποτελεσμάτων,
- Πιθανή παροχή εγγύησης για την ποιότητα της λύσης που βρίσκεται,
- Δυσκολία εφαρμογής,
- Πόσο καλά κλιμακώνεται ανάλογα με το μέγεθος προβλήματος,

Αφετέρου, πρέπει να εξεταστούν και τα χαρακτηριστικά του προβλήματος, καθώς επίσης και οι συνθήκες κάτω από τις οποίες το πρόβλημα καθίσταται επιλύσιμο:

- Το μέγεθος του προβλήματος.
- Απαιτούμενη ποιότητα λύσης.
- Διαθέσιμος χρόνος υλοποίησης μεθόδου.
- Διαθέσιμος χρόνος εκτέλεσης αλγορίθμου για την παραγωγή αποτελεσμάτων.

4.2.3.Ο Αλγόριθμος Con – ANT

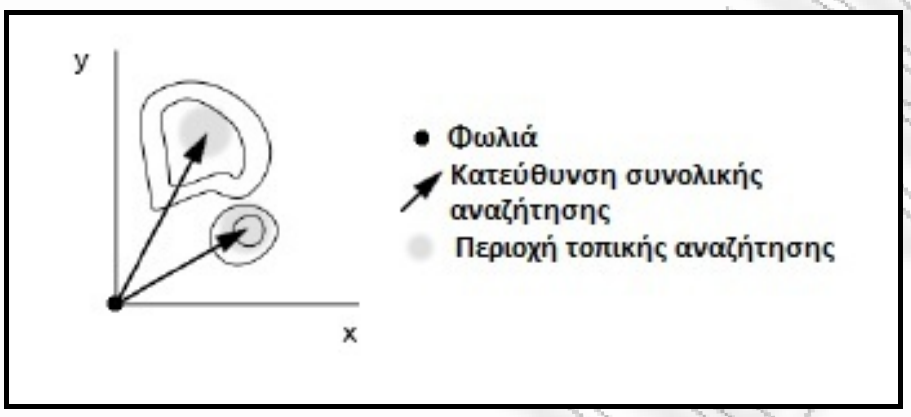
Ο αλγόριθμος Con-ANT [1] ήταν από τις πρώτες προσπάθειες στον τομέα της βελτιστοποίησης προβλημάτων συνεχών μεταβλητών. Συνιστάται για τοπική αναζήτηση σε μια «γειτονιά» πιθανών λύσεων γύρω από μια θέση που αποκαλείται φωλιά. Οι ερευνητές που εισήγαγαν τον αλγόριθμο επιχειρούν να βρουν την προαναφερθείσα λύση με τη χρήση μιας μεθόδου βελτιστοποίησης συνολικής αναζήτησης, όπως για παράδειγμα οι γενετικοί αλγόριθμοι. Στη συνέχεια γίνεται αναζήτηση στην επιλεγμένη γειτονιά με τη χρήση διανυσμάτων που εκκινούν από τη «φωλιά». Αρχικά, η αναζήτηση είναι ομοιόμορφα κατανεμημένη στη συνολική αναζήτηση σε μια ακτίνα και έχει τη μικρή συγκέντρωση φερορμόνης σε κάθε κατεύθυνση.



Εικόνα 2. Αρχική ομοικατευθυντική αναζήτηση αλγορίθμου Con-ANT

Αρχικά τα εικονικά μυρμήγκια επιλέγουν την κατεύθυνση τυχαία (Εικόνα 2). Στη συνέχεια κινούνται επίσης τυχαία μέσα σε μια ακτίνα, στην περιοχή όπου δείχνουν τα εκάστοτε διανύσματα, εκτελώντας μια τοπική αναζήτηση. Η ακτίνα αυτή μπορεί να μικρύνει προϊόντος του χρόνου, ώστε να κάνει την αναζήτηση πιο συγκεκριμένη γύρω από κάποια πιθανή λύση. Στη συνέχεια επιλέγεται η καταλληλότητα κάθε διαδρομής από τον αριθμό των ευρισκόμενων σε αυτή μυρμηγκιών και την ποσότητα φερορμόνης που έχει εναποτεθεί στη

διαδρομή αυτή (Εικόνα 3). Κάθε φορά που βρίσκεται μια καλύτερη από την μέχρι στιγμής βέλτιστη λύση από τον εν λόγω αλγόριθμο, τα διανύσματα προσανατολίζονται προς την κατεύθυνση της λύσης αυτής.



Εικόνα 3. Κατευθυνόμενη Αναζήτηση Αλγορίθμου Con-ANT

5. Υλοποίηση / Εκτέλεση Πειραμάτων

Στα πλαίσια της παρούσας εργασίας, εκτός από τη θεωρητική μελέτη του προβλήματος ένταξης μονάδων σε χρονικό ορίζοντα 24ώρου, έγινε και ο σχεδιασμός και η υλοποίηση ενός πρότυπου συστήματος, που μοντελοποιεί αρχικά μια εγκατάσταση μονάδων παραγωγής ενέργειας, και στη συνέχεια εφαρμόζει τον αλγόριθμο Con-ANT για τη εύρεση μιας αρκετά κοντινής στη βέλτιστη λύσης για τον οικονομικό προγραμματισμό λειτουργίας των προαναφερθεισών μονάδων.

Στη συνέχεια αυτού του κεφαλαίου γίνεται περιγραφή της διεξαχθείσας πρότυπης υλοποίησης, καθώς γίνεται αναφορά στη γλώσσα προγραμματισμού και τα εργαλεία που χρησιμοποιήθηκαν, περιγράφεται η δομή της παραχθείσας εφαρμογής, καθώς και περιγράφεται η λειτουργία της.

Στο δεύτερο μέρος αυτού του κεφαλαίου παρουσιάζονται τα πειραματικά στοιχεία, όπως αυτά προέκυψαν μετά από σειρά δοκιμών, κάτω από διαφορετικές συνθήκες και παραμέτρους του εν λόγω προβλήματος.

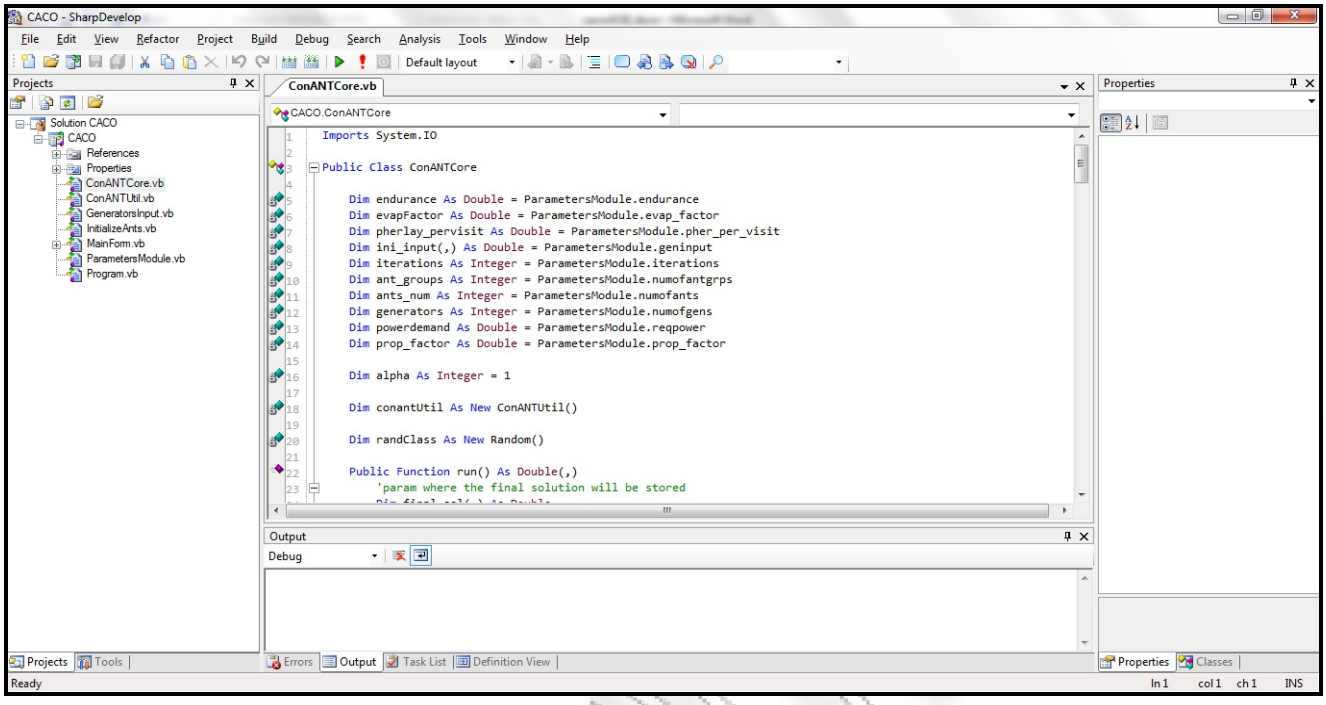
5.1 Πρότυπη Υλοποίηση

Στα πλαίσια της πρότυπης υλοποίησης που διεξήχθη σχεδιάστηκε και αναπτύχθηκε μια εφαρμογή η οποία μοντελοποιεί τη συμπεριφορά ενός συστήματος μονάδων παραγωγής, υλοποιεί τον αλγόριθμο Con-ANT και τελικά τον εφαρμόζει στο προαναφερθέν μοντέλο για την παραγωγή πλησίον της βέλτιστης λύσεων. Η διαδικασία αυτή περιγράφεται αναλυτικά στις ακόλουθες παραγράφους.

5.1.1 Χρησιμοποιούμενα Εργαλεία Ανάπτυξης

Για την ανάπτυξη τόσο του μοντέλου της εγκατάστασης μονάδων παραγωγής, όσο και του αλγορίθμου Con-ANT για την επίλυση του προβλήματος ένταξης των μονάδων χρησιμοποιήθηκε η γλώσσα προγραμματισμού Visual Basic .NET (VB.NET) της εταιρείας Microsoft Corporation [97].

Αναφορικά με το περιβάλλον ανάπτυξης, επελέγη το περιβάλλον ανάπτυξης SharpDevelop [98] το οποίο είναι ένα ανοιχτού κώδικα περιβάλλον ανάπτυξης. Στην Εικόνα 4 φαίνεται το εν λόγω περιβάλλον κατά τη χρήση του για το project της συγκεκριμένης εργασίας.



Εικόνα 4. Το περιβάλλον ανάπτυξης του SharpDevelop

5.1.2 Δομή Εφαρμογής / Αλγορίθμου

Σε αρχιτεκτονικό επίπεδο η ανάπτυξη της εφαρμογής χωρίζεται σε δύο μέρη: αφενός στην μοντελοποίηση των μονάδων μεμονωμένα και σαν σύνολο, και αφετέρου στην υλοποίηση του αλγορίθμου Con-ANT. Τα δύο αυτά μέρη παρουσιάζονται στη συνέχεια.

5.1.2.1. Μοντελοποίηση Μονάδων Παραγωγής

Στην εφαρμογή που αναπτύχθηκε η κάθε μονάδα παραγωγής προσδιορίζεται από μια σειρά από χαρακτηριστικά. Τα χαρακτηριστικά αυτά είναι τα ακόλουθα:

- **Ελάχιστη Παραγόμενη Ισχύς**: Προσδιορίζει την ελάχιστη δυνατή παραγόμενη ισχύ από την κάθε μονάδα.

- **Μέγιστη Παραγόμενη Ισχύς**: Προσδιορίζει την μέγιστη δυνατή παραγόμενη ισχύ από την κάθε μονάδα.

- **Αντικειμενική Συνάρτηση Κόστους**: Στα πλαίσια της εργασίας αυτής γίνεται η παραδοχή ότι η συνάρτηση που δίνει το κόστος λειτουργίας της μονάδας συναρτήσει της παραγόμενης ισχύος προσεγγίζεται ικανοποιητικά από μια πολυωνυμική συνάρτηση δευτέρου βαθμού. Έτσι η αντικειμενική συνάρτηση κόστους κάθε μονάδας χαρακτηρίζεται από τρεις σταθερές a, b, c που αποτελούν τους τρεις συντελεστές του προαναφερθέντος πολυωνύμου:

$$FC = a * x^2 + b * x + c$$

- **Αποδοτικότητα Ισχύος**: Χαρακτηριστικό μέγεθος που καθορίζει το ποσοστό της παραγόμενης ισχύος που είναι τελικά διαθέσιμο για εκμετάλλευση, και είναι μικρότερο της ονομαστικά παραγόμενης ισχύος

λόγω των διαφόρων απωλειών που υπεισέρχονται στο σύστημα παραγωγής.

- **Κλίση ισχύος**: Ο ρυθμός με τον οποίο μπορεί να μεταβάλλεται η παραγόμενη ισχύς κάθε μονάδας παραγωγής.
- **Χρόνος Εκκίνησης**: Ο χρόνος που απαιτείται για την εκκίνηση της συγκεκριμένης μονάδας παραγωγής.
- **Χρόνος Τερματισμού**: Ο χρόνος που απαιτείται για τον τερματισμό της συγκεκριμένης μονάδας παραγωγής.
- **Κόστος τερματισμού**: Το κόστος που σχετίζεται με τη διαδικασία τερματισμού της συγκεκριμένης μονάδας παραγωγής.
- **Κόστος εκκίνησης (Z)** : Το κόστος που σχετίζεται με τη διαδικασία εκκίνησης της συγκεκριμένης μονάδας παραγωγής όταν αυτή είναι σε θερμοκρασία λειτουργίας.
- **Κόστος Εκκίνησης (K)** : Το κόστος που σχετίζεται με τη διαδικασία εκκίνησης της συγκεκριμένης μονάδας παραγωγής όταν αυτή είναι σε κατάστασης αδράνειας για μεγάλο χρονικό διάστημα.

5.1.2.2. Υλοποίηση Αλγορίθμου Con-ANT

Κατά την υλοποίηση του αλγορίθμου Con-ANT, αναλύθηκε και υλοποιήθηκε η τοπική αναζήτηση, η συνολική αναζήτηση και φυσικά η διαδικασία εναπόθεσης, συγκέντρωσης και εξάτμισης της «φερορμόνης».

Σε πρώτο στάδιο μοιράζονται τα «εικονικά μυρμήγκια» σε ομάδες οι οποίες αναλαμβάνουν είτε την τοπική είτε την συνολική αναζήτηση. Κάθε ομάδα περιλαμβάνει τόσα εικονικά μυρμήγκια όσες και οι μονάδες παραγωγής που καλούμαστε να εντάξουμε σε λειτουργία στο εν λόγω πρόβλημα.

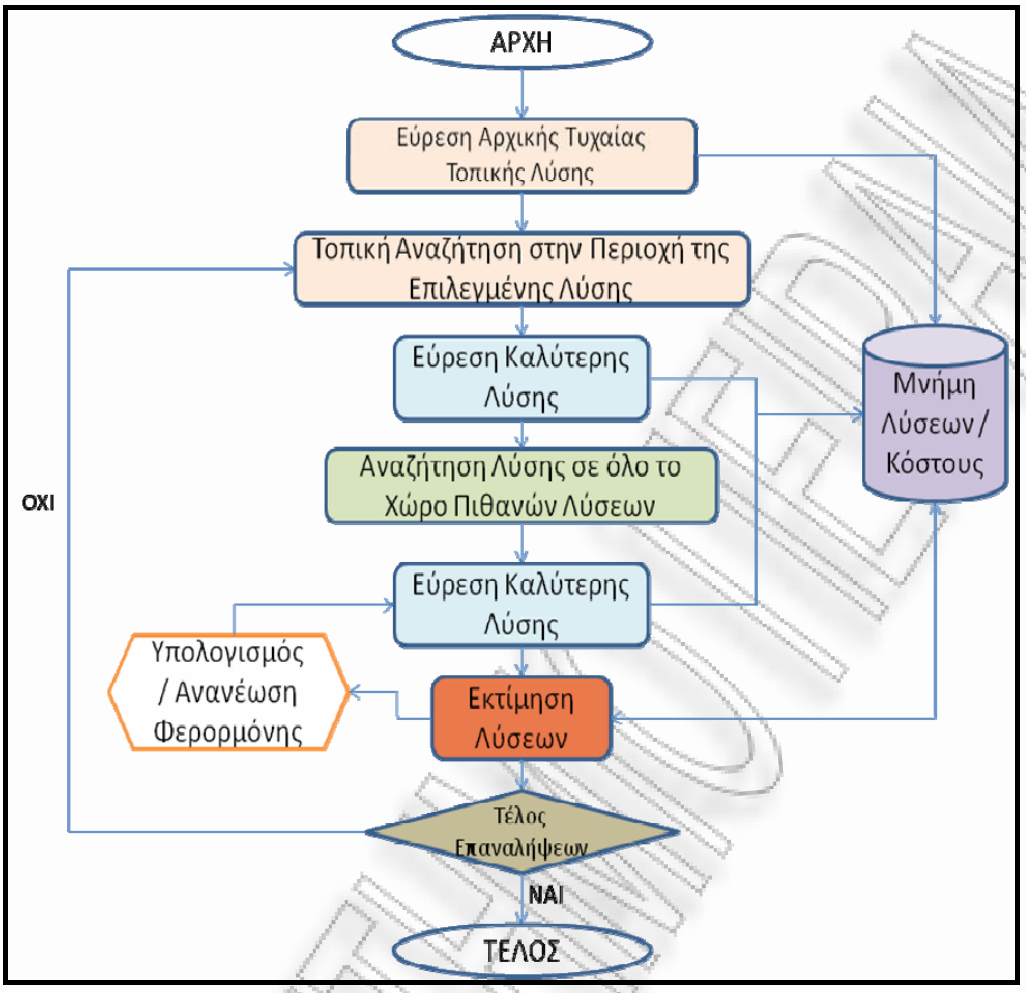
Αρχικά η εκτέλεση της τοπικής αναζήτησης εκκινεί από μια με τυχαίο τρόπο παραχθείσα λύση (διαδικασία FindGlobalRandomSolution). Στη συνέχεια υπολογίζεται η ακτίνα του χώρου αναζήτησης (διαδικασία CalculateSearchRadius) και εκτελείται αναζήτηση μέσα στο χώρο των πιθανών λύσεων που ορίζεται λαμβάνοντας ως κέντρο την τυχαία παραχθείσα αρχική λύση και ακτίνα αυτήν που υπολογίστηκε από την διαδικασία CalculateSearchRadius. Για κάθε λύση που βρίσκεται βάσει τον περιορισμών και των απαιτήσεων που έχουν τεθεί υπολογίζεται το κόστος της συγκεκριμένης λύσης (διαδικασία CalculateSolutionCost) και σε περίπτωση που το κόστος αυτό είναι μικρότερο από το μέχρι στιγμής χαμηλότερο κόστος της αναζήτησης που εκτελείται η εν λόγω λύση σημαίνεται ως η βέλτιστη που βρέθηκε.

Στη συνέχεια ακολουθεί η «συνολική αναζήτηση», η αναζήτηση δηλαδή σε ολόκληρο το χώρο των πιθανών λύσεων του προβλήματος. Κάθε ομάδα μυρμηγκιών που έχει αναλάβει την συνολική αναζήτηση επιλέγει τυχαία νέες περιοχές αναζήτησης μέσα από όλο το χώρο και αναζητά σε αυτές. Για την εύρεση νέας περιοχής λύσεων χρησιμοποιείται η διαδικασία FindGlobalRandomSolution.

Σε όλα τα στάδια εύρεσης λύσης στο πρόβλημα υπολογίζεται η υπάρχουσα φερορμόνη, εναποτίθεται επιπρόσθετη και φυσικά στο τέλος κάθε επανάληψης υπολογίζεται και η εξάτμιση της φερορμόνης (διαδικασία CalculatePheromon).

Αφού ολοκληρωθεί και η τοπική και η συνολική αναζήτηση υπολογίζεται η βέλτιστη λύση που βρέθηκε για κάθε επανάληψη. Η διαδικασία αυτή επαναλαμβάνεται για όσες επαναλήψεις οριστούν από τον χρήστη της εφαρμογής.

Στην Εικόνα 5 φαίνεται σχηματικά η λογική ροή του αλγορίθμου Con-ANT όπως αυτός υλοποιήθηκε στην εργασία αυτή.



