ΔΙΕΘΝΕΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

«Βιβλιογραφική παρουσίαση όλων των λύσεων του Προβλήματος του Περιοδεύοντος Πωλητή με τη χρήση Ανταγωνιστικών Νευρωνικών Δικτύων, Χαρτών Αυτό-οργάνωσης (Self-organizing Maps, SOM)»

«Εικόνα»

Του φοιτητή
Σαρηκυριακίδη Σταύρου
Αρ. Μητρώου: 164738

Επιβλέπων
Γουλιάνας Κωνσταντίνος
Βαθμίδα
Αναπληρωτής Καθηγητής

Θεσσαλονίκη 2023

Τίτλος Π.Ε. Βιβλιογραφική παρουσίαση όλων των λύσεων του Περιοδεύοντος Πωλητή με τη χρήση Ανταγωνιστικών Νευρωνικών Δικτύων, Χαρτών Αυτό-οργάνωσης (Self-organizing Maps, SOM)

Κωδικός Π.Ε. 22167

Ονοματεπώνυμο φοιτητή. Σαρηκυριακίδης Σταύρος

Ονοματεπώνυμο εισηγητή. Γουλιάνας Κωνσταντίνος

Ημερομηνία ανάληψης Π.Ε. 29-03-2022

Ημερομηνία περάτωσης Π.Ε. 20-01-2023

Βεβαιώνω ότι είμαι ο συγγραφέας αυτής της εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, έχω καταγράψει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών, εικόνων και κειμένου, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επιπλέον, βεβαιώνω ότι αυτή η εργασία προετοιμάστηκε από εμένα προσωπικά, ειδικά ως πτυχιακή εργασία, στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του ΔΙ.ΠΑ.Ε.

Πρόλογος

Ο κύριος λόγος για τον οποίο επιλέχθηκε αυτό το συγκεκριμένο θέμα πτυχιακής εργασίας αφορά το ενδιαφέρον του συγγραφέα στη Μηχανική μάθηση και πιο συγκεκριμένα στα τεχνητά νευρωνικά δίκτυα. Οι Χάρτες Αυτο-οργάνωσης (SOM) που είναι ένας τύπος τεχνητών νευρωνικών δικτύων χωρίς επιτήρηση, χρησιμοποιούνται σε προβλήματα οπτικοποίησης και ανάλυσης δεδομένων όπως το πρόβλημα του περιοδεύοντος πωλητή (TSP) το οποίο είναι ίσως το πιο γνωστό συνδυαστικό πρόβλημα βελτιστοποίησης. Οι Χάρτες Αυτο-οργάνωσης αποτελούν το μεγαλύτερο μέρος της παρούσας πτυχιακής εργασίας.

Περίληψη

Αντικείμενο της παρακάτω πτυχιακής εργασίας είναι η ανάλυση του Προβλήματος του Περιοδεύοντος Πωλητή (Travelling Salesman Problem, TSP) και η βιβλιογραφική παρουσίαση όλων των λύσεων του Προβλήματος που έχουν προταθεί με τη χρήση Χαρτών Αυτο-οργάνωσης (Self-Organizing Maps, SOM). Οι στόχοι της παρούσας πτυχιακής εργασίας είναι δυο. Ο πρώτος είναι να γίνει κατηγοριοποίηση και ανάλυση των μοντέλων SOM. Ο δεύτερος είναι η σύγκριση των μοντέλων σε κάθε μια από τις κατηγορίες με σκοπό την εύρεση του πιο βέλτιστου αλγορίθμου με βάση κάποια κριτήρια. Το παρόν σύγγραμμα χωρίζεται σε τρείς κύριες θεματικές ενότητες. Η πρώτη ενότητα ασχολείται με την επεξήγηση του TSP και των SOM. Η δεύτερη ενότητα εστιάζει στα διαφορετικά μοντέλα SOM που λύνουν το TSP. Η τρίτη ενότητα της εργασίας περιλαμβάνει τα υπολογιστικά αποτελέσματα και τις συγκρίσεις των μοντέλων που παρουσιάστηκαν. Στον επίλογο γίνεται η παρουσίαση του καλύτερου μοντέλου.

Λέξεις κλειδιά : Πρόβλημα Περιοδεύοντος Πωλητή, TSP, Χάρτες Αυτο-οργάνωσης, SOM

«Graphic solution of the Travelling Salesman Problem using competitive neural networks, Self-Organizing Maps (SOM)»

Stavros Sarikyriakidis

Abstract

The subject of the following dissertation is the analysis of the Travelling Salesman Problem (TSP) and the bibliographic presentation of all the solutions of the Problem that have been proposed using Self-Organization Maps (SOM). The objectives of this dissertation are two. The first objective is to categorize and analyze the SOM models. The second is to compare the models in each of the categories with the aim of finding the most optimal algorithm based on some criteria. This work is divided into three main thematic sections. The first section deals with the explanation of TSP and SOM. The second section focuses on the different SOM models that solve the TSP. The third section of the paper includes the computational results and comparisons of the presented models. In the epilogue, the best model is presented.

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Πινάκων

Συντομογραφίες

ΔΙΠΑΕ	Διεθνές Πανεπιστήμιο Ελλάδος
Π.Ε.	Πτυχιακή Εργασία
TSP	Travelling Salesman Problem
SOM	Self-Organizing Map
TSPLIB	TSP library
EN	Elastic Net

Κεφάλαιο 1ο: Εισαγωγή

Το Πρόβλημα του Περιοδεύοντος Πωλητή ή όπως είναι ευρέως γνωστό Travelling Salesman Problem είναι ένα από τα πιο μελετημένα προβλήματα συνδυαστικής βελτιστοποίησης, λόγω της μεγάλης υπολογιστικής πολυπλοκότητας του. Λαμβάνοντας υπόψη ένα σύνολο πόλεων και το κόστος του ταξιδιού (ή της απόστασης) μεταξύ κάθε πιθανών ζευγών, ο στόχος του Travelling Salesman Problem είναι να βρει τον καλύτερο δυνατό τρόπο επίσκεψης σε όλες τις πόλεις και να επιστρέψετε στο σημείο εκκίνησης ελαχιστοποιώντας το κόστος ταξιδιού (ή της απόστασης του ταξιδιού) .

Ένας Χάρτης Αυτο-οργάνωσης ή αλλιώς Self-Organizing Map είναι ένας τύπος είναι ένας τύπος τεχνητού νευρικού δικτύου που εκπαιδεύεται χρησιμοποιώντας μάθηση χωρίς επιτήρηση. Το Self-Organizing Map διαφέρει από άλλα τεχνητά νευρωνικά δίκτυα καθώς εφαρμόζει ανταγωνιστική μάθηση και χρησιμοποιεί μια λειτουργία γειτονιάς για να διατηρήσει τις τυπολογικές ιδιότητες του χώρου εισόδου.

Το παρόν σύγγραμμα χωρίζεται σε επτά κεφάλαια και το κάθε κεφάλαιο σε επιμέρους ενότητες.

Το πρώτο κεφάλαιο «Εισαγωγή» περιέχει τις βασικές έννοιες οι οποίες απασχολούν την παρούσα πτυχιακή εργασία.

Το δεύτερο κεφάλαιο «Self-Organizing maps & Travelling Salesman Problem» περιγράφει το πώς λειτουργεί το SOM στο TSP.

Στο τρίτο κεφάλαιο «Τα μοντέλα SOM» περιέχει τις κατηγορίες των SOM μοντέλων που λύνουν το TSP.

Το τέταρτο κεφάλαιο «Άλλες εφαρμογές» αναφέρει κάποιες άλλες εφαρμογές του TSP αλλά δεν κατηγοριοποιούνται.

Το πέμπτο κεφάλαιο «Αποτελέσματα και συγκρίσεις» περιγράφει συγκρίσεις μεταξύ μοντέλων καθώς και υπολογιστικά αποτελέσματα κάποιων.

Το έκτο κεφάλαιο «Σχολιασμός πινάκων» παρουσιάζει τους πίνακες που χρησιμοποιήθηκαν.

Το έβδομο κεφάλαιο και τελευταίο κεφάλαιο «Επίλογος» αναφέρει τα τελικά συμπεράσματα της πτυχιακής εργασίας.

Κεφάλαιο 2ο: Self-Organizing maps & Travelling Salesman Problem

2.1 Εισαγωγή

Αυτό το κεφάλαιο επιδιώκει να θέσει ένα θεωρητικό υπόβαθρο γύρω από TSP και το SOM. Υπάρχουν αρκετές παραλλαγές διαφορετικών TSP όμως θα επικεντρωθούμε πως το κλασσικό SOM λύνει το πρόβλημα αυτό.

2.2 Travelling Salesman Problem

Η απαρχή του TSP δεν είναι σαφείς. Ένα εγχειρίδιο για πλανόδιους πωλητές το 1832 αναφέρει το πρόβλημα και περιλαμβάνει παραδείγματα περιηγήσεων μέσω της Γερμανίας και της Ελβετίας, αλλά δεν περιέχει καμία μαθηματική προσέγγιση. Τον 19ο αιώνα το TSP ορίστηκε από τον Ιρλανδό μαθηματικό William Rowan Hamilton και το Βρετανό μαθηματικό Thomas Kirkman. Ο Hamilton δημιούργησε το Icosian Puzzle (το όνομα είναι από το αρχαίο ελληνικό είκοσι, ενώ αναφέρεται στη διεθνή βιβλιογραφία και ως Icosian game), που περιλαμβάνει την εύρεση ενός κύκλου, έτσι ώστε κάθε κόμβος να επισκέπτεται μία μόνο φορά, κανένας κόμβος να μην επισκέπτεται δεύτερη φορά και το τελικό σημείο να είναι ίδιο με το αρχικό. Ο κύκλος αυτός ονομάστηκε κύκλος του Hamilton ο οποίος είναι ένα μονοπάτι με την ιδιότητα αυτή. Οπότε, ένα μονοπάτι του Hamilton είναι ένα μονοπάτι σε μη κατευθυνόμενο γράφημα το οποίο επισκέπτεται κάθε κόμβο ακριβώς μια φορά. Η γενική μορφή του TSP φαίνεται να έχει μελετηθεί για πρώτη φορά κατά την δεκαετία του 1930 στη Βιέννη και στο Χάρβαρντ κυρίως από τον Karl Menger ο οποίος καθόρισε και το πρόβλημα. Στην δεκαετία του 1950 και 1960, έγινε όλο και πιο δημοφιλές στους επιστημονικούς κύκλους της Ευρώπης και της Αμερικής [1]. Η πολυπλοκότητα αυτού του προβλήματος είναι ότι από ένα πλήθος πόλεων και πάνω, ο αριθμός όλων των πιθανών διαδρομών είναι αρκετά μεγάλος για να τις εξετάσουμε τις διαδρομές μία προς μία. Για παράδειγμα, έστω ότι αναζητάμε τη λύση ενός προβλήματος TSP για N πόλεις. Ξεκινώντας από την πόλη που ορίζεται ως πρώτη στη διαδρομή (αρχική πόλη) έχουμε N-1 επιλογές για το ποιά θα είναι η δεύτερη πόλη της διαδρομής μας, N-2 επιλογές για την τρίτη πόλη και ούτω κάθε εξής. Επομένως, υπάρχουν συνολικά (N-1)! πιθανές διαδρομές που αποτελούν όλες λύσεις του προβλήματος. Τέλος, στο TSP αναζητείται η ελάχιστη από όλες αυτές τις λύσεις.

2.3 Self-Organizing maps

Έχουν αναπτυχθεί πολλοί ευρετικοί μέθοδοι προκειμένου να επιτευχθεί μια σχεδόν βέλτιστη λύση του TSP. Ο καθηγητής και ερευνητής της Φινλανδίας Dr. Teuvo Kohonen [73],[74] εισήγαγε ένα νέο τύπο νευρωνικού δικτύου το 1982 εισήγαγε έναν αυτο-οργανωτικό χάρτη με μάθησης χωρίς επίβλεψη, που προορίζεται για εφαρμογές στις οποίες είναι σημαντική η διατήρηση μιας τοπολογίας μεταξύ εισόδου και εξόδου. Ένα SOM διαφέρει από ένα τυπικό τεχνητό νευρωνικό δίκτυο τόσο στην αρχιτεκτονική όσο και στις ιδιότητες του αλγορίθμου. Πρώτον, η δομή του περιλαμβάνει ένα γραμμικό πλέγμα 2D νευρώνων με ένα στρώμα, αντί για μια σειρά στρωμάτων. Όλοι οι κόμβοι σε αυτό το πλέγμα συνδέονται απευθείας με τον φορέα εισόδου, αλλά όχι μεταξύ τους, πράγμα που σημαίνει ότι οι κόμβοι δεν γνωρίζουν τις τιμές των γειτόνων τους και ενημερώνουν μόνο το βάρος των συνδέσεών τους ως συνάρτηση

των δεδομένων εισόδου. Επίσης το SOM χρησιμοποιεί ανταγωνιστική μάθηση. Ο αλγόριθμος έχει ως εξής:

1. Αρχικοποίηση κάθε βάρους κόμβου *w_ij* με τυχαία τιμή.
2. Επιλογή τυχαίου vector εισόδου *x_k*.
3. Επανάληψη του 4ου και 5ου βήματος για όλους τους κόμβους του χάρτη:
4. Υπολογισμός Ευκλείδειας απόστασης μεταξύ του vector εισόδου *x(t)* και του vector βάρους *w_ij* που σχετίζονται με τον πρώτο κόμβο, όπου *t, i, j* = 0.
5. Καταγραφή του κόμβου που παράγει την μικρότερη απόσταση t.
6. Εύρεση του Best Matching Unit (BMU), δηλαδή τον κόμβο με την μικρότερη απόσταση από τους υπολογισμένους κόμβους.
7. Ορισμός της τοπολογικής γειτονιάς *βij(t)* και της ακτίνας *σ(t)* του BMU στον χάρτη.
8. Επανάληψη για όλους τους κόμβους στην γειτονία BMU: Ενημέρωση *w_ij* του πρώτου κόμβου στη γειτονιά BMU προσθέτοντας ένα κλάσμα διαφοράς μεταξύ του vector εισόδου *x(t)* και του βάρους *w(t)* του νευρώνα.
9. Επανέλαβε ολόκληρη την επανάληψη μέχρι να φτάσει στο όριο της επανάληψης *t=n*.

Το πρώτο βήμα είναι η φάση αρχικοποίησης και τα υπόλοιπα είναι η φάση εκπαίδευσης. Τα βάρη μέσα στην γειτονιά ενημερώνονται ως εξής:

$$w_{ij}(t+1) = w_{ij} + a_i(t)\beta_{ij}\left[x(t) - w_{ij}\right] \quad (1)$$

όπου *a(t)* είναι ο ρυθμός εκμάθησης με τιμές μεταξύ [0-1] και το $\beta_{ij}(t)$ είναι η μέθοδος γειτονίας, *t* είναι η τρέχων επανάληψη, *n* είναι το όριο επανάληψης.

Η ακτίνα και ο ρυθμός εκμάθησης μειώνονται κατά τη διάρκεια του χρόνου:

$$\sigma(t) = \sigma_0 * exp\frac{-t}{\lambda}, \quad όπου\ t = 1,2,\dots n \quad (2)$$

$$\alpha(t) = \alpha_0 * exp\frac{-t}{\lambda}, \quad όπου\ t = 1,2,\dots n \quad (3)$$

Η μέθοδος γειτονίας υπολογίζεται:

$$\beta_{ij}(t) = exp\frac{-d^2}{2\sigma^2(t)}, όπου\ t = 1,2,\dots n \quad (4)$$

Η Ευκλείδεια απόσταση κάθε κόμβου και του τρέχων εισόδου υπολογίζεται:

$$\left\|\vec{x} - \vec{w}_{ij}\right\| = \sqrt{\sum_{t=0}^{n}\left[\vec{x}(t) - \vec{w}_{ij}(t)\right]^2} \quad (5)$$

Το *d* είναι η απόσταση μεταξύ του κόμβου και του BMU και υπολογίζεται:

$$d = min\left(\left\|\vec{x} - \vec{w}_{ij}\right\|\right) = min\left(\sqrt{\sum_{t=0}^{n}\left[\vec{x}(t) - \vec{w}_{ij}(t)\right]^2}\right) \quad (6)$$

2.4 Εφαρμογή SOM στο TSP

Η πρώτη εφαρμογή SOM στο TSP έγινε από τους Bernard Angeniol, Gael de la Croix vaubois και Jean-Yves le texier το 1988 [56], λίγα χρόνια μετά την εισαγωγή του SOM από τον Kohonen. Μια ευκολονόητη άποψη του αλγορίθμου έχει ως εξής: ένα βήμα επανάληψης

αποτελείται με την παρουσίαση μια πόλης. Ο κόμβος που βρίσκεται πλησιέστερα στην πόλη που παρουσιάζεται κινείται προς την κατεύθυνση της και προκαλεί τους γείτονές του στο ring να κάνουν το ίδιο, αλλά με μειωμένη ένταση κατά μήκος του ring. Αυτή η συσχέτιση μεταξύ της κίνησης των γειτονικών κόμβων, ενστικτωδώς οδηγεί σε μια ελαχιστοποίηση της απόστασης μεταξύ δύο γειτόνων, δίνοντας έτσι μια σύντομη διαδρομή. Οι κόμβοι θα γίνουν προοδευτικά ανεξάρτητοι μεταξύ τους και, τελικά, ο καθένας θα συνδεθεί σε μία πόλη. Οι πόλεις στον αλγόριθμο έχουν τον αριθμό από 1 έως *M*. Κάθε πόλη *i* δείχνεται από τις δυο συντεταγμένες $(x_1^i,\ x_2^i)$. Οι κόμβοι έχουν τον αριθμό από 1 έως *N*. Κάθε κόμβος *j* στο ring χαρακτηρίζεται από τις δυο συντεταγμένες $(c_1^j,\ c_2^j)$. Κάθε κόμβος επίσης σχετίζεται με δυο γείτονες στο ring: κόμβοι *j* - 1 (mod *N*) και *j* + 1 (mod *N)*. Στην αρχή της διαδικασίας μόνο ένας κόμβος υπάρχει ο οποίος βρίσκεται στο σημείο (0,0). Ο αριθμός των κόμβων *N* μεγαλώνει στη συνεχεία με βάση τη διαδικασία δημιουργίας κόμβου. Σε κάθε βήμα περνιέται και μια *i* πόλη. Κάθε ολοκληρωμένη επανάληψη παίρνει *M* διαδοχικά βήματα: για *i* = 1 έως *i* = *M,* έτσι επιλεγεί κάθε πόλη μια φορά σε προκαθορισμένη σειρά. Αυτή η σειρά μπορεί να επιλεχθεί στη τύχη πριν την εκκίνηση της προσομοίωσης και έπειτα τροποποιείται κατά την διάρκεια της διαδικασίας της χαλάρωσης. Διαφορετική αρχική σειρά οδηγεί σε διαφορετικές λύσεις. Μια gain παράμετρος χρησιμοποιείται η οποία μειώνεται ανάμεσα σε δυο ολοκληρωμένες επαναλήψεις. Χρειάζονται μερικές επαναλήψεις από την υψηλή gain τιμή προς την χαμηλή τιμή για το τελικό αποτέλεσμα. Εξετάζοντας την πόλη *i* που περιλαμβάνει τα ακόλουθα βήματα:

1.3 Βρες τον κόμβο j_c ο οποίος είναι ο κοντινότερος στην πόλη *i*: για κάθε κόμβο *j*, υπολόγισε το ενδεχόμενο:

$$V_j = (x_1^i - c_1^j)^2 + (x_2^i - c_2^j)^2 \quad (1)$$

και καθόρισε το j_c από τον ανταγωνισμό:

$$V_{jc} = minV_j \quad (2)$$

2.3 Μετακίνησε τον κόμβο j_c και τους γείτονες του στο ring προς την πόλη *i*. Η απόσταση όπου κάθε κόμβος θα μετακινηθεί ορίζεται από μια συνάρτηση *f(G,n)*, όπου *G* είναι η gain παράμετρος και *n* είναι η απόσταση που μετρήθηκε ανάμεσα στους κόμβου *j* και j_c:

$$n = \inf(j - j_c(mod\ N), j_c - j(mod\ N)) \quad (3)$$

Κάθε κόμβος *j* μετακινείται από την θέση του $(c_1^j,\ c_2^j)$ σε μια καινούρια:

$$c_k^j \leftarrow c_k^j + f(G,n) \cdot (x_k^i - c_k^j)^2 \quad (4)$$

Η συνάρτηση *f* ορίζεται:

$$f(G,n) = (\tfrac{1}{\sqrt{2}}) \cdot \exp(\tfrac{-n^2}{G^2}) \quad (5)$$

Το οποίο σημαίνει:

- όταν G → ∞, όλοι οι κόμβοι μετακινούνται προς την πόλη *i* με την ίδια δύναμη $(\tfrac{1}{\sqrt{2}})$
- όταν G → 0, μόνο ο κόμβος j_c μετακινείται προς την πόλη *i*.

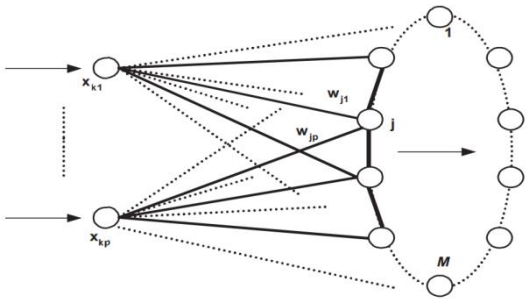
Μείωση του gain στο τέλος μιας ολοκληρωμένης έρευνας:

$$G \leftarrow (1 - a) \cdot G \quad (6)$$

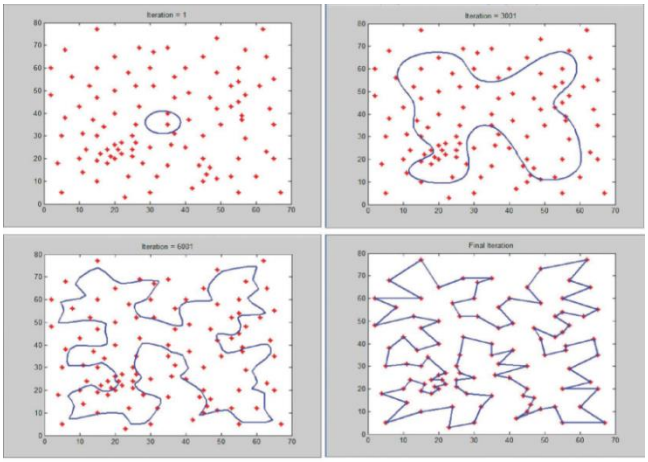
όπου α είναι η μοναδική παράμετρος που πρέπει να τροποποιηθεί. Σχετίζεται άμεσα με τον αριθμό των πλήρων ερευνών που απαιτούνται για την επίτευξη του αποτελέσματος, επομένως με την ποιότητά του. Το gain μειώνεται από ένα υψηλό αρχικό G_i το οποίο εξασφαλίζει μεγάλες κινήσεις για όλους τους κόμβους σε κάθε βήμα επανάληψης, σε ένα επαρκώς χαμηλό G_f για το οποίο σταθεροποιείται το δίκτυο. Αυτή η προσέγγιση έχει τα εξής χαρακτηριστικά: Πρώτον, ο συνολικός αριθμός κόμβων και συνδέσεων σε μια παράλληλη εφαρμογή σύνδεσης θα είναι ανάλογος μόνο με τον αριθμό των πόλεων στο πρόβλημα, με αποτέλεσμα να κλιμακώνεται πολύ καλά με το μέγεθος του προβλήματος. Δεύτερον, μόνο μία παράμετρος, η οποία ελέγχει άμεσα τον συνολικό αριθμό επαναλήψεων, πρέπει να συντονιστεί. Τέλος, οι τυπικές τιμές αυτής της παραμέτρου εξασφαλίζουν μια καλή λύση σε εύλογο χρονικό διάστημα για τις προσομοιώσεις που έγιναν. Μόνο η παράμετρος α χαρακτηρίζει τη μείωση του κέρδους πρέπει να ρυθμιστεί. Σχετίζεται άμεσα με τον αριθμό των πλήρων ερευνών που απαιτούνται για την επίτευξη του αποτελέσματος, επομένως με την ποιότητά του. Μια χαμηλή τιμή του α δίνει έναν καλύτερο μέσο όρο, αλλά μια υψηλή τιμή έχει το πλεονέκτημα ότι το βέλτιστο αποτέλεσμα είναι μερικές φορές εφικτό να επιτευχθεί. Δίνει επίσης μια καλή λύση σε πολλές προσπάθειες σε λιγότερο χρόνο από μία προσπάθεια με χαμηλή τιμή α. Σε κάθε περίπτωση, μια καλή μέση λύση (λιγότερο από 3% μεγαλύτερη από τη βέλτιστη) μπορεί να επιτευχθεί σε 2 δευτερόλεπτα σε κλασικό hardware. Πραγματοποιήθηκαν προσομοιώσεις σε μικρά σύνολα πόλεων.

Το Σχήμα 2.1 δείχνει ένα σχήμα ενός δικτύου τύπου SOM για το TSP. Ένας δακτύλιος νευρώνων εξόδου, που συμβολίζεται με 1, 2, έως *M*, χρησιμοποιείται για να χαρακτηρίσει τον χάρτη. Οι νευρώνες εισόδου, που λαμβάνουν τα δεδομένα της πόλης εισόδου (ας πούμε, τιμές συντεταγμένων), είναι πλήρως συνδεδεμένοι με κάθε νευρώνα εξόδου.

Το Σχήμα 2.2 απεικονίζει το βήμα αρχικοποίησης, ένα ενδιάμεσο και την τελική κατάσταση της εκπαιδευτικής διαδικασίας. Η λύση αναπαρίσταται στο τέταρτο γράφημα. Τα χαρακτηριστικά αυτού του αλγορίθμου κατά τη διάρκεια κάθε γύρου επεξεργασίας μπορούν να εμφανιστούν σε δισδιάστατα επίπεδα όπως φαίνεται στο σχήμα. Οι κόκκινες κουκίδες είναι οι πόλεις ενός TSP που είναι οι συντεταγμένες μιας τοποθεσίας. Ο μπλε κύκλος είναι ο δακτύλιος που συνήθως ξεκινά από μια τυχαία τοποθεσία. Μετά από αυτό, ο κύριος αλγόριθμος του SOM αρχίζει να παρακολουθεί την επόμενη τοποθεσία και το convex-hull εξαπλώνεται στον άξονα x & y κατά τη διάρκεια επαναλήψεων που εκτελούν τον αλγόριθμο. Η τελική επανάληψη δείχνει την καλύτερη λύση για τον χάρτη TSP.



Σχήμα 2.1: Αρχιτεκτονική SOM-TSP



Σχήμα 2.2: Τα στάδια της λύσης του TSP

2.5 Επίλογος

Εκ πρώτης όψεως το TSP είναι ένα πολύ σημαντικό πρόβλημα που έχει ως στόχο την ελαχιστοποίηση του κόστους εκπληρώνοντας όλα τα σημεία που πρέπει να επισπευτεί. Το SOM είναι ένα νευρωνικό δίκτυο το όποιο λύνει αυτό πρόβλημα.

Κεφάλαιο 3ο: Τα μοντέλα SOM

3.1 Εισαγωγή

Σε αυτό το κεφάλαιο θα παρουσιαστούν αρχικά οι κατηγορίες των μοντέλων SOM που επιλύουν το TSP καθώς και τις υποκατηγορίες που θα χωριστούν ύστερα τα μοντέλα.

3.2 Κατηγορίες

Τα μοντέλα κατηγοριοποιήθηκαν ανάλογα με την πολυπλοκότητα του αλγορίθμου που χρησιμοποίησε το κάθε ένα μοντέλο. Ορισμένα μοντέλα είναι πιο περίπλοκα για να κατασκευαστούν και άλλα με λιγότερες τροποποιήσεις. Υπάρχουν δυο κατηγορίες μοντέλων. Η πρώτη είναι οι εμπλουτισμένοι Χάρτες Αυτό-Οργάνωσης που περιλαμβάνει το βασικό SOM με κάποιες αναβαθμίσεις στον αλγόριθμο και στην δεύτερη κατηγορία είναι τα υβριδικά μοντέλα. Υπάρχει και μια ακόμη κατηγορία η οποία αν και χρησιμοποιεί τεχνική SOM, αντί για το κλασσικό TSP, λύνει το πρόβλημα των πολλαπλών πωλητών και όχι για έναν αποκλειστικά.

Ένας υβριδικός αλγόριθμος είναι ένας αλγόριθμος που συνδυάζει δύο ή περισσότερους άλλους αλγόριθμους που επιλύουν το ίδιο πρόβλημα. Δεν έχει να κάνει απλώς στον συνδυασμό πολλαπλών αλγορίθμων για την επίλυση ενός διαφορετικού προβλήματος - πολλοί αλγόριθμοι μπορούν να θεωρηθούν ως συνδυασμοί απλούστερων κομματιών - αλλά μόνο στον συνδυασμό αλγορίθμων που επιλύουν το ίδιο πρόβλημα, αλλά διαφέρουν σε άλλα χαρακτηριστικά, κυρίως την απόδοση. Τα πλεονεκτήματα των υβριδικών μεθόδων είναι οι υψηλές επιδόσεις, καλύτερα υπολογιστικά αποτελέσματα και μεγαλύτερη ευελιξία σε δεδομένα μεγάλης κλίμακας. Τα μειονεκτήματα ενός υβριδικού αλγορίθμου είναι κυρίως η πολυπλοκότητα των αλγορίθμων στην κατασκευή του πρωταρχικού μοντέλου, επειδή το δίκτυο είναι ένας συνδυασμός δύο ή περισσοτέρων μεθόδων, επομένως είναι δύσκολο να κατασκευαστεί. Επίσης, ορισμένοι αλγόριθμοι δεν έχουν καλά αποτελέσματα σχετικά με το χρόνο που αφιερώνεται για να ολοκληρωθούν. Τα υβριδικά μοντέλα περιλαμβάνον Greedy, Evolutionary, Genetic, Memetic, Chaos, και Fuzzy αλγόριθμους που θα αναλυθούν σε αργότερο κεφάλαιο.

Greedy

Οι Greedy αλγόριθμοι ή στα Ελληνικά άπληστοι αλγόριθμοι έχουν την ιδεολογική δομή που δημιουργεί μια λύση κομμάτι-κομμάτι, δηλαδή βήμα βήμα, επιλέγοντας πάντα το επόμενο κομμάτι που προσφέρει το πιο προφανές και άμεσο όφελος. Οπότε για προβλήματα όπου επιλέγεται η βέλτιστη λύση τοπικά και οδηγεί επίσης σε παγκόσμιες λύσεις είναι η καλύτερη εφαρμογή για Greedy αλγόριθμους. Το πλεονέκτημα της χρήσης ενός άπληστου αλγορίθμου είναι ότι οι λύσεις σε μικρές περιπτώσεις του προβλήματος μπορεί να είναι απλές και κατανοητές. Το μειονέκτημα είναι ότι είναι απολύτως πιθανό ότι οι βέλτιστες βραχυπρόθεσμες λύσεις μπορεί να οδηγήσουν στο χειρότερο δυνατό μακροπρόθεσμο αποτέλεσμα.

Evolutionary

Ένας Evolutionary ή στα Ελληνικά εξελικτικός αλγόριθμος είναι μια εξελικτική εφαρμογή υπολογιστών που λύνει προβλήματα με τη χρήση διαδικασιών που μιμούνται τις συμπεριφορές των ζωντανών πραγμάτων. Ως εκ τούτου, χρησιμοποιεί μηχανισμούς που συνήθως συνδέονται με τη βιολογική εξέλιξη, όπως η αναπαραγωγή, η μετάλλαξη και ο ανασυνδυασμός. Το

μειονέκτημά του είναι ότι απαιτεί υπολογιστική ισχύ. Οι εξελικτικοί αλγόριθμοι λειτουργούν σε μια διαδικασία φυσικής επιλογής που μοιάζει με Darwinian, δηλαδή οι πιο αδύναμες λύσεις εξαλείφονται ενώ οι ισχυρότερες, πιο βιώσιμες επιλογές διατηρούνται και επαναξιολογούνται στην επόμενη εξέλιξη με στόχο την επίτευξη βέλτιστων ενεργειών για την επίτευξη των επιθυμητών αποτελεσμάτων.