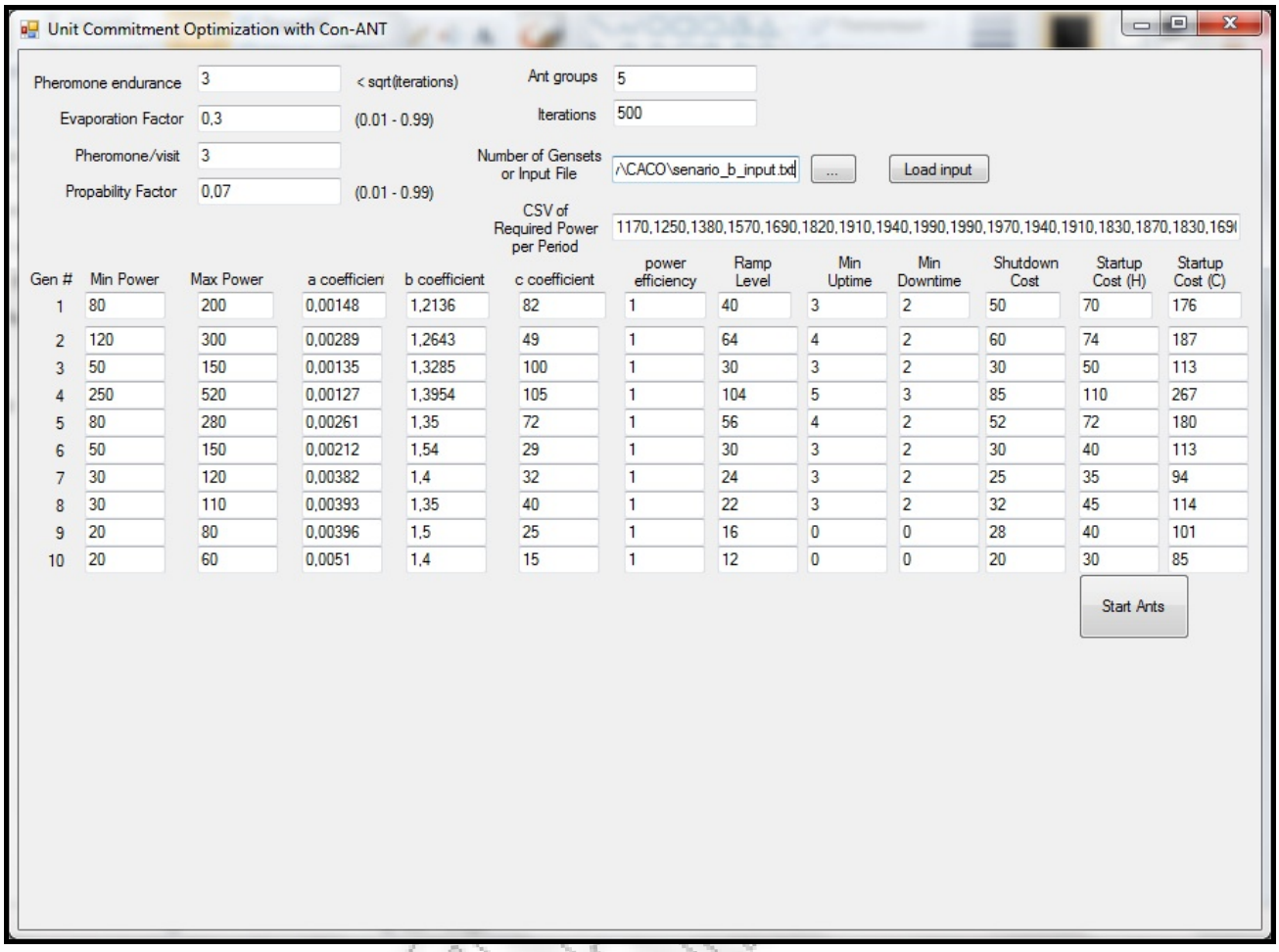
Εικόνα 5. Διάγραμμα Ροής Αλγορίθμου Con-ANT

5.1.3 Ανάλυση Εφαρμογής

5.1.3.1. Περιγραφή Λειτουργικότητας

Η υλοποίηση που περιγράφηκε στην προηγούμενη παράγραφο έχει ως αποτέλεσμα τη δημιουργία ενός εκτελέσιμου αρχείου (.exe file), το οποίο αποτελεί την εφαρμογή που εκτελούμε για την μοντελοποίηση και εύρεση λύσης στο πρόβλημα ένταξης μονάδων παραγωγής σε χρονικό ορίζοντα εικοσιτετραώρου. Στην παράγραφο αυτή περιγράφεται η λειτουργικότητα αυτής της εφαρμογής.

Κατά την εκκίνηση της εφαρμογής ο χρήστης εισάγεται στην αρχική οθόνη εισαγωγής παραμέτρων. Ο οθόνη αυτή φαίνεται στην Εικόνα 6.



Εικόνα 6. Αρχική Οθόνη Εισαγωγής Δεδομένων Εφαρμογής

Όπως φαίνεται και στην Εικόνα 6 εισάγονται στην εφαρμογή στο στάδιο αυτό οι παρακάτω απαραίτητες παράμετροι για την εφαρμογή:

- **Ant groups**: Ο αριθμός των ομάδων εικονικών μυρμηγκιών που θα διατρέξουν το χώρο των πιθανών λύσεων για την εύρεση της πλησιέστερης στη βέλτιστη λύσης. Σημειώνεται ότι κάθε ομάδα έχει ίσο αριθμό μυρμηγκιών με τον αριθμό των προς ένταξη μονάδων παραγωγής.
- **Iterations**: Ο αριθμός των επαναλήψεων του αλγορίθμου Con-ANT.
- **CSV of Required Power per Period**: Μια λίστα τιμών διαχωρισμένων με κόμματα, που αντιπροσωπεύουν την απαιτούμενη ισχύ από το σύνολο των μονάδων παραγωγής ανά στοιχειώδη περίοδο λειτουργίας.
- **Pheromone Endurance**: Παράμετρος που καθορίζει το πόσο ανθεκτική είναι η φερορμόνη που αφήνουν τα εικονικά μυρμήγκια κατά την εύρεση των λύσεων.
- **Evaporation factor**: Παράμετρος που καθορίζει το πόσο γρήγορα εξατμίζεται η φερορμόνη που εναποθέτουν τα εικονικά μυρμήγκια. Περιορίζεται μεταξύ των τιμών 0,01 και 0,99.
- **Pheromone/visit**: Η ποσότητα φερορμόνης που εναποθέτουν τα μυρμήγκια ανά «επίσκεψη» σε μια τιμή λύσης.

- **Probability factor**: Παράμετρος που καθορίζει την πιθανότητα ένα μυρμήγκι από λάθος να μην επιλέξει το μονοπάτι / λύση που έχει τη μεγαλύτερη συγκέντρωση φερορμόνης. Περιορίζεται μεταξύ των τιμών 0,01 και 0,99.

Επιπρόσθετα ορίζονται τα χαρακτηριστικά κάθε γεννήτριας όπως αυτά ορίστηκαν στην Παράγραφο 5.1.2.1 και εδώ αναφέρονται ως εξής:

- **Min Power**: Ελάχιστη παραγόμενη ισχύς από την συγκεκριμένη μονάδα παραγωγής.
- **Max Power**: Μέγιστη παραγόμενη ισχύς από την συγκεκριμένη μονάδα παραγωγής.
- **A coefficient**: Συντελεστής a του πολυωνύμου που κατά προσέγγιση προσδιορίζει την αντικειμενική συνάρτηση κόστους (FC = a*x^2+b*x+c) της συγκεκριμένης μονάδας παραγωγής.
- **B coefficient**: Συντελεστής b του πολυωνύμου που κατά προσέγγιση προσδιορίζει την αντικειμενική συνάρτηση κόστους (FC = a*x^2+b*x+c) της συγκεκριμένης μονάδας παραγωγής.
- **C coefficient**: Συντελεστής c του πολυωνύμου που κατά προσέγγιση προσδιορίζει την αντικειμενική συνάρτηση κόστους (FC = a*x^2+b*x+c) της συγκεκριμένης μονάδας παραγωγής.
- **Power efficiency**: Παράμετρος που καθορίζει την αποδοτικότητα σε ισχύ της συγκεκριμένης γεννήτριας / μονάδας παραγωγής. Περιορίζεται μεταξύ των τιμών 0,01 και 0,99.
- **Κλίση ισχύος**: Ο ρυθμός με τον οποίο μπορεί να μεταβάλλεται η παραγόμενη ισχύς κάθε μονάδας παραγωγής.
- **Χρόνος Εκκίνησης**: Ο χρόνος που απαιτείται για την εκκίνηση της συγκεκριμένης μονάδας παραγωγής.
- **Χρόνος Τερματισμού**: Ο χρόνος που απαιτείται για τον τερματισμό της συγκεκριμένης μονάδας παραγωγής.
- **Κόστος τερματισμού**: Το κόστος που σχετίζεται με τη διαδικασία τερματισμού της συγκεκριμένης μονάδας παραγωγής.
- **Κόστος εκκίνησης (Z)** : Το κόστος που σχετίζεται με τη διαδικασία εκκίνησης της συγκεκριμένης μονάδας παραγωγής όταν αυτή είναι σε θερμοκρασία λειτουργίας.
- **Κόστος Εκκίνησης (K)** : Το κόστος που σχετίζεται με τη διαδικασία εκκίνησης της συγκεκριμένης μονάδας παραγωγής όταν αυτή είναι σε κατάστασης αδράνειας για μεγάλο χρονικό διάστημα.

5.1.3.2. Περιγραφή Παραμέτρων

Πολύ σημαντικός για την ορθή εκτέλεση του προγράμματος, αλλά και την ουσιαστική είναι ο ορισμός και η κατανόηση των διαθέσιμων παραμέτρων για την μοντελοποίηση / προσομοίωση του αλγορίθμου μυρμηγκιών από την εφαρμογή.

Πιο συγκεκριμένα πολύ σημαντικό ρόλο στην εκτέλεση του αλγορίθμου των μυρμηγκιών παίζει η διάρκεια της φερορμόνης, ο χρόνος δηλαδή που τα μυρμήγκια που ακολουθούν μπορούν να «μυρίζουν» τη φερορμόνη που εναπόθεσε ένα μυρμήγκι στο

παρελθόν. Αυτό στην περίπτωση του αντικειμένου αυτής της εργασίας συνεπάγεται στον αριθμό των «κύκλων εκτέλεσης» του αλγορίθμου για τους οποίους μια εναπόθεση φερορμόνης είναι ενεργή για τον υπόλοιπο πληθυσμό «ενεργών» μυρμηγκιών.

Η προσομοίωση του προαναφερόμενου μεγέθους μπορεί να γίνει με δύο τρόπους:

- Με καθορισμό ενός αριθμού επαναλήψεων του αλγορίθμου για τον οποίο η φερορμόνη είναι ενεργή.
- Τον ρυθμό «εξάτμισης» της φερορμόνης ανά κύκλο εκτέλεσης, καθώς και την ποσότητα εναποτιθέμενης φερορμόνης ανά κύκλο εκτέλεσης.

Στην πρώτη περίπτωση ο ορισμός και η λειτουργικότητα είναι απλή, στη δεύτερη σε κάθε κύκλο αφαιρείται ποσό που καθορίζεται από το ρυθμό εξάτμισης μέχρι τελικού μηδενισμού του ποσού εναπόθεσης ανά «πέρασμα μυρμηγκιού» από κάποια θέση-λύση του προβλήματος.

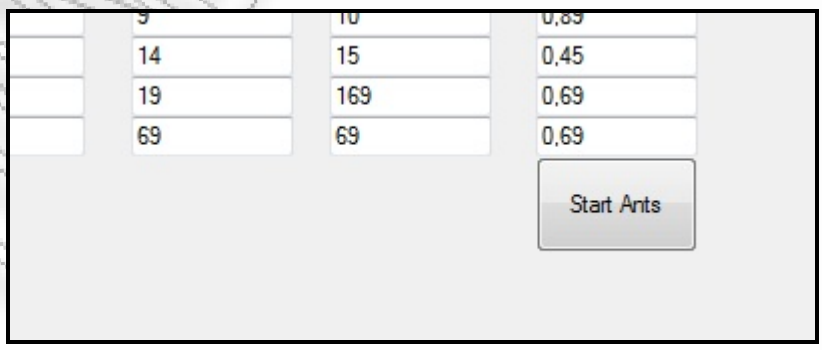
Στα πλαίσια της εφαρμογής αυτής, και για λόγους πληρότητας υλοποιήθηκαν παράμετροι εισόδου και για τους δύο τρόπους προσέγγισης του θέματος της εξάτμισης της φερορμόνης. Πιο συγκεκριμένα η παράμετρος «Pheromone Endurance» είναι ο αριθμός των κύκλων του αλγορίθμου για τον οποίο η φερορμόνη θεωρείται ενεργή. Η παράμετρος Pheromone/visit αντιπροσωπεύει το ποσό εναπόθεσης/επίσκεψη μυρμηγκιού (για την περίπτωση της δεύτερης λύσης/τρόπου προσέγγισης) και αντίστοιχα η παράμετρος evaporation factor είναι ο ρυθμός εξάτμισης της προαναφερθείσας ποσότητας φερορμόνης.

Για λόγους απλότητας στην εφαρμογή και στην διενέργεια του πειραματικού μέρους χρησιμοποιήθηκε ο πρώτος τρόπος καθορισμού της επίδρασης της φερορμόνης (Pheromone Endurance).

Η παράμετρος Probability factor αντιπροσωπεύει την πιθανότητα ένα μυρμήγκι παρότι «μύρισε» ποιο μονοπάτι έχει τη μεγαλύτερη συγκέντρωση φερορμόνης (και άρα θα έπρεπε να αποτελέσει την επιλογή του μυρμηγκιού) να επιλέξει διαφορετικό μονοπάτι από ¨λάθος εκτέλεση» του αλγορίθμου. Μαθηματικά αυτή η παράμετρος περιορίζεται μεταξύ 0 και 1, αλλά για ύπαρξη λογικής συνέχειας στην εκτέλεση του αλγορίθμου επιλέγονται πολύ χαμηλές τιμές για την παράμετρο αυτή (π.χ. 0,05 ήτοι 5% πιθανότητα για σφάλμα). Αν επιλεγεί μεγάλη τιμή για αυτήν την παράμετρο ο αλγόριθμος στην ουσία γίνεται μη αποδοτικός.

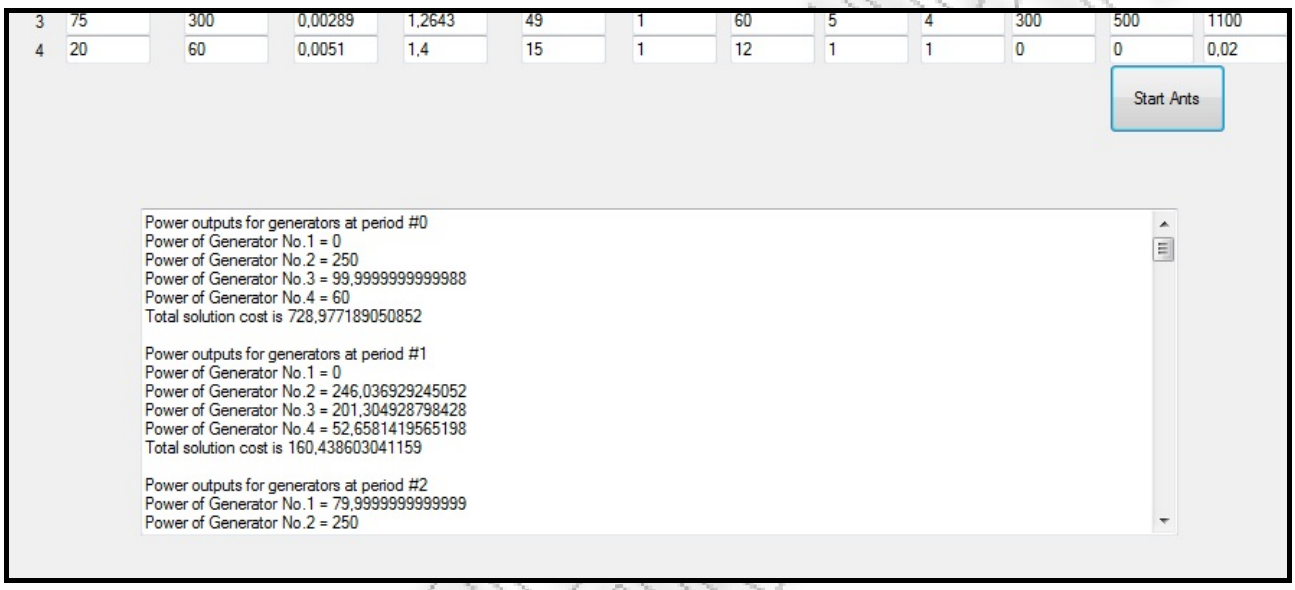
5.1.3.3. Περιγραφή Εκτέλεσης

Μετά την ορθή συμπλήρωση των παραμέτρων εισόδου της εφαρμογής που αφορούν τόσο τον αλγόριθμο όσο και τα χαρακτηριστικά των γεννητριών, εκκινούμε τον αλγόριθμο Con-ANT μέσω του button «Start Ants» που φαίνεται καλύτερα και στην Εικόνα 7.



Εικόνα 7. Button Εκκίνησης Αλγορίθμου

Μετά την ολοκλήρωση της εκτέλεσης του αλγορίθμου Con-ANT σύμφωνα με το διάγραμμα ροής που περιγράφηκε στην παράγραφο 5.1.2.2 εμφανίζεται στην οθόνη της εφαρμογής μια περιοχή κειμένου στην οποία καταγράφονται οι ισχείς λειτουργίας για όλες τις μονάδες παραγωγής, με τις οποίες το σύστημα βρέθηκε να έχει το ελάχιστο κόστος λειτουργίας για τις δεδομένες απαιτήσεις και περιορισμούς. Η περιοχή αυτή των αποτελεσμάτων φαίνεται καλύτερα στην Εικόνα 8.



Εικόνα 8. Περιοχή Αποτελεσμάτων Εφαρμογής

Εκτός από την παρουσίαση των ισχύων λειτουργίας των μονάδων παραγωγής η εφαρμογή καταγράφει σε ένα αρχείο το κόστος λειτουργίας της βέλτιστης ευρεθείσας λύσης, όπως αυτή καταγραφόταν κατά τη διάρκεια των αναζητήσεων τόσο τοπικά όσο και συνολικά, για όλες τις επαναλήψεις που ορίστηκε να εκτελεσθεί ο αλγόριθμος. Αυτό γίνεται για να δοθεί η δυνατότητα στον ερευνητή που εκτελεί την εφαρμογή της πρότυπης υλοποίησης να αποτυπώσει την πορεία του αλγορίθμου και το κατά πόσο ο αλγόριθμος αυτός συγκλίνει σε κάποια λύση κατά το δυνατόν πλησίον της βέλτιστης.

Το αρχείο αυτό ονομάζεται genset_costs_per_iteration_bestsolution.txt και δημιουργείται στο φάκελο από τον οποίον εκκινήθηκε το πρόγραμμα της πρότυπης υλοποίησης.

5.2 Εκτέλεση Πειραμάτων

Για την επαλήθευση τόσο της λειτουργικότητας αυτής καθαυτής της εφαρμογής, όσο και όσων θεωρητικά μελετήθηκαν σχετικά με το υπό μελέτη πρόβλημα, διεξήχθη μια σειρά πειραμάτων.

Για την εκτέλεση των λειτουργικών και ποιοτικών πειραμάτων μελετήθηκαν δύο διαφορετικές διατάξεις μονάδων παραγωγής και στη συνέχεια εφαρμόστηκε ο αλγόριθμος Con-ANT για την εύρεση της βέλτιστης δυνατής λύσης. Στη συνέχεια περιγράφονται οι δύο μελετώμενες διατάξεις και παρατίθενται τα αποτελέσματα εκτέλεσης της πρότυπης

εφαρμογής αντίστοιχα. Τέλος, γίνεται μια εκτίμηση των αποτελεσμάτων και εξαγωγή των αντίστοιχων συμπερασμάτων.

Στη συνέχεια προσομοιώθηκε μια ήδη μελετημένη εγκατάσταση γεννητριών, η οποία αναλύεται στην ερευνητική προσπάθεια που παρουσιάζεται στο [99], και στη συνέχεια εκτελείται ώστε να εξαχθούν συγκριτικά συμπεράσματα για την αποτελεσματικότητα τόσο του αλγορίθμου όσο και αυτής καθαυτή της εφαρμογής.

5.2.1 Περιγραφή Πειραματικού Συστήματος

Σενάριο Α

Η πρώτη διάταξη αφορά σύστημα μονάδων παραγωγής αποτελούμενο από 5 γεννήτριες με χαρακτηριστικά που δίνονται στον παρακάτω Πίνακα 1.

	Χαρακτηριστικά Ισχύος			**Χαρακτηριστικά Κόστους**		
A/A	**Min P (MW)**	**Max P (MW)**	**Power Loss**	**A coeff.**	**B coeff.**	**C coeff.**
1	210	290	0.76	3	4	16
2	220	390	0.65	10	11	68
3	170	320	0.71	12	4	121
4	170	430	0.44	23	2	65
5	300	490	0.80	7	123	34

Πίνακας 1. Χαρακτηριστικά Μονάδων Παραγωγής Σεναρίου Α

Για το σενάριο Α η ζητούμενη ισχύς ορίζεται στα 1220 MW.

Σενάριο Β

Η δεύτερη διάταξη αφορά σύστημα μονάδων παραγωγής αποτελούμενο από 10 γεννήτριες με χαρακτηριστικά που δίνονται στον παρακάτω Πίνακα 2.

	Χαρακτηριστικά Ισχύος			**Χαρακτηριστικά Κόστους**		
A/A	**Min P (MW)**	**Max P (MW)**	**Power Loss**	**A coeff.**	**B coeff.**	**C coeff.**
1	210	290	0.76	3	4	16

2	220	390	0.65	10	11	68
3	170	320	0.71	12	4	121
4	170	430	0.44	23	2	65
5	300	490	0.80	7	123	34
6	325	450	0.65	13	2	44
7	250	290	0.85	120	32	2
8	150	220	0.7	12	24	7
9	100	180	0.63	2	198	234
10	290	370	0.35	7	2	56

Πίνακας 2. Χαρακτηριστικά Μονάδων Παραγωγής Σεναρίου Β

Για το σενάριο Β η ζητούμενη ισχύς ορίζεται στα 2450 MW.

Αλγόριθμος Con-ANT

Και για τα δύο σενάρια ορίζονται οι παρακάτω παράμετροι αναφορικά με τον αλγόριθμο Con-ANT:

- **Ant groups**: 5, 10, 20
- **Iterations**: 500
- **Pheromone Endurance**: 3
- **Evaporation factor**: 0.3
- **Pheromone/visit**: 3
- **Probability factor**: 0.07

Σημειώνεται ότι θα διεξαχθούν δοκιμές για διάφορες τιμές της παραμέτρου Ant groups (5, 10 και 20 ομάδες μυρμηγκιών αντίστοιχα) για να διαπιστωθεί η επίπτωση του αριθμού των μυρμηγκιών στην απόδοση και στην ταχύτητα σύγκλισης του αλγορίθμου.

5.2.2 Εκτέλεση Πειραμάτων

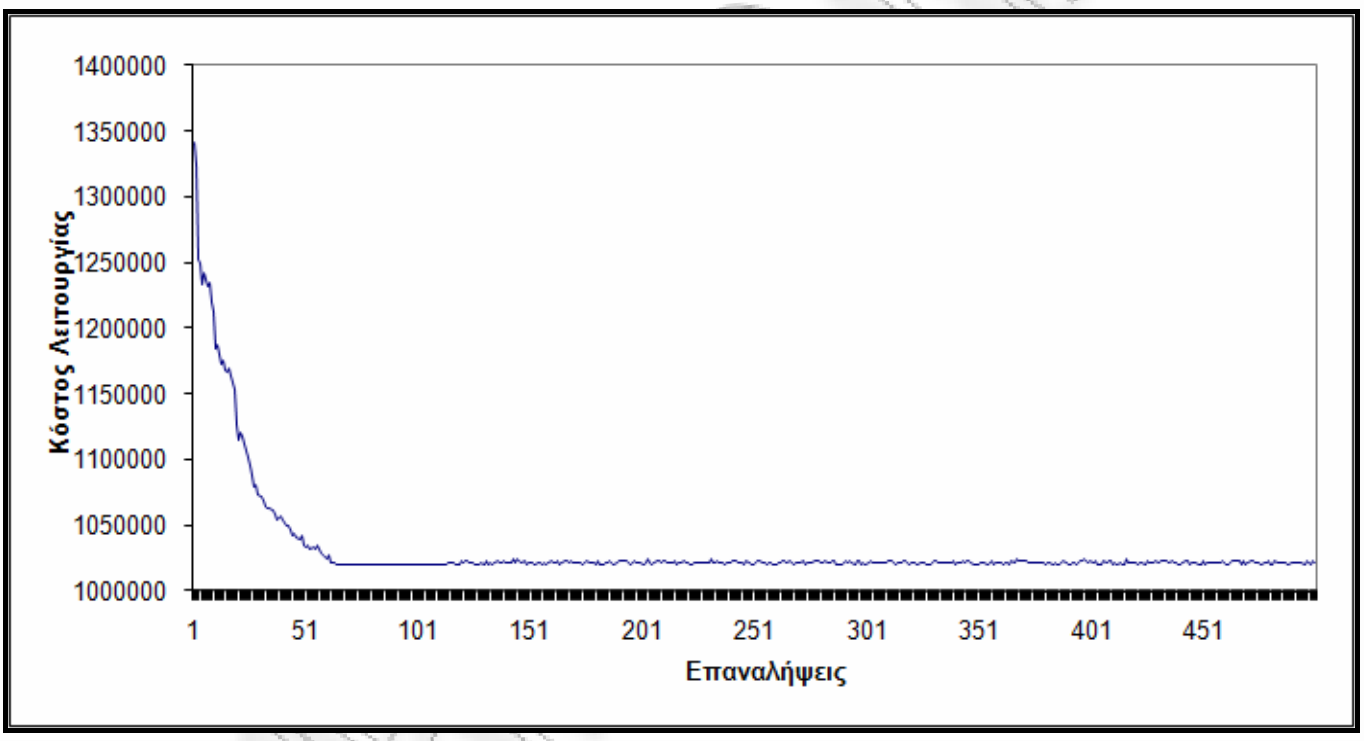
Για την εκτέλεση των πειραμάτων παραμετροποιήθηκε κατάλληλα η εφαρμογή ώστε να προσομοιώσει τα διάφορα σύνολα μονάδων και απαιτήσεων / περιορισμών που περιγράφηκαν. Στη συνέχεια εκτελέστηκε ο αλγόριθμος Con-ANT και παράχθηκαν τα παρακάτω αποτελέσματα

Σενάριο Α, 5 ομάδες μυρμηγκιών

Ο βέλτιστος συνδυασμός παραγόμενων ισχύων από τις δοθείσες μονάδες βρέθηκε να είναι ο παρακάτω:

A/A	Ισχύς Λειτουργίας (MW)
1	248,77
2	262,52
3	194,07
4	191,51
5	323,13

Στην Εικόνα 9 φαίνεται η πορεία του κόστους συναρτήσει των επαναλήψεων του αλγορίθμου.



Εικόνα 9. Σενάριο Α, 5 Ομάδες Μυρμηγκιών

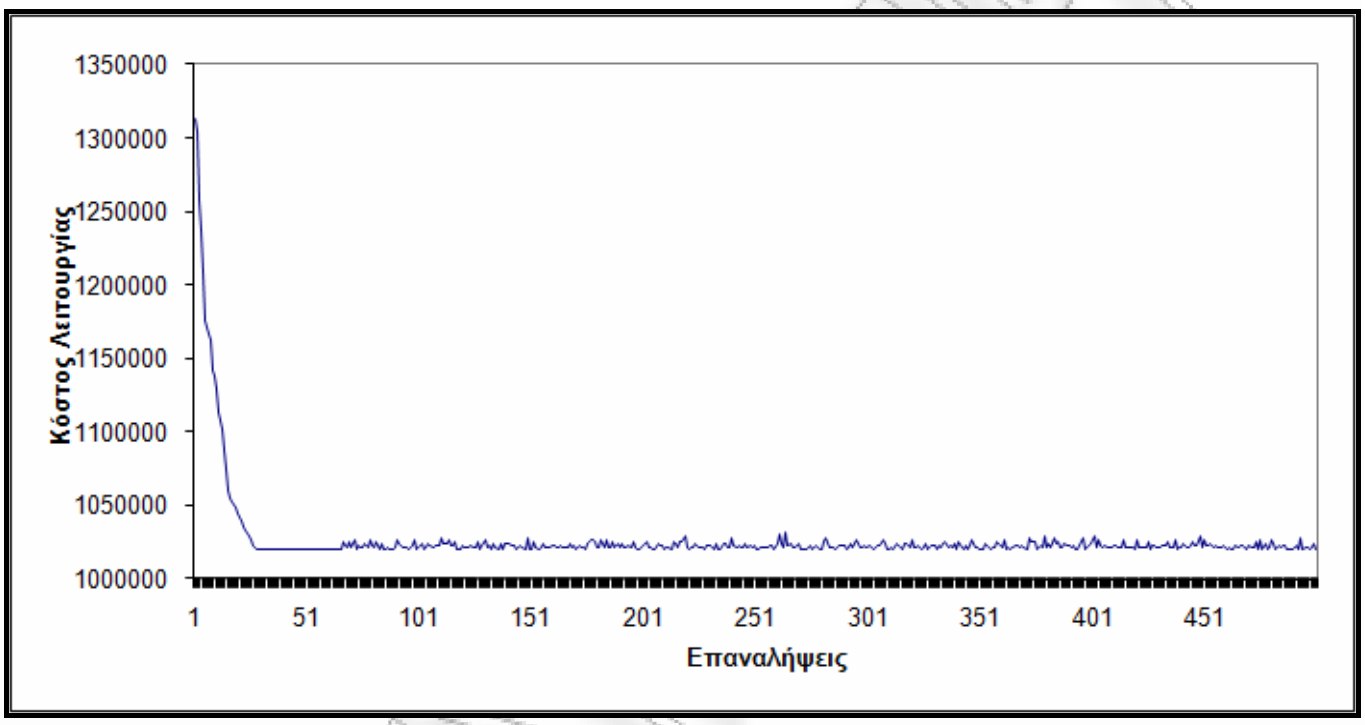
Σενάριο Α, 10 ομάδες μυρμηγκιών

Ο βέλτιστος συνδυασμός παραγόμενων ισχύων από τις δοθείσες μονάδες βρέθηκε να είναι ο παρακάτω:

A/A	Ισχύς Λειτουργίας (MW)
1	245,75
2	245,09

3	202,89
4	203,84
5	322,43

Στην Εικόνα 10 φαίνεται η πορεία του κόστους συναρτήσει των επαναλήψεων του αλγορίθμου.



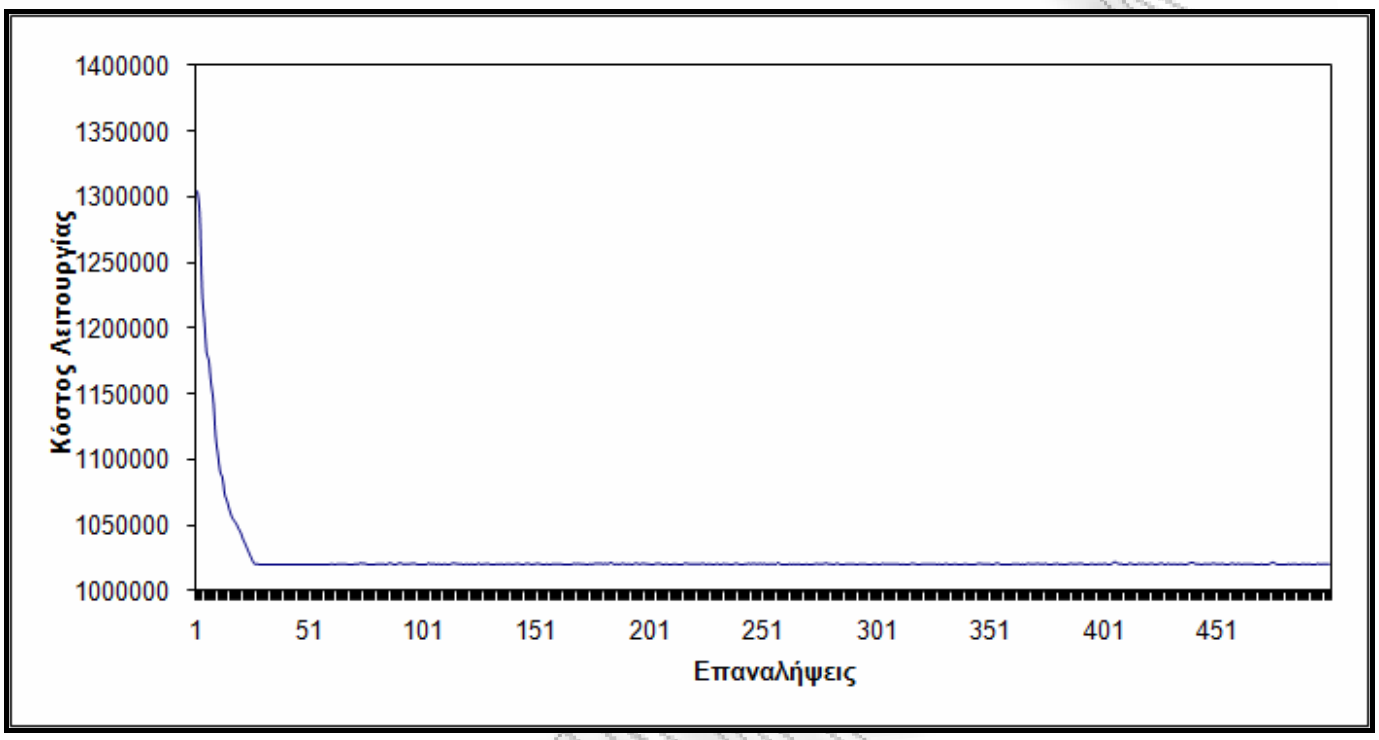
Εικόνα 10. Σενάριο Α, 10 Ομάδες Μυρμηγκιών

Σενάριο Α, 20 ομάδες μυρμηγκιών

Ο βέλτιστος συνδυασμός παραγόμενων ισχύων από τις δοθείσες μονάδες βρέθηκε να είναι ο παρακάτω:

Α/Α	**Ισχύς Λειτουργίας (MW)**
1	248,95
2	251,89
3	194,80
4	202,99
5	321,37

Στην Εικόνα 11 φαίνεται η πορεία του κόστους συναρτήσει των επαναλήψεων του αλγορίθμου.



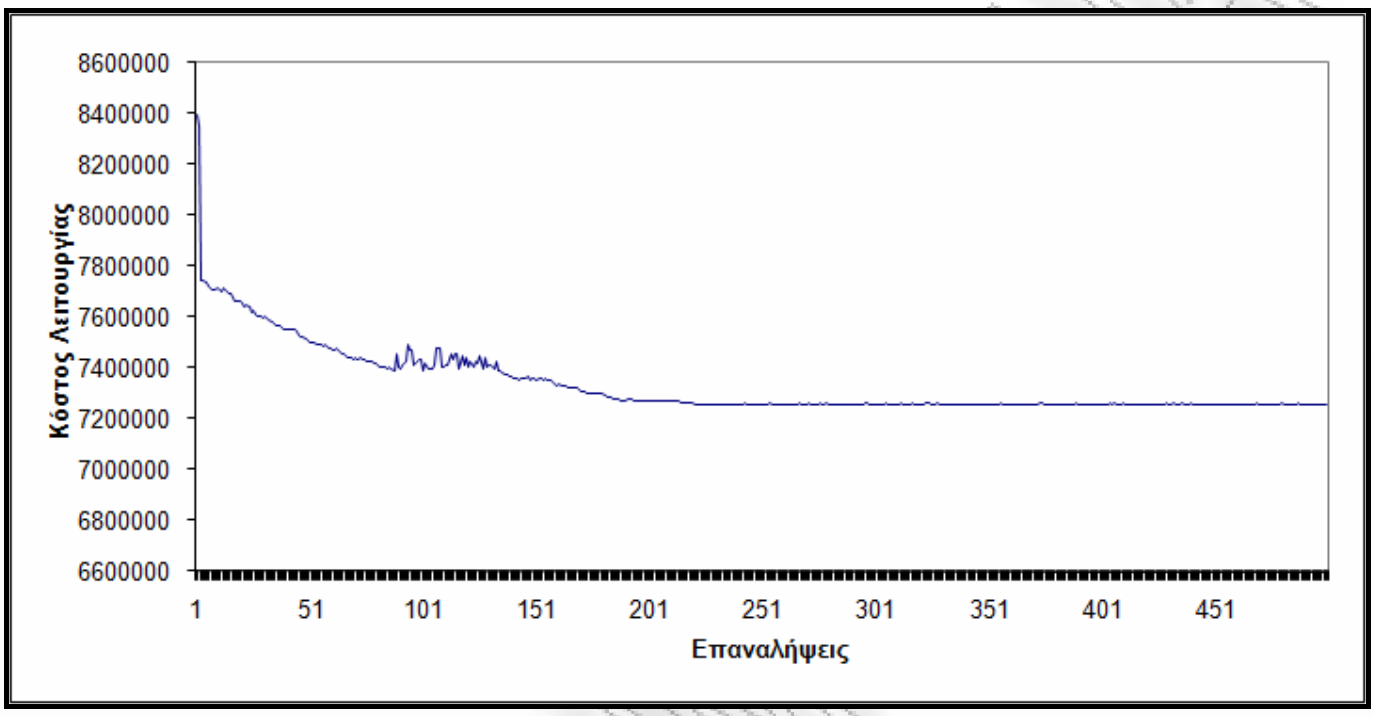
Εικόνα 11. Σενάριο Α, 20 Ομάδες Μυρμηγκιών

Σενάριο Β, 5 ομάδες μυρμηγκιών

Ο βέλτιστος συνδυασμός παραγόμενων ισχύων από τις δοθείσες μονάδες βρέθηκε να είναι ο παρακάτω:

Α/Α	Ισχύς Λειτουργίας (MW)
1	244,33
2	280,95
3	192,26
4	212,65
5	318,43
6	342,31
7	267,25
8	167,27
9	117,25
10	307,30

Στην Εικόνα 12 φαίνεται η πορεία του κόστους συναρτήσει των επαναλήψεων του αλγορίθμου.



Εικόνα 12. Σενάριο Β, 5 Ομάδες Μυρμηγκιών

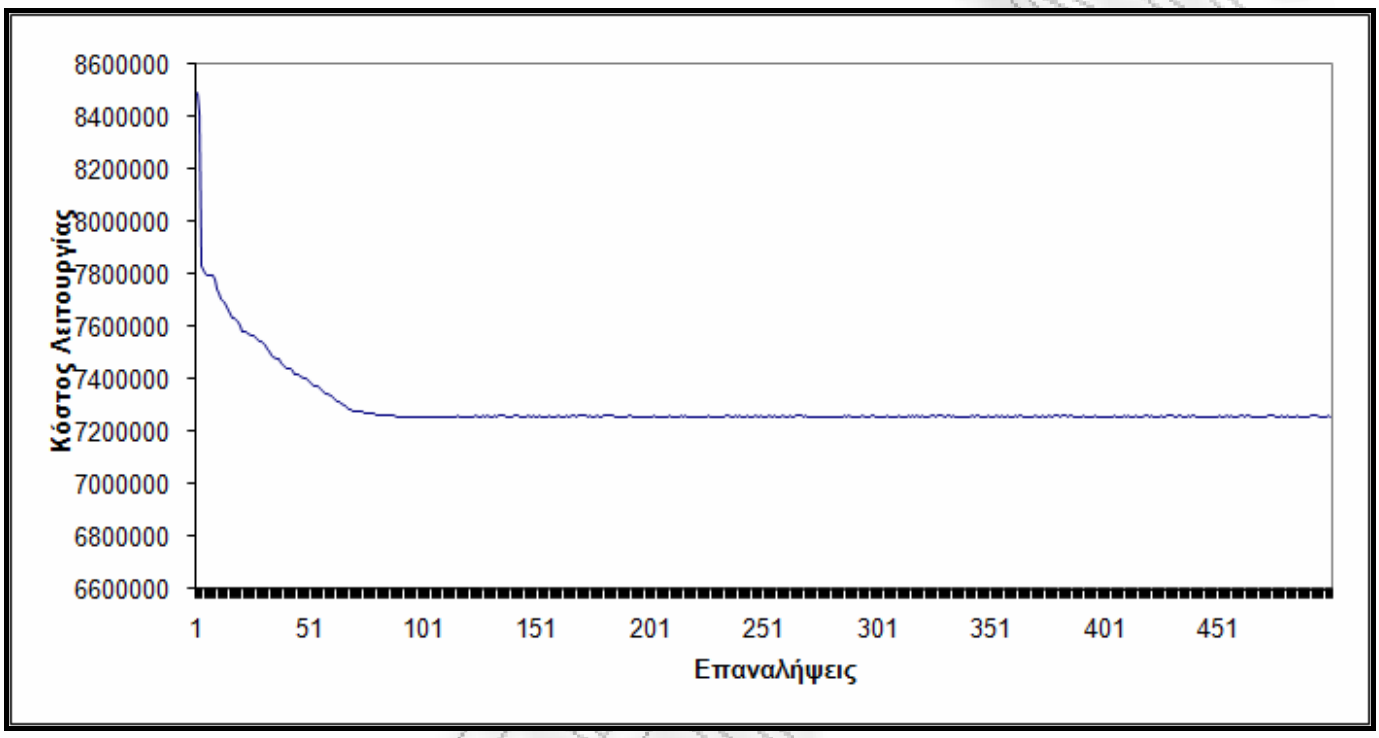
Σενάριο Β, 10 ομάδες μυρμηγκιών

Ο βέλτιστος συνδυασμός παραγόμενων ισχύων από τις δοθείσες μονάδες βρέθηκε να είναι ο παρακάτω:

A/A	Ισχύς Λειτουργίας (MW)
1	246,03
2	276,56
3	200,50
4	203,02
5	318,96
6	343,19
7	267,83
8	167,97
9	117,90

10	308,04

Στην Εικόνα 13 φαίνεται η πορεία του κόστους συναρτήσει των επαναλήψεων του αλγορίθμου.



Εικόνα 13. Σενάριο Β, 10 Ομάδες Μυρμηγκιών

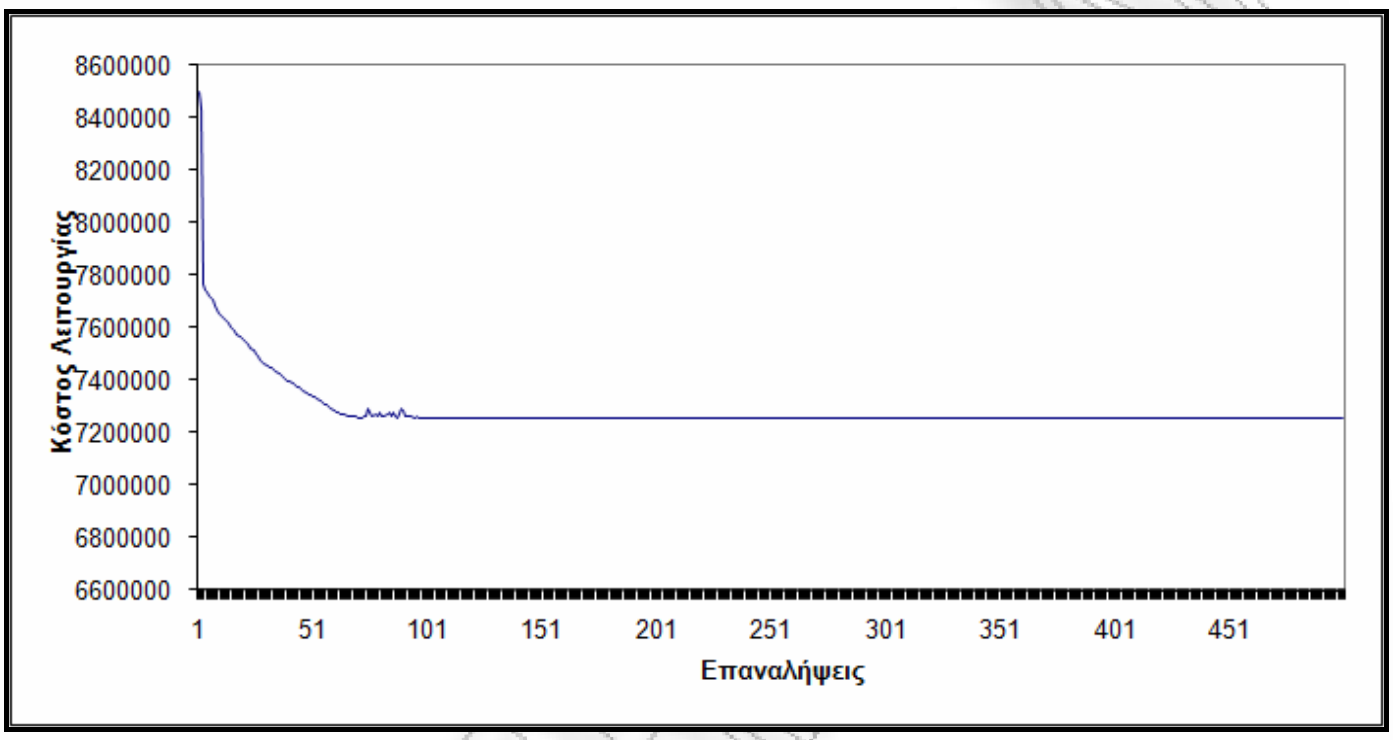
Σενάριο Β, 20 ομάδες μυρμηγκιών

Ο βέλτιστος συνδυασμός παραγόμενων ισχύων από τις δοθείσες μονάδες βρέθηκε να είναι ο παρακάτω:

Α/Α	**Ισχύς Λειτουργίας (MW)**
1	244,86
2	285,40
3	198,95
4	198,74
5	317,88
6	342,83
7	267,84
8	167,83
9	117,83

10	307,83

Στην Εικόνα 13 φαίνεται η πορεία του κόστους συναρτήσει των επαναλήψεων του αλγορίθμου.