Genetic

Ένας Genetic ή στα Ελληνικά γενετικός αλγόριθμος είναι μια ευρετική μέθοδος αναζήτησης που χρησιμοποιείται στην τεχνητή νοημοσύνη και τους υπολογιστές. Χρησιμοποιείται για την εύρεση βελτιστοποιημένων λύσεων σε προβλήματα αναζήτησης με βάση τη θεωρία της φυσικής επιλογής και την εξελικτική βιολογία. Οι γενετικοί αλγόριθμοι είναι εξαιρετικοί για αναζήτηση μέσω μεγάλων και σύνθετων συνόλων δεδομένων. Υπάρχουν αρκετά μειονεκτήματα στη χρήση γενετικών αλγορίθμων, μπορεί να είναι δαπανηρά στην εφαρμογή τους, μπορεί να είναι δύσκολο να κατανοηθούν και μπορεί να είναι δύσκολο να εντοπιστούν σφάλματα

Memetic

Οι Memetic ή στα Ελληνικά μιμητικοί αλγόριθμοι μπορούν να θεωρηθούν ως ένας γάμος μεταξύ μιας παγκόσμιας τεχνικής με βάση τον πληθυσμό και μιας τοπικής αναζήτησης που γίνεται από κάθε ένα από τα άτομα. Είναι ειδικοί τύποι γενετικών αλγορίθμων με τοπική αναρρίχηση. Όπως και οι γενετικοί αλγόριθμοι, οι μιμητικοί αλγόριθμοι είναι μια προσέγγιση με βάση τον πληθυσμό.

Chaotic

Ένας χαοτικός χάρτης αυτο-οργάνωσης μπορεί να παραχθεί αντικαθιστώντας τις γραμμικές νευρικές μονάδες του συμβατικού αυτοοργανούμενου χάρτη με νευρικές μονάδες ικανές να παράγουν χάος.

Fuzzy

Η Fuzzy ή αλλιώς η ασαφής λογική είναι μια προσέγγιση στην επεξεργασία μεταβλητών που επιτρέπει την επεξεργασία πολλαπλών πιθανών τιμών αλήθειας μέσω της ίδιας μεταβλητής. Η ασαφής λογική προσπαθεί να λύσει προβλήματα με ένα ανοιχτό, ανακριβές φάσμα δεδομένων και ευρετικών που καθιστά δυνατή την απόκτηση μιας σειράς ακριβών συμπερασμάτων. Η ασαφής λογική έχει σχεδιαστεί για να επιλύει προβλήματα λαμβάνοντας υπόψη όλες τις διαθέσιμες πληροφορίες και λαμβάνοντας την καλύτερη δυνατή απόφαση δεδομένης της εισόδου.

Simulated annealing

Το Simulated annealing (SA) είναι μια πιθανοτική τεχνική για την προσέγγιση του ολικού βέλτιστου μιας δεδομένης συνάρτησης. Συγκεκριμένα, είναι μια μεταευρετική προσέγγιση της καθολικής βελτιστοποίησης σε ένα μεγάλο χώρο αναζήτησης για ένα πρόβλημα βελτιστοποίησης. Χρησιμοποιείται συχνά όταν ο χώρος αναζήτησης είναι διακριτός.

Elastic Net

Η EN γραμμική παλινδρόμηση χρησιμοποιεί τις ποινές τόσο από τις τεχνικές λάσου όσο και από τις τεχνικές κορυφογραμμής για να τακτοποιήσει τα μοντέλα παλινδρόμησης. Η τεχνική

συνδυάζει τόσο τις μεθόδους παλινδρόμησης λάσου όσο και κορυφογραμμής μαθαίνοντας από τις αδυναμίες τους για να βελτιώσει την κανονικοποίηση των στατιστικών μοντέλων.

3.3 Εμπλουτισμένα μοντέλα SOM

Convex-hull

Το Convex-hull ενός συνόλου σημείων σε έναν δισδιάστατο Ευκλείδειο χώρο ορίζεται ως το μικρότερο (από άποψη περιοχής) κυρτό πολύγωνο που περιλαμβάνει όλα τα σημεία του συνόλου.

3.3.1 TOPSOM

Το TOpology Preserving Self-Organizing Map (TOPSOM) [4] είναι ένα διπλό στρώμα SOM, το όποιο παίρνει ως είσοδο τις συντεταγμένες πόλεων $X = [\vec{x}_1; \vec{x}_2; \cdots; \vec{x}_N]$, όπου N είναι ο αριθμός των πόλεων που θα εισέρθουν. Οι νευρώνες στο στρώμα εξόδου δείχνονται από το matrix $W = [\vec{w}_1; \vec{w}_2; \cdots; \vec{w}_M]$ όπου M είναι ο αριθμός των νευρώνων εξόδου. $M = aN$ όπου a είναι σταθερά και $\vec{w}_j = [w_{j1}, w_{j2}]$ είναι οι συντεταγμένες του νευρώνα εξόδου με δείκτη j. Οι νευρώνες στο στρώμα εξόδου μπορούν να μπουν σε έναν πίνακα και επομένως η τοπολογική σειρά στους νευρώνες εξόδου μπορεί να δημιουργηθεί. Κατά τη διαδικασία εκμάθησης του, το TOPSOM ρυθμίζει το στρώμα εξόδου, χαρτογραφεί τους νευρώνες στα nodes των πόλεων εισόδου και στη συνέχεια σχηματίζει μια διαδρομή με ελάχιστο μήκος που διασχίζει όλες τις πόλεις και να επιστρέφει στο σημείο εκκίνησης. Ο στόχος αυτής της μεθόδου είναι να ικανοποιήσει δύο περιορισμούς. Το Index topology-preserving and Convex Hull. Στο index topology-preserving, το τελικό tour που διασχίζει όλες τις πόλεις εισόδου παράγεται από την σειρά των index των σχετιζόμενων νευρώνων εξόδου τους. Το tour θα είναι κοντινό αν ο νευρώνας εξόδου μαζί με διπλανά index είναι γείτονες και στο φυσικό σύστημα συντεταγμένων επίσης. Έτσι έχουμε βάλει στόχο να εφαρμόσουμε στα nodes του επιπέδου εξόδου που έχουν συνεχόμενα index κοντά το ένα στο άλλο στο φυσικό σύστημα συντεταγμένων. Όμως είναι δύσκολο να ικανοποιήσουμε το Index topology-preserving. Ο κύριος λόγος είναι ο εξής. Οι νευρώνες εξόδου θα ανταγωνιστούν ο ένας τον άλλον για να κάνουν fit στις πόλεις εισόδου. Κατά τη διάρκεια του ανταγωνισμού η τοπολογία του ring μπορεί να καταστραφεί δηλαδή οι νευρώνες εξόδου δίπλα στο ring δεν είναι γείτονες στο φυσικό σύστημα συντεταγμένων. Επομένως το τελικό tour θα είναι μεγαλύτερο από αυτό με topology-preserving. Για να λυθεί αυτό έχουμε το ring distance στον κανόνα εκμάθησης

$$d(j, K) = \min(|j - K|, M - |j - K|) \qquad (4)$$

όπου το $d(j, K)$ περιγράφει τις index διαφορές μεταξύ νευρώνων και νικητή νευρώνα.

Με το convex-hull μπορεί αποδοτικά να μειώσει το πρόβλημα των path crossing το όποιο είναι ένα μεγάλο εμπόδιο στο να φτάσει το optimal route. Κατά τη διάρκεια ανταγωνιστικής εκπαίδευσης η συχνότητα επίσκεψης κάποιων νευρώνων εξόδου που βρίσκονται στο convex-hull μπορεί να αλλάξουν. Για να λυθεί αυτό φτιάχτηκε ένας elastic competitive Hebbian κανόνας εκμάθησης. Τα βήματα του αλγορίθμου έχουν ως εξής:

Κανονικοποίηση και τυχαία αρχικοποίηση εφαρμόζεται σε διάφορα πεδία μηχανικής μάθησης λαμβάνοντας ως πλεονεκτήματα την μείωση υπολογισμού και επιτάχυνση της προσέγγισης. Εισάγεται μια αναλογική μέθοδος κλιμάκωσης min-max:

$$x_{ij} = \frac{x_{ij} - c[j]}{max} \quad (1)$$

Όπου $c[j] = maxX(:, j)$ και $d[j] = minX(:, j)$. Με αυτόν τον τρόπο οι οριζόντιες και κατακόρυφες συντεταγμένες των πόλεων εισόδου κανονικοποιούνται σε [0,1].

Επιλογή νικητή όπου οι πόλεις τροφοδοτούνται στην είσοδο με τυχαία σειρά σε t χρόνο. Προστίθεται περισσότερη τυχαιότητα με σκοπό να βρει καλύτερο tour. Μετά οι νευρώνες εξόδου ανταγωνίζονται μεταξύ τους με βάση την Ευκλείδεια απόσταση. Ο νικητής νευρώνας είναι $\vec{w}_K(t)$, η κοντινότερη πόλη εισόδου $\vec{x}_i(t)$ και ορίζεται:

$$K = argmin\left\|\vec{x}_i(t) - \vec{w}_j(t)\right\|_2^2 j \in [1, M] \quad (2)$$

Βελτιωμένη διαδικασία εκμάθησης Hebbian λαμβάνοντας υπόψη τους υπολογιστικούς πόρους, ενημερώνονται μόνο ο νικητής νευρώνας και οι γείτονες του, του οποίου το index $j \in R(j; K, t)$. Το set της γειτονιάς μπορεί να δηλωθεί ως:

$$R(j; K, t) = \{j \| j - K| \leq max(\gamma^t M/2\,, \alpha/2)\} \quad (3)$$

Παρουσιάζεται ένας Elastic Hebbian κανόνας εκμάθησης με σκοπό να διατηρήσει την τοπολογία των node να μπορούν να διασχίζονται έτσι ώστε η τοπολογική σειρά των nodes να είναι καλά διατηρημένη στο φυσικό σύστημα συντεταγμενων.

$$W(t+1) = W(t) + \eta(t) G(j, K, t)\left(\vec{x}_i(t) - W(t)\right) \quad (5)$$

όπου

$$G(j, K, t) = \begin{cases} exp\left\{\frac{-d^2(j,K)}{2\sigma^2(t)}\right\}, & if \;\; j \in R(j; K, t) \\ 0, & otherwise. \end{cases} \quad (6)$$

Μόνο ο νικητής νευρώνας $\vec{w}_K(t)$ και οι γείτονες του $\vec{w}_j(t), j \in R(j; K, t)$ θα ενημερωθούν. Το $\sigma(t)$ στην εξίσωση (6) φθείρεται συνέχεια. Ο ρυθμός εκμάθησης $\eta(t) = \eta(0)\beta^t$, όπου β είναι αναλόγια που μειώνεται ανάμεσα σε (0,1).

Input: The coordinates of input cities **X**
Output: The synaptic weights of output neurons **W**
1: *Normalization and Random Initialization*
Proportional Min-Max Scaling using Eq. (1)
Initialize M output neurons randomly
$t \leftarrow 1$
2: **while** the termination condition is not achieved **do**
3: *Winner selection*
Calculate the winner neuron's index according to Eq. (2)
4: *Improved Hebbian Learning process*
Find the winning neighborhood $R(j, K, t)$ following Eq. (3) and update the neurons using Eqs. (5) and (6)
5: $t \leftarrow t + 1$
6: **end while**
7: *Final tour construction*
Get the final route traversing all input cities. The order of traversing all the input cities is determined by the order of its associated (*i.e.* nearest) neuron.

Σχήμα 3.1: Το γενικό πλαίσιο του TOPSOM

Οι διαφορές ανάμεσα με το κλασσικό SOM είναι ότι κατά την αρχικοποίηση, το παραδοσιακό SOM συνήθως οργανώνει τους νευρώνες εξόδου σε ένα κύκλωμα μη διασταύρωσης ή πολύγωνο αυστηρά. Αντ 'αυτού, το προτεινόμενο TOPSOM λασκάρει τον περιορισμό ενός ring χωρίς διασταύρωση και επεκτείνει τις λύσεις σε μεγάλο βαθμό. Δεύτερον οι κανόνες εκμάθησης είναι διαφορετικοί. Το παραδοσιακό SOM ενημερώνει τους νευρώνες εξόδου λαμβάνοντας υπόψη τις αποστάσεις τους στον νικητή νευρώνα στο σύστημα φυσικών συντεταγμένων. Το TOPSOM ενημερώνει τους νευρώνες εξόδου με έναν πιο δυναμικό τρόπο λαμβάνοντας υπόψη την ring απόσταση (4) για τον νικητή νευρώνα. Από την άλλη όμως το παραδοσιακό SOM ενημερώνει όλους τους νευρώνες κατά την διάρκεια την εκμάθησης. Σε αντίθεση όμως με το TOPMSOM το όποιο ενημερώνει μόνο τους νευρώνες εξόδου στην εξίσωση (3) λαμβάνοντας υπόψη το κενό μνήμης.

3.3.2 ORC-SOM

Στο overall-regional competitive SOM (ORC-SOM) [5] υπάρχουν δυο κανόνες ανταγωνισμού, το overall competition, και το regional competition τα όποια παρουσιάζονται στον ανταγωνιστικό αλγόριθμο εκμάθησης του κυκλικού κλασσικού SOM. Το overall competition χρησιμοποιείται για να δημιουργήσει το outlining της διαδρομής, καθώς το regional competition είναι για να το βελτιστοποιήσει για να αποκτήσει την βέλτιστη ή/και κοντά στη βέλτιστη λύση του TSP. Εισάγεται μια ακτίνα για να αποφασίσει εάν μια πόλη εισόδου πρέπει να χρησιμοποιηθεί για outlining ή για refining του τωρινού tour, και ποικίλλει σταδιακά από μηδέν έως κάποια θετική τιμή στη διαδικασία μάθησης για να επικεντρωθεί περισσότερο στο overall competition για outline στην αρχή της αναζήτησης και να εστιάσει περισσότερο στο regional competition για refining του tour έτσι ώστε να μπορεί να ανακαλυφθεί η βέλτιστη και/ή κοντά στη βέλτιστη λύση. Το ORC-SOM θεωρείται και ως μια απλή επέκταση του κλασσικού SOM με έναν ενσωματωμένο overall-regional competitive κανόνα εκμάθησης. Η βασική του ιδέα είναι ως εξής: Το κλασσικό SOM απλά μαθαίνει μια πόλη εισόδου X κάνοντας μετακίνηση τους νευρώνες προς το μέρος του με μια μετατόπιση καθ' όλη την διάρκεια της διαδικασίας εκμάθησης. Στο ORC-SOM η διαδικασία εκμάθησης δεν πρέπει να θεωρείται ίδια. Πρέπει να ξεκίνα από το outlining προς το refining της διαδρομής. Επίσης σε κάθε επανάληψη αν μια πόλη εισόδου βρίσκεται μακριά από το τωρινό tour θα πρέπει να ενημερώνει το tour λιγότερο έτσι ώστε να πραγματοποιείται το outlining ενώ αν βρίσκεται κοντά στο τωρινό tour θα πρέπει να ενημερώνει το tour περισσότερο έτσι ώστε να πραγματοποιείται το refining. Για αυτό τον σκοπό ορίστηκε το λ-region μια X πόλης. Το λ-region, που εμφανίζεται ως $\varepsilon_\lambda(X)$, ορίζεται ως η περιοχή στο διάστημα εισόδου του οποίου η Ευκλείδεια απόσταση από την πόλη εισόδου X είναι μικρότερη ή ίση με το λ.

Ο αλγόριθμος εκμάθησης του ORC-SOM που λύνει το Ευκλείδειο 2-D TSP για N πόλεις:

Είσοδος: συντεταγμένες πόλεων N σε δισδιάστατο χώρο.

Έξοδος: ένα βέλτιστο ή σχεδόν βέλτιστο tour.

Παράμετρος:

κυκλικό SOM με δύο εισόδους και κατάλληλους νευρώνες εξόδου M, $M > N$

αριθμός n_{max} βρόχων για τερματισμό.

Διαδικασία εκμάθησης:

1. Αρχικοποίηση *M* νευρώνων ώστε να είναι στατιστικά ομοιόμορφα διαμορφωμένοι σε ορθογώνιο frame ορισμένοι από τις μικρότερες και τις μεγαλύτερες συντεταγμένες των πόλεων του TSP και το *cd* να είναι το διαγώνιο length του frame. Αριθμός επαναλήψεων *n* = 0. Το *cd* είναι το διαγώνιο μήκος του ορθογώνιου πλαισίου που ορίζεται από τις μικρότερες και τις μεγαλύτερες συντεταγμένες των πόλεων στο TSP.
2. Επιλογή πόλης *X* από τις πόλεις του TSP τυχαία και προώθηση στο ORC-SOM.
3. Εύρεση του νικητή νευρώνα *i(X)* κοντινότερο στην πόλη εισόδου *X* με βάση την Ευκλείδεια απόσταση δηλαδή:

$$i(X) = argmin\|X(n) - W_j(n)\| \quad j = 1,2,\cdots,M \tag{1}$$

4. Εκπαίδευση νευρώνων χρησιμοποιώντας την εξής φόρμουλα:

$$W_j(n+1) = W_j(n) + Z\left(n, d_{X,i(X)}, \lambda(n)\right)\eta(n)h_{j,i(X)}(n)\left[X(n) - W_j(n)\right] \tag{2}$$

Για j = *i(X),i(X)* ± *1,i(X)* ± *2,…,i(X)* ± *σ(n)*, όπου *η(n)* είναι ο ρυθμός εκμάθησης, $h_{j,i(X)}(n)$ είναι η λειτουργιά γειτονιάς δεδομένου ότι

$$h_{j,i(X)}(n) = exp\left(-d_{j,i(X)}^2 / 2\,\sigma^2(n)\right) \tag{3}$$

Και το Displacement Coefficient Function (DCF) $Z\left(n, d_{X,i(X)}, \lambda(n)\right)$ δεδομένου ότι

$$Z\left(n, d_{X,i(X)}, \lambda(n)\right) = exp\{-\frac{d_{X,i(X)}^2}{2\lambda^2(n)} + \frac{1}{2}\} \tag{4}$$

Στις παραπάνω εξισώσεις το $d_{j,i(X)}$ είναι η απόσταση ανάμεσα στον νευρώνα *i(X)* και νευρώνα *j* στον κύκλο, $d_{X,i(X)}$ είναι η Ευκλείδεια απόσταση ανάμεσα στο *X* και $W_{i(X)}$ στην είσοδο, *σ(n)* είναι το effective-width και *λ(n)* είναι η συνάρτηση ακτίνας που υπολογίζεται:

$$\lambda(n) = \frac{cd}{4}\left(\frac{2}{1+exp\left(\frac{-n}{2}\right)} - \frac{1}{2}\right) \tag{5}$$

Για μια ομαλή μετάβαση από outlining σε refining της διαδρομής για κάθε νευρώνα, ορίζεται η ακτίνα *λ* να είναι μια αυξανόμενη function σε σχέση με την επανάληψη *n*.

5. Ενημέρωση το *σ(n)* και τον ρυθμό εκμάθησης *η(n)* με προκαθορισμένη στρατηγική μείωσης. Αν ο προκαθορισμένος αριθμός των επαναλήψεων n_{max} δεν εχει εκτελεστεί πηγαινε στο βήμα 2 με *n* = *n* + 1.
6. Ταξινόμηση των πόλεων από την έμφυτη συχνότητα των νευρώνων τις οποίες οι πόλεις έχουν καθοριστεί σε one-to-one σχέση και μετά δημιούργησε το tour του TSP. Αν δεν υπάρχει άλλη πόλη χαρτογραφημένη πηγαίνει στο βήμα 1 με μεγαλύτερο *M*, αλλιώς βγάζει στην έξοδο το tour.

3.3.3 CHSOM

Για το SOM, η μέθοδος αρχικοποίησης έχει μεγάλη επίδραση στο χρόνο επεξεργασίας για την επίτευξη μιας λογικής διαμόρφωσης τοπολογικού χάρτη για TSP. Συνήθως, όλοι οι νευρώνες παράγονται τυχαία πάνω από το χώρο δεδομένων εισόδου. Το κύριο μειονέκτημα μιας τέτοιας μεθόδου είναι ότι όλοι οι νευρώνες αρχικά ανακατεύονται. Έτσι απαιτεί μεγαλύτερο χρόνο επεξεργασίας για να επιτευχθεί μια τοπολογική διαμόρφωση που εξασφαλίζει τη γειτονιά των νευρώνων. Η άλλη μέθοδος για την αρχικοποίηση των νευρώνων είναι η δημιουργία τους σε

ορθογώνιο πλαίσιο ή σε κύκλο. Το προτεινόμενο SOM αρχικοποιείται για πρώτη φορά με πόλεις στο convex-hull ενός προβλήματος. Ταυτόχρονα, προκειμένου να διατηρηθεί η convex-0hull ιδιότητα, μια θεμελιώδης αρχή δημιουργίας και διαγραφής νευρώνα εισάγεται στο προτεινόμενο SOM. Ο αλγόριθμος CHSOM [6] (Convex-Hull SOM) περιγράφεται ως εξής:

Αρχικοποίηση μεθόδου: Για να κάνουμε χρήση της convex-hull ιδιότητας του TSP, αρχικά βρίσκουμε το convex-hull σε ένα set πόλεων. Μετά χρησιμοποιούνται οι πόλεις που βρίσκονται στο convex-hull για να αρχικοποιήσουμε το SOM. Με άλλα λόγια το SOM έχει k νευρώνες σε περίπτωση που ο αριθμός των πόλεων στο convex-hull είναι k. Η τιμή του νευρώνα παίρνει την ίδια τιμή της αντίστοιχης πόλης.

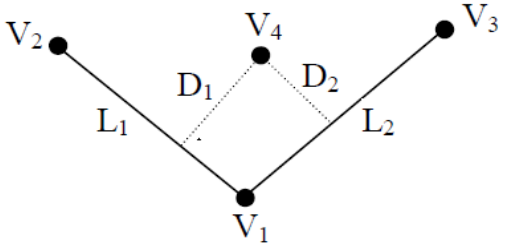
Δημιουργία νευρώνα: Η διαδικασία εκμάθησης χρησιμοποιεί την μέθοδο ενημέρωσης:

$$w_i(t+1) = w_i(t) + \alpha(t)h_i(t)\big(x_i(t) - w_i(t)\big) \quad (1)$$

$$h_i(t) = exp\left(\frac{-\|r_i - r_c\|^2}{2\sigma^2(t)}\right) \quad (2)$$

όπου *α(t)* είναι η παράμετρος εκμάθησης που παίρνει τιμές [0,1] και μειώνεται προς το μηδέν καθώς το *t* περνά, το *w(t)* είναι weight vector. Η μέθοδος γειτονιάς $h_i(t)$ έχει συνήθως σχήμα καμπάνας. Όπου r_i και r_cείναι τα weight vectors του τρέχον νευρώνα και του νικητή νευρώνα και *σ(t)* είναι η παράμετρος width.

Όμως κατά τη διάρκεια της εκμάθησης μόνο οι πόλεις που δεν βρίσκονται στο convex-hull εισέρχονται στους νευρώνες εισόδου. Πρώτα μια πόλη εισέρχεται στον νευρώνα. Τότε επιλέγεται ο νικητής με βάση τον κανόνα ανταγωνισμού. Υπάρχουν δυο περιπτώσεις στη δημιουργία νευρώνα. Η πρώτη είναι όταν ένας νευρώνας που αρχικοποιήθηκε γίνεται ο νικητής. Εκείνη την στιγμή δημιουργείται ένας νευρώνας λαμβάνοντας την ίδια τιμή με τον νευρώνα αρχικοποίησης. Στο επόμενο βήμα, ο νέος νευρώνας που δημιουργήθηκε εισέρχεται στο ring σαν ο αριστερός ή ο δεξιός γείτονας του νικητή. Προτείνετε μια αρχή για να προσδιοριστεί η θέση του δηλαδή αριστερός ή δεξιός γείτονας του νικητή. Υπάρχουν δυο γραμμές L1 & L2 που πέρανε από τον νικητή στο ring. Το L1 περνά μέσα από την αριστερή γειτονιά και το L2 από την δεξιά γειτονιά. Γίνεται υπολογισμός της απόστασης της πόλης εισόδου στις δυο γραμμές, D1(to L1) και D2(to L2). Τότε η θέση του νέου νευρώνα ως αριστερός ή δεξιός γείτονας, αποφασίζετε από το D1 και το D2. Αν D1>D2 ο νέος νευρώνας που δημιουργήθηκε εισέρχεται στο ring ως ο δεξιός γείτονας και αντίστροφα το αντίστοιχο. Το Σχήμα 3.2 δείχνει αυτήν την ιδιότητα. Η άλλη περίπτωση λέει ότι ένας νευρώνας επιλέγεται για νικητής για δυο διαφορετικές πόλεις στην ίδια μελέτη. Η θέση που εισέρχεται στο ring γίνεται τυχαία και όταν δημιουργηθεί οι γείτονες του θα ενημερωθούν με την προηγούμενη εξίσωση (1).



Σχήμα 3.2: Προσδιορισμός θέσης CHSOM

Διαγραφή νευρώνα: Ένα node διαγράφεται αν δεν έχει επιλεχθεί ως νικητής από μια πόλη κατά τη διάρκεια τριών ολόκληρων μελετών.

Μέθοδος γειτονίας: Χρησιμοποιείται η εξίσωση (2) ως μέθοδος γειτονιάς. Ωστόσο, μόλις ένας γείτονας είναι ένας νευρώνας αρχικοποίησης, η διαδικασία ενημέρωσης της γειτονιάς παραλείπει αυτόν τον νευρώνα αρχικοποίησης. Με άλλα λόγια. ενημερώνονται μόνο εκείνοι οι γείτονες που δεν είναι νευρώνες αρχικοποίησης.

Οι παράμετροι *α(t)* και *σ(t)* που χρησιμοποιήθηκαν στις εξισώσεις (1) και (2), ενημερώνονται με τις εξής εξισώσεις:

$$\alpha(t) = a(0)\left(1 - \frac{t}{T}\right) \tag{3}$$

$$\sigma(t) = 1 + (\sigma(t) - 1)\left(1 - \frac{t}{T}\right) \tag{4}$$

Όπου *α(0)* και *σ(0)* αρχικοποιούνται με 1 και 10 αντίστοιχα.

3.3.4 Expanding Property SOM

Η διαδικασία αυτής της μεθόδου [7] περιγράφεται ως εξής: μέσα σε κάθε επανάληψη, ο διεγερμένος νευρώνας τραβιέται προς την πόλη εισόδου καθώς και προς το convex-hull. Η προηγούμενη προσαρμογή, μαζί με τη συνεργατική προσαρμογή των γειτονικών νευρώνων, θα ανακαλύψει σταδιακά τη σχέση τοπολογικής γειτονιάς των δεδομένων εισόδου. Επίσης μαζί με τη συνεργατική προσαρμογή, μπορεί να προσεγγίσει την convex-hull ιδιότητα του TSP. Για να γίνει αυτό, είναι λογικό να προωθήσουμε τον διεγερμένο νευρώνα μακριά από το κέντρο των πόλεων. Η διαδικασία για την εφαρμογή της ιδέας περιγράφεται ως εξής:

1. Οι συντεταγμένες των πόλεων δηλώνονται ως $[x_{k1}, x_{k2}]^T$ για *(k=1,2,…,S)*. Το κέντρο των πόλεων χαρτογραφείται με την προέλευση *O'*.
2. Τυχαία ρύθμιση των vector βαρών $\vec{w}_j(t) = [x_{k1}, x_{k2}]^T$ για *(j=1,2,…,M)* και αρχικοποίηση των επαναλήψεων *t* = 0.
3. Επιλογή τυχαίας πόλης, $\vec{x}_k(t) = [x_{k1}(t), x_{k2}(t)]^T$ για *(k=1,2,…,S)* και τροφοδότηση της πόλης στους νευρώνες εισόδου.
4. Εύρεση του νικηφόρου νευρώνα *m(t)*, κοντά στην πόλη $\vec{x}_k(t)$ σύμφωνα με την Ευκλείδεια μετρική.
5. Εκπαίδευση του νευρώνα *m(t)* και τους γείτονές του μέσα στο πλάτος *σ(t)* χρησιμοποιώντας τον εξής κανόνα μάθησης:

$$\vec{w}_j(t+1) = c_J(t)\left[\vec{w}_j(t) + \alpha_j(t)\left(\vec{x}_k(t) - \vec{w}_j(t)\right)\right] \tag{1}$$

όπου ο ρυθμός εκμάθησης $a_j(t) \in [0,1](1 \le j \le M)$ προσδιορίζεται:

$$a_j(t) = \eta(t)h_{j,m(t)} = \eta(t)max\{0,1 - \frac{d_{j,m(t)}}{\sigma(t)+1}\} \tag{2}$$

όπου $\eta(t) \in [0,1]$ και

$$c_j(t) = \begin{cases} 1 + \frac{\alpha_j(t)\|\vec{x}_k(t) - \vec{w}_j(t)\|}{\alpha_j(t)\|\vec{x}_k(t)\| + \left(1-\alpha_j(t)\right)\|\vec{w}_j(t)\|}, \alpha_j(t) \le \frac{1}{2} \\ 1 + \frac{\left(1-\alpha_j(t)\right)\|\vec{x}_k(t) - \vec{w}_j(t)\|}{\alpha_j(t)\|\vec{x}_k(t)\| + \left(1-\alpha_j(t)\right)\|\vec{w}_j(t)\|}, \alpha_j(t) \le \frac{1}{2} \end{cases} \tag{3}$$

Η απόδειξη του βρίσκεται το paper του μοντέλου.

6. Ενημέρωση της παράμετρο πλάτους *σ(t)* και της παράμετρο μάθησης *η(t)* σύμφωνα με ένα προκαθορισμένο σχήμα μείωσης και εάν η εκμάθηση δεν τερματιστεί, μετάβαση στο βήμα 3 με *t* = *t* + 1.
7. Υπολογισμός της τιμής δραστηριότητας κάθε πόλης $\vec{x}_k(t)$ με βάση την φόρμουλα:

$$\alpha(x_k) = m_k - \lambda\{\psi(0)\|\vec{x}_k - \vec{w}_{m_k}\| + \psi\left(\frac{1}{3}\right)[\|\vec{x}_k - \vec{w}_{m_k} + 1\| - \|\vec{x}_k - \vec{w}_{m_k} - 1\|] + \psi\left(\frac{2}{3}\right)[\|\vec{x}_k - \vec{w}_{m_k} + 2\| - \|\vec{x}_k - \vec{w}_{m_k} - 2\|]\} \quad (4)$$

όπου m_k είναι ο νικητής νευρώνας που αντιστοιχεί στο $\vec{x}_k$ και $\lambda \in [-0.5,0.5]$ πρέπει να οριστεί μικρή τιμή ώστε να επηρεάζει μόνο την τοπική σειρά των πόλεων του ίδιου νικητή νευρώνα m_k. Στην εξίσωση (4) υιοθετείτε η έξης Mexican hat εξίσωση:

$$\psi(x) = (1 - x^2)e^{\frac{x^2}{2}} \quad (5)$$

8. Μπαίνουν σε σειρά οι πόλεις με τις τιμές δραστηριότητάς τους$\alpha(x_k)$ *(k=1,2,…,S)* και στη συνέχεια σχηματίζει μια περιοδεία για το TSP.

Ο κανόνας εκμάθησης ορίζεται στην εξίσωση (2) και είναι το κύριο κλειδί σε αυτήν την μέθοδο. Η κύρια διαφορά με προηγούμενους κανόνες εκμάθησης SOM είναι ότι έχει έναν επιπρόσθετο παράγοντα πολλαπλασιασμού $c_j(t)$. Αν αυτό είναι μεγαλύτερο του 1 διώχνει τον νευρώνα μακριά από την προέλευση. Αυτό το expanding αποτέλεσμα, μαζί με τη συνεργατική προσαρμογή των νευρώνων των γειτόνων, χρησιμοποιείται για να μάθει την convex-hull ιδιότητα του TSP. Η convex-hull ιδιότητα φαίνεται στο $c_j(t)$ το όποιο περιέχει πληροφορία για την απόσταση μεταξύ $\vec{x}_k, \vec{w}_{m_k}$ και τις αποστάσεις από την προέλευση $\|\vec{x}_k(t)\|$ και $\|\vec{w}_k(t)\|$.

3.3.5 ESOM

Η βασική ιδέα του Expanding SOM (ESOM) [8] είναι να ενσωματώσει την convex-hull ιδιότητα έμμεσα στον κανόνα εκμάθησης με ελάχιστο πρόσθετο υπολογισμό. Αποκτά την convex-hull ιδιότητα σταδιακά, καθώς η τοπολογική γειτονιά μεταξύ των πόλεων επιθεωρείται και διατηρείται. Ως εκ τούτου, το ESOM προσπαθεί να ικανοποιήσει τις επαρκείς και τις απαραίτητες συνθήκες των βέλτιστων διαδρομών. Αυτό πραγματοποιείται με τον ακόλουθο τρόπο: Σε μια ενιαία επανάληψη ενημέρωσης, εκτός από το να τραβήξει τον υπάρχοντα νευρώνα προς την πόλη εισόδου επεκτείνεται προς το convex-hull. Αξίζει να σημειωθεί ότι η προσέγγιση του convex-hull μπορεί να προσεγγιστεί με την απομάκρυνση από το κέντρο των πόλεων. Το κίνητρο του αναπτυγμένου ESOM είναι να προσεγγίσει μια βέλτιστη διαδρομή προσπαθώντας να ικανοποιήσει τόσο την επαρκή κατάσταση (δηλ. the neighborhood preserving property) όσο και μια απαραίτητη προϋπόθεση (δηλαδή η ιδιότητα Convex-Hull). Ονομάζεται expanding SOM (ESOM) επειδή αποδίδει την convex-hul ιδιότητα μέσω της επέκτασης των νευρώνων προς τα έξω από το κέντρο όλων των πόλεων.

Τα βήματα του αλγορίθμου είναι τα εξής:

1. Χαρτογράφηση όλων των συντεταγμένων των πόλεων $[x'_{k1}, x'_{k2}]^T$ για *(k = 1,..,n)* σε έναν κύκλο C_R κεντραρισμένο στην πηγή με ακτίνα $(R \leq 1)$. Οι νέες συντεταγμένες υποδηλώνονται με $[x_{k1}, x_{k2}]^T$. Το κεντρο των πολεων είναι χαρτογραφημένο στην πηγή (το κέντρο του κύκλου).

2. Ανάθεση των weight vector τυχαία $\vec{w}_j(0)$ για *(j=1,…,M)* μέσα στοC_R και *t*=0.
3. Επιλογή τυχαίας πόλης, $\vec{x}_k(t) = [x_{k1}(t), x_{k2}(t)]^T$ και τροφοδότηση της πόλης στους νευρώνες εισόδου.
4. Εύρεση του νικηφόρου νευρώνα *m(t)*, κοντά στην πόλη $\vec{x}_k(t)$ σύμφωνα με την Ευκλείδεια μετρική.
5. Εκπαίδευση του νευρώνα *m(t)* και των γειτόνων του μέσα στο αποτελεσματικό πλάτος *σ(t)* χρησιμοποιώντας την εξής φόρμουλα:

$$\vec{w}_j(t+1) = c_j(t)\left[\vec{w}_j(t) + \alpha_j(t)\left(\vec{x}_k(t) - \vec{w}_j(t)\right)\right] \quad (1)$$

όπου ο ρυθμός εκμάθησης $a_j(t) \in [0,1](1 \le j \le M)$ προσδιορίζεται:

$$a_j(t) = \eta(t)h_{j,m(t)} = \eta(t)max\{0,1 - \frac{d_{j,m(t)}}{\sigma(t)+1}\} \quad (2)$$

$$c_j(t) = \left[1 - 2\alpha_j(t)\left(1 - \alpha_j(t)\right)\beta_j(t)\right]^{\frac{-1}{2}} \quad (3)$$

όπου το $\beta_j(t)$ είναι η convex-hull ιδιότητα και ορίζεται ως:

$$\beta_j(t) = 1 - \vec{x}_k(t)^T\vec{w}_j(t) - \sqrt{(1 - \|\vec{x}_k(t)^2\|)\left(1 - \left\|\vec{w}_j(t)^2\right\|\right)} = 1 -$$

$$\left[x_{k1}(t), x_{k2}(t), \sqrt{1 - \|\vec{x}_k(t)^2\|}\right]\left[w_{j1}(t), w_{j2}(t), \sqrt{1 - \left\|\vec{w}_j(t)^2\right\|}\right] \quad (4)$$

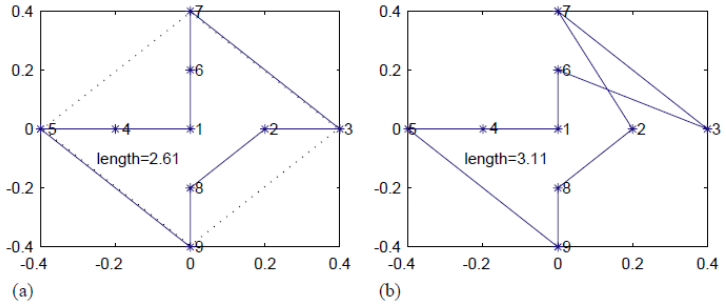
6. Ενημέρωση της παράμετρο πλάτους *σ(t)* και της παράμετρο μάθησης *η(t)* σύμφωνα με ένα προκαθορισμένο σχήμα μείωσης και εάν η εκμάθηση δεν τερματιστεί, μετάβαση στο βήμα 3 με *t* = *t* + 1.
7. Υπολογισμός της τιμής δραστηριότητας κάθε πόλης $\vec{x}_k(t)$ με βάση την φόρμουλα:

$$\alpha(x_k) = m_k - \frac{3}{26}\{\left\|\vec{x}_k - \vec{w}_{m_k}\right\| + \frac{2}{3}\left[\left\|\vec{x}_k - \vec{w}_{m_k} + 1\right\| - \left\|\vec{x}_k - \vec{w}_{m_k} - 1\right\|\right] +$$
$$\frac{1}{2}\left[\left\|\vec{x}_k - \vec{w}_{m_k} + 2\right\| - \left\|\vec{x}_k - \vec{w}_{m_k} - 2\right\|\right]\} \quad (5)$$

όπου m_k είναι ο νικητής νευρώνας που αντιστοιχεί στο $\vec{x}_k$

8. Ταξινόμηση των πόλεων με τις τιμές δραστηριότητας τους και μετά φτιάχνει το TSP tour.

Το $\left[x_{k1}(t), x_{k2}(t), \sqrt{1 - \|\vec{x}_k(t)^2\|}\right]$ και το $\left[w_{j1}(t), w_{j2}(t), \sqrt{1 - \left\|\vec{w}_j(t)^2\right\|}\right]$ στην εξίσωση (4) περιέχουν πληροφορίες για την απόσταση μεταξύ $\vec{x}_k(t)$ και $\vec{w}_j(t)$ καθώς και τις αποστάσεις από τις πηγές τους. Η εξίσωση (3) δείχνει την συνεργασία ανάμεσα στους διπλανούς νευρώνες, η τιμή 1 περιορίζει την τιμή να είναι γύρω στο 1.0 και η τιμή της δύναμης -0.5, εξασφαλίζει ότι ο υπάρχων νευρώνας δεν θα βγει έξω από τον κύκλο C_R. Τα βήματα 7 και 8 στο ESOM αποτελούν μια υλοποίηση της διαδικασίας χαρτογράφησης λύσεων. Στοχεύει να αποφέρει μια καλή διαδρομή από το ESOM μετά τη μάθηση.



Σχήμα 3.3: Η ιδιότητα convex-hull

Στο Σχήμα 3.3 (α) εφαρμόζεται η ιδιότητα convex-hull και η βέλτιστη διαδρομή είναι πολύ καλύτερη σε σχέση με το (β) που δεν εφαρμόζεται.

Τοπολογία Δακτυλίου

Κατά την τοπολογία δακτυλίου ή ring topology οι νευρώνες δημιουργούν σε τυχαίο σημείο έναν κλειστό βρόγχο ενωμένοι μεταξύ τους και ψάχνουν τις πόλεις που θα ενωθούν. Η διαφορά με το convex-hull κατά την διάρκεια της εκτέλεσης του προγράμματος, είναι ότι κατά την τοπολογία δακτυλίου το ring ανοίγει προς τα έξω ενώ αντίθετα το "ring" του convex-hull είναι ήδη ανοιχτό και κλείνει προς τα μέσα. Οι διαφορές τους είναι σημαντικές

3.3.6 MGSOM

Αρχικοποιεί έναν τυχαίο κυκλικό ring νευρώνων. Στη συνέχεια εκτείνει τους νευρώνες προς την τοποθεσία της πόλης. Χρησιμοποιώντας Ευκλείδεια απόσταση βρίσκει τον πλησιέστερο κόμβο. Τα πλεονεκτήματα του:

- Εύκολη υλοποίηση.
- Γρήγορο υπολογισμό.
- Στιβαρή εφαρμοστότατα.
- Παραγωγή καλών λύσεων.

Η αρχιτεκτονική του MGSOM [9] περιγράφεται ως εξής:

1. Αρχικοποίηση: Ορισμός n ως ο αριθμός των πόλεων. Οι εισαγωγές του αλγορίθμου είναι οι καρτεσιανές συντεταγμένες από ένα σετ n πόλεων. $m(t)$ είναι ο αριθμός των nodes σε σχήμα ρόμβου (αρχικοποίηση $m(0) = n$). Η gain παράμετρος ορίζεται $\sigma_0 = 10$. $t = 1$ και t_{min} είναι επιθυμητή minimum τιμή επαναλήψεων.
2. Τυχαιοποίηση: Ξεκίνημα υπολογισμού με διαφορετική αρχική σειρά πόλεων και χαρακτηρισμός των πόλεων 1,…, n. Το i είναι το περιεχόμενο της πόλης που παρουσιάζεται και αρχικοποιείται με $i = 1$ και επαναφορά της κατάστασης κατοίκισης όλων των node σε false. $Inhibit[j] = false for j = 1, \dots, m$.
3. Προσαρμογή παραμέτρων: Ο ρυθμός εκμάθησης α_k στο όποιο δεν χρειάζεται να οριστεί μια αρχική τιμή για αυτήν την παράμετρο διότι εξαρτάται μόνο από τον αριθμό των επαναλήψεων, και η απόκλιση της μεθόδου γειτονιάς σ_k αντίστοιχα το όποιο αρχικοποιείτε με την τιμή 10.

$$\alpha_k = \frac{1}{\sqrt[4]{k}} \quad (1)$$

$$\sigma_k = \sigma_{k-1} \times (1 - 0.01 \times k), \sigma_0 = 10 \quad (2)$$

4. Ανταγωνισμός: Μέσα από μια διαδικασία ανταγωνισμού, επιλέγεται το κοντινότερο node της πόλης i με βάση την Ευκλείδεια απόσταση. Αυτός ο διαγωνισμός γίνεται ανάμεσα μόνο στα nodes που δεν έχουν επιλεχτεί ως ο νικητής στη τρέχουσα επανάληψη. Επομένως ο νικητής νευρώνας J βγαίνει από:

$$J = argmin_j \|x_i - y_j\| \quad for\{j|Inhibit[j] = false\} \tag{3}$$

Ύστερα αλλάζει το status σε true.

5. Προσαρμογή: Με βάση τις εξισώσεις (4) και (5) ενημερώνονται τα vector του νικητή και των γειτόνων και όσων vectors διατηρούνται σε χρόνο *t*. Το μήκος της γειτονιάς του node *J* περιορίζεται σε 40% του *m(t)*. $C_j(t)$ είναι ένας μετρητής σήματος για την επιθεώρηση του ιστορικού μάθησης. Ο μετρητής σημάτων του νικητή ενημερώνεται και άλλοι μετρητές σήματος διατηρούν τις τιμές τους στο χρόνο *t*.

$$f(\sigma, d) = e^{\left(\frac{-d^2}{\sigma^2}\right)} \tag{4}$$

$$y_j^{new} = y_j^{old} + af(\sigma, d)\left(x_i - y_j^{old}\right) \tag{5}$$

$$C_j(t+1) = \begin{cases} C_j(t) + 1, & if i = J \\ C_j(t), & otherwise \end{cases} \tag{6}$$

6. Εισαγωγή node: Εισάγει ένα node κάθε T_{int} φορές εκμάθησης δια του παρόντος ο χάρτης μπορεί να μεγαλώσει έως και *m(t)=2n*. Στο $t = kT_{\text{int}}$ όπου *k* είναι ένας θετικός ακέραιος, ορίζεται ένα node *p* του οποίου ο μετρητής έχει την μέγιστη τιμή.

$$C_p(t) \geq C_j(t), \quad for \ \ all \ j \tag{7}$$

Αν εκεί υπάρχει πολλαπλός maximum μετρητής τότε επιλέγεται ένα node τυχαία. Αυτή είναι μια επιθεώρηση του ιστορικού εκμάθησης. Παίρνουμε τον γείτονα $f = p - 1$ όπου *f = mod m(t)*. Ένα node *r* εισάγεται ανάμεσα στο *p* και στο *f*. Ο synaptic vector του *r* ορίζεται:

$$y_r = 0.5\left(y_p + y_f\right) \tag{8}$$

Οι τιμές μετρητή του *p* και του *r* ξανά ορίζονται ως εξής:

$$C_p(t+1) = 0.5C_p(t), C_r(t+1) = 0.5C_p(t) \tag{9}$$

Το μεσοδιάστημα της εισαγωγής T_{int} είναι μια παράμετρος ελέγχου του αλγορίθμου. Μετά την εισαγωγή, ο αριθμός των nodes αυξάνεται: *m(t + 1) = m(t) + 1*.

7. *i = i* + 1. Αν *i < n* + 1 πήγαινε στο βήμα 4 αλλιώς όρισε *t = t* + 1 και συνέχισε τα βήματα.
8. Τεστ σύγκλισης: επιστροφή στο βήμα 2 για όσο $t \leq t_{min}$.

Για να πετύχουμε καλύτερη ποιότητα στην λύση, στις προσομοιώσεις από το MGSOM εκτελούνται τα εξής βήματα:

1. Τρέξιμο 10 προσομοιώσεων με t_{min}= 20 επαναλήψεις χρησιμοποιώντας MGSOM.
2. Επιλογή την καλύτερη λύση από την λύση των 10 προσομοιώσεων από πριν.

3. Τρέξιμο μια προσομοίωση με t_{min}= 10 επαναλήψεις εκπαίδευσης με βάρος matrix που σχετίζεται με την καλύτερη λύση που επιλέχθηκε από το προηγούμενο βήμα ως αρχικό βάρος matrix.

Η πολυπλοκότητα του αλγορίθμου (για κάθε τιμή του *t*) μπορεί να αξιολογηθεί ως εξής: Για κάθε πόλη υπάρχουν *0,3m* (μέσος όρος γειτονικού length) λειτουργίες για τον υπολογισμό της neighborhood function. Επομένως, η συνάρτηση πολυπλοκότητας του χρόνου για κάθε επανάληψη μπορεί να δοθεί από *T(m,n) = n(0.3m).* Επιπλέον, η μείωση της πολυπλοκότητας καθοδηγείται από το neighborhood function variance που όταν αναλυθεί μπορούμε να δούμε ότι ο χρόνος επεξεργασίας του αλγορίθμου μειώνεται γρήγορα (λόγω της επιλεκτικής ενημέρωσης). Αυτή η λειτουργία είναι κυρίως ελκυστική κατά την επεξεργασία μεγάλων συνόλων δεδομένων.

3.3.7 EGSOM

Η δομή του EGSOM [10] δημιουργείται με βάση τον αριθμό νικητή κάθε κόμβου και τις καταστάσεις των γειτόνων του. Κάθε πόλη αντιστοιχεί σε μια είσοδο και τυχαία εφαρμόζεται στο EGSOM. Τα πλεονεκτήματα είναι η εύκολη εφαρμοσιμότητα, ο γρήγορος υπολογισμός, οτι παράγει καλές λύσεις και ότι είναι γρήγορος σε χρόνο επεξεργασίας αλγόριθμο. Ο αλγόριθμος EGSOM είναι ο ίδιος με τον MGSOM με την διαφορά ότι στο συγκεκριμένο μοντέλο χρειάζονται $νευρώνες = 1.5 \times αριθμός\ πόλεων$ ενώ στο προηγούμενο μοντέλο ο αριθμός είναι διπλάσιος. Μικρή διαφορά αλλά αυτό που μας ενδιαφέρει είναι τα υπολογιστικά αποτελέσματα.

3.3.8 MSTSP

Το modified SOM applied to the TSP (MSTSP) [11] έχει την ίδια αρχιτεκτονική και παρόμοια φιλοσοφία με το MGSOM και το EGSOM. Η κύρια διαφορά είναι ότι το MSTSP δεν έχει το χαρακτηριστικό εισαγωγή ενός κόμβου όπου αυξάνεται ο αριθμός των κόμβων. Η τροποποιημένη προσέγγιση MSTSP περιγράφεται συνοπτικά ως εξής:

1. Αρχικοποίηση.
2. Τυχαιοποίηση.
3. Προσαρμογή παραμέτρων.
4. Προσαρμογή.
5. Ανταγωνισμός.
6. Βήμα 4 ή 7 ανάλογα με το αποτέλεσμα μιας συνθήκης.
7. Τεστ σύγκλισης.

Η πολυπλοκότητα του αλγορίθμου MSTSP για κάθε επανάληψη (για κάθε τιμή *k*) μπορεί να αξιολογηθεί ως εξής: Για κάθε πόλη, υπάρχουν *m* λειτουργίες για την εύρεση του κοντινότερου node σε αυτήν την πόλη, *0.2m* (μέσο μήκος γείτονα) λειτουργίες για τον υπολογισμό της συνάρτησης γειτονιάς. Επομένως, η συνάρτηση χρονικής πολυπλοκότητας για κάθε επανάληψη μπορεί να δοθεί από *n(m+0.2m),* και η συνάρτηση χρονικής πολυπλοκότητας του αλγορίθμου MSTSP για κάθε προσομοίωση μπορεί να δοθεί από το *T(m,n) = (10*20+10) * n (m+0.2m).* Επιπλέον, η μείωση της πολυπλοκότητας καθοδηγείται από τη διακύμανση της συνάρτησης γειτονιάς. Αναλύοντας την εξέλιξη της διακύμανσης, μπορεί να φανεί ότι ο χρόνος επεξεργασίας του αλγορίθμου μειώνεται γρήγορα (λόγω της επιλεκτικής ενημέρωσης). Αυτή η δυνατότητα είναι κυρίως ελκυστική κατά την επεξεργασία μεγάλων συνόλων δεδομένων.

Ευρετικά

Σε αυτήν την κατηγορία θα μπουν κυρίως τα εμπλουτισμένα μοντέλα SOM τα οποία είναι πιο γενικά και έχουν διαφορετικά χαρακτηριστικά.

3.3.9 Enhanced SOM Solution

Είναι μια βελτίωση του κλασικού SOM [12] χρησιμοποιώντας συντονισμό υπερπαραμέτρου για να προσαρμόσει τον αλγόριθμο και να δημιουργήσει δύο πρόσθετα χαρακτηριστικά για να βρει λύσεις με καλύτερα αποτελέσματα. Κατά την εξερεύνηση του αλγορίθμου που χρησιμοποιείται σε αυτό το μοντέλο, εντοπίζονται για πρώτη φορά οι κύριες υπερπαράμετροι του (μέγεθος πληθυσμού, αριθμός επαναλήψεων, ποσοστό μάθησης και οι μειώσεις για τα τελευταία δύο) και τις επιπτώσεις τους στις τελικές βαθμολογίες. Για να κατανοήσουμε την επιρροή καθενός από αυτούς τους παράγοντες και να βρούμε την καλύτερη διαμόρφωσή τους, χρησιμοποιείται ένας σχεδιασμός ενός παράγοντα για να συντονίσει την προτεινόμενη τεχνική, δηλαδή, μεταβάλλοντας περίπου ένα χαρακτηριστικό τη φορά για να εξετάσει την απομονωμένη του επιρροή. Η αναζήτηση των υπερπαραμέτρων που ταιριάζει καλύτερα στο σύνολο δεδομένων οδήγησε στον baseline αλγόριθμο. Η πρώτη βελτίωση στον baseline αλγόριθμο ήταν να αλλάξει ο τρόπος επιλογής του πρώτου κόμβου που εξετάστηκε. Εκτός από την τυχαία επιλεγμένη αρχική πόλη, ο αλγόριθμος αναγκάστηκε να ξεκινήσει από την πόλη στην πιο κεντρική θέση και στην πιο απομακρυσμένη θέση από το κεντροειδές όλων των πόλεων. Ως δεύτερη τροποποίηση, αφού χρησιμοποίησε τον συντονισμό των υπερπαραμέτρων και την αναγνώριση του σημαντικότερου χαρακτηριστικού ως μεγέθους του πληθυσμού, ο αλγόριθμος βελτιώθηκε με βάση την παραλλαγή αυτού του υπερπαραμέτρου σε κάθε επανάληψη SOM. Η λύση υπολογίστηκε χρησιμοποιώντας το F1 score που είναι είναι ο αρμονικός μέσος όρος της ακρίβειας και της ανάκλησης, όπου το F1 score φτάνει την καλύτερη τιμή του σε 1 και το χειρότερο σκορ στο 0 [13]. Επιλέχθηκε ως baseline για την ακόλουθη διαμόρφωση: 100000 επαναλήψεις, 0.9997 για τον ρυθμό μείωσης της αρχικής γειτονιάς, 0.8 για το ρυθμό εκμάθησης, 0.99997 για το για τον ρυθμό μείωσης εκμάθησης και 6 για τον συντελεστή πολλαπλασιαστή μεγέθους του πληθυσμού.

3.3.10 Modares, Somhom & Enkawa

Αυτό το μοντέλο [14] προτείνει έναν αλγόριθμο SOM για την επίλυση ενός σχετικά αποτελεσματικού προβλήματος TSP. Επιθεωρώντας απλώς τα δεδομένα πόλης εισόδου για κανονικότητες και μοτίβα και στη συνέχεια προσαρμόζοντας τον εαυτό του ώστε να ταιριάζει στα δεδομένα εισόδου μέσω συνεργατικής προσαρμογής των συναπτικών βαρών, ένα τέτοιο δίκτυο δημιουργεί την τοπική απόκριση στα δεδομένα εισόδου και έτσι αντικατοπτρίζει την τοπολογική διάταξη των πόλεων εισόδου. Αυτός ο χάρτης διατήρησης γειτονιάς οδηγεί στη συνέχεια σε ένα προβλεπόμενο tour στο TSP που εξετάζεται. Από κάθε πόλη, το προκύπτων tour προσπαθεί να επισκεφθεί την πλησιέστερη πόλη. Η συντομότερη υπο-περιήγηση μπορεί διαισθητικά να οδηγήσει σε μια καλή περιήγηση για το TSP. Είναι ένας από τους παλαιότερους αλγόριθμους SOM που επιλύουν το TSP. Ο προτεινόμενος αλγόριθμος μπορεί να περιγραφεί ως εξής:

1. Αρχικοποίηση: *N* είναι ο αριθμός των πόλεων και *M* ο αριθμός τον nodes. *(M=2N), a=0.03* όπου είναι ο ρυθμός εκμάθησης, *μ=0.6* και η αρχική τιμή gain παραμέτρου $G_0 = 10$.

2. Τυχαιοποίηση: Ταξινομεί με τυχαία σειρά τις πόλεις. *i* είναι η πόλη που παρουσιάζεται. *i*=1 και $Inhibit[j] = false$ για *j*=1,…,M
3. Επιλογή νικητή: Χρησιμοποιείται η Ευκλείδεια απόσταση στον ανταγωνισμό. Αυτός ο ανταγωνισμός γίνεται μόνο στα nodes που δεν έχουν επιλεχθεί ως νικητές στην παρούσα επανάληψη.

$$J = argmin_j \|x_i - y_j\| \qquad for\{j|Inhibit[j] = false\} \tag{1}$$

Ύστερα αλλάζει σε true το status.

4. Προσαρμογή: Με βάση την εξίσωση (2), ενημερώνεται το node *J* και οι γείτονες του που χρησιμοποιούν την εξίσωση γειτονιάς *f(G,d)* που ορίζεται με την εξίσωση (4), όπου *d* είναι η βασική απόσταση που μετριέται κατά μήκος του δακτυλίου μεταξύ των κόμβων *j* και *J* στην εξίσωση (3).

$$y_j(t+1) = y_j(t) + \mu f(\cdot)(x_i + y_j) \tag{2}$$

$$d = min[\|j - J\|, M - \|j - J\|] \tag{3}$$

$$f(G,d) = \begin{cases} \exp\left(\frac{-d^2}{G^2}\right) & d < 0.2 * M \\ 0 & otherwise \end{cases} \tag{4}$$

5. *i* = *i* + 1. Αν $i \leq N$ πήγαινε στο βήμα 3 αλλιώς *G = (1 - α)*G* ; *i = 1*; Πήγαινε στο βήμα 6.
6. Τεστ σύγκλισης: Για κάθε πόλη, εάν η θέση του αντίστοιχου κόμβου βρίσκεται σε αποδεκτή απόσταση (για παράδειγμα 0.001), τότε σταματάει. Διαφορετικά, πηγαίνει στο Βήμα 2.

3.3.11 Matsuyama

Ο Matsuyama [14],[15] πρότεινε έναν ενεργό ανταγωνιστικό μηχανισμό μάθησης για την επίλυση του TSP και ανέφερε ότι η μέθοδος μπορεί να επεκταθεί στην επίλυση πιο περίπλοκων προβλημάτων με μια μέτρια αλλαγή στο βασικό πρόγραμμα. Σε αυτή τη διαδικασία, ο αριθμός των κόμβων ορίζεται αρχικά ως τέσσερις φορές ο αριθμός των πόλεων και η λειτουργία ενημέρωσης ορίζεται αρκετά διαφορετικά από τη γενική λειτουργία ενημέρωσης ως εξής:

$$Y_j(t+1) =: Y_j(t) + \varepsilon(t) f\big((j-J),t\big) H\big(\|j-J\| \leq L(t)\big) \times \Big(X_i - Y_j(t)\Big) + \alpha(t)\Big(Y_{j+1} - 2Y_j(t) + Y_{j-1}(t)\Big) \tag{1}$$

Όπου

$$\varepsilon(t) = min\{\in (0)(1+\delta)^{\frac{t}{M}}, \varepsilon_{max}\} \tag{2}$$

$$a(t) = max\{\alpha(0) - \beta t, a_{min}\} \geq 0 \tag{3}$$

$$f(k,t) = exp\{\frac{-k^2}{(2\sigma^2(t))}\} \tag{4}$$

$$L(t) = 3\sigma(t), \sigma(t) = \sigma(0)(1-s)^{t/M} \tag{5}$$

Το $\varepsilon(t)$ είναι ο ρυθμός εκμάθησης, $\alpha(t)$ είναι ο συντελεστής σύζευξης μεταξύ κόμβων, *f(k,t)* είναι μια εκθετική συνάρτηση γειτονιάς, *L(t)* είναι συνάρτηση που μειώνεται σταδιακά με την πάροδο του χρόνου και *H(B)* είναι η συνάρτηση δείκτη που είναι 1 εάν το όρισμα είναι αληθές και 0 εάν όχι. Στην εξίσωση (1) το $f\big((j-J),t\big)H\big(\|j-J\| \leq L(t)\big)$ είναι η κατανομή

νικητή. Η διαδικασία αυτής της λειτουργίας είναι περίπλοκη. Όσον αφορά τη λειτουργία ενημέρωσης, το χαρακτηριστικό γνώρισμα είναι ότι περιλαμβάνει έναν νέο όρο που δημιουργεί μια δύναμη από τους γειτονικούς κόμβους. Η τιμή αυτής της δύναμης είναι μέγιστη στην αρχή της προσομοίωσης, όταν ο σωστός διαχωρισμός των κόμβων είναι εξαιρετικά σημαντικός, μειώνεται γραμμικά με το χρόνο. Παρ 'όλα αυτά, αυτή η δύναμη δεν έχει πλέον επίδραση στην επίτευξη των αποτελεσμάτων, αλλά είναι μόνο για να κάνει το ring πιο ομαλό κατά τη διάρκεια της προσομοίωσης.

3.3.12 SDP

Επιτρέπει τους χρηστές που δεν έχουν την εμπειρία σε υψηλή υπολογιστική απόδοση να αυξήσουν την απόδοση της προσομοίωσης του νευρωνικού δικτύου με παραλληλοποίηση. Με βάση την (Structural Data Parallel Simulation) SDP προσέγγιση [16] πρέπει να εντοπιστεί η προσέγγιση των δομών (ή ο αριθμός) δεδομένων για να το διανείμει μεταξύ των διαθέσιμων κόμβων επεξεργασίας από την διανομή σχήματος μιας γλώσσας υψηλής απόδοσης, όπως το HPF(High Performance Fortran). Το matrix των βαρών για τους νευρώνες αποθηκεύεται σε μια συστοιχία 2 διαστάσεων. Η πρώτη διάσταση ορίζει τον αριθμό του νευρώνα, η δεύτερη διάσταση περιέχει τον δείκτη (1 ή 2, που σημαίνει συντεταγμένη X ή Y). Περιέχει την απλή προσέγγιση της οποίας τα αποτελέσματα χρησιμοποιούνται για την εκλεπτυσμένη προσέγγιση. Η απλή προσέγγιση χρησιμοποιεί το πρόγραμμα Fortran 90 και προσπαθεί να το παραλληλοποιήσει απευθείας. Παίρνει το διαδοχικό πρόγραμμα ως βάση για παραλληλισμό. Απλώς λέει στο σύστημα, πώς πρέπει να διανεμηθούν τα δεδομένα. Με βάση την μέθοδο SDP για παραλληλισμό, ο υπολογισμός του TSP-SOM προγράμματος περιέχει 3 κομμάτια:

1. Παίρνει τις συντεταγμένες των πόλεων τυχαία.
2. Παίρνει τα αρχικά βάρη του δικτύου Kohonen δημιουργώντας έναν κύκλο.
3. Εκπαίδευση του δικτύου.

Εφαρμόστηκε μια απλοϊκή προσέγγιση για να έχουμε γρήγορα ένα πρακτικό αποτέλεσμα. Αυτό είναι μια απλή εφαρμογή του διαδοχικού αλγορίθμου με το Fortran και διανομή της δομής δεδομένων με HPF οδηγίες. Στην εξελιγμένη προσέγγιση χρησιμοποιήθηκαν τα αποτελέσματα της απλοϊκής προσέγγισης και βελτιστοποιήθηκε σε διάφορους προβληματικούς τομείς. Αυτό επηρεάζει τις αλλαγές στη δομή δεδομένων και τον κώδικα για παράλληλη εκτέλεση. Είναι σημαντικό να εξεταστεί για την επεξεργασία των δεδομένων η ευθυγράμμισή του στη μνήμη. Ωστόσο, για να επιτευχθεί η μέγιστη απόδοση, αυτή η μέθοδος είναι πολύ περιορισμένη. Εντοπίστηκαν τα ακόλουθα πέντε ζητήματα που χρήζουν ιδιαίτερης προσοχής και ανάλυσης:

- Αστοχίες local cache.
- Εξισορρόπηση φόρτου εργασίας.
- Καθυστέρηση της επικοινωνίας.
- Διακίνηση επικοινωνίας.
- Τοποθεσία δεδομένων.