Εικόνα 14. Σενάριο Β, 20 Ομάδες Μυρμηγκιών

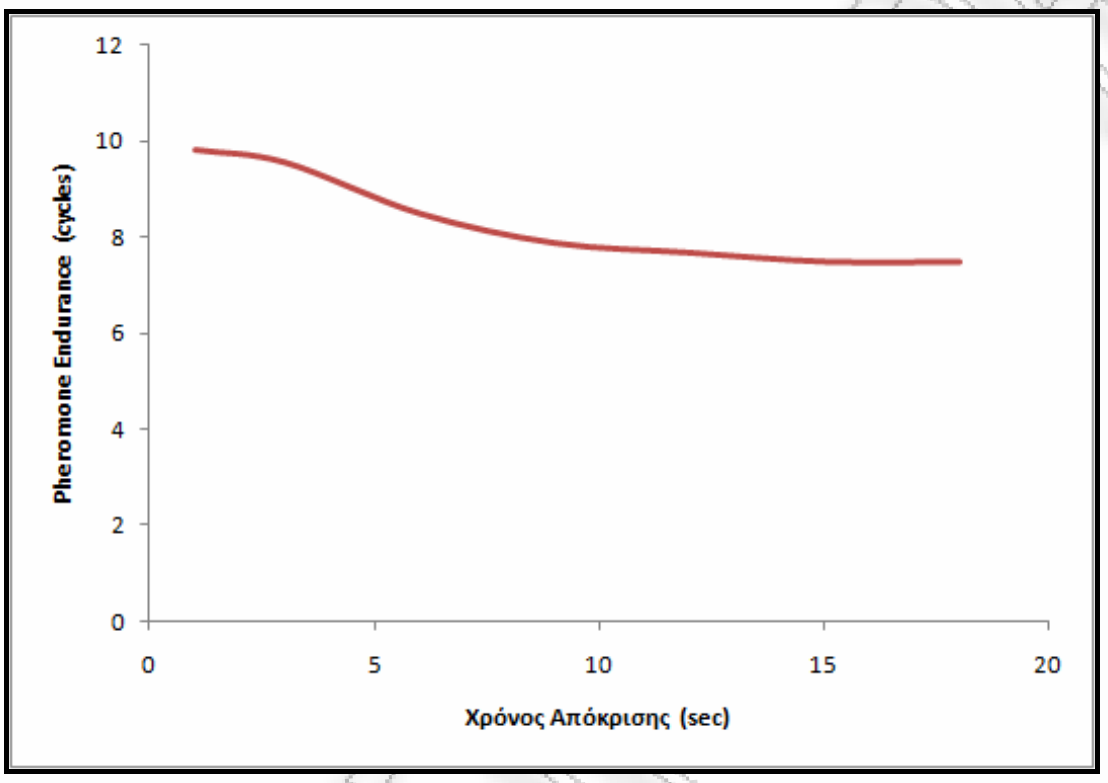
5.2.3 Εκτέλεση Δοκιμών Μεταβολής Παραμέτρων

Για λόγους πληρότητας της έρευνας έγιναν δοκιμές για το πώς επηρεάζει την εκτέλεση του αλγορίθμου πιθανή μεταβολή των παραμέτρων ανθεκτικότητας της φερορμόνης και της πιθανότητας λάθους επιλογής από τα μυρμήγκια.

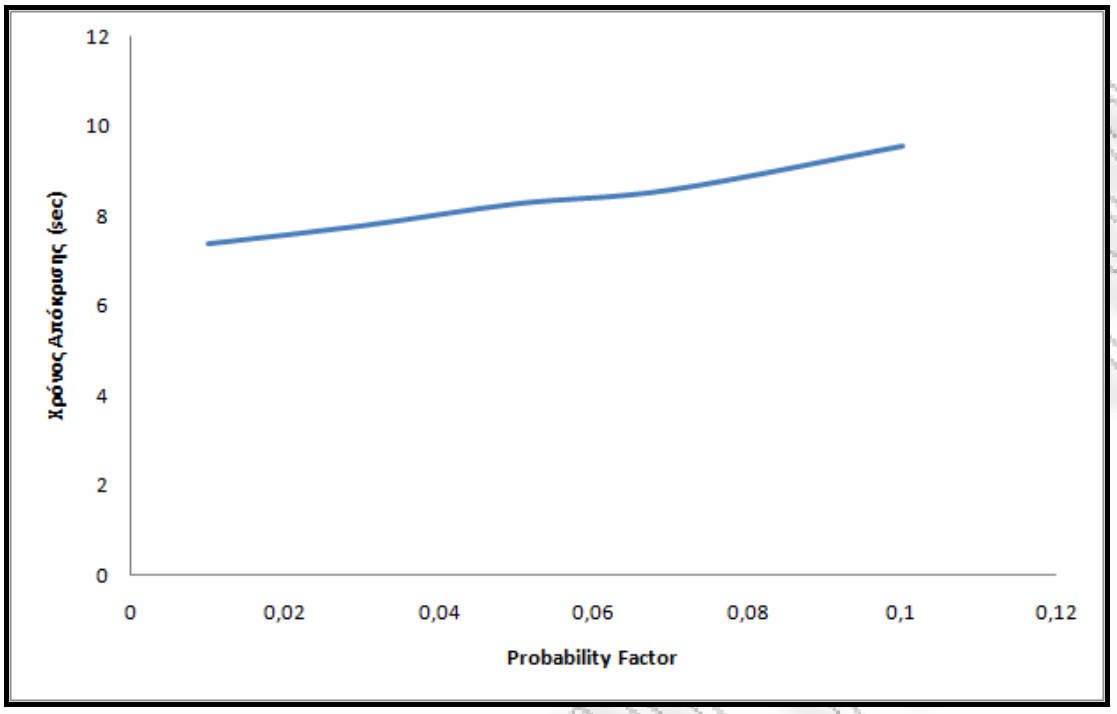
Πιο συγκεκριμένα, και όπως περιγράφηκε στην παράγραφο 5.1.3.2 η ανθεκτικότητα της φερορμόνης ελέγχεται στην συγκεκριμένη εφαρμογή από την παράμετρο Pheromone Endurance, ενώ η πιθανότητα λανθασμένης επιλογής από τα μυρμήγκια (επιλογή διαφορετικού «μονοπατιού» από αυτό με τη μεγαλύτερη συγκέντρωση φερορμόνης) από την παράμετρο Probability Factor.

Στο πρώτο στάδιο αυτών των δοκιμών διατηρήθηκε σταθερό το Probability Factor στην τιμή 0,07 και μεταβλήθηκε η ανθεκτικότητα της φερορμόνης από τον 1 έως και τους 20 κύκλους εκτέλεσης του αλγορίθμου. Όπως φαίνεται και στην Εικόνα 15 όσο μεγαλύτερη είναι η ανθεκτικότητα της φερορμόνης (μέχρι κάποιο σημείο) τόσο ταχύτερη είναι η απόκριση του αλγορίθμου. Αυτό είναι αναμενόμενο, καθώς μεγαλύτερη ανθεκτικότητα της φερορμόνης πρακτικά σημαίνει ότι τα επερχόμενα «προσομοιωμένα μυρμήγκια» θα εντοπίζουν ευκολότερα το «σωστό μονοπάτι», τη βέλτιστη δηλαδή μέχρι στιγμής λύση του προβλήματος. Αυτό εξηγείται από το γεγονός ότι η μεγαλύτερη ανθεκτικότητα συνεπάγεται μεγαλύτερη συγκέντρωση φερορμόνης στις επιλογές «βέλτιστων μονοπατιών/λύσεων».

Στο δεύτερο στάδιο κρατώντας σταθερή την παράμετρο Pheromone Endurance στην τιμή 12 (στην οποία από τα πειράματα του πρώτου σταδίου φαίνεται να σταθεροποιείται η απόδοση του αλγορίθμου) μεταβάλλαμε την πιθανότητα σφάλματος επιλογής (Probability Factor). Στην Εικόνα 16 φαίνεται η διακύμανση του χρόνου απόκρισης συναρτήσει της πιθανότητας σφάλματος επιλογής, και το αποτέλεσμα είναι αναμενόμενο, καθώς όσο μεγαλύτερη είναι η πιθανότητα σφάλματος στην επιλογή, τόσο αργότερα αναμένουμε να αποδώσει ο αλγόριθμος και αυτό αποτυπώνεται στο χρόνο απόκρισης του αλγορίθμου (και επαγωγικά της εφαρμογής) ξεκάθαρα.



Εικόνα 15 Χρόνος απόκρισης συναρτήσει της ανθεκτικότητας της φερορμόνης



Εικόνα 16 Χρόνος απόκρισης συναρτήσει της πιθανότητας σφάλματος επιλογής

5.2.4 Προσομοίωση Εγκατάστασης Σε Χρονικό Ορίζοντα 24 Ωρών

5.2.4.1 Προσομοίωση Εγκατάστασης 4 Γεννητριών

Στο πρώτο μέρος της συγκριτικής δοκιμής μελετήθηκε ένα σύστημα 4 γεννητριών με διαφορετικά χαρακτηριστικά, τα οποία δίνονται παρακάτω:

Ελάχιστη Ισχύς	Μέγιστη Ισχύς	Παράμετρος a	Παράμετρος b	Παράμετρος c	Απόδοση Γεννήτριας	Κλίση ισχύος	Χρόνος Εκκίνησης	Χρόνος Τερματισμού	Κόστος τερματισμού	Κόστος εκκίνησης (Z)	Κόστος Εκκίνησης (K)	Cold Start	Αρχική Κατάσταση
25	80	0,00396	1,5	25	1	16	4	2	80	150	350	4	-5
60	250	0,00261	1,35	72	1	50	5	3	110	170	400	5	8
75	300	0,00289	1,2643	49	1	60	5	4	300	500	1100	5	8
20	60	0,0051	1,4	15	1	12	1	1	0	0	0,02	0	-6

Στη συγκεκριμένη εγκατάσταση έγινε μελέτη ένταξης μονάδων παραγωγής σε χρονικό ορίζοντα 24 ωρών. Οι απαιτήσεις σε ισχύ στις 24 αυτές ώρες δίνονται από την παρακάτω αλληλουχία τιμών απαιτούμενης ισχύος σε MW:

Αύξων Αριθμός Χρονικής Περιόδου (hr)	Απαιτούμενη Ισχύς(MW)
1	410
2	500
3	575
4	650

5	**555**
6	**450**
7	**400**
8	**445**
9	**535**
10	**600**
11	**540**
12	**495**
13	**450**
14	**516**
15	**585**
16	**625**
17	**530**
18	**465**
19	**405**
20	**492**
21	**568**
22	**610**
23	**550**
24	**483**

Εκτελώντας τον αλγόριθμο Con-ANT για την εγκατάσταση που περιγράφηκε στην ενότητα αυτή και τις απαιτήσεις ισχύος μέσα σε ένα εικοσιτετράωρο προέκυψαν οι παρακάτω ισχείς εξόδου για καθεμία από τις 4 γεννήτριες, ανά ώρα λειτουργίας σε χρονικό ορίζοντα εικοσιτετραώρου:

# Γεννήτριας	1	2	3	4
# Ώρας 1	0,00	176,11	173,89	60,00
2	0,00	223,42	216,58	60,00
3	80,00	220,78	214,22	60,00
4	80,00	250,00	260,00	60,00
5	80,00	210,25	204,75	60,00
6	80,00	186,62	183,38	0,00
7	80,00	160,37	159,63	0,00
8	80,00	184,01	180,99	0,00
9	80,00	199,76	195,24	60,00
10	80,00	233,93	226,07	60,00
11	80,00	202,37	197,63	60,00
12	80,00	210,27	204,73	0,00

13	80,00	186,63	183,37	0,00
14	80,00	221,30	214,70	0,00
15	80,00	226,06	218,94	60,00
16	80,00	247,06	237,94	60,00
17	80,00	197,12	192,88	60,00
18	80,00	194,50	190,50	0,00
19	80,00	162,97	162,03	0,00
20	80,00	208,69	203,31	0,00
21	80,00	217,10	210,90	60,00
22	80,00	239,19	230,81	60,00
23	80,00	207,64	202,36	60,00
24	80,00	203,97	199,03	0,00

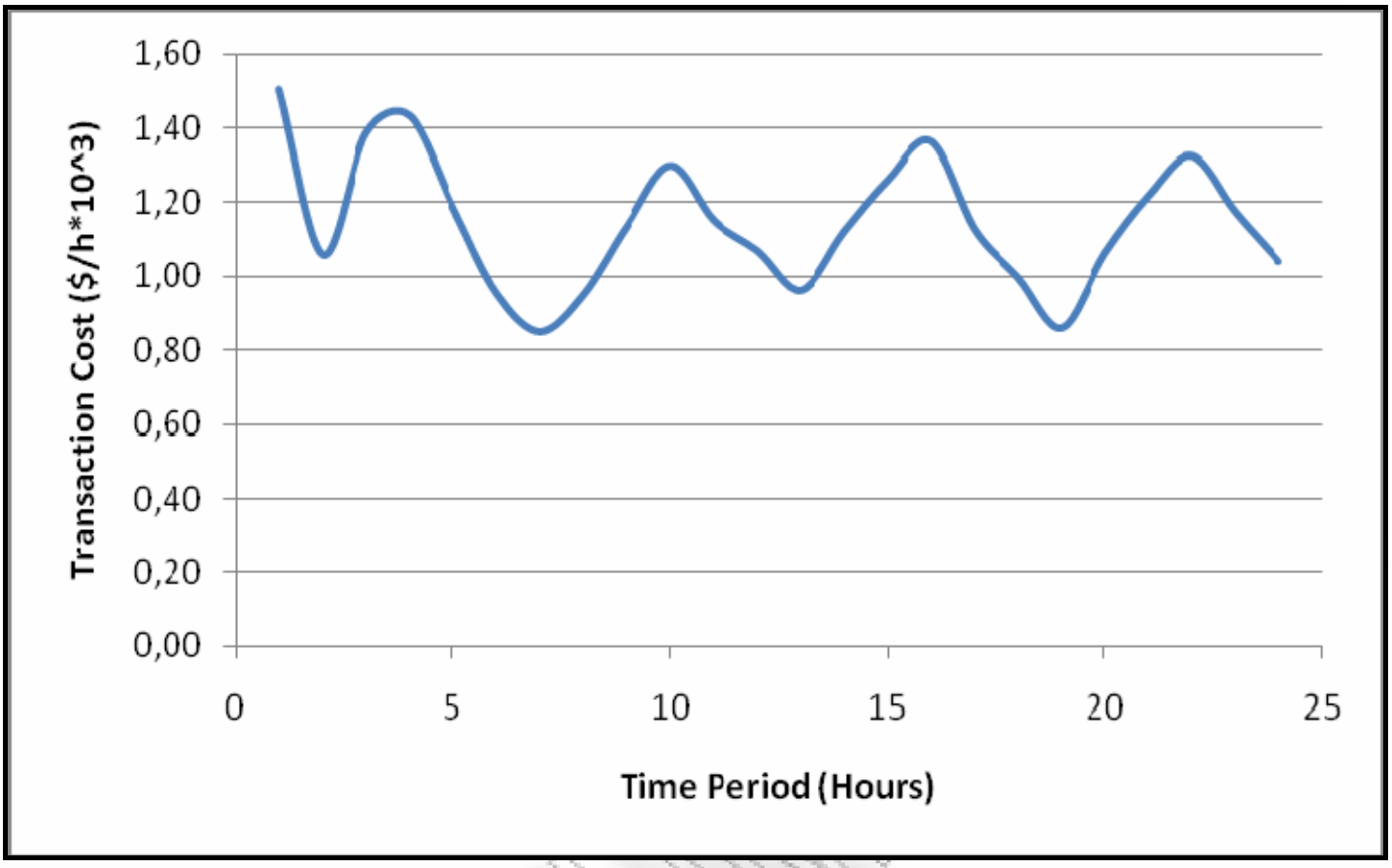
Ο χρόνος που χρειάστηκε για την επεξεργασία των δεδομένων αρχικοποίησης, την εκτέλεση του αλγορίθμου και την εξαγωγή αποτελεσμάτων ήταν κατά μέσο όρο 8,72 δευτερόλεπτα. Για την εξάλειψη του στατιστικού σφάλματος σημειώνεται ότι διεξήχθησαν 10 σειρές εκτελέσεων. Ο χρόνος αυτός επετεύχθη σε έναν υπολογιστή με επεξεργαστή Pentium 4, 3.2 GHz και 2GB RAM, ο οποίος έτρεχε λειτουργικό σύστημα Windows XP.

Ακολουθεί ο πίνακας συγκριτικής παρουσίασης των αποτελεσμάτων 3 διαφορετικών αλγορίθμων στο συγκεκριμένο πρόβλημα (Dynamic Programming – DP, Branch&Bound – B&B και Ant Colony) καθώς του Con-ANT που μελετήθηκε στην παρούσα εργασία.

Περίοδος	**Φορτίο**	**Dynamic Programming**		**Branch & Bound**		**Ant Colony**		**Con-ANT**	
		Κατάσταση	**Κόστος**	**Κατάσταση**	**Κόστος**	**Κατάσταση**	**Κόστος**	**Κατάσταση**	**Κόστος**
1	410	0111	0.8643	1111	1.2116	1111	1.2116	0111	1.504
2	500	0111	1.0796	1111	1.0575	1111	1.0575	0111	1.059
3	575	0111	1.2770	1111	1.2374	1111	1.2374	1111	1.387
4	650	1111	1.7834	1111	1.4334	1111	1.4334	1111	1.433
5	555	1111	1.1879	1111	1.1879	1111	1.1879	1111	1.187
6	450	1110	0.9632	1110	0.9632	1110	0.9632	1110	0.953
7	400	1110	0.8504	1110	0.8504	1110	0.8504	1110	0.850
8	445	1111	0.9353	1111	0.9353	1111	0.9353	1110	0.951
9	535	1111	1.1395	1111	1.1395	1111	1.1395	1111	1.139
10	600	1111	1.3009	1111	1.3009	1111	1.3009	1111	1.300
11	540	1111	1.1515	1111	1.1515	1111	1.1515	1111	1.151

12	495	1111	1.0460	1111	1.0460	1111	1.0460	1110	1.070
13	450	1111	0.9461	1111	0.9461	1111	0.9461	1110	0.963
14	516	1111	1.0946	1111	1.0946	1111	1.0946	1110	1.122
15	585	1111	1.2626	1111	1.2626	1111	1.2626	1111	1.262
16	625	1111	1.3660	1111	1.3660	1111	1.3660	1111	1.365
17	530	1111	1.1276	1111	1.1276	1111	1.1276	1111	1.127
18	465	1111	0.9788	1111	0.9788	1111	0.9788	1110	0.998
19	405	1111	0.8512	1111	0.8512	1111	0.8512	1110	0.861
20	492	1111	1.0392	1111	1.0392	1111	1.0392	1110	1.063
21	568	1111	1.2200	1111	1.2200	1111	1.2200	1111	1.219
22	610	1111	1.3267	1111	1.3267	1111	1.3267	1111	1.326
23	550	1111	1.1757	1111	1.1757	1111	1.1757	1111	1.175
24	483	1111	1.0188	1111	1.0188	1111	1.0188	1110	1.041
ΚΟΣΤΟΣ	$/ημέρα *10^4		2,698640		2.692194		2.692194		2.6979
ΧΡΟΝΟΣ	sec		2,04		77.28		3.42		8.71

Στην Εικόνα 17 απεικονίζεται η εξέλιξη του κόστους λειτουργίας των γεννητριών με τις ευρεθείσες από τον αλγόριθμο λύσεις συναρτήσει της αντίστοιχης χρονικής περιόδου (ώρας) μέσα στο εικοσιτετράωρο.