3.3.13 CAN

Το Co-Adaptive neural Network (CAN) [17],[18] είναι μια άλλη προσέγγιση που λύνει TSP χρησιμοποιώντας SOM. Το κύριο χαρακτηριστικό που χρησιμοποιεί ονομάζεται φάση συνεργασίας. Αυτή η φάση συνεργασίας σημαίνει ότι κατά την διάρκεια της διαδικασίας την εκμάθησης, διάφορες είσοδοι συνεργάζονται στο να διαλέξουν τον νικητή νευρώνα. Οι φάσεις

του ανταγωνισμού και της συνεργασίας διαφέρουν στο πως οι νευρώνες αντιδρούν όταν ένας νευρώνας είναι ο νικητής σε διάφορες εισόδους. Στην φάση του ανταγωνισμού εάν ένας νευρώνας ήταν ο νικητής μια φορά, μόνο η γειτονιά θα μετακινηθεί. Εάν ο νευρώνας επιλέχτηκε για περισσότερες από μια φορές τότε κανένας από τους νευρώνες θα μετακινηθεί. Στην συνεργατική διαδικασία κανένας από τους νικητές νευρώνες επιτρέπεται να μετακινηθεί παραπάνω από μια φορά. Οι γείτονες δεν επιτρέπεται να μετακινηθούν επίσης. Και οι δυο φάσεις χρησιμοποιούνται, στην αρχή η ανταγωνιστική φάση και η μέθοδος εκμάθησης αλλάζει στην φάση συνεργασίας μετά από κάποια στιγμή. Η αρχικοποίηση των νευρώνων σε αυτόν τον αλγόριθμο είναι ο ίδια με τον βασικό SOM. Η επιλογή του νικητή είναι επίσης παρόμοια, αλλά με μια αλλαγή που υποτίθεται ότι βελτιώνει την ταχύτητα: ο νικητήριος νευρώνας αναζητείται κοντά στον νευρώνα που ήταν ο νικητής στην προηγούμενη επανάληψη. Κάθε β επανάληψη το set από το οποίο αναζητείται ο νικητής νευρώνας επεκτείνεται σε όλο το set των νευρώνων. Τα βάρη των νευρώνων που πρέπει να μετακινηθούν κατά την επανάληψη t ενημερώνονται χρησιμοποιώντας:

$$w_{jk} \leftarrow w_{jk} + f\left(g_{jt}, d(j,K)\right)(x_{ik} - w_{ik}) \qquad k=1,2 \qquad (1)$$

$$f\left(g_{jt}, d(j,K)\right)(x_{ik} - w_{ik}) = exp\left(-\left(d\left(j,K\right)/g_{ij}\right)^2\right)/R \qquad (2)$$

$$g_{ij} = G_t \qquad (3)$$

Η γειτονιά του νικητήριου νευρώνα (το σύνολο των νευρώνων που θα κινηθούν) είναι

$$S = [j | d(j,K) < d^*, j = 1, \dots, M; j \neq K \qquad (4)$$

όπου

$$d^* = min(2G_t + 1, D^*, M/2) \qquad (5)$$

Η φάση συνεργασίας χρησιμοποιείται αντί για την φάση ανταγωνισμού όταν $G_t < G_{cross}$. Σε αυτές τις εξισώσεις, το w_iείναι το διάνυσμα βαρών, το f είναι η συνάρτηση ενεργοποίησης, το d είναι η βασική απόσταση, το K είναι ο νικητήριος νευρώνας, το G_t είναι το gain στην επανάληψη t. Εάν πληρείται μία από τις ακόλουθες συνθήκες, ο αλγόριθμος τερματίζεται.

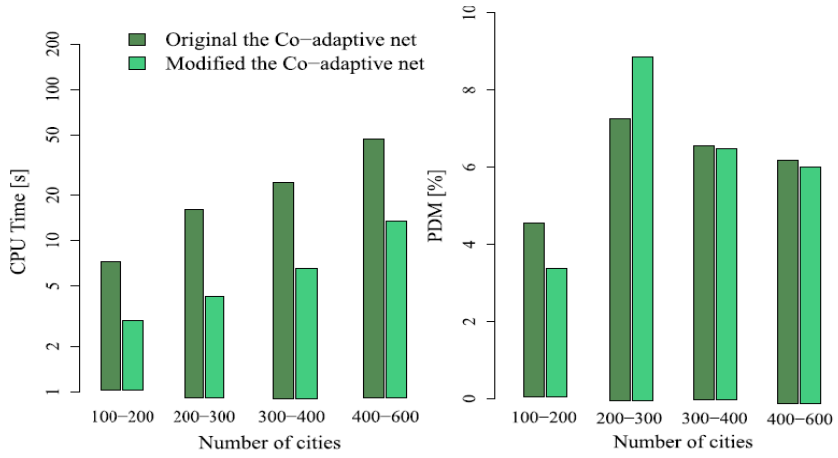
- Η νευρώνες είναι αρκετά κοντά στα guards, ή το μέγιστο σφάλμα μεταξύ των guards και των νικηφόρων νευρώνων τους είναι E_{max}.
- Τα βάρη δεν άλλαξαν κατά την τελευταία επανάληψη.
- $G_t \leq 0.01$.

Στο αρχικό έγγραφο του CAN, το μοντέλο συγκρίθηκε με πολλά άλλα μοντέλα. Τα συμπεράσματα είναι ότι ο αλγόριθμος CAN είναι πολύ ανώτερος από τους Matsuyama, Guilty Net [19] και Elastic Net [20] όσον αφορά την λύση από το βέλτιστο και υπολογιστικό χρόνο. Επίσης, ο αλγόριθμος CAN παράγει αποτελέσματα που είναι ανώτερα από του Somhom όσον αφορά την απόκλιση λύσης από τη βέλτιστη. Το CAN είναι περίπου 10 φορές ταχύτερο από το Somhom.

3.3.14 Modified CAN

Το Modified CAN [21] είναι παρόμοιο όπως το πρωτότυπο CAN, αλλά με κάποιες τροποποιήσεις που δίνουν καλύτερα αποτελέσματα. Η πρώτη είναι η μέθοδος αρχικοποίησης όπου παραμένει όπως ήταν στο πρωτότυπο όπου ο αλγόριθμος αρχικοποιεί ένα δακτύλιο ως

ένα μικρό κύκλο γύρω από το κεντροειδές των πόλεων. Επίσης, ο αρχικός κανόνας προσαρμογής τροποποιείται για να εξετάσει την b-condition, τότε ο κανόνας συνδυάζεται με Multi Scale Neighborhood Functions (MSNF) που χρησιμοποιήθηκε από τους Murakoshi και Sato [22]. Έγιναν μετατροπές στην διαδικασία της επιλογής του νικητή νευρώνα και κατά την προσαρμογή. Το δίκτυο Co-adaptive απαιτεί μεγαλύτερο αριθμό βημάτων προσαρμογής, λέγοντας αυτό, τοποθετώντας το MSNF στη λειτουργία της γειτονιάς αυξάνει την ποιότητα της λύσης και τον αριθμό των απαιτούμενων βημάτων προσαρμογής. Αυτή η τροποποίηση του κανόνα προσαρμογή b-condition μπορεί να χρησιμοποιηθεί για κανόνες χωρίς να μειώσει το μέγεθος της γειτονιάς. Η προσαρμογή του δικτύου τερματίζεται με την κατάσταση $G < 0,01$ σε όλες τις παραλλαγές του αλγορίθμου. Για προβλήματα με λιγότερο από πενήντα πόλεις, ο τροποποιημένος αλγόριθμος Co-adaptive net παρέχει καλύτερα αποτελέσματα. Το Co-adaptive net χρησιμοποιεί τον νικητή νευρώνα, και ο αλγόριθμος αποφεύγει την προσαρμογή των νικητών και των γειτονικών κόμβων, πράγμα που σημαίνει ότι οι κόμβοι μετακινούνται με λιγότερη συχνότητα. Ο αλγόριθμος είναι αρκετά περίπλοκος, το οποίο μπορεί να θεωρηθεί ως ένα αδύναμο σημείο σε ενδεχόμενης παραλληλοποίηση. Στο Σχήμα 3.4 γίνονται συγκρίσεις μεταξύ CAN και Modified CAN και τα αποτελέσματα χωρίζονται σε υπολογιστικό χρόνο και το δεύτερο γράφημα δείχνει την ποσοστιαία απόκλιση από το βέλτιστο μήκος του tour της μέσης τιμής του διαλύματος.



Σχήμα 3.4: Συγκρίσεις CAN και Modified CAN

3.3.15 KNIES

Η μέθοδος Kohonen Network Incorporating Explicit Statistics (KNIES) [23] εκμεταλλεύεται πλήρως τα στατιστικά στοιχεία. Το δίκτυο αποτελείται από έναν σταθερό αριθμό νευρώνων M και η είσοδος του δικτύου είναι οι συντεταγμένες κάθε κόμβου της πόλης, ο οποίες ανήκουν στο σύνολο των πόλεων. Οι κόμβοι της πόλης χαρτογραφούνται στους νευρώνες του ring και η σειρά των νευρώνων στο ring αντιπροσωπεύει τη σειρά των κόμβων της πόλης. Δεδομένου ότι το KNIES χρησιμοποιεί πλήρως τα πλεονεκτήματα του SOM, μπορεί να διατηρήσει τη δομή της γειτονιάς μεταξύ των κόμβων της πόλης. Η βασική διαφορά μεταξύ του SOM και του KNIES-TSP είναι το γεγονός ότι κάθε επανάληψη στη φάση εκπαίδευσης περιλαμβάνει δύο ξεχωριστές μονάδες: το τμήμα προσέλκυσης και το τμήμα διασποράς (εισαγωγής και διαγραφής αντίστοιχα). Στο τμήμα προσέλκυσης, ένα υποσύνολο των νευρώνων μεταναστεύει προς το σημείο δεδομένων που παρουσιάστηκε στο νευρωνικό δίκτυο. Αυτή η φάση είναι ουσιαστικά ίδια με τη φάση μάθησης του SOM. Ωστόσο, μετά από αυτή τη φάση, οι υπόλοιποι

νευρώνες που δεν έχουν εμπλακεί στο τμήμα προσέλκυσης συμμετέχουν σε μετανάστευση. Πράγματι, αυτοί οι νευρώνες απομακρύνονται τώρα από τις τρέχουσες θέσεις τους με τρόπο που εξασφαλίζει ότι οι παγκόσμιες στατιστικές ιδιότητες των σημείων δεδομένων κατοικούν στους νευρώνες. Έτσι, αν και στο SOM οι νευρώνες βρίσκουν μεμονωμένα τους τόπους τους τόσο στατιστικά όσο και τοπολογικά, στο KNIES διατηρούν συλλογικά το μέσο τους να είναι ο μέσος όρος των σημείων δεδομένων που αντιπροσωπεύουν. Ο συμβολισμός για τον αλγόριθμο έχει ως εξής:

- **Νίκη**(*j*) καταγράφει πόσες φορές ο νευρώνας *j* κερδίζει τον διαγωνισμό σε οποιαδήποτε epoch. Μηδενίζεται στην αρχή κάθε epoch και μετά την αφαίρεση της αναστολής.
- **Εισαγωγή**(*j*) εισάγει έναν νευρώνα στην ίδια θέση με τον νευρώνα *j*, ο οποίος ευρετηριάζεται ως *j* + 1.
- **Περιορισμός**(*j*) είναι ένας πίνακας Boolean που δείχνει εάν ο νευρώνας *j* αναστέλλεται ή όχι.
- **Διαγραφή**(*j*) διαγράφει νευρώνα *j* από το τρέχον set.

3.3.16 KNIES DECOMPOSE

Το KNIES Decompose [24] βασίζεται στις θεμελιώδεις αρχές του SOM του Kohonen και στην πρόσφατη παραλλαγή του KNIES. Η τελική περιοδεία επιτυγχάνεται συνδυάζοντας τα Hamiltonian μονοπάτια που το KNIES HPP (KNIES λύση στο πρόβλημα Hamiltonian μονοπατιών) κατασκευάζει για τις συστάδες που καθορίζονται σύμφωνα με τη μέθοδο ομαδοποίησης του Kohonen. Το Ευκλείδειο πρόβλημα του πλανόδιου πωλητή (TSP) είναι ένας στενός ξάδερφος του Ευκλείδειου Hamiltonian προβλήματος (HPP). Η αποσύνθεση σε υποπροβλήματα είναι μια ευρέως χρησιμοποιούμενη στρατηγική για την επίλυση μεγάλων μαθηματικών προγραμμάτων. Είναι ευκολότερο να λύσουμε τα υποπροβλήματα επειδή το μέγεθος κάθε υποπροβλήματος είναι πολύ μικρότερο και οι λύσεις τους μπορούν συνήθως να συνδυαστούν για να προσεγγίσουν τη λύση του αρχικού προβλήματος. Η στρατηγική για τη μείωση της πολυπλοκότητας μιας μεγάλης κλίμακας προβλήματος TSP είναι να αποσυντίθεται ή να χωριστεί σε μικρότερα υποπροβλήματα HPP, τα οποία είναι ευκολότερα να επιλυθούν. Στην ουσία, λύνεται ένα μεγάλο TSP δημιουργώντας μικρότερα HPPs. Μόλις επιλυθούν αυτές οι μικρότερες περιπτώσεις HPP μεμονωμένα, η λύση στο αρχικό TSP επιτυγχάνεται με την επιδιόρθωση των λύσεων στα υποπροβλήματα HPP. Η διαίρεση σε μικρότερα υποπρογράμματα HPP επιτυγχάνεται με τη συσσώρευση των πόλεων του αρχικού προβλήματος με τρόπο που οι δομικές ιδιότητες της προβλημάτων διατηρούνται. Τα βήματα του KNIES Decompose είναι:

1. Χωρισμός των πόλεων σε συστάδες για ένα δεδομένο αριθμό συστάδων.
2. Προσδιορισμός μιας σειράς για την επίσκεψη των συστάδων.
3. Αναγνώριση μιας πόλης για εισαγωγή και μια πόλη αποχώρησης από την συστάδα. Ακολουθεί την σειρά από το βήμα 2 και σχηματίζει γέφυρες μεταξύ των συστάδων.
4. Εύρεση Hamiltonian μονοπατιού σε κάθε συστάδα μεταξύ της εισόδου και των πόλεων εξόδου.
5. Σύνδεση Hamiltonian μονοπατιών χρησιμοποιώντας τα μονοπάτια που λαμβάνονται στο βήμα 3.

Instance	Decompositon	KL	KG
att532	6.15	6.74	6.80
bier127	6.59	2.76	3.08
eil51	3.49	2.86	2.86
eil76	6.49	4.98	5.48
eil101	6.84	4.66	5.63
kroA200	5.66	5.72	6.57
lin105	2.18	1.98	1.29
pcb442	7.99	11.07	10.44
pr107	10.84	0.73	0.42
pr124	3.22	0.08	0.49
pr136	1.93	4.53	5.15
pr152	3.24	0.97	1.29
rat195	8.35	12.24	11.92
rd100	4.89	2.09	2.62
st70	3.67	1.51	2.33
Average Relative Deviations	5.41	4.19	4.42

Σχήμα 3.5: Συγκρίσεις KNIES DECOMPOSE, KNIES TSP και KNIES TSP Global

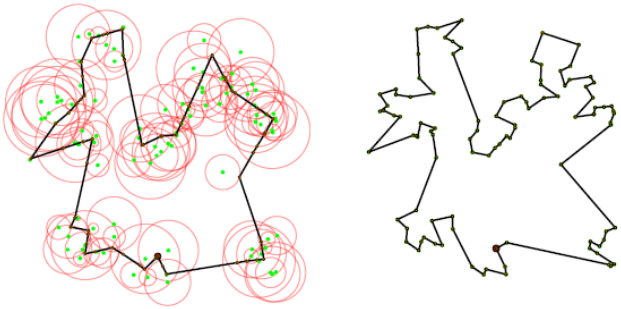
Στο Σχήμα 3.5 παρατηρούμε την σχετική απόκλιση από το βέλτιστο μήκος περιήγησης (σε τοις εκατό). Το κλασσικό KNIES φαίνεται να έχει τα καλύτερα αποτελέσματα και στο KNIES DECOMPOSE δείχνει καθώς αυξάνεται ο αριθμός των πόλεων, φαίνεται να έχει καλύτερα αποτελέσματα.

3.3.17 GSOA

Η προτεινόμενη μέθοδος ονομάζεται Growing Self-Organizing Array (GSOA) [25] και είναι μια μη εποπτευόμενη διαδικασία μάθησης για προβλήματα δρομολόγησης όπου οι βασικές αρχές της ακολουθούν τις υπάρχουσες εργασίες σχετικά με το SOM για το TSP, αλλά παρακινείται κυρίως από τον προγραμματισμό συλλογής δεδομένων όπου ένα ρομπότ όχημα καλείται να συλλέξει δεδομένα από ένα δεδομένο σύνολο σημείων ανίχνευσης. Σε αυτόν τον τύπο προβλήματος, ενδέχεται να μην είναι απαραίτητο να επισκεφθεί τους ιστότοπους με ακρίβεια και το ρομπότ όχημα μπορεί να χρησιμοποιήσει τηλεπισκόπηση ή ασύρματη επικοινωνία για να συλλέξει τα απαιτούμενα δεδομένα. Το GSOA είναι μια αναπτυσσόμενη δομή συστοιχίας N η οποία προσαρμόζεται επαναληπτικά στις τοποθεσίες των αισθητήρων S. Η διαδικασία εκμάθησης ξεκινά με έναν μόνο κόμβο v_1, η οποία θέση μπορεί να αρχικοποιηθεί ως θέση εκκίνησης s_1 ή μπορεί να ρυθμιστεί στα κέντρα των τοποθεσιών του αισθητήρα S. Η διαδικασία μάθησης εκτελείται σε έναν μικρό αριθμό epoch μάθησης, όπου κάθε epoch μάθησης είναι μια προσαρμογή του δακτυλίου των κόμβων N σε όλες τις τοποθεσίες αισθητήρων S. Οι θέσεις των αισθητήρων επιλέγονται με τυχαία σειρά. Για κάθε $s \in S$ καθορίζεται ένας νέος νικητής κόμβος που στη συνέχεια προσαρμόζεται προς το s. Μετά το τέλος ενός μόνο epoch μάθησης), ο δακτύλιος έχει n νέους κόμβους και όλοι οι προηγούμενοι κόμβοι στο δακτύλιο αφαιρούνται για να εξισορροπηθεί ο αριθμός των κόμβων με τον αριθμό των αισθητήρων. Η διαδικασία μάθησης επαναλαμβάνεται στη συνέχεια για έναν καθορισμένο αριθμό epoch μάθησης. Η διαδικασία αποτελείται από τρία μέρη:

- Προσδιορισμός κόμβου νικητή.
- Προσαρμογή του νικητή.
- Εξαγωγή της λύσης μετά από κάθε epoch εκμάθησης.

Το Σχήμα 3.6 δείχνει ένα παράδειγμα εξέλιξης GSOA έχοντας τα πράσινα σημεία που δείχνουν τις τοποθεσίες και γύρω τους έναν κόκκινο κύκλο που είναι οι σένσορες με τη δική τους ακτίνα επιτρέποντας στο ring να κινηθεί προς τις πόλεις.



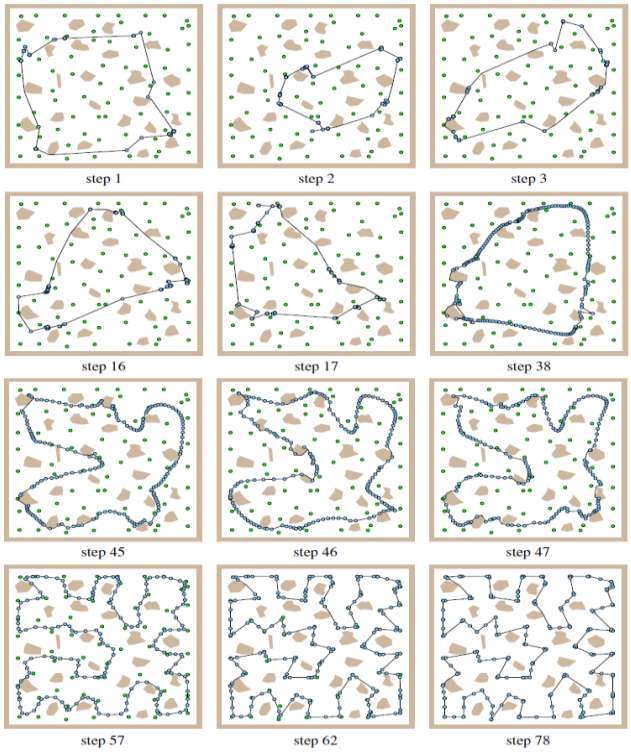
Σχήμα 3.6: Παράδειγμα GSOA

3.3.18 Obstacles + SOM

Αυτή η εργασία [26] παρουσιάζει έναν αλγόριθμο SOM που λύνει το TSP με εμπόδια στο χάρτη. Τα πειραματικά αποτελέσματα που παρουσιάζονται δείχνουν ότι μπορεί να χρησιμοποιηθεί ακόμη και μια απλή προσέγγιση της συντομότερης διαδρομής μεταξύ των εμποδίων και ο αλγόριθμος SOM είναι σε θέση να βρει μια λύση με ανταγωνιστική ποιότητα σε μια λύση του Ευκλείδειου TSP. Η χρησιμοποιούμενη προσέγγιση της συντομότερης διαδρομής είναι πιο υπολογιστικά εντατική από τον υπολογισμό της Ευκλείδειας απόστασης. . Η προτεινόμενη προσέγγιση βασίζεται στη γεωμετρική ερμηνεία και εφαρμογή δομών και αλγορίθμων από το πεδίο της υπολογιστικής γεωμετρίας (computational geometry CG) που μπορεί να θεωρηθεί σχετικά μακριά από το πεδίο των νευρωνικών δικτύων. Οι αλγόριθμοι που χρησιμοποιήθηκαν είναι οι εξής:

- Αλγόριθμος γραφήματος ορατότητας,
- Ένας κυρτός αλγόριθμος διαμέρισης πολυγώνων,
- Ένας αλγόριθμος τοποθεσίας σημείου,
- Διαδικασία ευθείας βάδισης σε κυρτό χώρισμα.

Το πλεονέκτημα της προτεινόμενης λύσης είναι ότι αυτοί οι αλγόριθμοι είναι σχετικά εύκολοι να εφαρμοστούν ή μπορούν να αντικατασταθούν από προυπολογισμένες δομές.



Σχήμα 3.7: Παράδειγμα του TSP έχοντας εμπόδια

3.4 Υβριδικά μοντέλα

Greedy

3.4.1 Self-Organizing Iterative

Το Self-Organizing Iterative [27] είναι ένας συνδυασμός από το Nearest Neighbour Algorithm from Both End Points (NND) [28] και Greedy αλγόριθμο [29]. Υπάρχουν δυο μέρη στον αλγόριθμο. Στο πρώτο οι τιμές προτεραιότητας των άκρων καθορίζονται και εντοπίζεται μια αρχική λύση. Στη συνέχεια, ο αλγόριθμος NND χρησιμοποιείται από επιλεγμένες κορυφές και οι τιμές προτεραιότητας των άκρων ενημερώνονται εξετάζοντας πόσες φορές χρησιμοποιείται μια άκρη σε μια λύση. Στο δεύτερο μέρος του αλγορίθμου, προκειμένου να βελτιωθεί η υπάρχουσα λύση, χρησιμοποιείται ένας αλγόριθμος επανάληψης. Ενημέρωση των τιμών προτεραιότητας των άκρων και στη συνέχεια ταξινόμηση με φθίνουσα σειρά. Μετά από αυτό, εκτελείτε ο Greedy αλγόριθμος. Ο αλγόριθμος The Nearest Neighbour Algorithm from Both End Points (NND) ξεκινά με μια κορυφή που επιλέχθηκε τυχαία στο γράφημα. Στη συνέχεια, ο αλγόριθμος συνεχίζει με την πλησιέστερη κορυφή που δεν έχει επισκεφτεί την αρχική κορυφή. Θα υπάρχουν δύο κορυφές. Προστίθεται μια κορυφή στο tour έτσι ώστε αυτή η κορυφή να μην έχει επισκεφθεί πριν και είναι η πλησιέστερη κορυφή σε αυτές τις δύο κορυφές. Ενημερώνονται οι τελικές κορυφές. Στο συγκεκριμένο paper τα nodes δηλαδή οι κόμβοι αναφέρονται ως κορυφές. Ο αλγόριθμος τελειώνει μετά την επίσκεψη σε όλες τις κορυφές. Τα βήματα του αλγορίθμου είναι τα εξής:

1. Επιλογή μιας τυχαίας κορυφής στο γράφημα.
2. Επίσκεψη της πλησιέστερης κορυφής που δεν έχει επισκεφθεί σε αυτήν την κορυφή.
3. Επίσκεψη της πλησιέστερης κορυφής που δεν έχει επισκεφθεί στις δυο κορυφές και ενημέρωση της τελευταίας κορυφής.

4. Υπάρχει άλλη κορυφή που δεν την έχει επισκεφτεί; Αν ναι, βήμα 3.
5. Πήγαινε στη τελευταία κορυφή από την άλλη τελευταία κορυφή.

Ο Greedy αλγόριθμος σταδιακά κατασκευάζει ένα tour επιλέγοντας επανειλημμένα το συντομότερο άκρο και προσθέτοντάς το στο tour, εφόσον δεν δημιουργεί κύκλο με λιγότερο από N άκρα ή αυξάνει την τιμή οποιουδήποτε node περισσότερο από 2. Δεν πρέπει να προσθέσουμε το ίδιο άκρο δύο φορές. Τα βήματα του αλγορίθμου είναι τα εξής:

1. Ταξινόμηση όλων των άκρων σε φθίνουσα σειρά.
2. Επιλογή του συντομότερου άκρου και πρόσθεση του στο tour εάν δεν παραβιάζει κανέναν από τους παραπάνω περιορισμούς.
3. Υπάρχουν N άκρα στο tour; Εάν όχι, βήμα 2.

3.4.2 A different Hybrid approach

Το Ant Colony Optimization (ACO) είναι ένα metaheuristic με βάση τον πληθυσμό που μπορεί να χρησιμοποιηθεί για να βρει κατά προσέγγιση λύσεις σε δύσκολα προβλήματα βελτιστοποίησης. Στο ACO, ένα σύνολο πρακτόρων λογισμικού που ονομάζονται τεχνητά μυρμήγκια αναζητούν καλές λύσεις σε ένα πρόβλημα βελτιστοποίησης. Για να εφαρμοστεί το ACO, το πρόβλημα βελτιστοποίησης μετατρέπεται σε πρόβλημα εύρεσης της καλύτερης διαδρομής σε ένα επιβαρυμένο γράφημα. Τα τεχνητά μυρμήγκια κατασκευάζουν διαδοχικά λύσεις μετακινούμενα στο γράφημα. Η διαδικασία κατασκευής της επίλυσης είναι στοχαστική και είναι μεροληπτική από ένα μοντέλο φερομόνης, δηλαδή ένα σύνολο παραμέτρων που σχετίζονται με τα συστατικά γραφήματος (είτε κόμβους είτε άκρες) των οποίων οι τιμές τροποποιούνται κατά το χρόνο εκτέλεσης από τα μυρμήγκια. Η γενική ιδέα του συνδυασμού ACO και Artificial neural network (ANN) είναι να αφήσει τα ants να κατασκευάσουν ένα tour το όποιο ύστερα θα βελτιωθεί εφαρμόζοντας ένα SOM. Αν και ο αλγόριθμος ACO είναι γρηγορότερος στο να συγκλίνει προς την καλή αλλά όχι τόσο καλή λύση, η σκέψη είναι να χρησιμοποιηθεί το ANN ως ένα είδος τοπικής αναζήτησης. Η διαδικασία έχει ως εξής:

1. Αρχικοποίηση ACO και SOM με τις παραμέτρους.
2. Επίλυση του TSP με το αρχικοποιημένο ACO.
3. Εξαγωγή του καλύτερου tour στο ACO και εισαγωγή του στο SOM.
4. Επίλυση του SOM.
5. Επιστροφή της λύσης όταν το training του SOM τελειώσει..

Συνοπτικά, η υβριδική προσέγγιση [30] αποτελείται από τη διαδοχική επεξεργασία ενός TSP από το ACO και SOM. Η βελτιωμένη απόδοση προκύπτει από την ειδική εφαρμογή του νευρωνικού δικτύου, γεγονός που οδηγεί στην τοπική ευθυγράμμιση των διασταυρώσεων στο tour που παρέχεται από το ACO. Αυτή η συμπεριφορά επιτυγχάνεται μέσω της προσομοίωσης μιας προηγμένης κατάστασης στη συνολική διαδικασία του αλγορίθμου SOM, έτσι ώστε οι μικρότερες ποσότητες νευρώνων να μετακινούνται. Επομένως, το SOM χρησιμοποιείται ως τοπική τεχνική βελτιστοποίησης για να βελτιώσει το tour που βρέθηκε από το ACO.

Evolutionary

3.4.3 CVOA + SOM

Αυτό το μοντέλο περιλαμβάνει την εφαρμογή μια νέας υβριδικής μεθόδου μεταξύ Coronavirus Optimization Algorithm (CVOA) [31] και του SOM αλγορίθμου για να λύσει το Ευκλείδειο

TSP. Η σχεδιασμένη προσέγγιση βασίζεται στην εξέταση έπειτα μιας τυχαίας διαδρομής της πόλης *n* ως είσοδο. Συμφώνα με κάθε διαδρομή που δημιουργήθηκε, οι παράμετροι προσαρμογής υπολογίζονται για να επιλέξουν τον κόμβο νικητή του οποίου η απόσταση είναι ελάχιστη στο σημείο της διαδρομής. Στη συνέχεια εφαρμόζεται η διαδικασία προσαρμογής για την ανίχνευση των γειτόνων προκειμένου να δημιουργηθεί μια νέα υποψήφια λύση. Η εφαρμογή της προτεινόμενης μεθόδου είναι ένας εξελικτικός αλγόριθμος που βασίζεται σε πιθανοτικές διαδικασίες που απαιτεί έναν ορισμό της συνάρτησης αντιγραφής, η οποία προτείνεται να εισάγει νέα άτομα που πρέπει να υποβληθούν στην CVOA εξέλιξη. Η συνάρτηση αντιγραφής σχεδιάζεται έτσι για να δημιουργήσει μια υποψήφια λύση από τον αλγόριθμο SOM. Οι κύριες ιδιότητες του CVOA είναι οι εξής:

- Οι πιθανότητες και οι παράμετροι ορίζονται, επίσης ενημερώνονται από την επιστημονική κοινότητα.
- Η εξερεύνηση του χώρου αναζήτησης αντιμετωπίζεται, εφόσον ο μολυσμένος πληθυσμός δεν είναι μηδενικός.
- Ο υψηλός ρυθμός επέκτασης εξασφαλίζει καλύτερη χρήση του χώρου αναζήτησης οδηγώντας στην εντατικοποίηση της ανάλυσης.

Επίσης, η έννοια των παράλληλων στελεχών αναδιαμορφώνεται με την εφαρμογή των αλγορίθμων επεξεργασίας σε διαφορετικούς επεξεργαστές προκειμένου να δημιουργηθεί η διαφοροποίηση των αναλύσεων. Ο αλγόριθμος βελτιστοποίησης Coronavirus ξεκάνει με το να παράξει έναν τυχαίο αρχικό πληθυσμό από τον μεμονωμένο Patient-Zero (PZ). Το επόμενο βήμα απασχολεί την εύρεση των νεκρών ατόμων εξαρτώμενο από την πιθανότητα θανάτου (P_DIE) και αφαιρώντας τους από τον πληθυσμό που έχει μολυνθεί. Αυτό το βήμα εντατικοποίησης επίσης ρυθμίζεται για την εύρεση ατόμων που έχουν αναρρώσει οι όποιοι μπορεί να μολύνουν άλλους, εξαρτώντας από τις πιθανότητες εξάπλωσης (P_SUPERSPREADER). Επίσης, για τη διασφάλιση διαφοροποίησης των ατόμων, είναι σκόπιμο να καθοριστεί μια πιθανότητα μετανάστευσης (P_TRAVEL) για την ανίχνευση ατόμων που επιτρέπουν την εξάπλωση του covid σε άλλες λύσεις. Η ενημέρωση των πληθυσμών εξασφαλίζεται για κάθε επανάληψη. Κάθε νεκρό άτομο προσαρτάται στον νεκρό πληθυσμό για να αποφευχθεί η επαναχρησιμοποίηση. Μετά από κάθε επανάληψη, ο μολυσμένος πληθυσμός στέλνεται στον πληθυσμό που έχει αναρρώσει. Το άτομο που έχει αναρρώσει μπορεί να επανεμφανιστεί μολυσμένος εάν αποκτηθεί η πιθανότητα επανεμφάνισης (P_REINFECTION). Έτσι, κάθε άτομο υποβάλλεται σε μέτρα απομάκρυνσης ή απομόνωσης εάν επιτευχθεί η πιθανότητα απομόνωσης. Τέλος, κάθε μολυσμένο άτομο αποστέλλεται σε νέο μολυσμένο πληθυσμό. Το κριτήριο τερματισμού αναμένεται εάν δεν υπάρχει εξέλιξη της λοίμωξης, που εξηγείται από το μειωμένο μέγεθος του μολυσμένου πληθυσμού και του νέου μολυσμένου πληθυσμού ή εάν τα άτομα του νέου μολυσμένου πληθυσμού δεν μπορούν πλέον να μεταδώσουν τον ιό. Γενικά, το κριτήριο τερματισμού επιτυγχάνεται καθ 'όλη τη διάρκεια μιας συγκεκριμένης διάρκειας που ονομάζεται PANDEMIC_DURATION.

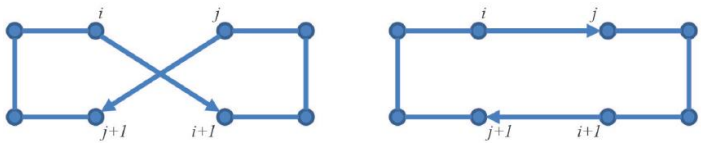
```
define infectedPopulation, newInfectedPopulation as set of Individual
define dead, recovered as list of individual
define PZ, bestIndividual, currentBestIndividual, aux as Individual
define time as integer
time ← 0, PZ ← infectPatientZero()
infectedPopulation ← PZ , bestIndividual ← PZ
while time < PANDEMIC_DURATION AND sizeof(infectedPopulation) >0 do
    dead ← die(infectedPopulation)
        for all i ϵ infectedPopulation do
          aux ← infect(i, recovered, dead)
          if not null(aux) then
             newInfectedPopulation ← aux
          end if
    endfor
    currentBestIndividual ← selectBestIndividual(newInfectedPopulation)
    if fitness(currentBestIndividual) < fitness(bestIndividual)
       bestIndividual ← currentBestIndividual
    end if
    recovered ← infectedPopulation , clear(infectedPopulation)
    newInfectedPopulation ← infectedPopulation , time ← time + 1
end while
return bestIndividual
```

Σχήμα 3.8: Διαδικασία CVOA

3.4.4 Hybrid

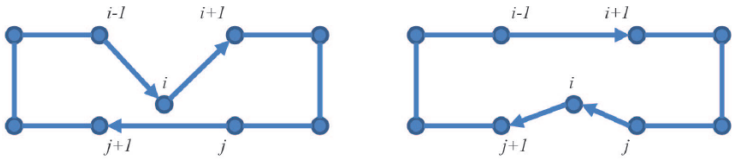
Αυτή η υβριδική προσέγγιση [32] χρησιμοποιεί το local search του οποίου η συμπεριφορά αφού βρει εφικτά αποτελέσματα παράγει έναν μηχανισμό για αλλαγή απάντησης και μετά σταματά να απαντάει όταν αποδεκτές τιμές έχουν αποκτηθεί. Ο μηχανισμός αλλαγής αποτελεσμάτων για αυτό το TSP πρόβλημα μπορεί να πετύχει αλλάζοντας την θέση από το path που συνδέει τους 2 υπάρχοντες πελάτες στο νέο route που συνδέει τον πελάτη με το νέο σημείο. Αν αυτές οι αλλαγές έχουν ως αποτέλεσμα να έχει μικρότερη απόσταση από την παλιά λύση, αντικαθιστά την παλιά λύση με την καινούρια. Κάνει αυτό επανειλημμένα μέχρι να καλύψει τον υποχρεωτικό αριθμό rounds ή βρει αποδεκτές συνθήκες. Τα 3 routes που χρησιμοποιούνται είναι τα εξής:

- 2-Opt exchange operator: σκοπεύει να μειώσει την απόσταση χωρίς να διασταυρώνονται routes. Ο μηχανισμός θα αλλάξει την διαδρομή *i* με *i*+1 και *j* με *j*+1 σε *i* με *j* και *i*+1 με *j*+1.



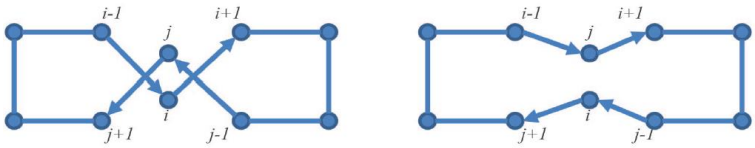
Σχήμα 3.9: 2-Opt exchange operator

- Relocate Operator: θα αλλάξει 2 routes συνδεδεμένα το ένα με το άλλο σε συγκεκριμένη στιγμή χωρίς να ξαναπερνάν. Τότε θα δημιουργήσει ένα καινούριο route σε αυτό το σημείο αλλάζοντας το ένα η το άλλο route σε 2 route συνδεδεμένα σε αυτό το σημείο. Ο μηχανισμός θα αλλάξει την διαδρομή *i*-1, *i* και *i*, *i*+1 σε *i*-1, *i*+1 καθώς και την διαδρομή *j*, *j*+1 σε *j*, *i* και *i*, *j*+1.



Σχήμα 3.10: Relocate Operator

- Exchange operator: αλλάζει 2 συνδεδεμένα routes μεταξύ τους σε συγκεκριμένο σημείο προς 2 routes συνδεδεμένα σε ένα νέο σημείο. Τα δυο παλιά route συνδεδεμένα στο νέο σημείο θα μετατραπούν σε 2 routes συνδεδεμένα στο παλιό point. Ο μηχανισμός θα αλλάξει την διαδρομή *i*-1, *i* και *i*, *i*+1 σε *i*-1, *j* και *j*, *i*+1 αντίστοιχα για την άλλη διαδρομή *j*-1, *j* και *j*, *j*+1 σε *j*-1, *i* και *i*, *j*+1.



Σχήμα 3.11: Exchange operator

Ο αλγόριθμος έχει ως εξής:

Αρχικοποίηση: *N* = αριθμός πόλεων, *M* = Αριθμός κόμβων. Δημιουργία κυκλικού node δικτύου με τυχαίο παραγόμενο κέντρο. Ανάθεση default τιμών για *μ*, *G* και *α*. Οι τιμές τους για μικρού μεγέθους πρόβλημα είναι 0.1, 20, 0.99 και για μεγάλου μεγέθους 0.1, 100, 0.9 αντίστοιχα. Δημιουργία μιας μικρής απόστασης για κάθε πόλη. Δημιουργία μεταβλητής *t* ως η συχνότητα επεξεργασίας κύκλων και ανάθεση *t* = 1. Δημιουργία περιορισμένης μεταβλητής θέτοντας $Inhibit_j = 0$ για κάθε node.

Τυχαία συχνότητα πόλεων: Δημιουργία ενός set από τυχαία συχνότητα πόλεων.

Επιλογή πόλης για επεξεργασία: Επιλογή της πόλης στη *t*-σειρά από το set.

Επιλογή νικητή node: Υπολογισμός τοποθεσίας από την τυχαία πόλη x_i για εύρεση νικητή node *J* χρησιμοποιώντας:

$$J = ArgMin_j(\|x_i - y_j\|) \quad (1)$$

Θέτει $Inhibit_j = 1$ για τον νικητή node. Αν $t \leq N$ επιλέγει τον νικητή node από κάθε node όπου $Inhibit_j = 0$ μονο.

Τροποποίηση: Μετακίνηση κάθε node χρησιμοποιώντας:

$$y_j = y_j + \Delta y_j \quad (2)$$

Εξέταση περιορισμού: Αν όλες οι πόλεις είναι κοντά, σταματά την διαδικασία. Εκτός από αυτό, πηγαίνει στο επόμενο βήμα.

Επεξεργασία αύξησης κύκλου: Αν *t* < *N* θέσε *t* = *t* + 1 και επιστροφή στο βήμα 3. Εκτός από αυτό, θέσε *t* = 1, $Inhibit_j = 0$ σε κάθε node και επιστροφή στο βήμα 2.

Με την προσθήκη 3 αλγορίθμων local search η τελική λύση είναι η εξής:

Αρχικοποίηση: Θέσε *t* να είναι αριθμός κύκλων επεξεργασίας, *t* = 0, max t είναι ο υψηλότερος αριθμός επαναλήψεων όπου η λύση δεν είναι καλύτερη και *max t* = 500000.

Αναβάθμιση αποτελέσματος από local search: Χρησιμοποιήθηκε η ακολουθία αποτελεσμάτων των πόλεων που έχουν την καλύτερη απάντηση σήμερα και άλλαξε την τοποθεσία χρησιμοποιώντας τυχαία τη βελτίωση χρησιμοποιώντας μια από τις τρεις μεθόδους local search. Κάθε local search περίγραμμα εκχωρείται *i* και *j* ως μια τυχαία συχνότητα πόλεων χωρίς πλεονασμό, και $i < j$.

2-Opt exchange operator: Αντικαταστήστε μια ακολουθία από i έως j με συχνότητες $\{j, j-1, j-2, \dots, i+2, i+1, i\}$.

Relocate Operator: Τυχαιοποίηση ακολουθίας μετεγκατάστασης συχνοτήτων από i έως j με συχνότητες $\{i+1, i+2, \dots, j-2, j-1, j, i\}$ ή αντικατάσταση $\{j, i, i+1, i+2, \dots, j-2, j-1\}$.

Exchange operator: Αντικατάσταση θέσης *i* με θέση *j* και αντικατάσταση *j* με θέση *i*.

Εξέταση λύσης: Εφόσον η πόλη έχει αναδιαταχτεί, αν η νέα λύση είναι καλύτερη από την αρχική λύση, αποθήκευσε της νέα λύση αντί για την παλιά και θέσε $t = 0$. Εκτός από αυτό, θέσε $t = t + 1$.

Επεξεργασία αύξησης κύκλου: Επιστροφή στο βήμα 2 ή σταματάει.

3.4.5 2opt + SOM

Αυτός ο αλγόριθμος [33] συνδυάζει το δίκτυο Kohonen και το βελτιώνει με τον αλγόριθμο 2opt. Το προηγούμενο είχε τον 2opt operator αλλά και άλλους 2 επίσης. Το συγκεκριμένο είναι πιο παλιό σε σχέση με το προηγούμενο υιοθετώντας μόνο αυτή την λειτουργιά και όχι παραπάνω. Τοποθετήθηκε στα υβριδικά και όχι στα εμπλουτισμένα λόγο του προηγούμενου μοντέλου έχοντας παρόμοια λειτουργία Ένας νευρώνας και μια πόλη ανατίθενται ο ένας στον άλλο. Όλοι οι νευρώνες οργανώνονται σε έναν vector. Αυτός ο vector αντιπροσωπεύει την ακολουθία των πόλεων που πρέπει να επισκεφθούν. Δυστυχώς, το Kohonen SOM χωρίς κάποιες τροποποιήσεις δεν είναι σε θέση να λύσει το TSP. Ο λόγος για αυτό είναι ότι εάν τα νευρικά βάρη ληφθούν ως συντεταγμένες των πόλεων, μπορεί να μην ισοδυναμούν με τις συντεταγμένες των πόλεων που δόθηκαν. Για την επίλυση του προβλήματος, έχει δημιουργηθεί ένας έγκυρος αλγόριθμος που θα τροποποιήσει την λύση Kohonen. Οι θέσεις των πόλεων και οι θέσεις των νευρώνων μπορεί να μην είναι ίσες. Ωστόσο, τα επαρκή νευρωνικά βάρη και οι συντεταγμένες των πόλεων είναι πολύ κοντά το ένα στο άλλο. Εφαρμόστηκε ένας αλγόριθμος που τροποποιεί τα νευρικά βάρη έτσι ώστε να ίσα με τις συντεταγμένες των πόλεων. Συνήθως το 2opt δεν αλλάζει την λύση πολύ. Ο αλγόριθμος 2opt επιλέγει ένα μέρος της διαδρομής, το αντιστρέφει και το εισάγει πίσω στον κύκλο. Εάν η νέα περιοδεία είναι μικρότερη από τον αρχικό κύκλο, τότε αντικαθίσταται. Ο αλγόριθμος σταματά όταν δεν μπορεί να γίνει άλλη βελτίωση.

```
procedure repair
begin
        Iterate through all neurons
        begin
                nearest_city = find the nearest city to current neuron
                if (nearest_city is not assigned to any neuron)
                        assign nearest_city and current neuron
                else
                        delete this neuron
        end
        Iterate through all cities
        begin
                if (current city is not assigned to any neuron)
                begin
                        create a new_neuron and assign it to current city
                        nearest_neuron = find the nearest neuron to current city
                        insert new_neuron before or after nearest_neuron, depending on which tour is locally
                        shorter
                end
        end
end
```

Σχήμα 3.12: Διαδικασία επιδιόρθωσης 2opt + SOM

Αυτός ο αλγόριθμος βελτιώθηκε από τους Rashid Ahmad και DoHyeun Kim [2] το 2014 προτείνοντας μια τροποποίηση στην μέθοδο γειτονιάς χρησιμοποιώντας Gaussian μέθοδο *h(t,r)*:

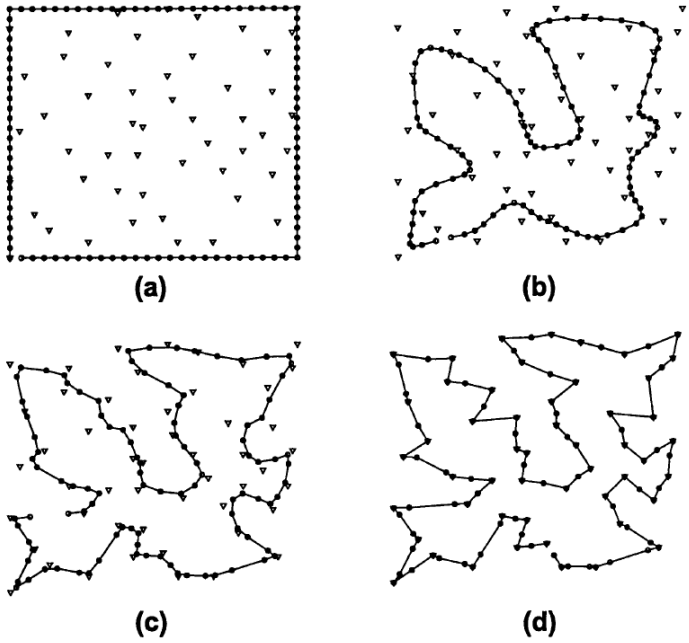
$$h(t,r) = \frac{a(1-r*f)}{[1+(\frac{t}{cdenom})^2]} \quad (1)$$

Αυτή η λειτουργία μειώνεται τόσο σε χωρικούς όσο και σε χρονικούς τομείς. Στον χωρικό τομέα, η αξία του είναι η μεγαλύτερη όταν ο κόμβος *i* είναι ο κόμβος νικητής και μειώνεται σταδιακά με την αυξανόμενη απόσταση από το *i*. Η παράμετρος *α* ορίζει την αρχική τιμή του *h* καθώς η παράμετρος *f(0<f<1/r)* ορίζει τον ρυθμό μείωσης *h* στον χωρικό τομέα. Στον χρονικό τομέα το *t* ελέγχει την τιμή *h* όπου η παράμετρος *cdenom* ορίζει την ρυθμό της μείωσης.

3.4.6 SETSP

Οι προτεινόμενες τροποποιήσεις του SOM Efficiently applied to the TSP (SETSP) [34] στο SOM οδηγούν σε μείωση της πολυπλοκότητας του αλγορίθμου από την εκτέλεση σε n^2 βήματα για την εκτέλεση, τουλάχιστον, *n* βήματα, όπου *n* είναι το μέγεθος των δεδομένων εισόδου (αριθμός πόλεων), καθώς εξελίσσεται ο αλγόριθμος. Έτσι, ακόμη και για ένα μεγάλο σύνολο δεδομένων, ο αλγόριθμος εκτελεί ταχύτερα καθώς ο αριθμός των επαναλήψεων εξελίσσεται. Επιπλέον, η μείωση της πολυπλοκότητας καθοδηγείται από τη διακύμανση της λειτουργίας της γειτονιάς που μειώνεται. Αυτή η λειτουργία είναι κυρίως ελκυστική κατά την επεξεργασία μεγάλων συνόλων δεδομένων. Οι συνεισφορές ήταν: η τροποποίηση της δομής του δικτύου (Bar to Ring) και η αρχικοποίηση, ο ορισμός του κατωφλίου λειτουργίας της γειτονιάς και ο ορισμός νέων νόμων προσαρμογής παραμέτρων. Οι κύριες σημαντικές πτυχές του SETSP είναι η μείωση της πολυπλοκότητας του αλγορίθμου ανάλογα με τον αριθμό των επαναλήψεων και την ταχύτερη προσέγγιση. Η αρχιτεκτονική του αλγορίθμου έχει τα εξής βήματα:

1. Αρχικοποίηση: Σαν είσοδο παίρνει τις συντεταγμένες των πόλεων. Ο αριθμός των επιθυμητών νευρώνων για υπολογισμούς (αυτός ο αριθμός πρέπει να είναι μεγαλύτερος από τον αριθμό των πόλεων) και το ελάχιστο βήμα του ρυθμού επανάληψης a_{min}. Οι νευρώνες αρχικοποιούνται σε ένα ορθογώνιο πλαίσιο γύρω από τις πόλεις και η αρχική τιμή για το *σ* υπολογίζεται χρησιμοποιώντας:
$$\sigma_0 = \frac{l}{4\times c} \quad (1)$$
όπου *l* είναι ο αριθμός των νευρώνων και *c* είναι σταθερά (*c* = 8). Επομένως, μπορεί να φανεί ότι η αρχικοποίηση της διακύμανσης, στην περίπτωση αυτή, σχετίζεται άμεσα με τον αριθμό των νευρώνων εισόδου.
2. Τροποποίηση Παραμέτρων: Λύνεται ο ρυθμός εκμάθησης a_n και η εξέλιξη απόκλισης της γειτονιάς σ_n.
$$a_n = \frac{1}{\sqrt[3]{n}} \quad (2)$$
$$\sigma_n = \sigma_{n-1} \times (1 - 0.01 \times n) \quad (3)$$
3. Εκτίμηση Γειτονιάς: Η λειτουργία γειτονιάς αξιολογείται με βάση τον αριθμό των επαναλήψεων και της απόκλισης *σ*. Το εύρος των νευρώνων που πρέπει να ενημερώνονται σε κάθε βήμα καθορίζεται από το όριο που ορίζεται για τη λειτουργία γειτονιάς.
4. Ανταγωνισμός: Εύρεση νικητή νευρώνα.
5. Συνεργασία: Ο νικητής νευρώνας και η γειτονιά του ενημερώνονται με βάση το βήμα 3.
6. Τροποποίηση: Επιστροφή στο βήμα 2 για όσο $\alpha > \alpha_{min}$.

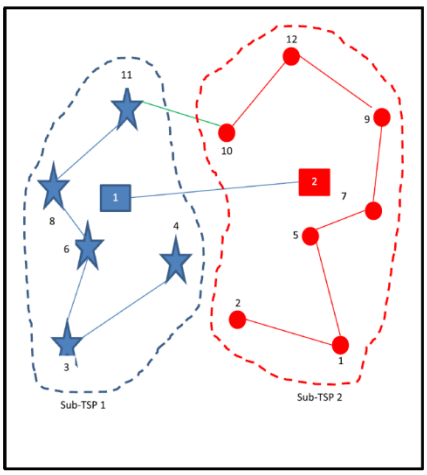


Σχήμα 3.13: Η εξέλιξη του SETSP από έξω προς τα μέσα

Genetic

3.4.7 Clustered SOM

Η προσέγγιση της λύσης [35] χρησιμοποιεί τόσο το SOM όσο και τον γενετικό αλγόριθμο (GA) και αποτελείται από τρία βήματα. Στο πρώτο βήμα, εφαρμόζεται ένα SOM ενός δεδομένου μεγέθους στα δεδομένα των συντεταγμένων των κόμβων. Σύμφωνα με το training process οι συντεταγμένες των πόλεων χρησιμοποιούνται ως δεδομένα εκπαίδευσης και συμβάλλουν επαναληπτικά στις κινήσεις των νευρώνων SOM ή του σχήματος του δικτύου. Σύμφωνα με την τελική θέση του χάρτη, οι πόλεις συγκεντρώνονται με βάση τον πλησιέστερο νευρώνα SOM σε αυτούς. Έτσι, οι πόλεις κάθε συμπλέγματος είναι γνωστές και τα sub-TSPs. Στη συνέχεια, το GA εφαρμόζεται σε κάθε υπο-πρόβλημα για να ελαχιστοποιηθεί το μήκος των υπο-διαδρομών. Χρησιμοποιείται ένα άλλο GA, το οποίο προσπαθεί να βρει μια καλή σειρά για τη σύνδεση των sub-tours. Το GA κωδικοποιεί τη σειρά επισκέψεων των πόλεων και των συστάδων ως strings μοναδικών αριθμών. Μετά από αυτό το βήμα, βρίσκεται μια λύση για ολόκληρο το αρχικό TSP. Ο αριθμός των συστάδων είναι ίσος με μια για κάθε 20 πόλεις. Παρατηρείται ότι το προτεινόμενο SOM με το συμπληρωματικό GA είναι ανώτερο όσον αφορά τόσο την αντικειμενική αξία όσο και τον χρόνο υπολογισμού με τους ανταγωνιστές τους. Πριν από κάθε εφαρμογή GA, οι παράμετροι της συντονίζονται με το σχεδιασμό των πειραμάτων και της μεθόδου δοκιμών και σφαλμάτων Taguchi [36]. Στο Taguchi, τρία επίπεδα θεωρούνται ως οι πιθανές τιμές κάθε παραμέτρου και διεξάγονται διάφορα πειράματα με ορισμένους σχεδιασμένους συνδυασμούς των επιπέδων. Συνεπώς, υπολογίζεται μια ποσότητα αναλογίας σήματος προς θόρυβο (S/N) για κάθε επίπεδο. Οι μικρότερες τιμές S/N είναι καλύτερες σε αυτό το πρόβλημα ελαχιστοποίησης. Ωστόσο, στην προσέγγιση δοκιμής και σφάλματος που χρησιμοποιείται για τον προσδιορισμό του μέγιστου αριθμού επαναλήψεων, τα πειράματα γίνονται με αυξανόμενο αριθμό *Maxit*, ξεκινώντας από 50 επαναλήψεις. Το Σχήμα 16 απεικονίζει μια λύση που κατασκευάστηκε σύμφωνα με αυτή τη μέθοδο για ένα μικρό TSP συνδέοντας τις πόλεις κάθε cluster και στη συνέχεια, τα clusters μεταξύ τους.



Σχήμα 3.14: Παράδειγμα λύσης Clustered SOM

3.4.8 Integrated SOM (ISOM)

Ονομάζεται το ολοκληρωμένο SOM (ISOM) [37], αφού ο κανόνας εκμάθησης ενσωματώνει τους τρεις μηχανισμούς μάθησης στη βιβλιογραφία SOM. Μέσα σε ένα ενιαίο βήμα εκμάθησης, ο υπάρχων νευρώνας αρχικά τραβιέται προς την πολύ εισόδου, και στη συνέχεια ωθείται στο convex-hull του TSP και τελικά τραβιέται προς το μεσαίο σημείο των δύο γειτονικών νευρώνων του. Το **evolved ISOM (eISOM)** χρησιμοποιεί γενετικό αλγόριθμο για να χειριστεί εγγενώς σκληρά ή χρονοβόρα προβλήματα. Στον γενετικό αλγόριθμο (genetic algorithm GA) κάθε άτομο αντιπροσωπεύει ένα πρόγραμμα μάθησης. Για κάθε σχήμα μάθησης, οι παράμετροι περιλαμβάνουν τον τύπο της φόρμουλας για τον υπολογισμό του συντελεστή επέκτασης και των παραμέτρων $a_i (i = 1 \acute{\epsilon} \omega \varsigma 4)$. Περιλαμβάνουν επίσης άλλες παραμέτρους στο ISOM όπως η ακτίνα *R*, ο συνολικός βρόχος μάθησης *L*, οι αρχικές τιμές και τα μειωμένα σχήματα του αποτελεσματικού πλάτους μάθησης *σ(t)* και τις παραμέτρους μάθησης $\eta_1(t)$ και $\eta_2(t)$. Χρησιμοποιείται η σχετική διαφορά μεταξύ του παραγόμενου μήκους του tour και του βέλτιστου μήκους tour για τη μέτρηση της ποιότητας του διαλύματος. Το eISOM είναι ένα από τα πιο ακριβή SOMs για το TSP με τετραγωνική πολυπλοκότητα υπολογισμού. Το ISOM για το TSP:

1. Μετατροπή των συντεταγμένων $[x'_{1i}, x'_{2i}]^2 (i = 1, \cdots, n)$ σε όλες τις πόλεις ώστε να είναι απλωμένες σε ένα κεντρικό κύκλο με ακτίνα $(R \leq 1)$. Εδώ n είναι ο αριθμός πόλεων. Από δω και περά $[x_{1i}, x_{2i}]^T$ δείχνουν τη νέα συντεταγμένη του $\vec{x}_i$.
2. Θέτει $t = 0$, $p = 2$ και το αρχικό βάρος των vector $\vec{w}_j(0) (j = 1, \cdots, n, \acute{\epsilon} \tau \sigma \iota M = n)$ με τυχαίες τιμές μέσα στον κύκλο.
3. Επιλογή πόλης τυχαία, $\vec{x}_k(t) = [x_{1k}(t), x_{2k}(t)]^T$ και βάλε την στους νευρώνες εισόδου.
4. Βρες τον νικητή νευρώνα εξόδου *m(t)*, κοντά στο $\vec{x}_k(t)$, σύμφωνα με την Euclidean metric:

$$m(t) = argmin \left\| \vec{x}_k(t) - \vec{w}_j(t) \right\|^2 \quad (1)$$

5. Εκπαίδευση νευρώνα *m(t)* και τους γείτονες μέσα στο effective width *σ(t)* χρησιμοποιώντας την εξής φόρμουλα:

$$\vec{w}_j(t+1) = c_j(t) \times \{ \vec{w}_j(t) + a_j(t) [\vec{x}_k(t) - \vec{w}_j(t)] \}$$

$$\frac{+\beta_j(t)}{2}\left[\vec{w}_{j-1}(t) + \vec{w}_{j+1}(t) - 2\vec{w}_j(t)\right] \quad (2)$$

όπου j = m(t), m(t) ± 1,…,m(t) ± σ(t). $c_j(t)$ είναι το expanding coefficient το οποίο θα συζητηθεί αργότερα. Οι ρυθμοί εκμάθησης $a_j(t)$ και $\beta_j(t)$ ορίζονται:

$$a_j(t) = \eta_1(t) \times \eta_{j,m(t)} \quad (3)$$

$$\beta_j(t) = \eta_2(t) \times \eta_{j,m(t)} \quad (4)$$

$$\eta_{j,m(t)} = \begin{cases} 1 - \frac{d_{j,m(t)}}{\sigma(t)+1} d_{j,m(t)} \leq \sigma(t), \\ 0 \qquad\qquad otherwise \end{cases} \quad (5)$$

το $\eta_1(t)$ και $\eta_2(t)$ είναι οι παράμετροι εκμάθησης του δικτύου, $\eta_{j,m(t)}$ είναι το neighborhood function και$d_{j,m(t)} = mod(|j - m(t)|, n)$ είναι η διαφορά ανάμεσα στους νευρώνες *m(t)* και *j* στο ring.

6. Ενημέρωσε το width *σ(t)* και τις παραμέτρους εκμάθησης $\eta_1(t)$ και $\eta_2(t)$ με προκαθορισμένο μειωνόμενο σχήμα. Αν ο προκαθορισμένος αριθμός από loops δεν έχει εκτελεστεί πήγαινε στο βήμα 3 με *t* = *t* + 1.
7. Υπολόγισε την τιμή για κάθε πόλη $\vec{x}_k(t)$ με βάση:

$$\alpha(\vec{x}_k) = m_k - \frac{3}{26}\{d(\vec{x}_k, \vec{w}_{mk}) + \frac{2[d(\vec{x}_k, \vec{w}_{mk+1}) - d(\vec{x}_k, \vec{w}_{mk-1})]}{3}\} \quad (6)$$

Όπου m_k k είναι ο νικητής νευρώνας που σχετίζεται με το $\vec{x}_k$.

8. Ταξινόμηση των πόλεων με βάση την τιμή και φτιάξε το tour για το TSP.