Εικόνα 17. Εξέλιξη Transaction Cost στις χρονικές περιόδους 24ώρου

5.2.4.2 Προσομοίωση Εγκατάστασης 10 Γεννητριών

Στο δεύτερο μέρος της συγκριτικής δοκιμής μελετήθηκε ένα σύστημα 10 γεννητριών με διαφορετικά χαρακτηριστικά, τα οποία δίνονται παρακάτω:

Ελάχιστη Ισχύς	Μέγιστη Ισχύς	Παράμετρος a	Παράμετρος b	Παράμετρος c	Απόδοση Γεννήτριας	Κλίση ισχύος	Χρόνος Εκκίνησης	Χρόνος Τερματισμού	Κόστος τερματισμού	Κόστος εκκίνησης (Z)	Κόστος Εκκίνησης (Κ)	Cold Start	Αρχική Τιμή
80	200	0,00148	1,2136	82	1	40	3	2	50	70	176	3	4
120	300	0,00289	1,2643	49	1	64	4	2	60	74	187	4	5
50	150	0,00135	1,3285	100	1	30	3	2	30	50	113	3	5
250	520	0,00127	1,3954	105	1	104	5	3	85	110	267	5	7
80	280	0,00261	1,35	72	1	56	4	2	52	72	180	3	5
50	150	0,00212	1,54	29	1	30	3	2	30	40	113	2	-3
30	120	0,00382	1,4	32	1	24	3	2	25	35	94	2	-3
30	110	0,00393	1,35	40	1	22	3	2	32	45	114	1	-3
20	80	0,00396	1,5	25	1	16	0	0	28	40	101	0	-1
20	60	0,0051	1,4	15	1	12	0	0	20	30	85	0	-1

Στη συγκεκριμένη εγκατάσταση έγινε μελέτη ένταξης μονάδων παραγωγής σε χρονικό ορίζοντα 24 ωρών. Οι απαιτήσεις σε ισχύ στις 24 αυτές ώρες δίνονται από την παρακάτω αλληλουχία τιμών απαιτούμενης ισχύος σε MW:

Αύξων Αριθμός Χρονικής Περιόδου (hr)	Απαιτούμενη Ισχύς(MW)
1	1170
2	1250
3	1380
4	1570
5	1690
6	1820
7	1910
8	1940
9	1990
10	1990
11	1970
12	1940
13	1910
14	1830
15	1870
16	1830
17	1690
18	1510
19	1420
20	1310
21	1260
22	1210
23	1250
24	1140

Εκτελώντας τον αλγόριθμο Con-ANT για την εγκατάσταση που περιγράφηκε στην ενότητα αυτή και τις απαιτήσεις ισχύος μέσα σε ένα εικοσιτετράωρο προέκυψαν οι παρακάτω ισχείς εξόδου για καθεμία από τις 10 γεννήτριες, ανά ώρα λειτουργίας σε χρονικό ορίζοντα εικοσιτετραώρου:

#Γεννήτριας	1	2	3	4	5	6	7	8	9	10
# Ώρας 1	200,00	211,15	0,00	428,85	0,00	150,00	120,00	0,00	0,00	60,00

2	200,00	211,14	0,00	428,86	0,00	150,00	120,00	0,00	80,00	60,00
3	200,00	205,03	150,00	414,97	0,00	150,00	120,00	0,00	80,00	60,00
4	200,00	290,00	150,00	520,00	0,00	150,00	120,00	0,00	80,00	60,00
5	200,00	227,94	150,00	466,43	235,63	150,00	120,00	0,00	80,00	60,00
6	200,00	232,33	150,00	476,83	240,83	150,00	120,00	110,00	80,00	60,00
7	200,00	254,57	150,00	520,00	265,43	150,00	120,00	110,00	80,00	60,00
8	200,00	270,00	150,00	520,00	280,00	150,00	120,00	110,00	80,00	60,00
9	200,00	300,00	150,00	520,00	280,00	150,00	120,00	110,00	80,00	60,00
10	200,00	300,00	150,00	520,00	280,00	150,00	120,00	110,00	80,00	60,00
11	200,00	300,00	150,00	520,00	280,00	150,00	120,00	110,00	80,00	60,00
12	200,00	270,00	150,00	520,00	280,00	150,00	120,00	110,00	80,00	60,00
13	200,00	254,55	150,00	520,00	265,45	150,00	120,00	110,00	80,00	60,00
14	200,00	234,66	150,00	482,13	243,21	150,00	120,00	110,00	80,00	60,00
15	200,00	243,84	150,00	502,80	253,36	150,00	120,00	110,00	80,00	60,00
16	200,00	234,60	150,00	482,08	243,32	150,00	120,00	110,00	80,00	60,00
17	200,00	220,82	150,00	451,08	228,10	150,00	120,00	110,00	0,00	60,00
18	200,00	179,83	150,00	358,14	182,37	150,00	119,66	110,00	0,00	60,00
19	199,98	179,17	150,00	330,39	151,33	139,56	102,07	107,49	0,00	60,00
20	200,00	150,76	150,00	283,87	143,11	131,84	93,16	97,27	0,00	60,00
21	200,00	143,25	150,00	270,70	136,93	122,88	85,97	90,27	0,00	60,00
22	200,00	145,80	150,00	280,12	145,15	133,12	0,00	95,82	0,00	60,00
23	200,00	152,20	150,00	293,34	151,46	141,99	0,00	101,01	0,00	60,00
24	199,98	145,46	0,00	273,88	142,87	130,77	91,24	95,81	0,00	60,00

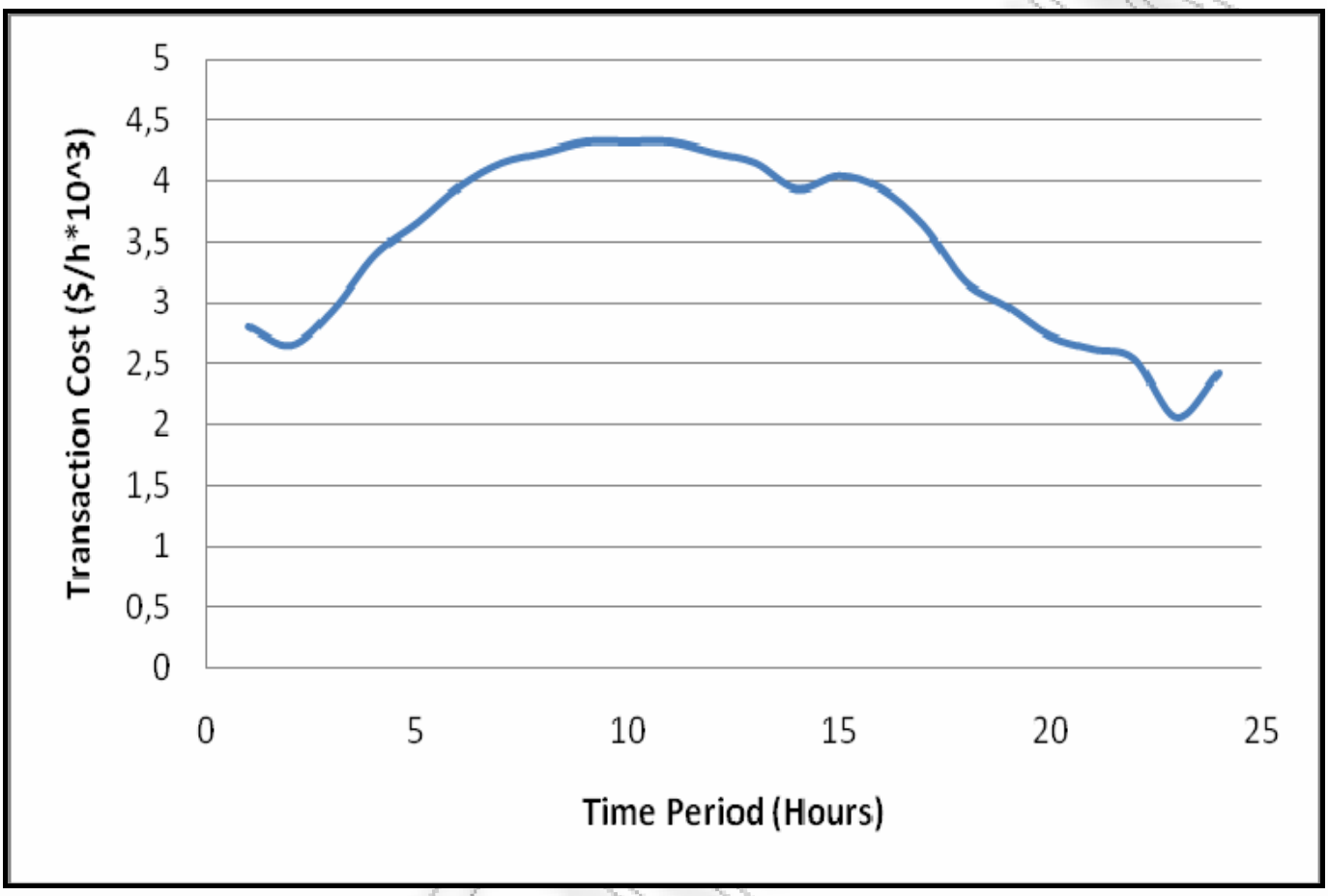
Ακολουθεί ο πίνακας συγκριτικής παρουσίασης των αποτελεσμάτων 3 διαφορετικών αλγορίθμων στο συγκεκριμένο πρόβλημα (Dynamic Programming – DP, Branch&Bound – B&B και Ant Colony) καθώς του Con-ANT που μελετήθηκε στην παρούσα εργασία.

Περίοδος	Φορτίο	Dynamic Programming		Branch & Bound		Ant Colony		Con-ANT	
		Κατάσταση	Κόστος	Κατάσταση	Κόστος	Κατάσταση	Κόστος	Κατάσταση	Κόστος

1	1170	1111101001	2.6381	1111111001	2.7258	1111111101	2.8496	1101011001	2.804
2	1250	1111111001	2.7191	1111111001	2.6061	1111111101	2.6062	1101011011	2.656
3	1380	1111111001	2.8894	1111111011	2.9817	1111111101	2.8870	1111011011	2.946
4	1570	1111111101	3.4260	1111111111	3.4098	1111111111	3.3968	1111011011	3.396
5	1690	1111111101	3.6074	1111111111	3.5787	1111111111	3.5787	1111111011	3.655
6	1820	1111111101	3.9488	1111111111	3.9064	1111111111	3.9064	1111111111	3.951
7	1910	1111111111	4.2474	1111111111	4.1464	1111111111	4.1464	1111111111	4.146
8	1940	1111111111	4.2297	1111111111	4.2297	1111111111	4.2297	1111111111	4.229
9	1990	1111111111	4.3782	1111111111	4.3782	1111111111	4.3782	1111111111	4.317
10	1990	1111111111	4.3782	1111111111	4.3782	1111111111	4.3782	1111111111	4.317
11	1970	1111111111	4.3171	1111111111	4.3171	1111111111	4.3171	1111111111	4.317
12	1940	1111111111	4.2297	1111111111	4.2297	1111111111	4.2297	1111111111	4.2297
13	1910	1111111111	4.1464	1111111111	4.1464	1111111111	4.1464	1111111111	4.1464
14	1830	1111111111	3.9325	1111111111	3.9325	1111111111	3.9325	1111111111	3.9325
15	1870	1111111111	4.0384	1111111111	4.0384	1111111111	4.0384	1111111111	4.0384
16	1830	1111111111	3.9325	1111111111	3.9325	1111111111	3.9325	1111111101	3.9325
17	1690	1111111111	3.5787	1111111111	3.5787	1111111111	3.5787	1111111101	3.635
18	1510	1111111111	3.1609	1111111111	3.1609	1111111111	3.1609	1111111101	3.170
19	1420	1111111111	2.9685	1101111111	2.9685	1101111111	2.9685	1111111101	2.970
20	1310	1111111111	2.7349	1101111111	2.7349	1101111111	2.7349	1111111101	2.731
21	1260	1111111111	2.6332	1101111111	2.6332	1101111111	2.6332	1111110101	2.626
22	1210	1111111111	2.5332	1101111111	2.5332	1101111111	2.5332	1111110101	2.546
23	1250	1111111111	2.6129	1101111111	2.6129	1101111111	2.6129	1101110101	2.062
24	1140	1101111111	2.3941	1101111111	2.3941	1101111111	2.3941		2.424
ΚΟΣΤΟΣ	$/ημέρα *10^4		8.3652		8.3476		8.3491		8.3709
ΧΡΟΝΟΣ	sec		13.05		383.56		112.54		11.75

Ο χρόνος που χρειάστηκε για την επεξεργασία των δεδομένων αρχικοποίησης, την εκτέλεση του αλγορίθμου και την εξαγωγή αποτελεσμάτων ήταν κατά μέσο όρο 11,75 δευτερόλεπτα. Για την εξάλειψη του στατιστικού σφάλματος σημειώνεται ότι διεξήχθησαν 10 σειρές εκτελέσεων. Ο χρόνος αυτός επετεύχθη σε έναν υπολογιστή με επεξεργαστή Pentium 4, 3.2 GHz και 2GB RAM, ο οποίος έτρεχε λειτουργικό σύστημα Windows XP.

Στην Εικόνα 18 απεικονίζεται η εξέλιξη του κόστους λειτουργίας των γεννητριών με τις ευρεθείσες από τον αλγόριθμο λύσεις συναρτήσει της αντίστοιχης χρονικής περιόδου (ώρας) μέσα στο εικοσιτετράωρο.



Εικόνα 18. Εξέλιξη Transaction Cost στις χρονικές περιόδους 24ώρου

5.2.5 Συμπεράσματα

Μελετώντας τα αποτελέσματα των πειραμάτων μπορούν να γίνουν αρκετές παρατηρήσεις και αντιστοίχως να εξαχθούν τα κατάλληλα συμπεράσματα. Αρχικά βλέπουμε ότι η μέθοδος βελτιστοποίησης με την χρήση κοινωνιών μυρμηγκιών έχει πολύ καλά αποτελέσματα και σε προβλήματα συνεχών μεταβλητών όπως αυτό που μελετήσαμε. Επίσης ο αλγόριθμος Con-ANT παρουσιάζει σύγκλιση σε μια «περιοχή λύσεων», η οποία σύγκλιση κρίνεται ικανοποιητική.

Σημειώνεται πως γίνεται αναφορά σε «περιοχή λύσεων» και όχι σε συγκεκριμένη λύση στην οποία συγκλίνει ο αλγόριθμος, καθώς ο αλγόριθμος αυτός εντάσσεται στους ευριστικούς αλγορίθμους, και παρότι μπορεί να φτάσει στην καλύτερη λύση που μπορεί να επιτύχει, η τυχαιότητα της αναζήτησης οδηγεί τον αλγόριθμο σε άλλες λύσεις υπο-βέλτιστες. Αυτό απεικονίζεται σε όλα τα διαγράμματα (Εικόνες 9 - 14) από τις υπάρχουσες διακυμάνσεις στις επαναλήψεις όπου έχει επιτευχθεί η σύγκλιση στη βέλτιστη δυνατή «περιοχή λύσεων».

Αναμενόμενο επίσης θεωρείται το γεγονός ότι το σύστημα αποτελούμενο από δέκα μονάδες παραγωγής (σενάριο Α), είναι αρκετά πιο πολύπλοκο από το σενάριο Β (πέντε μονάδες παραγωγής προς ένταξη), και ως εκ τούτου απαιτούνται περισσότερες επαναλήψεις

για την τελική σύγκλιση στην επιθυμητή περιοχή λύσεων. Αυτό είναι εμφανές συγκρίνοντας τα διαγράμματα των δύο σεναρίων που αφορούν τις ίδιες ομάδες μυρμηγκιών.

Τέλος, μια άλλη σημαντική παράμετρος για την αποτελεσματικότητα του αλγορίθμου Con-ANT είναι ο αριθμός των μυρμηγκιών που συμμετέχουν στην «περιήγηση» στον χώρο των πιθανών λύσεων, ή πιο συγκεκριμένα οι χρησιμοποιούμενες ομάδες μυρμηγκιών (παράμετρος εφαρμογής ***ant groups***). Όπως φαίνεται και στα διαγράμματα των Εικόνων 9 έως 14 όσο περισσότερα μυρμήγκια «περιηγούνται» στο χώρο των λύσεων, τόσο συντομότερη είναι σύγκλιση, αλλά και τόσο μικρότερες είναι οι διακυμάνσεις γύρω από την «περιοχή βέλτιστης λύσης» που έχει βρεθεί. Αυτό είναι κάτι αναμενόμενο, καθώς η αναζήτηση γίνεται πυκνότερη, με αποτέλεσμα να καταλήγει ο αλγόριθμος σε ένα ικανοποιητικό αποτέλεσμα πιο γρήγορα.

Στα πειράματα που μελετήθηκε η συμπεριφορά εγκαταστάσεων 4 και 10 μονάδων παραγωγής σε χρονικό ορίζοντα 24 ωρών παρατηρήθηκε η μεγαλύτερη καθυστέρηση στην εξαγωγή αποτελεσμάτων για την περίπτωση των 10 μονάδων παραγωγής, κάτι που φυσικά ήταν αναμενόμενο, δεδομένης της αυξημένης πολυπλοκότητας αναζήτησης λύσεων από τον αλγόριθμο Con-ANT. Σε κάθε περίπτωση καταλήξαμε σε λύσεις που καλύπτουν το απαιτούμενο φορτίο, εκτός από κάποιες συγκεκριμένες χρονικές περιόδους στην περίπτωση των 10 γεννητριών. Αυτό όμως ήταν apriori αδύνατο να συμβεί, καθώς η απαιτούμενη ισχύς ήταν μεγαλύτερη από το άθροισμα των μέγιστων παραγόμενων ισχύων από όλες τις διαθέσιμες μονάδες παραγωγής. Έτσι στα εν λόγω χρονικά διαστήματα οι γεννήτριες τέθηκαν σε λειτουργία μέγιστης παραγωγής ισχύος.

6. Σύνοψη - Συμπεράσματα

Στην εργασία αυτή μελετήθηκε το πρόβλημα της ένταξης μονάδων παραγωγής. Πιο συγκεκριμένα, και επειδή το πρόβλημα αυτό μπορεί να αντιμετωπιστεί είτε βραχυπρόθεσμα, είτε μακροπρόθεσμα, έμφαση δόθηκε στην ένταξη μονάδων παραγωγής σε χρονικό ορίζοντα εικοσιτετραώρου. Αρχικά μελετήθηκε γενικά το πρόβλημα, καθώς και οι προτεινόμενες στη βιβλιογραφία προσεγγίσεις για τη λύση του. Στη συνέχεια, επελέγη η οικογένεια αλγορίθμων αποικιών μυρμηγκιών για την επίλυση του προβλήματος, και πιο συγκεκριμένα ο αλγόριθμος Con-ANT, ο οποίος αφορά και προβλήματα συνεχών μεταβλητών.

Το επόμενο στάδιο του παρόντος πονήματος, ήταν η ανάλυση του προβλήματος για την εξαγωγή ενός μοντέλου, στο οποίο στηρίχθηκε και η τελική επίλυσή του. Εν συνεχεία σχεδιάστηκε και υλοποιήθηκε μια πρότυπη εφαρμογή η οποία περιλαμβάνει το μοντέλο του προβλήματος καθώς και τον αλγόριθμο Con-ANT ο οποίος εκτελείται για την εύρεση της λύσης με το μικρότερο κόστος λειτουργίας για δύο διαφορετικές εγκαταστάσεις: μία με 5 και μία με 10 μονάδες παραγωγής.

Στα πειράματα που εκτελέστηκαν για τις δύο εγκαταστάσεις μονάδων παραγωγής με την πρότυπη αυτή υλοποίηση χρησιμοποιήθηκαν 3 διαφορετικοί αριθμοί συμμετεχόντων μυρμηγκιών στην αναζήτηση λύσης. Τα αποτελέσματα των πειραμάτων κατέδειξαν την καταλληλότητα του αλγορίθμου Con-ANT για την επίλυση του ζητούμενου προβλήματος, καθώς παρατηρήθηκε σύγκλιση σε μια «περιοχή λύσεων». Επισημαίνεται ότι δεν υπήρξε απόλυτη σύγκλιση σε μια μοναδική λύση, κάτι που ήταν αναμενόμενο καθώς οι αλγόριθμοι αποικιών μυρμηγκιών ανήκουν στην οικογένεια των ευριστικών αλγορίθμων.

Ακρωνύμια

- ACS – Ant Colony System
- ACO – Ant Colony Optimization
- CACO – Continuous Ant Colony Optimization
- DACO – Discrete Ant Colony Optimization
- CIAC – Continuous Interacting Ant Colony
- LR – Lagrange Relaxation
- ALR – Augmented Lagrange Relaxation
- GA – Genetic Algorithms
- DP – Dynamic Programming
- UCP – Unit Commitment Problem
- PL – Priority List
- TSP – Travelling Salesman Problem
- API – Asynchronous Parallel Implementation
- EA – Evolutionary Algorithms
- ES – Evolutionary Strategies
- DE – Differential Evolution
- GP – Genetic Programming

Βιβλιογραφία

1. S. A. Kazarlis, A. G. Bakirtzis, and V. Petridis, “A genetic algorithm solution to the unit commitment problem,” IEEE Trans. Power Syst., vol. 11, pp. 83-92, Feb. 1996.
2. A. H. Mantawy, Y. L. Abded-Magid, and S. Z. Selim, “Integrating genetic algorithms, tabu search, and simulated annealing for the unit commitment problem,” IEEE Trans. Power Syst., vol. 14, no. 3, pp. 829-836, Aug. 1999.
3. A. Colorni, M. Dorigo et V. Maniezzo, Distributed Optimization by Ant Colonies, actes de la première conférence européenne sur la vie artificielle, Paris, France, Elsevier Publishing, 134-142, 1991.
4. M. Dorigo, Optimization, Learning and Natural Algorithms, PhD thesis, Politecnico di Milano, Italie, 1992.
5. S. Goss, S. Aron, J.-L. Deneubourg et J.-M. Pasteels, The self-organized exploratory pattern of the Argentine ant, Naturwissenschaften, volume 76, pages 579-581, 1989
6. J.-L. Deneubourg, S. Aron, S. Goss et J.-M. Pasteels, The self-organizing exploratory pattern of the Argentine ant, Journal of Insect Behavior, volume 3, page 159, 1990
7. C. Adami, “ Introduction to Artificial Life”, Published by TELOS, The Electronic Library of Science, Santa Clara, California, 1998, Pages 1-15.
8. M. Dorigo, and G. Di Caro, “ The Ant Colony Optimization meta-heuristic” In D. Corne, M. Dorigo, and F. Glover, editors, New Ideas in Optimization. McGraw-Hill, New York, NY, USA, 1999.
9. Krzysztof Socha, “ ACO for Continuous and Mixed-Variable Optimization “, IRIDIA, Universite Libre de Bruxelles, CP 194/6, Av. Franklin D. Roosevelt 50, 1050 Brussels, Belgium
10. R. Chelouah, and P. Siarry, “ Enhanced Continuous Tabu Search” , In S. Voss, S. Martello, I. H. Osman, and C. Roucairol, editors, Meta-Heuristics Advances and Trends in Local Search Paradigms for Optimization, Kluwer Academic Publishers, Boston, MA, USA, 1999, chapter 4, pages 49-61.
11. R. K. Kincaid, S. Grith, R. Sykes, and J. Sobieszczanski-Sobieski, “ A Bell-Curve Genetic Algorithm for Mixed Continuous and Discrete Optimization Problems” , In Proceedings of 43rd AIAA Structures, Structural Dynamics,and Materials Conference. AIAA, Denver, CO, USA, 2002.
12. M. Mathur, S. B. Karale, S. Priye, V. K. Jyaraman, and B. D. Kulkarni, “ Ant Colony Approach to Continuous Function Optimization” , Ind. Eng. Chem. Res., 39, 2000, pages 3814- 3822.

13. B. Bullnheimer, R. F. Hartl, and C. Strauss, " Applying the Ant System to the Vehicle Routing Problem, Metaheuristic: Advances and Trends in Local Search Paradigms for Optimization Voss, S., et al(Eds), Kluwer, 1999.

14. D. Costa, and A. Hertz, " Ants Can Colour Graphs" , Journal of the Operational Research Society, 1997, 48: pages 295-305.

15. P. Forsyth, and A. Wren, " An Ant System for Bus Driver Scheduling", Proc. Of the 7th Int. Workshop on Computer-Aided Scheuling of Public Transport,1997.

16. M. Dorigo, G. D. Caro, and L. M. Gambardella, "Ant Algorithm for Discrete Optimization" , Artificial Life 5 (3), 1999, pages 137-172.

17. M. Dorigo, E. Bonabeau, and G. Theraulaz, " Ant Algorithm and Stigmergy" , Future Generation Computer Systems 16, 2000, pages 851-871.

18. Mohammad Reza Jalali, Optimum Design and Operation of Hydrosystems by Ant Colony Optimization Algorithms; A New Meta-Heuristic Approach, PhD Thesis, Iran University of Science and Technology (IUST), Collage of Civil Engineering, 2005.