Το expanding coefficient ορίζεται:

$$c_j(t) = \left[\left[1.0 + b_j(t) \times e_j(t)\right]\right]^{\alpha_4} \quad (7)$$

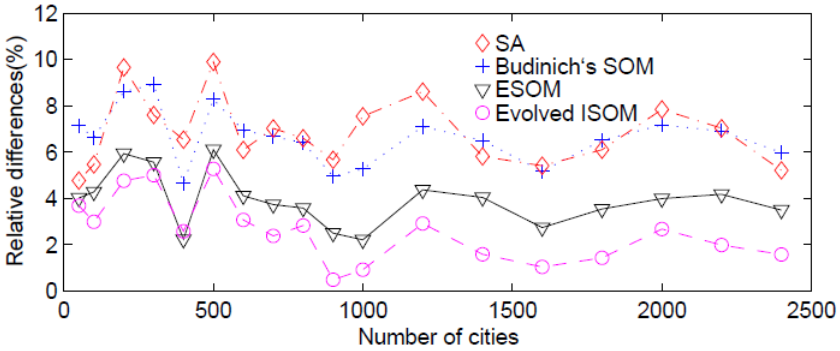
$$b_j(t) = \alpha_1 \times \alpha_j(t)^{\alpha_2} \times \left(1 - \alpha_j(t)\right)^{\alpha_3} \quad (8)$$

Το expanding coefficient για το eISOM ορίζεται:

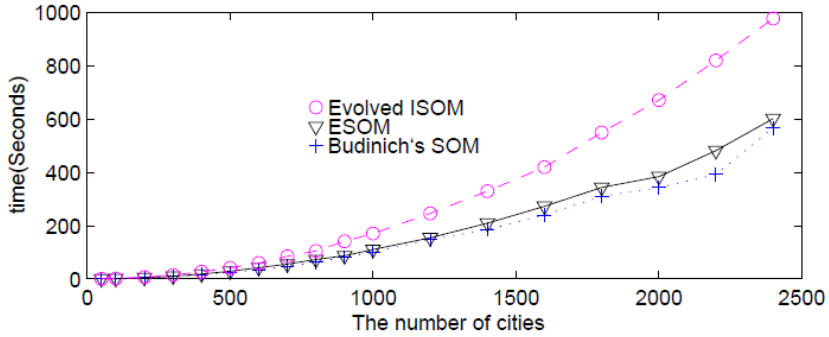
$$c_j(t) = 1 + a_j(t)^3 \left(1 - a_j(t)\right)^{0.25} \{\sum_{i=1}^{2}\left[a_j(t)x_{ik}(t) + \left(1 - a_j(t)\right)w_{ij}(t)\right]^2 - \left|\sum_{i=1}^{2} x_{ik}(t)w_{ij}(t)\right|\} \quad (9)$$

Το expanding term ορίζεται:

$$e_j(t) = \left\|\vec{w}_j(t+1)\right\|^2 - < \vec{w}_j(t), \vec{x}_k(t) > \quad (10)$$



Σχήμα 3.15: Σύγκριση της ποιότητας λύσης του eISOM, του ESOM, του SOM του Budinich και της προσέγγισης SA



Σχήμα 3.16: Η σύγκριση του μέσου χρόνου εκτέλεσης μεταξύ των τριών SOM

3.4.9 ART/SOFM

Αυτή η προσέγγιση [38] χρησιμοποιεί τον συνδυασμό Adaptive Resonance Theory (ART) [39] και Self Organizing Feature Maps (SOFM) για επίλυση μεγάλου όγκου TSP. Με αυτόν τον συνδυασμό επιτρέπει τον αριθμό των νευρώνων να είναι σταθερός. Το βασικό ART χρησιμοποιεί μια τεχνική μάθησης χωρίς επίβλεψη. Ο όρος "adaptive" και "resonance" που χρησιμοποιείται σε αυτό, σημαίνει ότι είναι δεκτικοί στη νέα μάθηση (δηλαδή προσαρμοστική) χωρίς να απορρίπτουν τις προηγούμενες ή τις παλιές πληροφορίες (δηλαδή τον συντονισμό). Τα ART δίκτυα είναι ευρέως γνωστά για να λύσουν τη δύσκολη επιλογή της σταθερότητας-πλαστικότητας, δηλαδή, η σταθερότητα αναφέρεται στη φύση τους για την απομνημόνευση της μάθησης και της πλαστικότητας αναφέρεται στο γεγονός ότι είναι ευπροσάρμοστα για την επίτευξη νέων πληροφοριών. Λόγω της φύσης του ART είναι πάντα σε θέση να μάθει νέα πρότυπα εισόδου χωρίς να ξεχνούν το παρελθόν. Ο αλγόριθμος χωρίζεται σε:

Συσταδοποίηση: Για το Clustering χρησιμοποιείται unsupervised learning οπου η συλλογή από εισαγωγές ομαδοποιείται σε κατηγορίες. Στη συνέχεια χρησιμοποιήθηκε ο σταθερός αριθμός των νευρώνων ίσο με τον αριθμό των πόλεων που έγιναν εισαγωγή. Ο νικητής νευρώνας για κάθε πόλη δεν αντιγράφτηκε αλλά αποκλείστηκε από τον επόμενο ανταγωνισμό. Η Συσταδοποίηση/Clustering χρησιμοποιείται συχνά για να διαιρέσει το TSP σε μικρότερα υπο-προβλήματα και να συνδυάσει τις υπο-λύσεις ξεχωριστά.

Εύρεση Hamiltonian μονοπατιών για κάθε συστάδα (εσωτερικά tour συστάδων): Ξεκινάει με μια δακτύλιο από N νευρώνες μαζί με τα βάρη τους. Η πρώτη είσοδος και οι νευρώνες ανταγωνίζονται μεταξύ τους για το ποιος θα είναι ο νικητής νευρώνας κάνοντας την Ευκλείδεια

απόσταση από την πόλη εισαγωγής. Όπου D είναι η Ευκλείδεια απόσταση και j ο νικητής νευρώνας.

$$D_j = min(D) \quad (1)$$

Μεταφέρεται ο νικητής νευρώνας και οι γείτονες του προς την πόλη i και γίνεται Gaussian συνάρτηση H για να βρεθεί η απόσταση με την οποία κινείται κάθε νευρώνας προς την π;ολη.

$$H(\sigma, d) = exp\left(\frac{-d^2}{\sigma^2}\right) \quad (2)$$

Όπου η παράμετρος σ μετράει τον βαθμό στον όποιο ο παρών νευρώνας συμμετέχει στην γειτονιά του νικητή νευρώνα στη διαδικασία εκμάθησης. Για να λυθεί το TSP χρησιμοποιείται το d για να αναπαραστήσει την κυκλική απόσταση από τον νικητή νευρώνα. Η κυκλική απόσταση ανάμεσα στον νευρώνα k και στον νικητή νευρώνα j είναι:

$$d = \min[|j-k|, |j+N-k|, |k+(N-j)|] \quad (3)$$

Πρέπει να δημιουργηθεί ένα κλειστό loop με Hamiltonian κύκλους, όπου ο πρώτος και ο N νευρώνας είναι γείτονες. Όμως για το συγκεκριμένο πρόβλημα πρέπει να βρεθούν Hamiltonian διαδρομές. Γι αυτό ορίζεται η απόσταση ανάμεσα στον k και στον νικητή νευρώνα j:

$$d = |(j-k)| \quad (4)$$

Ο κανόνας ενημέρωσης που χρησιμοποιείται για την αλλαγή των βαρών των νευρώνων είναι:

$$W_{xk} = W_{xk} + H(\sigma, d)(X_i - W_{xk}) \quad (5)$$

$$W_{yk} = W_{yk} + H(\sigma, d)\left(Y_i - W_{yk}\right) \quad (6)$$

και έπειτα ο νικητής νευρώνας μετακινείται προς την πόλη και αναγκάζει και τους γείτονες του να κάνουν το ίδιο. Ο νικητής νευρώνας περιορίζεται κάνοντας το βάρος του αρνητικό ώστε ο νικητής να μην ξανά ανταγωνιστεί άλλους σε αυτό το epoch.

Εύρεση Hamiltonian βρόχου για ενδοσυσταδικό tour: Για να λυθεί το πρόβλημα σύνδεσης άλλων cluster, κάθε cluster έχει πόλεις αρχής και τέλους. Έτσι ξέροντας ότι αν ένας νευρώνας είναι νικητής μιας πόλης η οποία ανήκει σε ένα k cluster τότε ο νικητής νευρώνας για άλλη πόλη του ιδίου cluster θα είναι άμεσος γείτονας του προηγούμενου νευρώνα.

Σύνδεση των συστάδων και εφαρμόζοντας heuristics για τη βελτίωση της λύσης: Το τελευταίο στάδιο λύνει το πρόβλημα ένωσης cluster με άλλα. Ωστόσο, η σύνδεση των tour σύμφωνα με την εσωτερική συστάδα δημιουργεί πολυάριθμες διασταυρώσεις. Έτσι, για να βελτιώσουμε την αρχική μας λύση, εκτελούμε heuristics για να αφαιρέσουμε τις διασταυρώσεις και να ανταλλάξουμε συνδέσμους. Κατά την αφαίρεση των διασταυρώσεων υπάρχουν δυο ειδών διασταυρώσεων που προκύπτουν στα τελικά tours: intra και inter συστάδες διασταυρώσεων δηλαδή μικρές και μεγάλες διασταυρώσεις.

Memetic

3.4.10 MEMETIC SOM

Το Memetic SOM [40] χρησιμοποιεί έναν εξελικτικό βρόχο που ενσωματώνει το SOM. Οι Memetic αλγόριθμοι είναι υβριδικοί evolutionary αλγόριθμοι, που συνδυάζουν τα πλεονεκτήματα των heuristic σύμφωνα με πληθυσμιακή αναζήτηση. Ο κύριος χειρίστης είναι ο SOM αλγόριθμος εφαρμοσμένος στο δικτυακό γράφο. Σε κάθε δημιουργία. ένας

προκαθορισμένος αριθμός από βασικές SOM επαναλήψεις παρουσιάζονται επιτρέποντας στο πτωτικό run να γίνει διακοπή και αν ενωθεί με την εφαρμογή άλλον χειριστών, οι οποίοι μπορεί να είναι άλλοι SOM χειριστές με τις δικές τους παραμέτρους, mapping and fitness evaluation και selection χειριστές. Ένας βρόχος κατασκευής, καθώς και ένας βρόχος βελτίωσης, ενσαρκώνονται σε μια δομή memetic βρόγχου. Οι λεπτομέρειες των χειριστών είναι οι εξής:

SOM operator: Το κλασσικό SOM εφαρμοσμένο σε ring δίκτυο και τις παραμέτρους του SOM.

Mapping operator: Δηλώνεται με το MAPPING, παράγει αποδεκτή TSP λύση και τροποποιεί το σχήμα του ring αντίστοιχα για κάθε παράγωγη. Αυτός ο operator χαρτογραφεί πόλεις στον κοντινότερο νευρώνα που δεν έχει δηλωθεί σε κάποια πόλη. Αυτό δημιουργεί ένα έγκυρο tour. Έπειτα ο operator μετακινεί νευρώνες προς την τοποθεσία των μονών ανατεθειμένων πόλεων (αν υπάρχει) και αποστέλλει συχνά δυο διαδοχικές πόλεις στο tour.

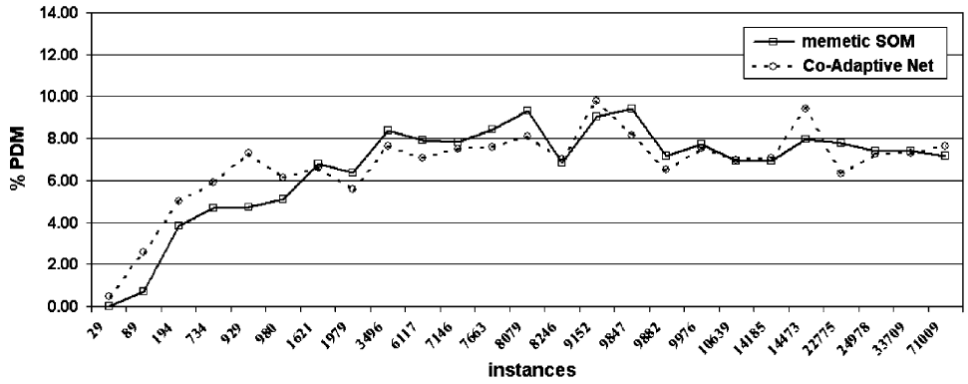
Fitness operator: Για κάθε "άτομο", αυτός ο operator δηλώνεται με το FITNESSTSP και αξιολογεί μια βαθμωτή ftiness τιμή που πρέπει να μεγιστοποιηθεί και η οποία χρησιμοποιείται από τον selection operator. Επιστρέφει την τιμή fitness = - length, όπου length είναι το length του tour καθορισμένο από τις πόλεις πάνω στο ring.

Selection operator: Βασισμένο στη μεγιστοποίηση του fitness, ο χειριστής δηλώνεται με SELECT αντικαθιστά τα χειρότερα άτομα, τα οποία έχουν τις χαμηλότερες τιμές fitness στον πληθυσμό, με τον ίδιο αριθμό των καλύτερων ατόμων, τα οποία έχουν τις υψηλότερες τιμές fitness στον πληθυσμό. Μια ελιτίστικη έκδοση SELECT_ELIT αντικαθιστά τα χειρότερα άτομα με το καλύτερο άτομο που συναντάται κατά τη διάρκεια του run.

Το Σχήμα 3.17 είναι η αναβάθμιση του SOM που παίρνει τις λύσεις στη γειτονιά από τις προηγούμενες λύσεις SOM και τις βελτιώνει τοπικά.

```
Initialize population with Pop individuals.
g = 0
While g < Gen
    Apply one or more standard SOM processes (denoted SOM), with their own
parameter settings, to each individual in population.
    Apply mapping operator MAPPING to each individual in population.
    Apply fitness evaluation operator FITNESSTSP to each individual in
population.
    Save the best individual encountered.
    Apply selection operator SELECT.
    Apply elitist selection operator SELECT_ELIT.
    g = g + 1
End while.
Report best individual encountered.
```

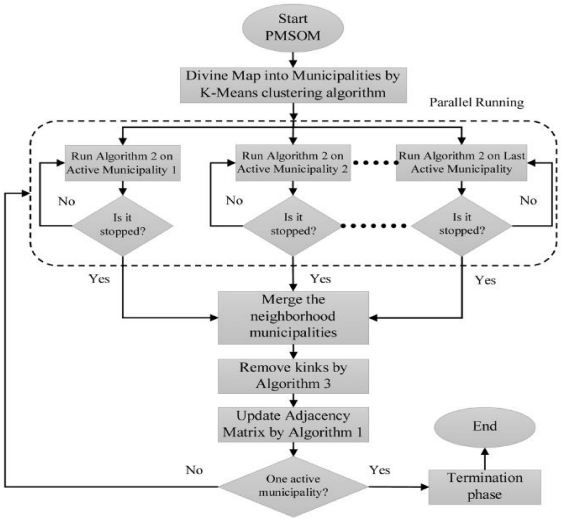
Σχήμα 3.17: Το Memetic loop που ενσωματώνει το SOM



Σχήμα 3.18: Ποσοστιαία απόκλιση του μέσου διαλύματος μεταξύ Memetic και CAN

3.4.11 PMSOM

Η προσέγγιση της παραλληλοποίησης Memetic SOM ή The Parallelized Memetic Self-Organizing Map (PMSOM) [41] συνίσταται στη διαίρεση των πόλεων σε δήμους, στην ανάπτυξη της καταλληλότερης λύσης από κάθε δήμο, ώστε να βρεθεί η καλύτερη συνολική λύση και, τέλος, στην ένωση των δήμων γειτονιάς με τη χρήση ενός χειριστή ανάμειξης για τον προσδιορισμό της τελικής λύσης. Το προτεινόμενο σύστημα είναι μια πιο γενικευμένη μορφή Memetic SOM ως ο τελευταίος νικητής των εφαρμογών νευρωνικών δικτύων για την επίλυση του Ευκλείδειου TSP.



Σχήμα 3.19: Το διάγραμμα ροής του PMSOM

Chaotic

3.4.12 CHAOSOM

Το CHAOSOM [42] χρησιμοποιεί Hodgkin-Huxley εξίσωση [43] όπου είναι το μαθηματικό μοντέλο που προσομοιάζει την πιθανή κίνηση σε squid giant axon.

$$\begin{cases} \frac{dV}{dt} = & I - 120.0m^3h(V-55.0) \\ & - 36.0n^4(C+72.0) \\ & - 0.24(V+49.387), \\ \frac{dm}{dt} = & \frac{0.1(-35-V)}{\exp\left(\frac{-35-V}{10}\right)-1}(1-m) \\ & - 4\exp\left(\frac{-60-V}{18}\right)m, \\ \frac{dh}{dt} = & 0.07\exp\left(\frac{-60-V}{20}\right)(1-h) \\ & - \frac{1}{\exp\left(\frac{-30-V}{10}\right)+1}h, \\ \frac{dn}{dt} = & \frac{0.01(-50-V)}{\exp\left(\frac{-50-V}{10}\right)-1}(1-n) \\ & - 0.125\exp\left(\frac{-60-V}{80}\right)n, \end{cases}$$

Σχήμα 3.20: Η εξίσωση Hodgkin-Huxley

Όπου *V* είναι δυναμικό μεμβράνης, *m* και *h* υποδηλώνουν συντελεστή ενεργοποίησης νατρίου και συντελεστή αδρανοποίησης νατρίου, αντίστοιχα και το *n* είναι συντελεστής ενεργοποίησης καλίου. Το *I* δηλώνει το τρέχον ερέθισμα.

$$I = I_0 + Asin2\pi ft \quad (1)$$

όπου I_0 = 20, A = 40 και f = 300.2

Ο βασικός αλγόριθμος του CHAOSOM είναι ο ίδιος με το SOM, ωστόσο, το σημαντικό χαρακτηριστικό του CHAOSOM είναι να ανανεώσει το learning rate και neighboring coefficient τη στιγμή που τα spikes δημιουργούνται χαοτικά από την Hodgkin-Huxley εξίσωση. Ο ρυθμός μετάδοσης $a_s(s)$ και ο γείτονας νευρώνας $Nc_s(s)$ είναι ως εξής:

$$a_s(s) = \frac{1}{ln(s+2)}$$

$$Nc_s(s) = \lfloor -\frac{1}{n_s}s + \frac{m}{6(p+1)} + 1 \rfloor \quad (2)$$

όπου *p* είναι ο αριθμός των spikes του *V*(t) από το σχήμα 3.20 και μεγαλώνει ανά ένα, και *ns* είναι η μειωμένη συχνότητα του γειτονικού συντελεστή Nc_s. Προκειμένου να προσαρμοστεί η σταθερά χρόνου της εξίσωσης στο σχήμα 3.20 στο CHAOSOM αλγόριθμο, χρησιμοποιείται η μεταβλητή $V_k = V(k \times \Delta t)\ για\ \Delta t = 1/150$. Το βήμα της παραμέτρου s αυξάνεται μονοτονικά με το χρόνο *t*, ωστόσο, αν το χαοτικό σήμα V_k παράγει ενα spike, το βήμα της παραμέτρου *s* ανανεώνεται σε μικρότερη τιμή από την ακόλουθη εξίσωση:

$$s = 500p \quad (3)$$

Επιπροσθέτως, την ιδία στιγμή των spikes, δρομολογείται το tour των δεδομένων εισόδου με βάση: κάθε δεδομένο εισόδου ανατίθεται στον νευρώνα με το vector του βάρους με τη σειρά. Τέλος το μικρότερο tour που δημιουργήθηκε ανάμεσα σε όλα τα tours κατά την διάρκεια μάθησης ορίζεται ως το αποτέλεσμα του CHAOSOM.

3.4.13 Chaotic Maps

Στα μαθηματικά, ένας χαοτικός χάρτης είναι ένας χάρτης (δηλαδή μια συνάρτηση εξέλιξης) που παρουσιάζει κάποιο είδος χαοτικής συμπεριφοράς. Οι χάρτες μπορούν να

παραμετροποιηθούν από μια παράμετρο διακριτού χρόνου ή συνεχούς χρόνου. Οι χαοτικοί χάρτες [44] εμφανίζονται συχνά στη μελέτη δυναμικών συστημάτων. Οι χαοτικοί χάρτες συχνά δημιουργούν fractal. Αν και ένα fractal μπορεί να κατασκευαστεί με επαναληπτική διαδικασία, ορισμένα fractal μελετώνται από μόνα τους, ως σύνολα και όχι από την άποψη του χάρτη που τα δημιουργεί. Αυτό συμβαίνει συχνά επειδή υπάρχουν πολλές διαφορετικές επαναληπτικές διαδικασίες για τη δημιουργία του ίδιου fractal. Σε αυτήν την εργασία διερευνείται η επίδραση εννέα διαφορετικών χαοτικών χαρτών στην ποιότητα των αποτελεσμάτων που προκύπτουν από ένα SOM που έχει χρησιμοποιηθεί για την επίλυση του TSP. Ο στόχος της είναι να συγκρίνει τους χαοτικούς χάρτες και να συμπεράνει ποιοι χάρτες ανταποκρίνονται καλύτερα σε έναν αριθμό επαναλήψεων στο μέγεθος του datset. Οι 9 διαφορετικοί χάρτες είναι: Arnold Cat Map, Burgers Map, Delayed Logistic, Dissipative Standard Map, Henon Map, Ikeda Map, Lozi Map, Sinai Map, και Tinkerbell Map. Οι χάρτες που χρησιμοποιούνται και τα αποτελέσματα δίνονται παρακάτω. Τα αποτελέσματα διαφέρουν ανάλογα με την κλίμακα του προβλήματος (μεγάλης ή μικρής κλίμακας) και τις επαναλήψεις. Είτε ένας χαοτικός χάρτης είχε καλύτερη απόδοση συμβολισμένος με (+) αλλά στατιστικά ασήμαντα καλύτερη απόδοση, καλύτερη με σημασία εντός διαστήματος εμπιστοσύνης 95% συμβολισμένος με (++) ή χειρότερη με (-) αλλά και πάλι στατιστικά ασήμαντη ή χειρότερη με σημασία εντός διαστήματος εμπιστοσύνης 95% με (- -). Ο χάρτης Henon αντιστοιχεί σε ένα χρονικά διακριτό δυναμικό σύστημα [45]. Ο Χάρτης Burgers είναι μια διακριτοποίηση ενός ζεύγους συζευγμένων διαφορικών εξισώσεων που χρησιμοποιήθηκαν από τον Burgers για να απεικονίσουν τη σημασία της έννοιας της διακλάδωσης στη μελέτη των υδροδυναμικών ροών [46],[47]. Τα πειράματα των χαρτών παρουσιάζονται στον παρακάτω πίνακα 7. Κλιμακώνονται από 1000 έως 1000000 επαναλήψεις και χωρίζονται σε προβλήματα μικρής και μεγάλης κλίμακας.

	Small problem			Big problem		
Iterations	1'000	10'000	1'000'000	1'000	10'000	1'000'000
Arnold Cat Map	-	+	-	+	-	-
Burgers Map	+	+	-	++	++	- -
Delayed Logistics	-	-	- -	+	++	- -
Dissipative Map	-	-	-	-	+	+
Hénon Map	+	++	- -	-	+	-
Ikeda Map	- -	+	-	+	+	-
Lozi Map	-	-	- -	-	+	-
Sinai Map	-	+	+	-	+	-
Tinkerbell Map	-	-	- -	++	++	- -

Πίνακας 3.1: Υπολογιστικά πειράματα Χαοτικών Χαρτών

Το **Henon map** συμβαδίζει σε ένα time-discrete dynamical σύστημα. Η εξίσωση του Henon map δίνεται:

$$x_{n+1} = 1 - a \cdot x_n^2 + y_n$$
$$y_{n+1} = \beta \cdot x_n \quad (1)$$

όπου α = 1.4, β = 0.3, με n πόλεις.

Το **Burgers map** είναι ο χωρισμός ενός ζευγαριού από διαφορετικές εξισώσεις που χρησιμοποιήθηκαν από τον Burgers για να απεικονίσει την σχέση διακλάδωσης υδροδυναμικής ροής. Η εξίσωση του Burgers map δίνεται:

$$x_{n+1} = (a \cdot x_n) - y^2 \quad (2)$$
$$y_{n+1} = (\beta \cdot y_n) + (x \cdot y)$$

όπου α = 0.75 και β = 1.75.

Το **Delayed Logistic map** προέρχεται από delayed-coupled χάρτες αλληλεπίδρασης τοπολογιών δικτύου. Η εξίσωση του Delayed Logistic map δίνεται:

$$x_{n+1} = a \cdot x_n \cdot (1 - y_n) \quad (3)$$

$$y_{n+1} = x_n$$

όπου α = 2.27.

Το **Tinkerbell map** συμβαδίζει σε discrete-time dynamical σύστημα. Η εξίσωση του Tinkerbell map δίνεται:

$$x_{n+1} = x_n^2 - y_n^2 + (a \cdot x_n) + (\beta \cdot y_n) \quad (4)$$

$$y_{n+1} = (2 \cdot x_n \cdot y_n) + (\rho \cdot x_n) + (\upsilon \cdot y_n)$$

όπου α = 0.9, β = -0.6013, ρ = 2.0, υ = 0.5.

Για όλους τους χάρτες το είτε το *x* είτε το *y* μπορούν να χρησιμοποιηθούν ως χαοτικοί ψευδοτυχαίοι αριθμοί. Στη συγκεκριμένη έρευνα χρησιμοποιείται μόνο το *x* εφαρμόζοντας τη διαδικασία modulo για να αποκτήσει αριθμούς μεταξύ [0,1].

3.4.14 Fuzzy

Στο Fuzzy SOM [48],[49] μερικές παράμετροι που σχετίζονται με την γειτονιά στο SOM αλλάζουν με membership function. Η παράμετρος learning rate παραλείπεται. Επίσης λαμβάνει υπόψη όλα τα δεδομένα εισόδου για κάθε βήμα επανάληψης και επομένως είναι πιο αποτελεσματικό στο να μειώσει ταλαντώσεις και να αποφύγει dead units. Το Fuzzy που χρησιμοποιείται είναι ένας συνδυασμός SOM και Fuzzy C Means clustering algorithm. Στη δομή νευρωνικού δικτύου, κάθε νευρώνας εξόδου απευθείας αντιστοιχεί σε μια πόλη στο δίκτυο των πόλεων. Ο αριθμός των νευρώνων εξόδου που χρησιμοποιούνται για την περιγραφή των πόλεων είναι γενικά αυθαίρετος. Ωστόσο, εάν ο αριθμός των νευρώνων είναι ίσος με τον αριθμό των πόλεων το πρόβλημα απλοποιείται. Όσο μεγαλύτερος είναι ο αριθμός των νευρώνων, τόσο μεγαλύτερη είναι η ακρίβεια του μοντέλου. Η ακρίβεια εξαρτάται από την πολυπλοκότητα του προβλήματος. Όσο πιο περίπλοκο είναι ένα πρόβλημα τόσο περισσότεροι νευρώνες χρειάζονται κατά την έξοδο. Ο αριθμός των νευρώνων εξόδου επιλέγεται χειροκίνητα. Τα συστατικά βάρους ορίζονται τυχαία και προσαρμόζονται σταδιακά χρησιμοποιώντας έναν Self organizing αλγόριθμο μαθησης και τελικά μια χαρτογράφηση γίνεται από την είσοδο στην έξοδο. Ο αλγόριθμος μάθησης αποτελείται από τα ακόλουθα βήματα:

1. Τυχαιοποίηση των weights για όλους τους νευρώνες.
2. Εισαγωγή όλων των patterns:$X_l = \{: X_{l1}, \dots, X: X_{ln}\}$, l=1,…,M.
3. Πάρε τυχαία ένα pattern εισόδου και υπολόγισε τις Ευκλείδειες αποστάσεις από κάθε pattern X_1 σε όλους τους νευρώνες εξόδου.

   $$d_{lj}(t) = \sqrt{\sum_{i=1}^{N} \left(X_{li} - W_{ij}(t)\right)^2} \quad (1)$$

   Όπου l = 1,…, M και j = 1,…, K.

4. Υπολόγισε την συμμετοχή του κάθε pattern σε όλους τους νευρώνες.

$$R_{lj}(t) = \frac{\{d_{lj}(t)\}^{-2}}{\sum_{m=1}^{K}\{d_{lm}(t)\}^{-2}} \quad (2)$$

Όπου l = 1,…, M και j = 1,…, K.

5. Βρες τον νικητή νευρώνα και τους γείτονες του νικητή.
6. Προσάρμοσε τα συναπτικά weights για κάθε νευρώνα με βάση τους υπολογισμούς συμμετοχών.

$$W_{ij}(t+1) = W_{ij}(t) + \frac{\sum_{l=1}^{M} R_{lj}(t)\left(X_{li} - W_{ij}(t)\right)}{\sum_{l=1}^{M} R_{lj}(t)} \quad (3)$$

7. Μείωσε τις τιμές των παραμέτρων η που είναι ο ρυθμός εκμάθησης και λ που είναι η ακτίνα γειτονιάς.
8. Καθόρισε την συνθήκη σταθερότητας του δικτύου.

$$max\{|W_{ij}(t+1) - W_{ij}(t)|\} < \varepsilon$$
$$1 \le i \le N$$
$$1 \le j \le N$$

Αν η συνθήκη του βήματος 8 ικανοποιηθεί ή ο αριθμός των επαναλήψεων ολοκληρωθεί τότε λήγει η διαδικασία εκμάθησης αλλιώς πηγαίνει στο βήμα 2 για ένα ακόμη loop εκμάθησης. Από τον παραπάνω αλγόριθμο φαίνεται ότι διευκολύνει την δυσκολία επιλογής παραμέτρων δικτύου. Στην παραπάνω διαδικασία εκμάθησης τα weights είναι προσαρμοσμένα μόνο μια φορά σε κάθε εκμάθηση loop και τα χαρακτηριστικά από όλο το input λαμβάνονται υπ όψιν μόλις τα weights προσαρμοστούν, έτσι η ταχύτητα εκμάθησης και η ακρίβεια εκτίμησης είναι και τα δύο βελτιωμένα.

Simulated Annealing

3.4.15 SOM + SA

Με το SOM επειδή σχηματίζεται ένας χάρτης ενώ τα δεδομένα που έχουν παρόμοια χαρακτηριστικά είναι διατεταγμένα κοντά και άλλα δεδομένα είναι διατεταγμένα σε μακρινό μέρος, η διαδρομή θα καθοριστεί σύμφωνα με τη σειρά των πόλεων που καθορίζονται τυχαία. Επομένως, η διαδρομή που λαμβάνεται υποδηλώνει γρήγορα ένα μοτίβο διαδρομής κοντά στο παγκόσμιο βέλτιστο. Ωστόσο, επειδή η ακρίβεια επιδεινώνεται σε τοπικό επίπεδο, το SA (Simulated Annealing), το οποίο είναι εξαιρετικό για τοπική αναζήτηση, εφαρμόζεται στη συνέχεια για να αντισταθμίσει αυτήν την περιοχή. Με αυτή τη διαδικασία, το SA εκτελείται χρησιμοποιώντας την λύση που έγινε από το SOM ως αρχική λύση του SA για να επιτευχθεί μια διαδρομή υψηλότερης ακρίβειας. Ο αλγόριθμος SA δίνεται:

1. Ορισμός παραμέτρων ως η αρχική θερμοκρασία.
2. Δημιουργία μιας αρχικής λύσης (διαδρομή) τυχαία.
3. Επιλογή δύο πόλεων τυχαίων και ανταλλαγή χρησιμοποιώντας 2-opt.
4. Συγκρίνετε πριν από την ανταλλαγή και μετά την ανταλλαγή. Εάν μια λύση μετά την ανταλλαγή βελτιώνεται, τότε αποδεχτείτε τη λύση μετά την ανταλλαγή. Τα κριτήρια αποδοχής:

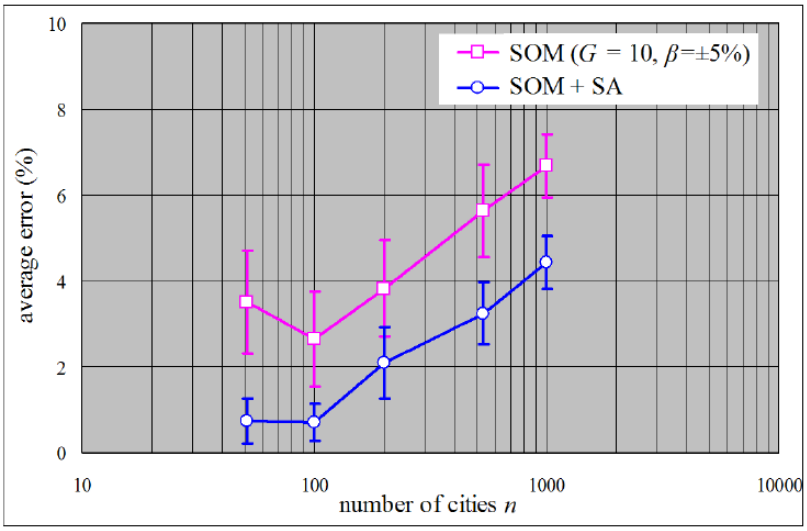
$$A(E, E', T) = \begin{cases} 1 & if \Delta E < 0 \\ exp\left(\frac{-f(x') - f(x)}{T}\right) & otherwise \end{cases} \quad (1)$$

5. Επανάληψη βήματος 3 και 4 μέχρι ο αριθμός των επαναλήψεων (L) σε κάθε θερμοκρασία.
6. Όταν επιτευχθεί η τελική θερμοκρασία, τερματίζει. Διαφορετικά, ενημέρωση της θερμοκρασίας σύμφωνα με το πρόγραμμα ψύξης. επιστροφή στο Βήμα 2.

Ο αλγόριθμος της διαδικασίας SOM + SA είναι ο εξής:

1. Εκτέλεση SOM αλγορίθμου.
2. Εκτέλεση διαδικασιών διέλευσης αποφάσεων και αφαίρεση διελεύσεων για τη λύση από το βήμα 1.
3. Εκτέλεση Αλγορίθμου SA χρησιμοποιώντας τη λύση από το βήμα 2 ως αρχική λύση.
4. Εκτέλεση διαδικασιών διέλευσης αποφάσεων και αφαίρεση διελεύσεων από το βήμα 3 να χρησιμοποιηθεί ως η τελική λύση.

Στο προτεινόμενο SOM + SA [50] η ρύθμιση μιας αρχικής θερμοκρασίας του SA ασκεί ιδιαίτερα μεγάλη επιρροή στην απόδοση αναζήτησης. Η υψηλότερη ακρίβεια της λύσης επιτυγχάνεται με τον καθορισμό κατάλληλων παραμέτρων της διαδικασίας SA. Η υπολογιστική πολυπλοκότητα και η ακρίβεια της λύσης για την TSP μεγάλης κλίμακας εκτιμήθηκαν, αποκαλύπτοντας την πιθανότητα να αποκτηθεί μια κατά προσέγγιση λύση για ένα πρόβλημα δεκάδων χιλιάδων πόλεων χρησιμοποιώντας έναν συνηθισμένο υπολογιστή.



Σχήμα 3.21: Σύγκριση του SOM με SOM + SA στην επίδραση του μεγέθους του προβλήματος στη μέση ακρίβεια του διαλύματος

Elastic net

3.4.16 EN + SOM

Το παρόν paper [51] παρουσιάζει έναν αλγόριθμο EN, ενσωματώνοντας τις ιδέες SOM. Η προτεινόμενη λύση βασίζεται σε μια δομή SOM αρχικοποιημένη με τόσους τεχνητούς νευρώνες όπως ο αριθμός των στόχων που πρέπει να επιτευχθούν. Στη διαδικασία ανταγωνιστικής χαλάρωσης, οι πληροφορίες σχετικά με την τροχιά που συνδέουν τους νευρώνες συνδυάζονται με την απόσταση των νευρώνων από τον στόχο. Ο αλγόριθμος ανόδου κλίσης προσπαθεί να γεμίσει την κοιλάδα τροποποιώντας παραμέτρους σε μια κατεύθυνση κλίσης ανόδου της ενεργειακής συνάρτησης. Το EN βασίζεται ουσιαστικά στον τρόπο με τον οποίο οι περιορισμοί παρουσιάζονται στην ενεργειακή του λειτουργία και συνεπώς μπορούν

να χρησιμοποιηθούν στη λύση μη γεωμετρικών προβλημάτων. Σε κάθε επανάληψη του αλγορίθμου EN, οι συντεταγμένες όλων των κόμβων υπολογίζονται ακόμη και αν δεν υπάρχει αλλαγή στις συντεταγμένες. Ωστόσο, σε SOM, ένα ή μερικά βάρη ενημερώνονται. Σε αυτή το μοντέλο, το SOM χρησιμοποιείται για την κατασκευή του δικτύου και τον ποσοτικό προσδιορισμό των κριτηρίων. Το EN χρησιμοποιείται για την εξαγωγή της διαδρομής TSP κάτω από ένα σύνολο βαρών και την εμφάνιση των αποτελεσμάτων. Κατά τη θεωρία, το EN αποτελείται από ένα μονοδιάστατο δίκτυο από n στοχαστικούς νευρώνες, καθένας από τα οποίους ορίζεται από ένα διατεταγμένο ζευγάρι αριθμών τυπικά ισοδύναμο με την ενεργοποίηση της μονάδας σε συμβατικά νευρωνικά δίκτυ και $P = [p_1, p_2, \dots, p_n]$ όπου δείχνει μια συντεταγμένη. Κατά τη διάρκεια της χαρτογράφησης, πρέπει να διατηρηθεί μια κατάλληλη σειρά των μονάδων. Για να επιτευχθεί αυτό, κάθε μονάδα υποβάλλεται σε δύο δυνάμεις. Η πρώτη επιτρέπει σε κάθε σημείο άκρης να προσελκύει κοντινές μονάδες, ενώ η δεύτερη το τραβά προς το μέσο των γειτονικών μονάδων του (ελαστική ένταση). Το δίκτυο λειτουργεί με επαναληπτική ενημέρωση της ενεργοποίησης της μονάδας *j-th*. Με την ενσωμάτωση των ιδεών του SOM και της στρατηγικής της ανόδου κλίσης στον αλγόριθμο EN, μια φάση EN βασισμένη στο SOM με την εξής μέθοδο: ο νικητής νευρώνας και η γειτονιά του ενημερώνονται ως

$$\Delta w_j = \eta \left(\sum_{\mu=1}^{N} [\left(\frac{\exp(\beta/2|\chi^\mu - w_j|^2)}{\sum_{k=1}^{n} \exp(\beta/2|\chi^\mu - w_k|^2} \right) (\chi^\mu - w_j) \right] + \gamma \Theta(w)) \qquad (1)$$

Όπου vectors βαρών $w_j = w_j(t+1) - w_j(t), (w) = w_{j+1}(t) - 2w_j(t) + w_{j-1}(t)$, $\gamma = 0.6$ είναι η ελαστική παράμετρος και είναι σταθερό όπως και ο ρυθμός εκμάθησης $\eta = 0.4$, $\beta \equiv 1/\sigma^2$, x^μ είναι το μοτίβο εισαγωγής.

3.5 Multiple

Το πρόβλημα πολλαπλών πωλητών είναι μια επέκταση του γνωστού Multiple-Depot Multiple-TSP του προβλήματος πλανόδιου πωλητή όπου ένας αριθμός πωλητών m χρησιμοποιείται για να εξυπηρετήσει ένα σύνολο n τοποθεσιών / πόλεων, που περιορίζονται στον αρχικό περιορισμό ότι κάθε τοποθεσία πρέπει να επισκεφθεί μόνο μία φορά. Οι πωλητές μοιράζονται μια ενιαία αποθήκη - γνωστή ως Multi-TSP μονής αποθήκης ή χρησιμοποιούν πολλαπλές αποθήκες - παραλλαγή γνωστή ως Multiple-Depot Multiple-TSP.

3.5.1 MinMax Multiple-TSP

Σε αυτό το μοντέλο [52] χρησιμοποιήθηκε το SOM για MinMax multiple-TSP, ένας εξελικτικός αλγόριθμος και η προσέγγιση ACO. Επιπλέον, οι αλγόριθμοι δοκιμάζονται μεμονωμένα αλλά και υβριδοποιούνται μαζί. Στην περίπτωση του SOM για πολλαπλά TSP, η τοπολογία του στρώματος εξόδου πρέπει να τροποποιηθεί έτσι ώστε αντί μιας ενιαίας διαδρομής που έχει το πρότυπο SOM, πρέπει να δημιουργηθούν αρκετές διαδρομές m. Ο εξελικτικός αλγόριθμος για την επίλυση Multiple TSP ήταν η αναπαράσταση two-part χρωμοσωμάτων. Λόγω της ανάγκης να εκτελεστούν πιο πολύπλοκες μεταλλάξεις μέσα σε ένα άτομο που ενεργοποιείται από την αναγκαιότητα εξισορρόπησης των περιηγήσεων, χρησιμοποιείται η τεχνική πολλαπλών χρωμοσωμάτων [53]. Αυτή η τεχνική μειώνει το μέγεθος του χώρου αναζήτησης εξαλείφοντας περιττές λύσεις. Επίσης, βελτιώνεται από την ευρετική προσέγγιση της τοπικής αναζήτησης 2-opt. Η προσέγγιση ACO χρησιμοποιεί το g-MinMaxACS που είναι μια παραλλαγή του Ant Colony System. Αντί να χρησιμοποιήσει ένα μόνο μυρμήγκι για να κατασκευάσει μια περιήγηση όπως στο TSP, ένα σύνολο μυρμηγκιών

χρησιμοποιείται για να δημιουργήσει μια ολοκληρωμένη λύση στο πρόβλημα Multiple TSP. Αρχικά, όλα τα μυρμήγκια ξεκινούν τις περιηγήσεις τους από την αποθήκη και ο πωλητής για να επισκεφθεί την επόμενη πόλη επιλέγεται τυχαία.

SOM για MinMax multiple-TSP

Η διαδικασία εκπαίδευσης για multiple TSP μπορεί να συνοψιστεί ως εξής. Σε κάθε επανάληψη μιας πόλης *x* επιλέγεται τυχαία και ο νικητής νευρώνας, ο οποίος υποδηλώνεται από το w^* επιλέγεται για να είναι αυτή στη μικρότερη Ευκλείδεια απόσταση. Στη συνέχεια, υπολογίζεται η ακτίνα της γειτονιάς του νικητήριου νευρώνα. Η τιμή της ακτίνας έχει αρχικά ρυθμιστεί ίση με τον αριθμό των νευρώνων, αλλά μειώνεται σε κάθε επανάληψη, χρησιμοποιώντας εκθετική αποσύνθεση:

$$\sigma(t) = \sigma_0 e^{-t/\lambda_r} \quad (1)$$

όπου *t* είναι η τρέχον επανάληψη, σ_0 υποδηλώνει την αρχική τιμή της ακτίνας και λ_r είναι μια σταθερά αποσύνθεσης ίση με τον συνολικό αριθμό επαναλήψεων *k* διαιρούμενο με *log(σ0)*. Οποιοιδήποτε κόμβοι στη γειτονιά των νικητών νευρώνων, εκτός από τον κόμβο που αντιστοιχεί στην αποθήκη που δεν αλλάζει ποτέ, αλλάζει το βάρος τους σύμφωνα με τους τύπους:

$$w_i = w_i + an(w^*, i)(x - w_i) \quad (2)$$

$$n(w^*, i) = e^{\frac{-dist(w^*,i)^2}{2*(\sigma(t)/10)^2}} \quad (3)$$

όπου *dist* υπολογίζει την απόσταση στο ring και όχι το διάστημα του βάρους. Ο ρυθμός εκμάθησης *α* επίσης διασπάται σε κάθε επανάληψη χρησιμοποιώντας την εκθετική λειτουργία αποσύνθεσης:

$$\alpha(t) = \alpha_0 e^{-t/\lambda_l} \quad (4)$$

όπου t είναι η τρέχον επανάληψη, α_0 υποδηλώνει τον αρχικό ρυθμό μάθησης και λ_l = *k* / $log(\alpha_0/\alpha_{min})$ οπου α_{min} είναι ο ελάχιστος ρυθμός μάθησης. Η τελική λύση είναι χτισμένη μετά την τελευταία επανάληψη του SOM: Για κάθε πόλη υπολογίζουμε και πάλι τον πλησιέστερο νευρώνα και οι περιηγήσεις σχηματίζονται με βάση τη σχετική σειρά αυτών των νευρώνων στην αρχική τοπολογία δακτυλίου.

Εξελικτικός αλγόριθμος

Στο evolutionary algorithm (EA) χρησιμοποιήθηκε η τεχνική chromosome με την οποία γίνονται μεταφορές ανάμεσα στους salesman που ανταλλάσουν πόλεις. Στοχεύοντας σε ισορροπημένες περιηγήσεις στην περίπτωση του Minmax MTSP, υιοθετήθηκε η μετατροπή για cross-tour η οποία λέει ότι αντί για να γίνονται ζευγάρια τυχαία από τα επιλεγμένα chromosome, γίνεται σορτάρισμα ανάλογα με το fitness και τα ζευγάρια δημιουργούνται επιλέγοντας τον πρώτο επιλαχόν από το ένα τέλος και ο άλλος επιλαχόν από το άλλος τέλος (μεγαλύτερο tour μαζί με το μικρότερο tour, δεύτερο μεγαλύτερο tour μαζί με το δεύτερο μικρότερο tour) έτσι δίνει την δυνατότητα για να συνεργαστούν οι καλύτεροι και να αναβαθμιστούν οι χειρότεροι.

Η ACO προσέγγιση

Σε αυτή την προσέγγιση επιλέγεται το g-MinMaxACS, που έδειξε ότι ξεπερνά τα πιο πολύπλοκα συστήματα όσον αφορά τα κριτήρια αξιολόγησης πολλαπλών αντικειμένων. Στο

g-MinMaxACS αντί να χρησιμοποιείτε ένα μυρμήγκι για να κατασκευάσει ένα tour όπως στο TSP, χρησιμοποιείται ένα σύνολο από m μυρμήγκια για να δημιουργήσουν μια πλήρη λύση στο πρόβλημα πολλαπλών TSP. Αρχικά, όλα τα μυρμήγκια ξεκινούν τις περιηγήσεις τους από την αποθήκη και ο πωλητής για να επισκεφθεί την επόμενη πόλη επιλέγεται τυχαία. Ο επιλεγμένος πωλητής επιλέγει την επόμενη πόλη που θα επισκεφθεί σύμφωνα με τον κανόνα μετάβασης από τον τυπικό αλγόριθμο Ant Colony System (ACS) που παρουσιάζεται:

$$s = \begin{cases} argmax_{u \in C} \tau(r,u) \cdot \eta^{\beta}(r,u), & if q \leq q_0 \\ S, & otherwise \end{cases} \quad (5)$$

όπου *τ(r, s)* είναι το μονοπάτι φερομόνης που σχετίζεται με την άκρη *(r, s), η(r, s)* υποδηλώνει τις ευρετικές πληροφορίες που αντιστοιχούν στην άκρη *(r, s)* που λαμβάνονται ως το αντίστροφο της μέτρησης κόστους (απόσταση), το β είναι μια παράμετρος που αντικατοπτρίζει τη σχετική σημασία της φερομόνης έναντι της απόστασης, q είναι ένας τυχαίος αριθμός που επιλέγονται στην περιοχή [0,1] και $q_0 \in [0, 1]$ είναι μια παράμετρος. Το *S* είναι μια τυχαία μεταβλητή με την κατανομή πιθανότητας που δίνεται από:

$$p(r,s) = \begin{cases} \frac{\tau(r,s) \cdot \eta^{\beta}(r,s)}{\sum_{u \in C} \tau(r,u) \cdot \eta^{\beta}(r,u)}, & if s \in C \\ 0, & otherwise \end{cases} \quad (6)$$

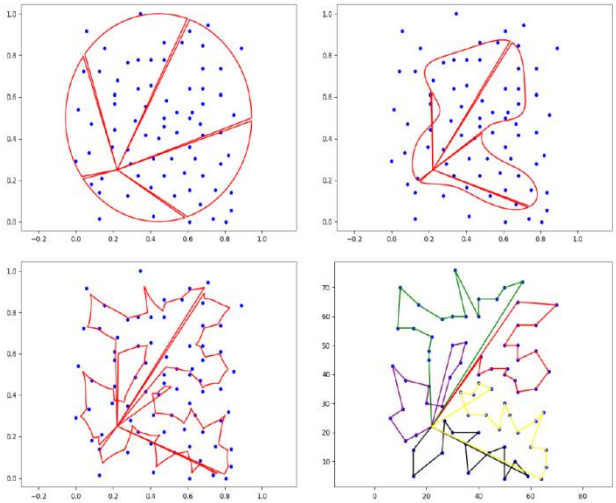
που ορίζει την πιθανότητα με την οποία ένα μυρμήγκι επιλέγει να μετακινηθεί δίπλα στον κόμβο *S*, ενώ βρίσκεται στον κόμβο *r*. Όταν ένα μυρμήγκι διασχίζει μια άκρη, η τοπική ενημερωμένη έκδοση φερομόνης λαμβάνει χώρα και το επίπεδο φερομόνης στην επισκεπτόμενη άκρη ενημερώνεται όπως στο πρότυπο ACS, ανεξάρτητα από το ποιος πωλητής το διασχίζει, χρησιμοποιώντας:

$$\tau(r,s) = (1 - \rho) \cdot \tau(r,s) + \rho \cdot \Delta\tau(r,s) \quad (7)$$

όπου $\rho \in (0,1)$ είναι η μια τοπική παράμετρος αποσύνθεσης φερομόνης και $\Delta\tau(r, s) = \tau_0$, όπου τ_0 είναι αρχικό επίπεδο φερομένης και ορίζεται ως

$$\tau_0 = (n \cdot L_{NN})^{-1}$$

που υποδηλώνει μια ποσότητα που ορίζεται στην πρώτη επανάληψη του αλγορίθμου δημιουργώντας με άπληστο τρόπο με βάση μια ευρετική περιήγηση πλησιέστερου γείτονα και υπολογίζοντας το μήκος της που δηλώνεται με L_{NN}, και *n* είναι ο αριθμός των πόλεων στο TSP.



Σχήμα 3.22: Multiple TSP παράδειγμα με 5 πωλητές

3.5.2 MTSP

Σε αυτό το μοντέλο [54] υπάρχουν *k* rings τα οποία το *k* είναι ο αριθμός των κόμβων πωλητή και *M* είναι οι πόλεις. Έχει 2 κόμβους στα στρώματα εισόδου, όπως όλα τα μοντέλα και *kM* κόμβους στο στρώμα εξόδου. Ο κανόνας του νικητή τροποποιείται και ένας νέος όρος καθιστά τους κόμβους πιο πιθανόν να είναι νικητές όταν είναι μακριά από την πόλη που προέρχονται. Ο αλγόριθμος μπορεί εύκολα να εφαρμοστεί για παράλληλο υπολογισμό. Συγκρίνεται με το συμβατικό SOM από τα Modares. Το MTSP και το συμβατικό TSP συναγωνίστηκαν με 2, 3 και 4 πωλητές και τα δύο. Η ποιότητα των αποτελεσμάτων χρησιμοποιώντας τη βελτιωμένη προσέγγιση SOM είναι καλύτερη από αυτές χρησιμοποιώντας τη συμβατική προσέγγιση SOM. Επιπλέον, η βελτιωμένη προσέγγιση SOM συγκλίνει ταχύτερα από τη συμβατική προσέγγιση SOM. Οι διαφορές τους είναι: ο καθορισμός των κόμβων *M* στο στρώμα εξόδου αντί των κόμβων 2*M* για να μειωθεί ο υπολογισμός του υπολογισμού, τροποποιώντας τη λειτουργία της γειτονιάς προσθέτοντας μια εξίσωση για να επιταχύνει τη σύγκλιση και να τροποποιήσει τους κανόνες ενημέρωσης προσθέτοντας μια εξίσωση για να αυξήσει τη σύγκλιση. Αλγόριθμος:

Ένα νευρωνικό δίκτυο δημιουργείται με 2 Nodes στο input layer και *kM* nodes στο output layer, το οποίο φτιάχνει ένα *k* rings και κάθε ring περιέχει *M* nodes.

Κανόνας νικητή:

$$D_{ij} = \left|A_i - W_j\right|[1 + T_2 + T_3] \quad (1)$$

$$T_2 = \frac{L_i - V}{1+V} \quad (2)$$

$$T_3 = \left|\frac{j - j_{mid}}{M}\right| \quad (3)$$

$$[n_k, J] = min\{D_{ij}\} \quad (4)$$

$$f(D_{ij}, G) = \begin{cases} e^{\left(\frac{-d_j^2}{G^2}\right), if D_{ij} < rM} \\ 0, otherwise \end{cases} \quad (5)$$

$$G = (1-a)^n G \quad (6)$$

$$a = a(0.998)^n \quad (7)$$

Η εξίσωση (1) είναι ο κανόνας νικητή δηλαδή η απόσταση ανάμεσα στον νευρώνα και στην πόλη εισόδου A και W είναι οι συντεταγμένες του νευρώνα στο ring. Ο όρος T_2 ελέγχει τη δίκαιη κατανομή του φόρτου εργασίας για κάθε πωλητή, όπου L, είναι το μήκος της διαδρομής. V είναι η μέση διάρκεια της περιοδείας. Ο όρος T_3 κάνει τους κόμβους να έχουν περισσότερες πιθανότητες να είναι νικητές όταν βρίσκονται μακριά από την προέλευση τους. Η εξίσωση (4) είναι δείχνει πως επιλέγεται ο νικητής νευρώνας όπου $[n_k, J]$ δείχνει των J κόμβο του n ring που είναι νικητής. Η εξίσωση (5) δείχνει τον μέθοδο γειτονιάς όπου G είναι η παράμετρος gain και r είναι μια σταθερά που δείχνει το εύρος της γειτονιάς, n είναι η επανάληψη και α είναι η αλλαγή gain rate.

3.6 Pure SOM Εφαρμογές

Σε αυτή την ενότητα θα παρουσιαστούν κάποιες άλλες εφαρμογές του SOM που λύνει το TSP προβλήματα όπου κάποια από αυτά είναι από τα σημαντικότερα που εξέλιξαν τον αλγόριθμο SOM. Αξίζει να αναφερθεί ότι σε κάποια από τα μοντέλα παραπάνω, κάποιες εφαρμογές από τις παρακάτω είτε αναφέρθηκαν, είτε συγκρίθηκαν, οπότε αξίζει να αναφερθούν διότι έπαιξαν ρόλο στην εξέλιξη. Κάποιες εφαρμογές χρησιμοποιούνται σε πραγματικά δεδομένα.

3.6.1 Kitaori, Murakoshi & Funakubo

Αυτή η εργασία [55] προτείνει μεθόδους που θα αναπτύξουν την ικανότητα του χάρτη χαρακτηριστικών αυτο-οργάνωσης του Kohonen (SOFM) να επιλύει προβλήματα βελτιστοποίησης και δείχνει επίσης πόσο χρήσιμο είναι το SOFM στην επίλυση προβλημάτων βελτιστοποίησης. Οι συγγραφείς επικεντρώθηκαν στο TSP ως τυπικό παράδειγμα προβλήματος βελτιστοποίησης. Το συμβατικό SOFM μπορεί να λύσει το TSP. Αλλά η λύση δεν είναι η βέλτιστη λύση επειδή η διαδρομή τέμνεται. Ως εκ τούτου, οι συγγραφείς προτείνουν νέες μεθόδους για να κρατήσουν το μονοπάτι από το να τέμνεται ανά πάσα στιγμή. Προσθέτοντας αυτές τις μεθόδους στον κανόνα της αλλαγής των συναπτικών δυνάμεων, το μήκος της διαδρομής βελτιώνεται μειώνοντας το χρόνο επανάληψης και αυξάνοντας το ρυθμό σύγκλισης. Ωστόσο, δεν μπορούσαν να εκτιμήσουν σωστά το μήκος της διαδρομής, επειδή ο ρυθμός σύγκλισης του συμβατικού μοντέλου επιδεινώθηκε καθώς αυξανόταν ο αριθμός των πόλεων.

3.6.2 Marco Budinich

Η μη εποπτευόμενη μάθηση [58] που εφαρμόζεται σε ένα μη δομημένο νευρωνικό δίκτυο μπορεί να δώσει κατά προσέγγιση λύσεις στο πρόβλημα του πλανόδιου πωλητή. Για 50 πόλεις στο αεροπλάνο, αυτός ο αλγόριθμος λειτουργεί όπως το elastic net των Durbin και Willshaw (1987) και βελτιώνεται όταν αυξάνεται ο αριθμός των πόλεων για να γίνει καλύτερος από το simulated annealing για προβλήματα με περισσότερες από 500 πόλεις. Σε όλες τις δοκιμές αυτός ο αλγόριθμος απαιτεί ένα κλάσμα του χρόνου που απαιτείται από το simulated annealing. Τα αποτελέσματα δείχνουν ότι το δίκτυο αυτό λειτουργεί ικανοποιητικά σε ένα αποδεδειγμένα δύσκολο πρόβλημα και ιδιαίτερα σε μεγάλα μεγέθη.

3.6.3 S.J. Kim, J.H. Kim & H.M. Choi

Αυτό το έγγραφο [57] προτείνει έναν αποτελεσματικό αλγόριθμο SOFM, για την επίλυση μεγάλης κλίμακας TSPs, στον οποίο ένας νικητής νευρώνας για κάθε πόλη δεν αντιγράφεται αλλά απλώς αποκλείεται στον επόμενο διαγωνισμό. Αυτή η προσέγγιση για τα TSPs που

χρησιμοποιούν SOFM είναι πολλά υποσχόμενο λόγω των ακόλουθων χαρακτηριστικών. Πρώτον, ο συνολικός αριθμός νευρώνων και συνδέσεων είναι μικρός και γραμμικά ανάλογος με τον αριθμό των πόλεων των TSPs, δίνοντας έτσι καλή επεκτασιμότητα για το μέγεθος του προβλήματος και είναι καλός για παράλληλη υλοποίηση υλικού. Δεύτερον, ο προτεινόμενος αλγόριθμος απαιτεί μόνο νευρώνες εξόδου N και συνδέσεις 2N, όπου N είναι ο αριθμός των πόλεων, και δεν απαιτεί επιπλέον αλγόριθμο δημιουργίας και διαγραφής νευρώνων, και έτσι δίνει 30% ταχύτερη σύγκλιση από τον συμβατικό αλγόριθμο για TSPs 30 πόλεων. Τέλος, λόγω της άμεσης εφαρμογής της πραγματικής προοπτικής, ο προτεινόμενος αλγόριθμος μπορεί να λάβει σχεδόν βέλτιστες λύσεις στο 92% των συνολικών δοκιμών εντός εύλογου χρονικού διαστήματος. Η προσομοίωση για TSPs των 1000 πόλεων δίνει επίσης καλά και ελπιδοφόρα αποτελέσματα για τον προτεινόμενο αλγόριθμο.

3.6.4 Tairi & Mohamed

Αυτή η μελέτη [59] προτείνει μια-πρώτα cluster-δεύτερον μια ευρετική διαδρομή χρησιμοποιώντας ένα δίκτυο SOM με ποικίλες παραμέτρους και έναν άπληστο αλγόριθμο για την ελαχιστοποίηση του κόστους ή της απόστασης της λύσης TSP-D (traveling salesman problem with drones) και τη μέτρηση του αντίκτυπου που έχει το ένα στο άλλο. Η μελέτη διεξήχθη χρησιμοποιώντας σύνολα δεδομένων TSP αναφοράς διαφόρων μεγεθών που προσαρμόστηκαν για να ταιριάζουν στο μοντέλο TSP-D. Η μελέτη δείχνει ότι ενώ μια προσέγγιση ελαχιστοποίησης του κόστους μπορεί να αποφέρει εξοικονόμηση απόστασης έως και 21%, μια προσέγγιση ελαχιστοποίησης απόστασης μπορεί στην πραγματικότητα να οδηγήσει σε υψηλότερο κόστος έως και 54%. Επιπλέον, ο αλγόριθμος SOM που χρησιμοποιήθηκε σε αυτή τη μελέτη αποδείχθηκε ότι δημιουργεί σημαντική εξοικονόμηση κόστους έως και 10,4% για δίκτυα που αποτελούνται από 194 πελάτες ή περισσότερους.

3.6.5 Waste Disposal

Σε αυτό το έγγραφο [3], εφαρμόστηκε SOM για την επίλυση του προβλήματος 20 waste disposal sites (WDS). Το δοχείο απορρίπτεται από πλαστικά απορρίμματα. Το WDS δεν επιλέγεται τυχαία, αλλά υπάρχει στην πόλη Niš, δηλαδή στο έγγραφο λύνει ένα πραγματικό πρόβλημα. Για να το απλοποιήσουμε είναι ένα πραγματικό δεδομένο εισόδου που χρησιμοποιείται ως είσοδος για την επίλυση του TSP αλλά με τις συντεταγμένες των αποβλήτων χρησιμοποιώντας το SOM του Kohonen.

3.6.6 Marine patrol

Αυτή η μέθοδος [61] βασίζεται σε μια λύση SOM για το TSP, αν και με σημαντικές αλλαγές. Οι τοποθεσίες των αναφερόμενων αιτημάτων Search and Rescue (SAR), μαζί με τις τοποθεσίες των αναφερόμενων περιστατικών παράνομων αλιευτικών δραστηριοτήτων χρησιμοποιούνται ως κατευθυντήριες γραμμές για τον σχεδιασμό της διαδρομής που πρέπει να ακολουθήσει το σκάφος. Ωστόσο, αντί να αναγκάσει τις διαδρομές περιπολίας να περάσουν ακριβώς σε αυτές τις τοποθεσίες, όπως θα συνέβαινε σε ένα TSP, η προτεινόμενη μέθοδος χρησιμοποιεί τις τοποθεσίες ως εκτιμητές πυκνότητας για το πού πρέπει να τοποθετηθεί η προσπάθεια περιπολίας. Στη συνέχεια αποκτά μια διαδρομή περιπολίας που διέρχεται από τις περιοχές με μεγαλύτερη πυκνότητα. Παίρνει κάποια δεδομένα από το Πορτογαλικό Ναυτικό.

3.6.7 Multi-Goal Path

Αυτή η εργασία [60] παρουσιάζει μια προσέγγιση SOM για το multi-goal path πρόβλημα με πολυγωνικούς στόχους. Το πρόβλημα είναι να βρεθεί η συντομότερη κλειστή διαδρομή χωρίς σύγκρουση για ένα κινητό ρομπότ που λειτουργεί σε ένα επίπεδο περιβάλλον που αντιπροσωπεύεται από έναν πολυγωνικό χάρτη *W*. Η ζητούμενη διαδρομή πρέπει να επισκεφθεί ένα δεδομένο σύνολο περιοχών όπου το ρομπότ λαμβάνει μετρήσεις για να βρει ένα αντικείμενο ενδιαφέροντος. Τα βάρη των νευρώνων θεωρούνται σημεία στο *W* και στη λύση βρίσκονται ως οι κατά προσέγγιση συντομότερες διαδρομές που συνδέουν τα σημεία (βάρη). Το προτεινόμενο SOM έχει μικρότερο αριθμό παραμέτρων από μια προηγούμενη προσέγγιση που βασίζεται στον SOM για το TSP. Επιπλέον, ο προτεινόμενος αλγόριθμος παρέχει καλύτερες λύσεις σε λιγότερο υπολογιστικό χρόνο για προβλήματα με μεγάλο αριθμό πολυγωνικών στόχων. Το πρόβλημα σχεδιασμού για τους στόχους πόντων μπορεί να διατυπωθεί ως το γνωστό TSP.

3.6.8 Orienteering Problem with Neighborhoods + SOM

Αυτή η εργασία [63], ασχολείται με το πρόβλημα προσανατολισμού (Orienteering problem, OP) με την μη εποπτευόμενη εκμάθηση του SOM. Προτείνει την επίλυση του OP με έναν νέο αλγόριθμο που βασίζεται στο SOM για το TSP. Και τα δύο προβλήματα είναι παρόμοια στην εύρεση μιας περιοδείας που επισκέπτεται τις συγκεκριμένες τοποθεσίες. Ωστόσο, το OP πρόκειται να καθορίσει την πιο πολύτιμη περιοδεία που μεγιστοποιεί τις ανταμοιβές που συλλέγονται με την επίσκεψη σε ένα υποσύνολο των τοποθεσιών, διατηρώντας παράλληλα τη διάρκεια της περιοδείας κάτω από τον καθορισμένο προϋπολογισμό ταξιδιού. Ο προτεινόμενος αλγόριθμος στοχαστικής αναζήτησης βασίζεται στην μη εποπτευόμενη μάθηση του SOM και κατασκευάζει μια εφικτή λύση κατά τη διάρκεια κάθε epoch μάθησης. Τα αναφερόμενα αποτελέσματα υποστηρίζουν τη σκοπιμότητα της προτεινόμενης ιδέας και δείχνουν ότι η απόδοση είναι ανταγωνιστική με τις υπάρχουσες ευρετικές μεθόδους. Επιπλέον, το βασικό πλεονέκτημα της προτεινόμενης προσέγγισης που βασίζεται στο SOM είναι η δυνατότητα αντιμετώπισης του γενικευμένου OP με γειτονιές, όπου οι ανταμοιβές μπορούν να συγκεντρωθούν ταξιδεύοντας οπουδήποτε στη γειτονιά των τοποθεσιών. Αυτή η γενίκευση προβλημάτων ταιριάζει καλύτερα στις αποστολές συλλογής δεδομένων με ασύρματη μετάδοση δεδομένων και επιτρέπει την εξοικονόμηση περιττών εξόδων ταξιδιού για την επίσκεψη στις συγκεκριμένες τοποθεσίες.