19. H. R. Maier, A. R. Simpson, A. C. Zecchin, W. K. Foong, K. Y. Phang, H.Y. Seah, and C. L. Tan, "Ant Colony Optimization for Design of Water Distribution Systems" J. Water Resour. Plng. and Mgmt., Volume 129, Issue 3, May-June 2003, pages 200-209.

20. P. G. Lowery, "Generating Unit Commitment by Dynamic Programming", IEEE Transactions on Power Apparatus and Systems Vol. PAS-85, No. 5, pp. 422-426, 1966.

21. W. J. Hobbs, G. Hermon, S. Warner, G. B. Sheble, "An Enhanced Dynamic Programming Approach for Unit Commitment", IEEE Transactions on Power Systems, Vol. 3, No. 3, pp. 1201-1205, 1988.

22. C. Cheng, C. W. Liu, and C. C. Liu, "Unit Commitment by Lagrangian Relaxation and Genetic Algorithms", IEEE Transactions on Power Systems, Vol. 15, No. 2, pp. 707-714, 2000.

23. A. Borghetti. A. Frangioni, F. Lacalandrs, C. A. Nucci, "Lagrangian Heuristics based on Disaggregated Bundle Methods for Hydrothermal Unit Commitment", IEEE Transactions on Power Systems, Vol. 18, No. 2, pp. 974, 2003.

24. Z. Liu, N. Li, C. Zhang, "Unit Commitment Scheduling Using a Hybrid ANN and Lagrangian Relaxation Method", in Proceedings of International Conference on Multimedia and Ubiquitous Engineering, pp. 481-484, 2008.

25. K. Tokoro, Y. Masuda, H. Nishini, "An Efficient Unit Commitment Schedule by Combining of Continuous Relaxation Method and Genetic Algorithm", IEEJ Transactions on Electronics, Information and Systems, Vol. 127. No. 9, pp. 1452-1459, 2007.

26. C. L. Chen, S. C. Wang, “Branch-and-Bound Scheduling for Thermal Generating Units”, IEEE Transactions on Energy Conversion, Vol. 8, No. 2, pp. 184-189, 1993.

27. G. Cote, M. A. Laughton, “Decomposition Techniques in Power System Planning: the Benders Partitioning Method”, Electrical Power and Energy Systems, Vol. 1, No. 1, pp. 57-64, 1979.

28. F. Zhuang, F. D. Galiana, “Unit Commitment by Simulated Annealing”, IEEE Transactions on Power Systems, Vol. 5, No. 1, pp. 311-317, 1990.

29. A. H. Mantawy, Y. L. Abdel-Magid, S. Z. Selim, “Unit Commitment by Tabu Search”, IEE Proceedings - Generation, Transmission and Distribution, Vol. 145, No. 1, pp. 56-64, 1998.

30. S. A. Kazarlis, A. G. Bakirtzis, V. Petridis, “A Genetic Algorithm Solution to the Unit Commitment Problem”, IEEE Transactions on Power Systems, Vol. 11, No. 1, pp. 83-92, 1996.

31. A. Rudolf, R. Bayrleithner, “A Genetic Algorithm for Solving the Unit Commitment Problem of a Hydro- Thermal Power System”, IEEE Transactions on Power Systems, Vol. 14, No. 4, pp. 1460-1468, 1999.

32. J. Valenzuela, A. E. Smith, “A Seeded Memetic Algorithm for Large Unit Commitment Problems”, Journal of Heuristics, Vol. 8, pp. 173-195, 2002.

33. G. Dudek, “Unit Commitment by Genetic Algorithm with Specialized Search Operators”, Electric Power Syst. Res., Vol.72, No. 3, pp. 299-308, Elsevier, 2004.

34. J. Maturana, M. C. Riff, “Solving the Short-Term Electrical Generation Scheduling Problem by an Adaptive Evolutionary Approach”, European J. of Operational Research, Vol. 179, pp. 677-691, 2007.

35. Y. M. Chen, W. S. Wang, “Fast Solution Technique for Unit Commitment by Particle Swarm Optimisation and Genetic Algorithm”, International Journal of Energy Technology and Policy, Vol. 5, No. 4, pp. 440-456, 2007.

36. S. Patra, S. K. Goswami, B. Goswami, “Differential Evolution Algorithm for Solving Unit Commitment with Ramp Constraints”, Electric Power Components and Systems, Vol. 36, No. 8, pp. 771-787, 2008.

37. N. P. Padhy, “Unit Commitment -A Bibliographical Survey”, IEEE Transactions on Power Systems, Vol. 19, No. 2, pp. 1196-2005, 2004.

38. S. Salam, “Unit Commitment Solution Methods”, Proceedings of World Academy of Science, Engineering and Technology, Vol. 26, pp. 600-605, 2007.

39. I. J. Raglend, N. P. Padhy, “Comparison of Practical Unit Commitment Solutions”, Electric Power Components and Systems, Vol. 36, No. 8, pp. 844-863, 2008.

40. A. E. Eiben, J. E. Smith, Introduction to Evolutionary Computing, Springer Verlag, 2003.

41. J. H. Holland, Adaptation in Natural and Artificial Systems, University of Michigan Press, 1975.

42. I. Rechenberg, Evolutionstrategie: Optimierung Technischer Systeme nach Prinzipien der Biologischen Evolution, Fomman Holzboog Verlag, 1973.

43. L. J. Fogel, A. J. Owens, M. J. Walsh, Artificial Intelligence through Simulated Evolution, Wiley, New York, 1966.

44. K. V. Price, R. M. Storn, J. A. Lampinen, Differential Evolution: A Practical Approach to Global Optimization, Springer, 2005.

45. A. Kele¸s, A. S¸. Etaner-Uyar, B. T¨urkay, "A Differential Evolution Approach for the Unit Commitment Problem", ELECO 2007: 5th International Conference on Electrical and Electronics Engineering, Bursa, pp. 132-136, 2007.

46. S. Damousis, A. G. Bakirtzis, "Genetic Algorithm Solution to the Economic Dispatch Problem", IEE Proceedings-C, Vol. 141, No. 4, pp. 377-382, 1996.

47. H. G. Beyer, H. P. Schwefel, "Evolution Strategies, A Comprehensive Introduction", Natural Computing, Vol. 1, pp. 3-52, Kluwer, 2002.

48. G. Pampara, A. P. Engelbrecht, N. Franken, "Binary Differential Evolution", IEEE Congress on Evolutionary Computation, pp. 1873-1879, 2006.

49. T. Gong, A. Tuson, "Differential Evolution for Binary Encoding", 11th Online World Conference on Soft Computing in Industrial Applications, 2006.

50. R. W. Hamming, "Error-detecting and Error-correcting Codes", Bell Systems Technical Journal, Vol. 29, pp. 147-160, 1950.

51. N. Krasnogor, J. Smith, "A Tutorial for Competent Memetic Algorithms: Model, Taxonomy and Design Issues", IEEE Transactions on Evolutionary Computation, Vol. 9, No. 5, pp. 474-488, 2005.

52. A. J.Wood, B. F. Wollenberg, Power Generation, Operation, and Control, Wiley, New York, 1996.

53. Sheble, G.B., G.N. Fahd, "Unit Commitment Literature Synopsis," IEEE Transactions on Power Systems, Vol.9, No.1, (February 1994), pp. 128-135.

54. Bertsekas, D.P., Nonlinear Programming, (Athena Scientific, 1995), pp.525.

55. Avriel, M., and B. Golany, eds., Mathematical Programming for Industrial Engineers, (Marcel Dekker, 1996), pp. 231-232.

56. Muckstadt, J.A., and S. Koenig, “An Application of Lagrangian Relaxation to Scheduling in Power-Generation Systems,” Operations Research, Vol. 25, No. 3, (May-June 1977), pp. 387-403.

57. Merlin, A., and P. Sandrin, “A New Method for Unit Commitment at Electricite de France,” IEEE Transactions on Power Apparatus and Systems, Vol. PAS-102, No. 5, (May 1983), pp. 1218-1225.

58. Bard, J.F., “Short-Term Scheduling of Thermal-Electric Generators Using Lagrangian Relaxation,” Operations Research, Vol. 36, No. 5, (September-October 1988), pp. 756-766.

59. Zhuang, F., and F.D. Galiana, “Towards a More Rigorous and Practical Unit Commitment by Lagrangian Relaxation,” IEEE Transactions on Power Systems, Vol. 3, No. 2, (May 1988), pp. 763-773.

60. Aoki, K., M. Itoh, T. Satoh, K. Nara, M. Kanezashi, “Optimal Long-Term Unit Commitment in Large Scale Systems Including Fuel Constrained Thermal and Pumped Storage Hydro,” IEEE Transactions on Power Systems, Vol. 4, No. 3, (August 1989), pp. 1065- 1073.

61. Batut, J., and A. Renaud, “Daily Generation Scheduling Optimization with Transmission Constraints: A New Class of Algorithms,” IEEE Transactions on Power Systems, Vol 7, No. 3, (August 1992), pp. 982-989.

62. Beltran, C., and F.J. Heredia, “Unit Commitment by Augmented Lagrangian Relaxation: Testing Two Decomposition Approaches,” Journal of Optimization Theory and Applications, 112(2), (Feb 2002), pp. 295-314.

63. Li, C., R.B. Johnson, A.J. Svoboda, C. Tseng, E. Hsu, “A Robust Unit Commitment Algorithm for Hydro-Thermal Optimization,” IEEE Transactions on Power Systems, Vol. 13, No. 3, (August 1998), pp. 1051-1056.

64. IEE Proceedings of Gener. Transm. Distrib. September 1994, “Thermal Unit Commitment Using Genetic Algorithms, by D. Dasgupta, and D.R. McGregor,” Vol 141, No. 5, pp. 459-465.

65. Orero, S.O., and M.R. Irving, “Combination of the Genetic Algorithm and Lagrangian Relaxation Decomposition Techniques for the Generation Unit Commitment Problem,” Electric Power Systems Research, 43, (1997), pp. 149-156.

66. Cheng, C., C. Liu, C. Liu, “Unit Commitment by Lagrangian Relaxation and Genetic Algorithms,” IEEE Transactions on Power Systems, Vol. 15, No. 2, (May 2000), pp 707- 714.

67. Duo, H., H. Sasaki, T. Nagata, H. Fujita, “Solution For Unit Commitment Using Lagrangian Relaxation Combined with Evolutionary Programming,” Electric Power Systems Research, 51, (1999), pp. 71-77.

68. Shaw, J.J., “A Direct Method for Security-Constrained Unit Commitment,” IEEE Transactions on Power Systems, Vol.10, No.3, (August 1995), pp. 1329-1342.

69. Xia, Q., Y.H. Song, B. Shang, C. Kang, N. Xiang, “Effective Decomposition and Coordination Algorithms for Unit Commitment and Economic Dispatch with Security Constraints,” Electric Power Systems Research, 53, (2000), pp 39-45.

70. Wang, S.J., S.M. Shahidehpour, D.S. Kirschen, S. Mokhtari, G.D. Irisarri, “Short- Term Generation Scheduling with Transmission and Environmental Constraints Using an Augmented Lagrangian Relaxation,” IEEE Transactions on Power Systems, Vol. 10, No 3, (August 1995), pp. 1294-1301.

71. C.F. Pang and H. C. Chen, “Optimal Short Term Thermal Unit Commitment”, IEEE Trans. On Power Apparatus and System, Vol PAS-95, No. 4, pp. 1336-1342, July-August 1976.

72. S. K. Tong, and S. M. Shahidehpour, ” Combination of Lagrangian Relaxation and Linear Programming Approaches for Fuel Constraints Unit Commitment Problems”, IEE Proc., Vol. 136, Pt. C, No. 3, May 1989.

73. S. K.Tong, and S. M. Shahidehpour, ” An Innovative Approach to Generation Scheduling in Large Scale Hydrothermal Power Systems with Fuel Constraints Units”, Proceedings of PICA ’89, pp. 188-196. Also accepted for publication, IEEE Trans. on Power Systems, 1990.

74. S. K. Tong, and S. M. Shahidehpour, ” Hydrothermal Unit Commitment with Probabilistic Constraints Using Scgmentation Method”, IEEE Trans. on Power Systems, Vol. 5. No. 1, pp. 276-282, Feb. 1990.

75. Z. Ouyang, and S. M. Shahidehpour, ” Short Term Unit Commitment Expert System”, Electric Power Systems Research, Vol. 19, No. 3, 1990.

76. S. Mokhtari, J. Singh, and B. Wollenberg, ” A Unit Commitment Expert System”, Proceedings of PICA ’87, pp. 400-405.

77. T. Terano, and H. Isoda, ” An Expert System for Power Generation Scheduling in Multi-Chain Hydro Power Stations”, Proceedings of IEEE AI ’88, pp. 551-556.

78. E. Bonabeau, M. Dorigo, G. Theraulaz, Swarm Intelligence, From Natural to Artificial Systems, Oxford University Press, Oxford, 1999.

79. S. Camazine, J.L. Deneubourg, N. Franks, J. Sneyd, G. Theraulaz, E. Bonabeau, Self-organization in Biological Systems, Princeton University Press, Princeton, NJ, 2000.

80. R. Chelouah, P. Siarry, A continuous genetic algorithm designed for the global optimization, J. Heuristics 6 (2000) 191– 213.

81. R. Chelouah, P. Siarry, Tabu search applied to global optimization, Eur. J. Oper. Res. 123 (2000) 256–270.

82. A. Colorni, M. Dorigo, V. Maniezzo, Distributed optimization by ant colonies, in: F. Varela, P. Bourgine (Eds.), Proceedings of the First European Conference on Artificial Life, ECAL'91, Elsevier, Paris, France, 1992, pp. 34–142.

83. J. Dréo, Modélisation de la mobilisation chez les fourmis, Mémoire de DEA, Université Paris7 & Université Libre de Bruxelles, 2001.

84. R. Eberhart, J. Kennedy, Y. Shi, Swarm Intelligence, Evolutionary Computation, Morgan Kaufmann, Los Altos, CA, 2001.

85. D.B. Fogel, Evolutionary Computation: The Fossil Record, IEEE Press, 1998, Chapter 7, pp. 481–484.

86. F. Glover, M. Laguna, Tabu Search, Kluwer Academic Publishers, Dordrecht, 1997.

87. C. Hewitt, Viewing control structures as patterns of passing messages, J. Artif. Intell. 8 (3) (1977) 323–364.

88. J. Holland, Outline for logical theory of adaptive systems, J. ACM 9 (3) (1962) 297–314.

89. R. Storn, K. Price, Differential evolution—a simple and efficient adaptive scheme for global optimization over continuous spaces, Technical Report TR95-012, International Computer Science Institute, Berkeley, CA.

90. P. Siarry, G. Berthiau, F. Durbin, and J. Haussy, " Enhanced Simulated Annealing for Globally Minimizing Functions of Many Continuous Variables", ACM Transactions on Mathematical Software, 1997, 23(2):209-228.

91. G. Bilchev, and I. C. Parmee. " The Ant Colony Metaphor for Searching Continuous Design Spaces" In T. C. Fogarty, editor, Proceedings of the AISB Workshop on Evolutionary Computation, volume 993 of LNCS, Springer-Verlag, Berlin, Germany, 1995, pages 25-39.

92. M. Wodrich and G. Bilchev, "Cooperative distributed search: the ant's way. Control & Cybernetics" , 1997, (3): pages 413-446.

93. J. Dreo and P. Siarry, " A New Ant Colony Algorithm Using the Heterarchical Concept Aimed at Optimization of Multiminima Continuous Functions" , In M. Dorigo, G. D. Caro, and M. Sampels, editors, Proceedings of the Third International Workshop on Ant Algorithms (ANTS'2002), volume 2463 of LNCS, Springer-Verlag, Berlin, Germany, 2002, pages 216-221.

94. C. Ling, S. Jie, O. Ling, and C. Hongjian, " A Method for Solving Optimization Problems in Continuous Space Using Ant Colony Algorithm" , Lecture Notes in Computer Science 2463, pages 288-289, 2002.

95. L. Y. Jun, W. T. Jun, " An Adaptive Ant Colony System Algorithm for Continuous-space Optimization Problems", Journal of Zhejiang University Science 4 (1), pages 40-46, 2003.

96. N. Monmarche, G. Venturini, and M.Slimane, " On how Pachycondyla apicalis ants suggest a new search algorithm", Future Generation Computer Systems, 2000, 16:937-946.

97. Visual Basic .NET: http://msdn.microsoft.com/en-us/vbasic/default.aspx

98. SharpDevelop IDE: http://www.icsharpcode.net/OpenSource/SD/

99. Sishaj P. Simon, Narayana Prasad Padhy and R.S. Anand, "An ant colony system approach for unit commitment problem", International Journal of Electrical Power & Energy Systems, Volume 28, Issue 5, June 2006, Pages 315-323

100. Jorge Valenzuela and Alice E. Smith, "A Seeded Memetic Algorithm for Large Unit Commitment Problems", Journal of Heuristics, Volume 8, Number 2 / March, 2002, Pages 173-195

Παράρτημα Α. Κώδικας Υλοποίησης Con-ANT

```
Imports System.IO

Public Class ConANTCore

      Dim endurance As Double = ParametersModule.endurance
      Dim evapFactor As Double = ParametersModule.evap_factor
      Dim pherlay_pervisit As Double = ParametersModule.pher_per_visit
      Dim generatorsData(,) As Double = ParametersModule.geninput
      Dim iterations As Integer = ParametersModule.iterations
      Dim ant_groups As Integer = ParametersModule.numofantgrps
      Dim ants_num As Integer = ParametersModule.numofants
      Dim generators As Integer = ParametersModule.numofgens
      Dim powerDemandPerPeriod() As Double = ParametersModule.reqpower
      Dim prop_factor As Double = ParametersModule.prop_factor

      Dim alpha As Integer = 1

      Dim conantUtil As New ConANTUtil()

      Dim randClass As New Random()

      Public Function run() As String
            Dim previousPeriodSolution(generators-1,1) As Double
            For initPrev=0 To generators-1
                  previousPeriodSolution(initPrev, 0)=0
                  previousPeriodSolution(initPrev, 1)=0
            Next
```

```
            Dim periodSolutionStore(powerDemandPerPeriod.GetUpperBound(0),generators-1,1)
As Double

            Dim periodCostStore(powerDemandPerPeriod.GetUpperBound(0)) As Double


            Dim perioddemand As Double = 0

            Dim periodsolution(,) As Double

            Dim periodcost As Double = 0

            Dim totalcost As Double = 0


            Dim ini_input(,) As Double = generatorsData.Clone


            Dim start As Date = Date.Now


            For i=0 To powerDemandPerPeriod.GetUpperBound(0)


                  If (i <> 0 ) Then

                        ini_input = conantUtil.AdaptConstraints(generatorsData,
previousPeriodSolution)

                  End If

                  perioddemand = powerDemandPerPeriod(i)


                  periodsolution = runPeriod(perioddemand, previousPeriodSolution,
ini_input)


                  ''debug ONLY

                  ''periodsolution = conantUtil.FindGlobalSolution(ini_input, perioddemand)

                  ''Dim asolution(,) As Double = {{74,74},{0,0},{280,280},{56,56}}

                  ''periodsolution = conantUtil.FindLocalSolution(0, 0, perioddemand,
ini_input, asolution, 4, 2)


                  Debug.WriteLine("Finished period with demand: " +
perioddemand.ToString())
```

```
                For solit=0 To generators-1
                      periodSolutionStore(i, solit, 0) = periodsolution(solit, 0)
                      periodSolutionStore(i, solit, 1) = periodsolution(solit, 1)
                Next

                periodcost = conantUtil.CalculateSolutionCost(geninput, periodsolution,
previousPeriodSolution)
                periodCostStore(i) = periodcost

                totalcost += periodcost

                previousPeriodSolution = periodsolution.Clone
            Next



            Dim finish As Date = Date.Now
            Dim duration As TimeSpan = finish.Subtract(start)

            'PREPARE RESULTS
            Dim resultText As String = ""
            For j=0 To powerDemandPerPeriod.GetUpperBound(0)
                resultText += "Power outputs for generators at period #" + j.ToString() +
System.Environment.NewLine
                For k=0 To generators-1
                    resultText += "Power of Generator No."+(k+1).ToString()+" =
"+periodSolutionStore(j, k, 0).ToString() + System.Environment.NewLine
                Next
                resultText += "Total solution cost is " + periodCostStore(j).ToString()
+ System.Environment.NewLine
                resultText += System.Environment.NewLine
            Next
```

```
            resultText += System.Environment.NewLine

            resultText += "All periods' cost = " + totalcost.ToString() +
System.Environment.NewLine + "Time to find solution: " + duration.ToString()



            return resultText


      End Function


      Public Function runPeriod(powerdemand As Double, previousPeriodSolution(,) As Double,
ini_input(,) As Double) As Double(,)

            'param where the final solution will be stored

            Dim final_sol(,) As Double

            Dim solution_pergen_perit(iterations-1,generators-1,1) As Double

            Dim ant_pergen_perit(ant_groups-1,generators-1) As Double

            Dim total_sols_cost(iterations-1) As Double

            Dim best_iteration As Integer = 0


            'GLOBAL SEARCH SPECIFIC

            Dim denom_sum As Double = 0

            Dim denominat(ant_groups-1) As Double

            Dim pheromone_perant(ant_groups-1) As Double


            Dim best_value As Double = 999999999999999999


            'randomly find an acceptable solution to start the Con-ANT search

            'Dim init_rand_sol(,) As Double =
conantUtil.RandomizeGeneratorOutput(ini_input, powerdemand)

            'debug

            Dim init_rand_sol(,) As Double = conantUtil.FindGlobalSolution(ini_input,
powerdemand)



            total_sols_cost(0) = conantUtil.CalculateSolutionCost(ini_input, init_rand_sol,
```

```
previousPeriodSolution)
            best_value = total_sols_cost(0)


            'add the solution as the first loop solution
            For gg=0 To generators-1
                  solution_pergen_perit(0,gg,0)= init_rand_sol(gg,0) 'first random solution
no loss
                  solution_pergen_perit(0,gg,1)= init_rand_sol(gg,1) 'first random solution
with loss
            Next
            'temporarily set this randomized solution as global solution
            final_sol = init_rand_sol.Clone


            'Dim minmax_noloss(generators-1,1) As Double
            'Dim minmax_withloss(generators-1,1) As Double
            'For i=0 To generators-1
            '     minmax_noloss(i,0) = ini_input(i,0)
            '     minmax_noloss(i,1) = ini_input(i,1)
            '     minmax_withloss(i,0) = ini_input(i,0)*ini_input(i,5)
            '     minmax_withloss(i,1) = ini_input(i,1)*ini_input(i,5)
            'Next


            Dim rad = conantUtil.CalculateSearchRadius(ini_input, generators)
            Dim local_groups As Integer = ant_groups\2


            For iter = 1 To iterations-1
                  Dim antiter_sol(ant_groups-1,generators-1,1) As Double
                  Dim antgroup_sumcost(ant_groups-1) As Double
                  Dim iteration_bestsolution_cost As Double = 999999999999999999
                  Dim iteration_best_antind As Integer = 0
```

```
                    'LOCAL SEARCH

                    For locant=0 To local_groups-1

                          'Note that we pass the last found final_sol as selected solution!

                          'iter and locant are passed for debug purposes

                          Dim local_ini_adaptation(,) As Double =
conantUtil.AdaptConstraints(ini_input, final_sol)

                          Dim local_newsol(,) As Double = conantUtil.FindLocalSolution(iter,
locant, powerdemand, local_ini_adaptation, final_sol, generators, rad)



                          For k=0 To generators-1

                                antiter_sol(locant,k,0)=local_newsol(k,0)

                                antiter_sol(locant,k,1)=local_newsol(k,1)

                          Next

                          Dim cost As Double = conantUtil.CalculateSolutionCost(ini_input,
local_newsol, previousPeriodSolution)



                          antgroup_sumcost(locant) += cost

                          If cost < iteration_bestsolution_cost Then

                                iteration_bestsolution_cost = cost

                                iteration_best_antind = locant

                          End If

                    Next

                    If iter > 1 Then

                          'GLOBAL SEARCH


                          'RE-INITIALIZE

                          denom_sum = 0


                          For globant = local_groups To ant_groups-1


                                'LETS COMMENT THAT AND TRY (debug) FOR THE
```

```
RandomizeGeneratorOutput METHOD

                            'Dim rand_sol() As Double =
conantUtil.FindGlobalRandomSolution(generators, minmax_noloss, powerdemand)

                            'debug

                            Dim rglob_sol(,) As Double =
conantUtil.FindGlobalSolution(ini_input, powerdemand)



                            'Calculate solution with respective powerloss

                            'Dim rglob_sol(generators,2) As Double

                            For m=0 To generators-1

                            '     rglob_sol(m,0) = rand_sol(m)

                            '     rglob_sol(m,1) = rand_sol(m)*ini_input(m,5)

                                  antiter_sol(globant,m,0) = rglob_sol(m,0)

                                  antiter_sol(globant,m,1) = rglob_sol(m,1)

                            Next

                            Dim gcost As Double =
conantUtil.CalculateSolutionCost(ini_input, rglob_sol, previousPeriodSolution)

                            antgroup_sumcost(globant) += gcost

                            If gcost < iteration_bestsolution_cost Then

                                  iteration_bestsolution_cost = gcost

                                  iteration_best_antind = globant

                            End If


                            pheromone_perant(globant) =
conantUtil.CalculatePheromon(generators , 0 ,iter , solution_pergen_perit , endurance ,
evapFactor , rglob_sol, 2*rad , pher_per_visit)



      denominat(globant)=antgroup_sumcost(globant)*pheromone_perant(globant)^alpha

                        denom_sum = denom_sum + denominat(globant)

                        Next
```

```
                        'Evaluate global solution
                        Dim minimum_global_sol As Double = 99999999999999999
                        Dim min_ant_ind As Integer = 0
                        Dim loopsol As Double = 0

                        If denom_sum > 0 Then 'if pheromone has been layed
                              For gant = local_groups To ant_groups-1
                                    ' evaluate with formula
                                    loopsol =
antgroup_sumcost(gant)*pheromone_perant(gant)^alpha / denom_sum
                                    If loopsol < minimum_global_sol Then
                                          minimum_global_sol = loopsol
                                          min_ant_ind = gant
                                    End If
                              Next
                        Else
                              For gant = local_groups To ant_groups-1
                                    ' assign value
                                    loopsol = antgroup_sumcost(gant)
                                    If loopsol < minimum_global_sol Then
                                          minimum_global_sol = loopsol
                                          min_ant_ind = gant
                                    End If
                              Next
                        End If

                        ' randomize a possible false path selection by the ant
                        If randClass.NextDouble()<prop_factor
                        loopsol = min_ant_ind
                        While (min_ant_ind=loopsol)
                              'find ANOTHER ant group index and assign it as the "minimum
```

```
cost" one
                                '(obviously wrong, but that was a mistake made by chance)
                                min_ant_ind = randClass.Next(local_groups, ant_groups-1)
                            End While
                        End If

                        If antgroup_sumcost(min_ant_ind)<iteration_bestsolution_cost
                        iteration_bestsolution_cost = antgroup_sumcost(min_ant_ind)
                        iteration_best_antind = min_ant_ind
                        End If

                    End If ' end of global search

                    ' FINAL EVALUATION
                    total_sols_cost(iter) = iteration_bestsolution_cost
                    For gen=0 To generators-1
                        solution_pergen_perit(iter,gen,0) =
antiter_sol(iteration_best_antind, gen, 0)
                        solution_pergen_perit(iter,gen,1) =
antiter_sol(iteration_best_antind, gen, 1)
                    Next

                    If iteration_bestsolution_cost < best_value and
iteration_bestsolution_cost>0 Then
                        best_value = iteration_bestsolution_cost
                        best_iteration = iter
                        For mt=0 To generators-1
                            final_sol(mt,0) = solution_pergen_perit(best_iteration,mt,0)
                        final_sol(mt,1) = solution_pergen_perit(best_iteration,mt,1)
                    Next
                    End If
```

```
            Next


            Try
                  Dim objReader As StreamWriter = New
StreamWriter(Environment.CurrentDirectory()+"/genset_costs_per_iteration_bestsolution.txt")
                  objReader.WriteLine("Lowest cost: " + best_value.ToString)
                  For logsol=0 To generators-1
                        objReader.WriteLine("Power of generator no" + logsol.ToString + " =
" + final_sol(logsol,0).ToString + " and with loss = " + final_sol(logsol,1).ToString)
            Next
                  For i=0 To iterations-1
                        objReader.WriteLine(total_sols_cost(i))
                  Next
              objReader.Close()
            Catch exe As Exception
                  MessageBox.Show(exe.Message)
            End Try


            return final_sol
      End Function
End Class
```

Πίνακας 3. Κλάση ConANTCore.vb

```
Imports System.Math

Public Class ConANTUtil

      Dim randClass As New Random()

      Public Function AdaptConstraints(generatorsData(,) As Double,
```

```
previousPeriodSolution(,) As Double) As Double(,)
            Dim newGeneratorsData(,) As Double = generatorsData.Clone


            For i=0 To newGeneratorsData.GetUpperBound(0)-1


                  If (previousPeriodSolution(i,0)<>0) Then
                        'new solution is limited within the ramp-range from the last-
period solution
                        'if solution-ramp>minimum, restrict minimum power to the
solution-ramp
                        if (previousPeriodSolution(i,0) - generatorsData(i,6)) >
generatorsData(i,0) Then
                              newGeneratorsData(i,0) = previousPeriodSolution(i,0) -
generatorsData(i,6)
                        End If
                        'if solution+ramp<maximum, restrict maximum power to the
solution+ramp
                        if (previousPeriodSolution(i,0) + generatorsData(i,6)) <
generatorsData(i,1) Then
                              newGeneratorsData(i,1) = previousPeriodSolution(i,0) +
generatorsData(i,6)
                        End If
                  End If


                  If newGeneratorsData(i,1) < newGeneratorsData(i,0) Then
                        newGeneratorsData(i,1) = newGeneratorsData(i,1)
                  End If


            Next


            return newGeneratorsData
      End Function
```

```
      Public Function CalculateSolutionCost(geninput(,) As Double, solution(,) As Double,
previousPeriodSolution(,) As Double) As Double

            Dim solutioncost As Double = 0

            For i=0 To solution.GetUpperBound(0)


                  If solution(i,0) <> 0 Then

                        'calculate functional cost

                        Dim cost_i As Double =
geninput(i,2)*solution(i,0)^2+geninput(i,3)*solution(i,0)+geninput(i,4)

                        'add to overall solution cost

                        solutioncost += cost_i

                  End If


                  If (solution(i,0) = 0 And previousPeriodSolution(i,0) <> 0) Then

                        ' shutting down, so add shut down cost

                        solutioncost += geninput(i,9)

                  End If

                  If (solution(i,0) <> 0 And previousPeriodSolution(i,0) = 0) Then

                        ' shutting down, so add startup cost

                        solutioncost += geninput(i,10)

                  End If

            Next

            return solutioncost

      End Function


      Public Function FindGlobalSolution(ini_input(,) As Double, requiredPower As Double)
As Double(,)

            Dim gensMaxIndex As Integer = ini_input.GetUpperBound(0)

            Dim randomsolution(gensMaxIndex,1) As Double


            'calculate mean maximum  power generation
```

```
            Dim totmax As Double = 0

            Dim totmin As Double = 0

            For d=0 To gensMaxIndex

                  totmax += ini_input(d,1)

                  totmin += ini_input(d,0)

            Next


            If totmax < requiredPower Then

                  MessageBox.Show("Generators after constraints adaptation not enough for
power demand: " + requiredPower.ToString()+". Returning maxed generators for this
period.")

                  For d=0 To gensMaxIndex

                        randomsolution(d,0) = ini_input(d,1)

                        randomsolution(d,1) = ini_input(d,5)*randomsolution(d,0)

                  Next

                  return randomsolution

            End If


            Dim meanmaxpergen = totmax/(gensMaxIndex+1)

            'now check how many generators working at "mean max" will cover power demand

            Dim musthavegens As Double = Math.Ceiling(requiredPower/meanmaxpergen)


            Dim numberofworkinggens As Integer =  gensMaxIndex+1

            Dim turnedOffgens As Integer = 0

            If numberofworkinggens-musthavegens >0 Then

                  numberofworkinggens =  Math.Round(randClass.Next(gensMaxIndex +1 -
musthavegens)) + musthavegens

                  turnedOffgens = gensMaxIndex+1-numberofworkinggens

            End If


            Dim foundSolution As Boolean = False
```

```
            Dim loopout As Integer = 200
            While Not foundSolution And loopout>0
                  gensMaxIndex = ini_input.GetUpperBound(0)
                  Dim totalsolution As Double = 0
                  For z=0 To gensMaxIndex
                        randomsolution(z,0) = randClass.Next(ini_input(z,0),
ini_input(z,1))
                        randomsolution(z,1) = randomsolution(z,0)*ini_input(z,5)
                        totalsolution += randomsolution(z,0)
                  Next
                  Dim actuallyTurnedOff As Integer = 0
                  Dim totminWithTurnoff As Double =0
                  Dim totmaxWithTurnOff As Double = 0
                  While actuallyTurnedOff < turnedOffgens
                        Dim offIndex As Integer =
Math.Round(randClass.NextDouble()*gensMaxIndex)
                        For g=0 To gensMaxIndex
                              If randomsolution(g,0)<>0 and g<>offIndex Then
                                    totminWithTurnoff += ini_input(g,0)
                                    totmaxWithTurnOff += ini_input(g,1)
                              End If
                        Next
                        If randomsolution(offIndex, 0) <> 0 and totmaxWithTurnOff >
requiredPower Then
                              totalsolution -= randomsolution(offIndex, 0)
                              randomsolution(offIndex, 0) = 0
                              randomsolution(offIndex, 1) = 0
                              actuallyTurnedOff +=1
                        End If
                  End While
```

```
                Dim diffToArrange As Double = requiredPower-totalsolution
                Dim pcnt As Double = 0
                totminWithTurnoff =0
                totmaxWithTurnOff = 0
                For g=0 To gensMaxIndex
                      If randomsolution(g,0)<>0 Then
                            totminWithTurnoff += ini_input(g,0)
                            totmaxWithTurnOff += ini_input(g,1)
                      End If
                Next

                Dim adaptedSolution As Boolean = False
                Dim loopoutinner As Integer = 200

                'work until the last NOT zero solution
                While randomsolution(gensMaxIndex, 0) = 0
                      gensMaxIndex -= 1
                End While
                'substract the balader last not zero solution
                totalsolution -= randomsolution(gensMaxIndex, 0)

                While Not adaptedSolution And loopoutinner>0
                      If requiredPower>totalsolution Then
                            'pcnt = diffToArrange/(totmaxWithTurnOff-totalsolution)
                            For f=0 To gensMaxIndex-1
                                  If (randomsolution(f,0)<>0) Then
                                        totalsolution -= randomsolution(f,0)
                                        randomsolution(f,0) +=
randClass.NextDouble()*(ini_input(f,1)-randomsolution(f,0))
                                        randomsolution(f,1) = ini_input(f,
5)*randomsolution(f,0)
```

```
                                    totalsolution += randomsolution(f,0)

                                End If

                            Next

                            If (requiredPower-totalsolution) > ini_input(gensMaxIndex,
0) And (requiredPower-totalsolution)<ini_input(gensMaxIndex, 1) Then

                                randomsolution(gensMaxIndex, 0) = requiredPower-
totalsolution

                                randomsolution(gensMaxIndex, 1) =
ini_input(gensMaxIndex, 5)*randomsolution(gensMaxIndex, 0)

                                totalsolution = requiredPower

                                adaptedSolution = true

                            End If

                        Else

                            'pcnt = diffToArrange / (totalsolution-totminWithTurnoff)

                            For f=0 To gensMaxIndex-1

                                If (randomsolution(f,0)<>0) Then

                                    totalsolution -= randomsolution(f,0)

                                    randomsolution(f,0) -=
randClass.NextDouble()*(randomsolution(f,0)-ini_input(f,0))

                                    randomsolution(f,1) = ini_input(f,
5)*randomsolution(f,0)

                                    totalsolution += randomsolution(f,0)

                                End If

                            Next

                            If (totalsolution-requiredPower) > ini_input(gensMaxIndex,
0) And (totalsolution-requiredPower)<ini_input(gensMaxIndex, 1) Then

                                randomsolution(gensMaxIndex, 0) = totalsolution-
requiredPower

                                randomsolution(gensMaxIndex, 1) =
ini_input(gensMaxIndex, 5)*randomsolution(gensMaxIndex, 0)

                                totalsolution = requiredPower

                                adaptedSolution = true

                            End If
```

```
                        End If
                        Dim lastTot As Double = 0
                        For f=0 To gensMaxIndex
                              lastTot+=randomsolution(f,0)
                        Next

                        If lastTot=requiredPower Then
                              foundSolution = true
                        End If
                        loopoutinner -= 1
                  End While

                  loopout -=1
            End While



            If loopout = 0 Then
                  throw new Exception("Could not adapt a solution to power demand: " +
requiredPower.ToString())
            End If

            return randomsolution
      End Function

      Public Function FindLocalSolution(iter As Integer, locant As Integer, powerdemand
as Double, ini_input(,) as Double, initial(,) As Double, generators As Integer, rad as
Double) As Double(,)
            Dim newsolution(generators-1,1) As Double
            Dim totaldiff As Double = 0
            Dim continuelooping As Boolean = True
```

```
            Dim loopout1 As Integer = 150
            Dim loopout2 As Integer = 150
            Dim maxGeneratorsIndex As Integer = generators-1

            While continuelooping and loopout1>0
                  continuelooping = false
                  For i=0 To maxGeneratorsIndex
                        Dim diff As Double = 0
                        newsolution(i,0) = 0
                        newsolution(i,1) = 0

                        If (initial(i,0) <> 0) Then 'find a "close" solution ONLY if
initial solution is NOT zero
                              While (newsolution(i,0)<ini_input(i,0) Or
newsolution(i,0)>ini_input(i,1)) and loopout2>0
                                    Dim signrand As Double = randClass.NextDouble()
                                    Dim sign As Integer = 1
                                    If signrand < 0.5 Then
                                          sign = -1
                                    End If

                                    'Check if rad is bigger than diff from initial
solution from boundaries
                                    '(no loss solution)
                                    Dim temprad As Double = rad
                                    Dim t1 As Double = initial(i,0)
                                    Dim t2 As Double = ini_input(i,0)
                                    Dim t3 as Double = ini_input(i,1)

                                    If t1=t2 Then
                                          sign=1
```

```
                        If (t3-t1) < rad Then
                              temprad = t3-t1
                        End If
                  Else If t3=t1 Then
                        sign=-1
                        If (t1-t2) < rad Then
                              temprad = t1-t2
                        End If
                  Else If (t1-t2)<rad Then
                        temprad = t1-t2
                  Else If (t3-t1)<rad Then
                        temprad = t3-t1
                  End If

                  diff = sign*temprad*randClass.NextDouble()

                  'select second (index=2) parameter of initial array -
the one with power loss
                  If (initial(i,0)=0) Then
                        newsolution(i,0) = initial(i,0)
                        newsolution(i,1) = initial(i,1)
                  Else If ((initial(i,0)+diff) < ini_input(i,1)) and
((initial(i,0)+diff) > ini_input(i,0)) Then
                        newsolution(i,0) = initial(i,0)+diff
                        newsolution(i,1) = initial(i,1)+diff
                  Else
                        newsolution(i,0) = initial(i,0)
                        newsolution(i,1) = initial(i,1)
                  End If

                  'HERE HERE HERE
```

```
                                    'just for debug purposes
                                    If (loopout2<50) Then
                                          Dim a As Double = newsolution(i,0)
                                          Dim b As Double = ini_input(i,0)
                                          Dim c As Double = ini_input(i,1)
                                          Dim d as Double = 0.0000
                                          loopout2-=0
                                    End If

                                    loopout2-=1
                              End While
                              totaldiff += diff
                        End If

                  Next

                  Dim totalnewpower As Double = 0

                  ''WRONG now ensure that found solution is not near bounds of "local"
area
                  'now adjust last sequentially generator so that total output is the
requested
                  Dim correctedRandomOutput As Boolean = False
                  For i=0 To maxGeneratorsIndex
                        If correctedRandomOutput = false and (newsolution(i,0)-totaldiff)
> ini_input(i,0) And (newsolution(i,0)-totaldiff) < ini_input(i,1) Then
                              newsolution(i,0) -= totaldiff
                              totalnewpower += newsolution(i,0)
                              correctedRandomOutput = True
                        Else
                              totalnewpower += newsolution(i,0)
```

```
                        End If

                Next


                ' now allowing for rounding errors (1%) check that new total power
equals the power demand

                If totalnewpower > powerdemand*1.01 Or totalnewpower < powerdemand*0.99
Then

                        continuelooping = True

                End If

                loopout1-=1

            End While


            If loopout1=0 or loopout2=0 Then

                throw new Exception("Could not find a new local solution for power
demand "+powerdemand.ToString()+" and locant group " +locant.ToString()+ " and iteration
"+iter.ToString()+" so looped out.")

            End If


            'debug

            For lak=0 To ini_input.GetUpperBound(0)

                If (newsolution(lak, 0) < ini_input(lak, 0) Or newsolution(lak,
0)>ini_input(lak, 1)) and  newsolution(lak, 0)<>0 Then

                        newsolution = newsolution

                End If

            Next


            return newsolution

      End Function


      Public Function RandomizeGeneratorOutput(ini_input(,) As Double, requiredPower As
Double) As Double(,)

            Dim randomsolution(geninput.GetUpperBound(0),1) As Double
```

```
            Dim totalpower As Double = 0
            Dim foundsolution As Boolean = False
            Dim loopout As Integer = 250

            'calculate mean maximum  power generation
            Dim totmax As Double = 0
            For d=0 To ini_input.GetUpperBound(0)
                  totmax += geninput(d,1)
            Next
            Dim meanmaxpergen = totmax/(ini_input.GetUpperBound(0))
            'now check how many generators working at "mean max" will cover power demand
            Dim musthavegens As Double = Math.Ceiling(requiredPower/meanmaxpergen)

            Dim numberofworkinggens =  geninput.GetUpperBound(0)
            If numberofworkinggens-musthavegens >0 Then
                  numberofworkinggens =
Math.Round(randClass.Next(ini_input.GetUpperBound(0) - musthavegens)) + musthavegens
            End If
            Dim zeroSelectionProbability As Double = 0.05*(ini_input.GetUpperBound(0)-
numberofworkinggens)

            While Not foundsolution And loopout > 0
                  If loopout < 60 Then
                        ' reduce zero probability because the system can not find
solution
                        zeroSelectionProbability = zeroSelectionProbability/2
                  End If
                  Dim zeroPowerGens As Integer = 0
                  For i=0 To ini_input.GetUpperBound(0)
                        Dim randpower As Double
                        If requiredPower-totalpower>ini_input(i,0) And requiredPower-
totalpower<ini_input(i,1) Then
```

```
                        randpower = requiredPower-totalpower

                    Else if zeroPowerGens < (geninput.GetUpperBound(0)-
numberofworkinggens) and randClass.Next() < zeroSelectionProbability Then

                        randpower = 0

                        zeroPowerGens +=1

                    Else If requiredPower-totalpower>ini_input(i,0) Then

                        randpower = randClass.Next(ini_input(i,0), ini_input(i,1))

                    Else

                        randpower = 0

                        zeroPowerGens +=1

                    End If

                    randomSolution(i,0) = randpower

                    randomSolution(i,1) = randpower*ini_input(i,5)

                    totalpower += randpower

                Next




                'if remaining power demand is within range of the last remaining
generator

                'we have an acceptable solution

                'If requiredPower-totalpower>geninput(geninput.GetUpperBound(0)-1,0)
And requiredPower-totalpower<geninput(geninput.GetUpperBound(0)-1,1) Then

                '    randomSolution(geninput.GetUpperBound(0)-1,0) = requiredPower-
totalpower

                '    'multiply by power loss parameter (sixth value in the input
array)

                '    randomSolution(geninput.GetUpperBound(0)-1,1) = (requiredPower-
totalpower)*geninput(geninput.GetUpperBound(0)-1,5)

                '    foundSolution = True

                'Else ' clear found solution and redo the process

                '    Array.Clear(randomsolution, 0, randomsolution.GetUpperBound(0))
```

```
                  '     totalpower=0
                  'End If
                  If requiredPower = totalpower Then
                        foundSolution = True
                  Else ' clear found solution and redo the process
                        Array.Clear(randomsolution, 0, randomsolution.GetUpperBound(0))
                        totalpower=0
                  End If

                  loopout -=1
            End While

            If loopout = 0 Then
                  throw new Exception(" Could not adapt initial solution to requested
power: " + requiredPower.ToString() + ". Process looped out! ")
            End If

            return randomSolution
      End Function

      Public Function FindGlobalRandomSolution(generators As Integer, minmax_noloss(,) As
Double, powerdemand As Double) As Double()
            Dim globalsol(generators-1) As Double
            Dim restpower As Double = powerdemand
            Dim powerdiff(generators-1) As Double
            Dim initialpowers(generators-1) As Double

            For mk=0 To generators-1
                  initialpowers(mk) = minmax_noloss(mk, 0)
            Next
```

```
            'Adding random power within range to each generator BUT THE LAST
            'in order to cover the remaining power demand (restpower parameter).
            Dim rpowermax As Double = 0
            Dim rpowerAddition As Double = 0
            Dim continuelooping As Boolean = True
            'maximum loops to try to find an acceptable global solution
            Dim loopout As Integer = 150
            While continuelooping And loopout > 0
                  restpower = powerdemand
                  'Subtracting all powers added to the initial solution passed from the
power demand
                  For i=0 To generators-1
                        restpower -= initialpowers(i)
                        powerdiff(i) = minmax_noloss(i,1) - initialpowers(i)
                  Next
                  For j=0 To generators-2
                        If restpower > powerdiff(j) Then
                              rpowermax = powerdiff(j)
                        Else
                              rpowermax = restpower
                        End If
                        rpowerAddition = rpowermax*randClass.NextDouble()
                        globalsol(j) = initialpowers(j)+rpowerAddition
                        'CAUTION: setting initial powers to the randomized ones so that
if the function is re-run
                        'due to lack of power, the power of each genset will be
increased.
                        initialpowers(j) = globalsol(j)
                        If restpower>=rpowerAddition Then
                              restpower -= rpowerAddition
                        End If
```

```
                    Next


                    If restpower > powerdiff(generators-1) Then
                          'Remaining more power to cover than the last generator can cover
withing range.
                          'Thus rerun this while loop
                    Else
                          globalsol(generators-1) = initialpowers(generators-1) + restpower
                          continuelooping = false
                    End If


                    loopout -=1
              End While


              If loopout = 0 Then
                    throw new Exception(" Could not randomize new global solution to
requested power. Process looped out! ")
              End If


              return globalsol
        End Function


        Public Function CalculateSearchRadius(geninput(,) As Double, generators As Integer)
As Double
              Dim tempdiff As Double = 0
              Dim mindiff As Double = 999999999999
              For i=0 To generators-1
                    tempdiff = geninput(i,1)-geninput(i,0)
                    If tempdiff<mindiff Then
                          mindiff = tempdiff
                    End If
```

```
            Next


            Dim rad As Double = mindiff/20

            'return on tenth of minimum difference divided by 2 to find the radius: /10/2
= /20

            return rad

      End Function


      Public Function CalculatePheromon(generators As Integer, initialpheromone As
Double, iteration As Integer, solution_pergen_perit(,,) As Double, endurance As Double,
evap_factor As Double, current_sol(,) As Double, doubleRad As Double, pher_pervisit As
Double) As Double

            Dim calc_iter As Integer = 1

            'If iteration >endurance Set iteration For calc To the difference, Then

            'more iterations are neglected as the previous pheromone will have evaporated

            If iteration > endurance Then

                  calc_iter = iteration - endurance

            End If


            Dim pheromone As Double = initialpheromone

            Dim evaporation As Double = 0

            Dim looppher As Double = 0

            For i=calc_iter To iteration

                  evaporation = (1-evap_factor)^(iteration - calc_iter)

                  For j=0 To generators-1

                        If System.Math.Abs(solution_pergen_perit(i,j,1) -
current_sol(j,1)) > doubleRad Then

                    looppher = 0

                Else

                    looppher = System.Math.Abs(solution_pergen_perit(i,j,1) -
current_sol(j,1))*evaporation/doubleRad

                End If
```

```
                pheromone += looppher

                  Next

            Next


            return pheromone

      End Function


End Class
```

Πίνακας 4. Κλάση ConANTUtil.vb

```
Imports System.IO

Imports System.Text.RegularExpressions

Imports System.Globalization


'

' Created by SharpDevelop.

' User: dalexo

' Date: 14/2/2010

' Time: 6:55 μμ

'

' To change this template use Tools | Options | Coding | Edit Standard Headers.

'

Public Partial Class MainForm


      Private inputfieldsCreated As Boolean = False

      Private inputContents As String = ""

      Private controlNum As Integer = 0


      'genlabel.Visible
```

```
      Public Sub New()

            ' The Me.InitializeComponent call is required for Windows Forms designer
support.

            Me.InitializeComponent()

            controlNum = Me.Controls.Count

            Me.Text = "Unit Commitment Optimization with Con-ANT"


            Dim ini As New InitializeAnts

            Me.gennolabel1.Visible=false

            Me.dynamicControl011.Visible=false

            Me.dynamicControl021.Visible=false

            Me.dynamicControl031.Visible=false

            Me.dynamicControl041.Visible=false

            Me.dynamicControl051.Visible=False

            Me.dynamicControl061.Visible=False

            Me.dynamicControl071.Visible=False

            Me.dynamicControl081.Visible=False

            Me.dynamicControl091.Visible=False

            Me.dynamicControl101.Visible=False

            Me.dynamicControl111.Visible=False

            Me.dynamicControl121.Visible=False


            Me.startAntsButton.Visible=False

            Me.resultsTextBox.Visible=False


            Me.numofantgrps.Text = "5"

            Me.numofiterations.Text = "500"

            Me.pheroendurance.Text = "3"

            Me.evaporationFactor.Text = "0,3"
```

```
            Me.propabilityFactor.Text = "0,07"

            Me.pherpervisit.Text = "3"

            Me.reqpower.Text = ""

            Me.numofgen.Text =
"G:\sh\meletes\del100415_598_antcolony\senario_a_input.txt"


      End Sub



      Public Function GetFileContents(ByVal FullPath As String, _

       Optional ByRef ErrInfo As String = "") As String


        Dim strContents As String

        Dim objReader As StreamReader

        Try


            objReader = New StreamReader(FullPath)

            strContents = objReader.ReadToEnd()

            objReader.Close()

            Return strContents

        Catch Ex As Exception

            ErrInfo = Ex.Message

            return ErrInfo

        End Try

    End Function


      Sub Label1Click(sender As Object, e As EventArgs)


      End Sub
```

```
      Sub Button1Click(sender As Object, e As EventArgs)


            if (openFileDialog1.ShowDialog() = DialogResult.OK) then
                  numofgen.Text = openFileDialog1.FileName
            Else
            Debug.WriteLine("Error selecting file!")
        End If







      End Sub


      Sub Label3Click(sender As Object, e As EventArgs)


      End Sub


      Sub AntgrplabelClick(sender As Object, e As EventArgs)


      End Sub


      Sub OpenFileDialog1FileOk(sender As Object, e As
System.ComponentModel.CancelEventArgs)


      End Sub


      Sub ParseaAndLoadFileContent(content As String)


      End Sub
```

```
Sub LoadbuttonClick(sender As Object, e As EventArgs)

      While Me.Controls.Count>controlNum
      Me.Controls.RemoveAt(Me.Controls.Count-1)
  End While
  Me.dynamicControl011.Text=""
      Me.dynamicControl021.Text=""
      Me.dynamicControl031.Text=""
      Me.dynamicControl041.Text=""
      Me.dynamicControl051.Text=""
  Me.dynamicControl061.Text=""
  Me.dynamicControl071.Text=""
  Me.dynamicControl081.Text=""
  Me.dynamicControl091.Text=""
  Me.dynamicControl101.Text=""
  Me.dynamicControl111.Text=""
  Me.dynamicControl121.Text=""

      Try

            Dim tboxwidth As Integer = dynamicControl011.Width
            Dim rowheight As Integer = gennolabel1.Height
            Dim iniheightoffset As Integer = 5
            Dim labelloc As Point = gennolabel1.Location
            Dim tbox1loc As Point = dynamicControl011.Location
            Dim tbox2loc As Point = dynamicControl021.Location
            Dim tbox3loc As Point = dynamicControl031.Location
            Dim tbox4loc As Point = dynamicControl041.Location
            Dim tbox5loc As Point = dynamicControl051.Location
```

```
Dim tbox6loc As Point = dynamicControl061.Location
Dim tbox7loc As Point = dynamicControl071.Location
Dim tbox8loc As Point = dynamicControl081.Location
Dim tbox9loc As Point = dynamicControl091.Location
Dim tbox10loc As Point = dynamicControl101.Location
Dim tbox11loc As Point = dynamicControl111.Location
Dim tbox12loc As Point = dynamicControl121.Location

Dim inputNum As Integer = 5
Try
      inputNum = Integer.Parse(numofgen.Text)
Catch
      inputNum = -1
End Try

Dim rows As String() = {}
Dim gensets As Integer
If (inputNum>0) Then
      gensets = inputNum
Else
      Dim sErr As String = ""
      inputContents = GetFileContents(numofgen.Text, sErr)
      If sErr = "" Then
      'MessageBox.Show("Input file loaded")
      ParseaAndLoadFileContent(inputContents)
Else
      Debug.WriteLine("Error retrieving file: " & sErr)
 End If
rows = Regex.Split(inputContents, "\r")
'first row is occupied with power requests so gensets are the rest rows
```

```
            gensets = rows.Length-1
            'assign first row to required powers
            Me.reqpower.Text = rows(0)
      End If

      For i=0 To gensets-1

            If (i=0) Then
                  If (inputNum<0) Then
                        Dim values As String() = Regex.Split(rows(i+1), ";")
                        dynamicControl011.Text = values(0)
                        dynamicControl021.Text = values(1)
                        dynamicControl031.Text = values(2)
                        dynamicControl041.Text = values(3)
                        dynamicControl051.Text = values(4)
                        dynamicControl061.Text = values(5)
                        dynamicControl071.Text = values(6)
                        dynamicControl081.Text = values(7)
                        dynamicControl091.Text = values(8)
                        dynamicControl101.Text = values(9)
                        dynamicControl111.Text = values(10)
                        dynamicControl121.Text = values(11)
                  End If
                  Me.gennolabel1.Visible=true
                   Me.dynamicControl011.Visible=true
                  Me.dynamicControl021.Visible=true
                  Me.dynamicControl031.Visible=true
                  Me.dynamicControl041.Visible=true
                  Me.dynamicControl051.Visible=True
                  Me.dynamicControl061.Visible=true
```

```
                        Me.dynamicControl071.Visible=true
                        Me.dynamicControl081.Visible=true
                        Me.dynamicControl091.Visible=true
                        Me.dynamicControl101.Visible=true
                        Me.dynamicControl111.Visible=true
                        Me.dynamicControl121.Visible=true
                    Else
                        Dim myLabel = New Label()
                    myLabel.Text = I+1
                    myLabel.Name = "dynamicLabel" & I+1
                    myLabel.TextAlign = gennolabel1.TextAlign
                    myLabel.Size = gennolabel1.Size
                    myLabel.Location = new Point(labelloc.X,
labelloc.Y+iniheightoffset+I*gennolabel1.Height)
                        Me.Controls.Add(myLabel)

                    Dim myTextBox1 = New TextBox()
                        myTextBox1.Name = "dynamicControl01" & I+1
                        myTextBox1.Size = dynamicControl011.Size
                        myTextBox1.Visible = True
                        myTextBox1.Location = new Point(tbox1loc.X,
tbox1loc.Y+iniheightoffset+I*dynamicControl011.Height)
                        Me.Controls.Add(myTextBox1)

                        Dim myTextBox2 = New TextBox()
                        myTextBox2.Name = "dynamicControl02" & I+1
                        myTextBox2.Size = dynamicControl011.Size
                        myTextBox2.Visible = True
                        myTextBox2.Location = new Point(tbox2loc.X,
tbox1loc.Y+iniheightoffset+I*dynamicControl011.Height)
                        Me.Controls.Add(myTextBox2)
```

```
                                Dim myTextBox3 = New TextBox()
                                myTextBox3.Name = "dynamicControl03" & I+1
                                myTextBox3.Size = dynamicControl011.Size
                                myTextBox3.Visible = True
                                myTextBox3.Location = new Point(tbox3loc.X,
tbox1loc.Y+iniheightoffset+I*dynamicControl011.Height)
                                Me.Controls.Add(myTextBox3)

                                Dim myTextBox4 = New TextBox()
                                myTextBox4.Name = "dynamicControl04" & I+1
                                myTextBox4.Size = dynamicControl011.Size
                                myTextBox4.Visible = True
                                myTextBox4.Location = new Point(tbox4loc.X,
tbox1loc.Y+iniheightoffset+I*dynamicControl011.Height)
                                Me.Controls.Add(myTextBox4)

                                Dim myTextBox5 = New TextBox()
                                myTextBox5.Name = "dynamicControl05" & I+1
                                myTextBox5.Size = dynamicControl011.Size
                                myTextBox5.Visible = True
                                myTextBox5.Location = new Point(tbox5loc.X,
tbox1loc.Y+iniheightoffset+I*dynamicControl011.Height)
                                Me.Controls.Add(myTextBox5)

                                Dim myTextBox6 = New TextBox()
                                myTextBox6.Name = "dynamicControl06" & I+1
                                myTextBox6.Size = dynamicControl011.Size
                                myTextBox6.Visible = True
                                myTextBox6.Location = new Point(tbox6loc.X,
tbox1loc.Y+iniheightoffset+I*dynamicControl011.Height)
                                Me.Controls.Add(myTextBox6)
```

```
                                Dim myTextBox7 = New TextBox()
                                myTextBox7.Name = "dynamicControl07" & I+1
                                myTextBox7.Size = dynamicControl011.Size
                                myTextBox7.Visible = True
                                myTextBox7.Location = new Point(tbox7loc.X,
tbox1loc.Y+iniheightoffset+I*dynamicControl011.Height)
                                Me.Controls.Add(myTextBox7)


                                Dim myTextBox8 = New TextBox()
                                myTextBox8.Name = "dynamicControl08" & I+1
                                myTextBox8.Size = dynamicControl011.Size
                                myTextBox8.Visible = True
                                myTextBox8.Location = new Point(tbox8loc.X,
tbox1loc.Y+iniheightoffset+I*dynamicControl011.Height)
                                Me.Controls.Add(myTextBox8)


                                Dim myTextBox9 = New TextBox()
                                myTextBox9.Name = "dynamicControl09" & I+1
                                myTextBox9.Size = dynamicControl011.Size
                                myTextBox9.Visible = True
                                myTextBox9.Location = new Point(tbox9loc.X,
tbox1loc.Y+iniheightoffset+I*dynamicControl011.Height)
                                Me.Controls.Add(myTextBox9)


                                Dim myTextBox10 = New TextBox()
                                myTextBox10.Name = "dynamicControl10" & I+1
                                myTextBox10.Size = dynamicControl011.Size
                                myTextBox10.Visible = True
                                myTextBox10.Location = new Point(tbox10loc.X,
tbox1loc.Y+iniheightoffset+I*dynamicControl011.Height)
                                Me.Controls.Add(myTextBox10)
```

```
                                Dim myTextBox11 = New TextBox()
                                myTextBox11.Name = "dynamicControl11" & I+1
                                myTextBox11.Size = dynamicControl011.Size
                                myTextBox11.Visible = True
                                myTextBox11.Location = new Point(tbox11loc.X,
tbox1loc.Y+iniheightoffset+I*dynamicControl011.Height)
                                Me.Controls.Add(myTextBox11)


                                Dim myTextBox12 = New TextBox()
                                myTextBox12.Name = "dynamicControl12" & I+1
                                myTextBox12.Size = dynamicControl011.Size
                                myTextBox12.Visible = True
                                myTextBox12.Location = new Point(tbox12loc.X,
tbox1loc.Y+iniheightoffset+I*dynamicControl011.Height)
                                Me.Controls.Add(myTextBox12)

                                If (inputNum<0) Then
                                    Try
                                        Dim values As String() = Regex.Split(rows(i+1),
";")
                                        myTextBox1.Text = values(0)
                                        myTextBox2.Text = values(1)
                                        myTextBox3.Text = values(2)
                                        myTextBox4.Text = values(3)
                                        myTextBox5.Text = values(4)
                                        myTextBox6.Text = values(5)
                                        myTextBox7.Text = values(6)
                                        myTextBox8.Text = values(7)
                                        myTextBox9.Text = values(8)
                                        myTextBox10.Text = values(9)
```

```
                                        myTextBox11.Text = values(10)

                                        myTextBox12.Text = values(11)

                                    Catch

                                        MessageBox.Show("Input file has wrong format:
\n it should contain 6 numbers separated with a ;")

                                    End Try


                                End If


                                startAntsButton.Location = New Point(tbox11loc.X,
tbox1loc.Y+iniheightoffset+gensets*dynamicControl011.Height)

                            startAntsButton.Visible = true


                        End If



                    Next


                    ReDim Preserve ParametersModule.geninput(gensets-1,11)

                inputfieldsCreated = True


            Catch ex As Exception

                    MessageBox.Show(String.Concat("Error occured during initial load of
input data!", ex.Message))

            End Try

        End Sub


        Sub StartAntsButtonClick(sender As Object, e As EventArgs)

            Try

                    'Dim style As NumberStyles = NumberStyles.Double Or
```

```
NumberStyles.AllowDecimalPoint Or NumberStyles.AllowThousands
                Dim style As NumberStyles = NumberStyles.Float
                Dim culture As CultureInfo = CultureInfo.InstalledUICulture

                'counter for the actual number of input rows (i.e. the number of
generators
                Dim si As Integer = 0

                Dim con As Control
                For Each con In Me.Controls
                      If (con.Name.Contains("dynamicControl")) Then
                            Dim nums As String = con.Name.Substring(14)
                            Dim numParamIndex As String = nums.Substring(0,2)
                            Dim numGensetIndex As String = nums.Substring(2)
                            ParametersModule.geninput(Integer.Parse(numGensetIndex,
style)-1,Integer.Parse(numParamIndex, style)-1)=con.Text
                      End If
                      If (con.Name.Contains("dynamicControl01")) Then
                            si = si+1
                      End If
                Next
                ParametersModule.numofgens = si
                Dim success As Boolean = true
                Try
                      ParametersModule.numofantgrps =
Integer.Parse(Me.numofantgrps.Text)
                Catch ex1 As Exception
                      MessageBox.Show(String.Concat("Please give correct number of ant
groups! ", ex1.Message))
                End Try

                ParametersModule.numofants =
```

```
ParametersModule.numofantgrps*ParametersModule.numofgens



                Try

                      ParametersModule.iterations =
Integer.Parse(Me.numofiterations.Text)

                Catch ex2 As Exception

                      MessageBox.Show(String.Concat("Please give correct number of
iterations! ", ex2.Message))

                End Try

                Try

                      ParametersModule.endurance = Double.Parse(Me.pheroendurance.Text)

                Catch ex3 As Exception

                      MessageBox.Show(String.Concat("Please give correct number of
pheromone endurance! ", ex3.Message))

                End Try

                Try

                      ParametersModule.evap_factor =
Double.Parse(Me.evaporationFactor.Text)

                      If ParametersModule.evap_factor <= 0 Or
ParametersModule.evap_factor>=1 Then

                            throw new Exception("Pheromone evaporation should be <0,99
and >0,01")

                      End If

                Catch ex4 As Exception

                      MessageBox.Show(String.Concat("Please give correct number of
pheromone evaporation! ", ex4.Message))

                End Try

                Try

                      ParametersModule.prop_factor =
Double.Parse(Me.propabilityFactor.Text)

                      If ParametersModule.prop_factor <= 0 Or
ParametersModule.prop_factor>=1 Then

                            throw new Exception("Propability factor should be <0,99 and
>0,01")
```

```
                        End If

                Catch ex5 As Exception

                        MessageBox.Show(String.Concat("Please give correct number of
propability factor!", ex5.Message))

                End Try

                Try

                        ParametersModule.pher_per_visit =
Double.Parse(Me.pherpervisit.Text)

                Catch ex6 As Exception

                        MessageBox.Show(String.Concat("Please give correct number of
pheromone per visit!", ex6.Message))

                End Try

                Try

                        Dim powerString() As String = Me.reqpower.Text.Split(New Char()
{","c})

                        Dim reqPowers(powerString.GetUpperBound(0)) As Double

                        For p=0 To powerString.GetUpperBound(0)

                              reqPowers(p) = Double.Parse(powerString(p))

                        Next

                        ParametersModule.reqpower = reqPowers

                Catch ex7 As Exception

                        MessageBox.Show(String.Concat("Please give correct number of
requested power csv!", ex7.Message))

                End Try


                Dim antcore As New ConANTCore


                Dim resultText As String = antcore.run()


                'resultsTextBox.Height = 70

                resultsTextBox.Location = New Point((Me.Width -
resultsTextBox.Width)/2, startAntsButton.Location.Y+100)
```

```
                resultsTextBox.Text = resultText
                resultsTextBox.Visible = True

            Catch gloexe As Exception
                MessageBox.Show(gloexe.Message)
                Debug.Write(gloexe.StackTrace)
            End Try
        End Sub

        Sub Label12Click(sender As Object, e As EventArgs)

        End Sub
End Class
```

Πίνακας 5. ***Κλάση MainForm.vb***

```
Public Module ParametersModule

    ' array with max powers
    ' first column: min power
    ' second column: max power
    ' third column: a-param
    ' fourth column: b-param
    ' fifth column: c-param
    ' sixth column: power loss
    Public geninput(,) As Double = {}

    ' number of generators
    Public numofgens As Integer
```

```
        ' each group with 'numofgens' ants
        Public numofantgrps As Integer
        ' numofants = numofantgrps*numofgens
        Public numofants As Integer
        ' number of iterations
        Public iterations As Integer
        ' endurance
        Public endurance As Double
        ' evaporation factor
        Public evap_factor As Double
        ' evaporation factor
        Public prop_factor As Double
        ' pheromone layed per visit at a solution
        Public pher_per_visit As Double
        ' requested power from all generators
        Public reqpower() As Double
End Module
```

Πίνακας 6. ***Κλάση ParametersModule.vb***