3.6.9 Orienteering Problem + SOM

Η προτεινόμενη προσέγγιση [62] είναι σε θέση να χρησιμοποιήσει άμεσα μια μη μηδενική ακτίνα επικοινωνίας και έτσι μπορεί να βρει λύσεις με υψηλότερες ανταμοιβές από την απαιτητική μεταευρετική για το OP και το Πρόβλημα Ομαδικού Προσανατολισμού (Team Orienteering Problem, TOP). Χωρίς να εξετάζουμε τις γειτονιές. Ως εκ τούτου, η προτεινόμενη προσέγγιση που βασίζεται σε SOM μπορεί να θεωρηθεί ως ευρετική κατασκευή και να συνδυαστεί με εξελικτικές τεχνικές και υπάρχουσες μεταευρετικές για τη βελτίωση της λύσης όχι μόνο στο OP και το TOP αλλά κυρίως στον προσανατολισμό προβλημάτων με γειτονιές που είναι κατάλληλες συνθέσεις για σενάρια συλλογής ρομποτικών πληροφοριών με μεμονωμένα ή με στόλο ρομποτικών οχημάτων. Η κύρια διαφορά μεταξύ του συνηθισμένου OP και της γενίκευσής του, το πρόβλημα προσανατολισμού με τις γειτονιές (Orienteering Problem with Neighborhoods, OPN) είναι ότι εκτός από το καθορισμένο υποσύνολο των

αισθητήρων που παρέχουν τις πιο πολύτιμες μετρήσεις, απαιτείται επίσης να προσδιοριστούν τα καταλληλότερα σημεία από τα οποία μπορούν να συλλεχθούν οι μετρήσεις (ανταμοιβές) από τους αισθητήρες. Και οι δύο συνθέσεις (το OP και το OPN) μοιράζονται την κύρια πρόκληση των προβλημάτων προσανατολισμού που είναι ο προσδιορισμός των υποσυνόλων των τοποθεσιών σύμφωνα με την περιοδεία που τις επισκέπτεται σε σχέση με τον δεδομένο προϋπολογισμό ταξιδιού.

3.6.10 DTSP + SOM

Η κύρια δυσκολία του TSP του Dubins [64] προκύπτει από το γεγονός ότι είναι απαραίτητο να προσδιοριστεί η ακολουθία των επισκέψεων στις τοποθεσίες μαζί με συγκεκριμένες επικεφαλίδες του οχήματος στις τοποθεσίες. Το προτεινόμενο έγγραφο είναι η πρώτη λύση που βασίζεται στο SOM του Dubins TSP, η οποία περιλαμβάνει προκλήσεις του υποκείμενου συνδυασμού TSP με τη συνεχή βελτιστοποίηση των επικεφαλίδων στις τοποθεσίες. Παρόλο που τα αποτελέσματα δεν παρουσιάζουν σημαντικά καλύτερες λύσεις του SOM από έναν πιο υπολογιστικά απαιτητικό Memetic αλγόριθμο, τα αποτελέσματα υποστηρίζουν τη σκοπιμότητα της προτεινόμενης ιδέας και την καλύτερη δυνατότητα κλιμάκωσης για μεγαλύτερα και πυκνότερα προβλήματα.

3.6.11 Cellular SOM

Σε αυτή την εργασία [65], προτείνεται ένα μαζικά παράλληλο κυτταρικό SOM, που ονομάζεται Cellular SOM για την αντιμετώπιση μεγάλης κλίμακας Euclidean TSPs. Η εφαρμογή του μοντέλου γίνεται σε πλατφόρμα CUDA GPU, με αξιολογήσεις που πραγματοποιούνται σε 52 περιπτώσεις TSP μεγάλου μεγέθους μέχρι 85900 πόλεις. Το προτεινόμενο μοντέλο ασχολείται με μια πολύ απλή διαδικασία κοντά στο αρχικό πρότυπο SOM. Το βασικό σημείο είναι ο τεράστιος παραλληλισμός που θα πρέπει να επιτρέψει μια σημαντική μείωση του χρόνου εκτέλεσης στο μέλλον, καθώς χιλιάδες πολλαπλά cores θα είναι διαθέσιμα σε ένα μόνο chip.

3.7 Επίλογος

Σε αυτό το κεφάλαιο παρουσιάστηκαν οι κατηγορίες των μοντέλων και τα μοντέλα που χωρίστηκαν σε υβριδικά και μη.

Κεφάλαιο 4ο: Άλλες εφαρμογές

4.1 Εισαγωγή

Σε αυτό το κεφάλαιο θα αναφερθούν υποκατηγορίες του TSP.

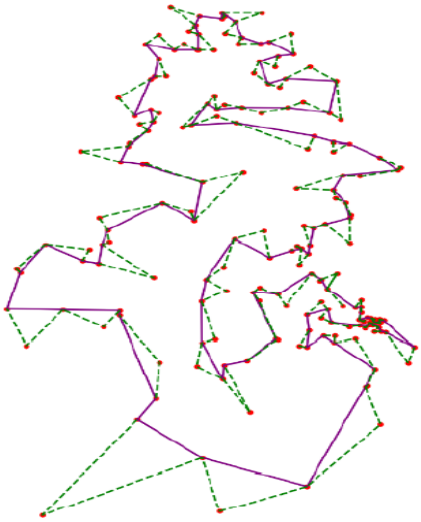
4.2 The vehicle routing problem (VRP)

Είναι ένα πολύ γνωστό πρόβλημα συνδυαστικής βελτιστοποίησης που ασχολείται με την εξεύρεση της βέλτιστης δρομολόγησης αγαθών μεταξύ μιας κεντρικής αποθήκης και ενός συνόλου πελατών. Το Capacitated Vehicle Routing Problem (CVRP) [66] εισάγει έναν περιορισμό στις δυνατότητες ενός οχήματος. Για να επεκταθεί η εφαρμογή SOM από το TSP στο CVRP, πρέπει να ληφθούν υπόψη δύο παράγοντες: ο αριθμός των οχημάτων στο πρόβλημα καθώς και ο περιορισμός της χωρητικότητας του οχήματος. Εφαρμόζοντας το SOM στο CVRP, υπάρχουν k δακτύλιοι νευρώνες (ένας δακτύλιος για κάθε ένα από τα οχήματα/διαδρομές), όπου κάθε δακτύλιος αποτελείται από M νευρώνες. Οι νευρώνες εξόδου πρέπει τώρα να αναπροσαρμόζονται τόσο από τον δακτύλιο (διαδρομή) όσο και από τη θέση εντός της διαδρομής. Για να ληφθεί υπόψη ο περιορισμός της ικανότητας του CVRP, ακολουθήθηκαν δύο προσεγγίσεις. Και οι δύο αυτές προσεγγίσεις ενσωματώνουν έναν μηχανισμό για την τιμωρία των διαδρομών που υπερβαίνουν την χωρητικότητα κατά την επιλογή του κόμβου που κερδίζει για μια πόλη εισόδου x. Μία από τις πρώτες εφαρμογές του SOM στο CVRP χρησιμοποιεί μια πιθανοτική προσέγγιση για την επιλογή του νικηφόρου κόμβου J [67]. Σε αυτόν τον αλγόριθμο κάθε φορά που μια πόλη εισόδου, x, παρουσιάζεται στο δίκτυο, ο συντομότερος κόμβος (υπολογισμένο από την Ευκλείδεια απόσταση) σε κάθε διαδρομή αναγνωρίζεται. Από τους K κόμβους, ο νικητής κόμβος επιλέγεται με τέτοιο τρόπο ώστε η πιθανότητα του νικητή κόμβου να επιλεχτεί από μια υπερφορτωμένη διαδρομή προς το μηδέν καθώς εξελίσσεται το δίκτυο. Η δεύτερη προσέγγιση ενσωματώνει έναν όρο για να τιμωρήσει τις υπερφορτωμένες διαδρομές κατά τη φάση του ανταγωνισμού του αλγορίθμου. Η παλαιότερη παραλλαγή αυτής της προσέγγισης πολλαπλασιάζει την Ευκλείδεια απόσταση με έναν παράγοντα «handicap» καθώς οι κόμβοι ανταγωνίζονται [68]. Με αυτή τη μέθοδο, ένα μεγαλύτερο «handicap» δείχνει ότι η τρέχουσα διαδρομή είναι κοντά ή πάνω από τον περιορισμό της χωρητικότητας. Σε επακόλουθες προσεγγίσεις SOM στο VRP, προστίθεται ένας όρος στην Ευκλείδεια απόσταση καθώς ανταγωνίζονται οι κόμβοι. Τα δημοσιευμένα αποτελέσματα της εφαρμογής του SOM στο CVRP που χρησιμοποιούν σύνολα προβλημάτων αναφοράς, ξεπερνούν τα αποτελέσματα από προηγούμενες προσεγγίσεις SOM [67-70].

4.3 TSP-D

Ο στόχος του προβλήματος του πλανόδιου πωλητή με drones (TSP-D) [71] είναι η αποτελεσματική δρομολόγηση ενός οχήματος που έχει εξοπλιστεί με ένα drone από μια αποθήκη έναρξης για να παραδώσει πακέτα σε ένα σύνολο πελατών και να επιστρέψει στην αποθήκη έναρξης με την ελάχιστη απόσταση που διανύθηκε. Το TSP-D έχει μελετηθεί ευρέως από την πρώτη εισαγωγή του το 2015 και έχει αποδειχθεί ότι μειώνει σημαντικά το χρόνο ταξιδιού σε σύγκριση με ένα κλασσικό TSP. Το μοντέλο που προτάθηκε σε αυτή τη μελέτη χρησιμοποιεί ένα SOM για να προσδιορίσει μια αρχική λύση στο υποκείμενο TSP. Στη συνέχεια, ένας greedy αλγόριθμος χρησιμοποιείται για τον προσδιορισμό των ιδανικών κόμβων πελατών που θα εξυπηρετηθούν από το drone προκειμένου να μειωθεί η συνολική

απόσταση που διανύθηκε από το όχημα. Τα δεδομένα που χρησιμοποιήθηκαν για τη μελέτη αυτή ανακτήθηκαν από μια εθνική βιβλιοθήκη TSP. Μετά τον προσδιορισμό μιας εφικτής διαδρομής για το όχημα, ρυθμίζοντας το δίκτυο SOM, η επόμενη φάση της λύσης είναι να καθορίσει τους ιδανικούς υποψήφιους για την εισαγωγή των drones. Η εισαγωγή του drone αναφέρεται στη διαδικασία της απόρριψης ενός από τους κόμβους από τη διαδρομή του φορτηγού και αντ 'αυτού, προστίθεται στην διαδρομή των drones. Ο στόχος του αλγορίθμου ήταν να προσδιοριστεί ποιοι κόμβοι, εάν επιλεχτούν για την παράδοση των drone, θα μειώσουν σημαντικά τη συνολική απόσταση που διανύθηκε με φορτηγό. Αυτή η διαδικασία εισαγωγής κόμβων drone επαναλαμβάνεται μέχρι να μην υπάρχουν πιθανοί υποψήφιοι για την παράδοση των drone, προκειμένου να ελαχιστοποιηθεί η απόσταση που διανύθηκε με φορτηγό. Το μέγεθος του δικτύου που χρησιμοποιήθηκε για αυτή τη μελέτη ήταν 8 φορές μεγαλύτερο του μεγέθους πληθυσμού, ο αριθμός των επαναλήψεων αρχικοποιήθηκε στις 100,000 αλλά σταμάτησε στις 24,487 επαναλήψεις. Δοκιμάστηκε σε 194 κόμβους πελατών. Τα αποτελέσματα έδειξαν ότι η αρχική λύση που δημιουργήθηκε χρησιμοποιώντας το SOM ήταν 7% υψηλότερο από την βέλτιστη λύση καθώς η λύση στο TSP-D ήταν 39% λιγότερη από τη βέλτιστη λύση TSP και υπολογίστηκε σε 28.8 δευτερόλεπτα. Η τελική λύση για το μεγάλο TSP-D προτείνει ότι ο συνδυασμός drone με ένα delivery φορτηγό μπορεί να βοηθήσει σημαντικά στην μείωση απόστασης της μεταφοράς και του χρόνου αν σχεδιαστεί αποτελεσματικά. Στο Σχήμα 4.1 οι κόκκινες τελείες είναι οι τοποθεσίες, η μωβ γραμμή είναι η διαδρομή που θα κάνει το φορτηγό και η πράσινη διακεκομμένη είναι η διαδρομή που θα κάνει το drone.



Σχήμα 4.1: TSP με drone και φορτηγό

4.4 Επίλογος

Σε αυτήν την ενότητα παρουσιάστηκαν κάποιες διαφορετικές εφαρμογές στο TSP.

Κεφάλαιο 5ο: Αποτελέσματα και συγκρίσεις

5.1 Εισαγωγή

Σε αυτό το κεφάλαιο θα γίνουν συγκρίσεις στις αποδόσεις βέλτιστων λύσεων και στις υπολογιστικές ταχύτητες κάποιων μοντέλων που αναλύθηκαν παραπάνω. Επίσης συγκρίνονται και κάποιες λύσεις θα δοθούν. Προκειμένου να επιλυθεί ένα TSP, στον αλγόριθμο χρειάζεται να εισαχθούν δεδομένα στο επίπεδο εισόδου και αυτά τα δεδομένα είναι TSPLIB. Το TSPLIB είναι μια βιβλιοθήκη δειγμάτων πόλεων/σημείων για το TSP (και σχετικά προβλήματα) από διάφορες πηγές και διάφορους τύπους. Ένα παράδειγμα βιβλιοθήκης είναι το 'qa194.tsp' όπου τα γράμματα ονομάζουν μια χώρα/σύνολο δεδομένων και οι αριθμοί δείχνουν τον αριθμό των πόλεων/κουκκίδων αυτού του συνόλου δεδομένων (δηλαδή Κατάρ και 194 πόλεις). Άλλα παραδείγματα είναι το «pbc1173» και το «bier127». Κάνοντας εισαγωγή μιας TSPLIB βιβλιοθήκης, ο κάθε αλγόριθμος παίρνει τα δεδομένα και εμφανίζει την λύση του στο τέλος την βέλτιστη λύση του. Κάποιοι αλγόριθμοι που ανταγωνίστηκαν με αυτούς που παρουσιάστηκαν προηγουμένως δεν διότι δεν είναι SOM αλλά αξίζει να αναφερθούν γιατί είναι αλγόριθμοι βελτιστοποίησης. Τα αποτελέσματα βγήκαν από τους δημιουργούς τον μοντέλων και συγκεντρώθηκαν. Κάποιες συγκρίσεις γίνανε από εμένα, παίρνοντας τα κοινά TSPs τους και βρίσκοντας τα ποσοστά.

5.2 Συγκρίσεις

Σε σύγκριση με τις παραδοσιακές μεθόδους TSP που βασίζονται σε SOM, το TOPSOM χαλαρώνει τον ισχυρό «κλειστό» περιορισμό κατά την αρχικοποίηση και είναι σε θέση να ικανοποιήσει τον περιορισμό του convex-hull χρησιμοποιώντας έναν EN Hebbian μηχανισμό κατά τη βελτιστοποίηση. Το TOPSOM έχει μέσο PDM (δηλαδή την ποσοστιαία απόκλιση της μέσης τιμής της λύσης στο best Known Solution) 4,16% και 8,02% σε μικρής κλίμακας δεδομένων στο ch130 και μεσαίας κλίμακας uy734, ενώ το δεύτερο καλύτερο (δηλαδή eISOM) παίρνει μόνο μέσο PDM 6,37% και 9,87%, αντίστοιχα. Ως αποτέλεσμα, το TOPSOM υπερτερεί του eISOM έως και 2,21% και 1,85%. Για τις περιπτώσεις μεγάλης κλίμακας όπως το ro2950 και το fi10639, το TOPSOM έχει μέσο PDM 14,53% και 21,94%, αντίστοιχα, ενώ το δεύτερο καλύτερο eISOM παίρνει μέσο PDM 15,67% και 23,45%, αντίστοιχα. Έτσι, το TOPSOM υπερτερεί του eISOM έως και 1,14% και 1,51%.Η υπεροχή του TOPSOM γίνεται εμφανής από την άποψη της ταχύτητας επίλυσης προβλημάτων, κατά 14,08% ταχύτερα από το παραδοσιακό SOM κατά μέσο όρο. Το Memetic SOM είναι ανώτερο από το CAN και το ORC-SOM στη δημιουργία των καλύτερων περιηγήσεων κατά μέσο όρο, υποδεικνύοντας ότι το Memetic SOM δεν είναι μόνο γρήγορο αλλά και ανώτερο στη λήψη ακριβέστερων λύσεων, ωστόσο μόνο για περιπτώσεις μικρής κλίμακας. Για περιπτώσεις μεγάλης κλίμακας, το ORC-SOM έχει 6,178% κατά μέσο όρο, το οποίο είναι το μικρότερο σε σύγκριση με εκείνα από το CAN και το Memetic SOM, σημειώνοντας βελτίωση 1,76% και 1,34% αντίστοιχα. Αυτό δείχνει ότι το ORC-SOM είναι κατάλληλο για πιο ακριβείς λύσεις με υψηλότερη υπολογιστική πολυπλοκότητα για προβλήματα μεγάλης κλίμακας σε σύγκριση με το CAN και το Memetic SOM. Το CHSOM έλαβε καλύτερα αποτελέσματα από τον αλγόριθμο του Leung για προβλήματα με 70, 107, 124, 195 και 442 πόλεις. Αν και το CHSOM δεν βρήκε καλύτερες λύσεις για τα KROA200 και ATT532, η ποιότητα της λύσης είναι πολύ κοντά σε αυτή του αλγορίθμου του Leung. Το Expanding Property SOM συναγωνίστηκε το ESOM του Leung και τις επιδόσεις και των δύο

δείχνουν ότι είναι πολύ πιο κοντά στα θεωρητικά όρια. Ο κύριος λόγος για τον οποίο το σφάλμα του Expanding Property SOM είναι ελαφρώς μεγαλύτερο από το ESOM του Leung είναι ότι ο προτεινόμενος κανόνας μάθησης είναι συγκριτικά πιο συντηρητικός από αυτόν του Leung. Έτσι, ο προτεινόμενος κανόνας μάθησης περιορίζει το εύρος της επέκτασης του convex-hull του διεγερμένου νευρώνα. Έχει ταχύτερο μέσο χρόνο εκτέλεσης, αλλά η ποιότητα της λύσης είναι κοντά. Οι ιδιότητες λύσης του ESOM κυμαίνονται από 4,05% έως 11,35%. Έτσι, η απόδοσή του αν και γίνεται λίγο χειρότερη καθώς ο αριθμός των πόλεων αυξάνεται, εξακολουθεί να είναι σταθερή. Το ESOM διαρκεί λίγο περισσότερο χρόνο εκτέλεσης από το SOM του Budinich. Λόγω του γεγονότος ότι το MGSOM και το EGSOM έχουν παρόμοιες προδιαγραφές, ορισμένα από τα TSPs που δοκιμάστηκαν είχαν το ίδιο αποτέλεσμα. Αλλά το MGOM έχει καλύτερη μέση απόκλιση από το βέλτιστο μήκος περιοδείας με συνολικό μέσο όρο 2,3215%, το EGOM με 2,4925% και το MSTSP με 2,4372%. Το Enhanced SOM Solution μετά την επιλογή της βασικής διαμόρφωσης υπερπαραμέτρου, χρησιμοποίησε τις πρόσθετες τροποποιήσεις που χρησιμοποιήθηκαν επιτυγχάνοντας ελαφρώς καλύτερα αποτελέσματα με βαθμολογία F1 10,07891. Το CAN ανταγωνίστηκε τα MEMETIC SOM, ORC-SOM, eISOM, ESOM και SETSP και είχε τα χειρότερα μέσα αποτελέσματα μεταξύ όλων των άλλων. Το Modified CAN συναγωνίστηκε το αρχικό και τα αποτελέσματα έδειξαν ότι ο χρόνος επεξεργασίας μειώθηκε κατά περισσότερο από το ήμισυ στο τροποποιημένο SOM και το PDM έχει καλύτερα αποτελέσματα μόνο όταν ο αριθμός των πόλεων είναι μεταξύ 200-300. Το αρχικό έχει καλύτερα αποτελέσματα για λιγότερες από 200 πόλεις, αλλά όταν είναι πάνω από 300 και οι δύο έχουν παρόμοια αποτελέσματα. Τα πειραματικά αποτελέσματα έδειξαν ότι σε μικρής κλίμακας TSP μόνο ο αλγόριθμος ESOM είναι πιο εμφανής και τα ποσοστά απόκλισης των KNIES και ORC-SOM δεν διαφέρουν πολύ. Αλλά μόλις η κλίμακα των πειραματικών δεδομένων είναι σχετικά μεγάλη, ο αλγόριθμος ORC-SOM έχει τη χαμηλότερη απόκλιση από τον βέλτιστο ρυθμό. Αυτό δείχνει ότι για προβλήματα TSP μεσαίας και μεγάλης κλίμακας, ο αλγόριθμος ORC-SOM είναι πράγματι ευκολότερο να βρεθεί η παγκοσμίως βέλτιστη διαδρομή. Όταν το μέγεθος δεδομένων TSP είναι μεγαλύτερο ή ίσο με 500, σε σύγκριση με τους άλλους δύο αλγόριθμους, το ORC-SOM έχει το μικρότερο βέλτιστο ποσοστό απόκλισης. Όταν το μέγεθος δεδομένων TSP είναι μικρότερο ή ίσο με 1000, οι χρόνοι εκτέλεσης διαφόρων αλγορίθμων είναι πολύ κοντά. Και όταν το μέγεθος δεδομένων TSP είναι μεγαλύτερο από 1000, ο αλγόριθμος ORC-SOM έχει τον μικρότερο χρόνο εκτέλεσης, δείχνοντας τα δικά του πλεονεκτήματα. Μετά την παραπάνω σύγκριση, δεν είναι δύσκολο να διαπιστωθεί ότι για δεδομένα TSP μικρής κλίμακας, το ORC-SOM δεν έχει εξαιρετικά χαρακτηριστικά σε σύγκριση με άλλους αλγόριθμους, ενώ για δεδομένα μεγαλύτερης κλίμακας, το ORC-SOM μπορεί όχι μόνο να βρει μια καλή διαδρομή αλλά και να έχει σχετικά μικρό χρόνο λειτουργίας. Επίσης, το KNIES είναι πιο περίπλοκο από το ESOM. Κατά συνέπεια, η απόδοση του ESOM είναι καλύτερη από την KNIES. Συγκρίνοντας αλγόριθμους, το KNIES-TSP εκτελείται τουλάχιστον σε *n*n* βήματα, ενώ το SETSP εκτελεί τουλάχιστον σε *n* βήματα, οπότε το SETSP έχει δείξει ανώτερη απόδοση σε σύγκριση με το KNIES. Η επιτυχία του KNIES DECOMPOSE αυξάνεται καθώς αυξάνεται το μέγεθος του προβλήματος. Εάν ληφθούν υπόψη τα att532, pcb442, kroA200 και rat195, οι μέσες σχετικές αποκλίσεις από τα βέλτιστα μήκη περιοδείας γίνονται 7,03, 8,94 και 8,93 για το DECOMPOSE KNIES, KNIES_TSP και KNIES_TSP_Global, αντίστοιχα. Εκτός από την ακρίβεια, το KNIES DECOMPOSE μειώνει τον χρόνο λειτουργίας που απαιτείται για το KNIES_TSP_Global στο att532, από 49.888,68 δευτερόλεπτα σε 3,02 δευτερόλεπτα, χωρίς επιδείνωση της ποιότητας του διαλύματος. Η εφαρμογή των Bernard Angeniol, Gael de la Croix vaubois και Jean-Yves le texier συγκρίθηκε

με την μέθοδο Elastic Net που παρουσίασαν οι Durbin και Willshaw. Τα αποτελέσματα για πέντε ομάδες 50 πόλεων παρουσιάζουν παρόμοια χαρακτηριστικά. Κατά μέσο όρο, και οι δύο προσεγγίσεις είναι ισοδύναμες. Τα αποτελέσματα σε σύγκριση με το Tank και το Hopfield δείχνουν ότι μια χαμηλή τιμή της παραμέτρου *a* δίνει έναν καλύτερο μέσο όρο, αλλά μια υψηλή τιμή έχει το πλεονέκτημα ότι μερικές φορές επιτυγχάνεται η βέλτιστη. Δίνει επίσης μια καλή λύση σε πολλές προσπάθειες σε λιγότερο χρόνο από μία προσπάθεια με χαμηλή τιμή *α*. Φαίνεται ότι το υβριδικό 2opt + SOM δεν είναι ένας πολύ ισχυρός αλγόριθμος για το TSP. Έχει ξεπεραστεί και από τους δύο αλγόριθμους: αλγόριθμους EA και Lin-Kerninghan. Το μοντέλο A different Hybrid approach ξεπερνά το ACO και το SOM σε επίπεδο 5%. Στο Hybrid μοντέλο, αυτή η μέθοδος ανταγωνιζόταν τον Zhou και ήταν ταχύτερη κατά 84% λαμβάνοντας υπόψη τα ίδια TSPs και 11% ταχύτερη από τον αλγόριθμο του Peker. Σε σύγκριση με την προσέγγιση που χρησιμοποιεί τα K-means και τον αλγόριθμο Firefly, παρατηρείται ότι το Clustered SOM με το συμπληρωματικό GA είναι ανώτερο τόσο από την άποψη της αντικειμενικής τιμής όσο και του χρόνου υπολογισμού. Το μοντέλο ART/SOFM έχει υπέρβαση 10% σε σχέση με το βέλτιστο μήκος περιήγησης στο TK2392 και το eISOM στο ίδιο TSP έχει πλεόνασμα 6,44%. Επίσης, στον ίδιο αριθμό πόλεων, στο PR2392 το TSP Memetic SOM έχει 7,34%, το ESOM έχει 8,49%, το eISOM έχει 6,44% και το ORC-SOM είναι 7,16%. Το Eli51 είναι ένα από τα κοινά TSPs που χρησιμοποιείται πολλές φορές. Το SETSP έχει υπολογιστικό αποτέλεσμα 435, κοντά στο βέλτιστο μήκος. Επιπλέον, το Fuzzy έχει το ίδιο όπως παραπάνω και στο MGSOM έχει 431 όπως και το EGSOM. Το μοντέλο Hybrid έχει 440. Το Self organizing Iterative έχει 470. Όσον αφορά την υπέρβαση %, τα αποτελέσματα του Eli51 στο eISOM είναι 2,56%, στο SETSP είναι 2,22%, στο MEMETIC είναι 2,52%, στο EGSOM είναι 1,39% καθώς και το MGSOM και το ORC-SOM 3.00%. Τα παραπάνω αποτελέσματα υποδεικνύουν προβλήματα TSP μικρής κλίμακας. Τώρα θα δείξω αποτελέσματα από μια μεγάλη κλίμακα προβλημάτων που σημαίνει περισσότερες πόλεις. Το TSP που ονομάζεται fi10639 έχει 10639 συνολικές πόλεις. Το TOPSOM όπως παρουσιάστηκε προηγουμένως έχει ποσοστιαία Απόκλιση του Μέσου διαλύματος ίση με 21,94%. Σε σύγκριση με το eISOM που έχει 23,45% και το παραδοσιακό SOM με 27,02%. Το MEMETIC έχει 6,93%. Το TSP berlin52 έχει 52 πόλεις, οπότε είναι ένα πρόβλημα μικρής κλίμακας. Το βέλτιστο μήκος περιήγησης του παραδοσιακού SOM είναι 7544. Το αποτέλεσμα του μήκους του Self organizing Iterative σε αυτό το TSP είναι 7959. Το eISOM έχει μέσο μήκος 8148 και το TOPSOM τιμή 7995. Επίσης, στο Hybrid μοντέλο, η μέση τιμή είναι 8208. Επιπλέον, στο A different hybrid approach, η μέση τιμή είναι 7669. Ένα κοινό χαρακτηριστικό των περισσότερων μοντέλων είναι η Ευκλείδεια απόσταση που χρησιμοποιείται για να βρεθεί ο πλησιέστερος γείτονας. Μόνο στο A different hybrid approach πήρε το TSP από το Welch Two Sample t-test. Εκτός από αυτό, όλοι τους χρησιμοποίησαν παρουσίες TSPLIB. Επιπλέον, το TOPSOM χρησιμοποίησε National TSP's δεδομένα. Το CHAOSOM συγκρίθηκε με το SOM και το ποσοστιαίο σφάλμα της βέλτιστης απόστασης στο CHAOSOM έχει δείξει καλύτερα αποτελέσματα. Το Fuzzy συγκρίθηκε με EA και τον αλγόριθμο του Lin Kerninghan, και αυτός ο αλγόριθμος παρέχει σημαντικά καλύτερα αποτελέσματα για το TSP καθώς αυξάνεται ο αριθμός των πόλεων. Ο μέσος ρυθμός βελτίωσης της ακρίβειας της λύσης (μέσο σφάλμα για το βέλτιστο) στη διαδικασία SOM + SA εκτιμάται ως περίπου 2%, αλλά είναι διάσπαρτος ανάλογα με το μέγεθος των προβλημάτων. Το PMSOM συγκρίθηκε με το MEMETIC SOM και τα αποτελέσματα έδειξαν ότι είναι σε θέμα ταχύτητας επεξεργασίας είναι πολύ πιο γρήγορο γιατί χρησιμοποιεί clusters όσο ο αριθμός των πόλεων αυξάνεται και PDM% 4,53 έναντι 5,87 του MEMETIC SOM.

5.3 Επίλογος

Συνοψίζοντας, σύλλεξα μερικά μοντέλα που θεωρώ ότι έχουν τα καλύτερα SOMs και στη συνέχεια συγκέντρωσα τα κοινά TSPs τους, και μετά από συγκρίσεις έβγαλα κάποια συμπεράσματα τα οποία θα αναφερθούν σε επόμενο κεφάλαι

Κεφάλαιο 6ο: Σχολιασμός πινάκων

Ο πίνακας 6.1 και ο πίνακας 6.4 αποτελούν σύνοψη όλων των μοντέλων. Πρώτον, ο πίνακας αποτελείται από τον τύπο του μοντέλου. Επιπλέον Στην επόμενη στήλη εμφανίζεται η ημερομηνία κυκλοφορίας του μοντέλου, το Σύνολο δεδομένων, διαφορετικά, ο αριθμός των πόλεων που δοκιμάστηκαν, προκειμένου να ληφθεί η απόδοση του αλγορίθμου με το συνδυασμό των επαναλήψεων. Επίσης, παρουσιάζεται η απόδοση του μοντέλου, δηλαδή σε τι σύνολα δεδομένων έδειξε να αποδίδει καλύτερα. Επιπλέον, δίνεται η βιβλιοθήκη των TSPs που χρησιμοποιήθηκαν. Ο Πίνακας 6.6 εμφανίζει μια σύνοψη των πολλαπλών πωλητών και έχει τις ιδιότητες όπως οι προηγούμενοι πίνακες. Πολλά μοντέλα εκτελούνται και είναι ικανά να δώσουν καλύτερα αποτελέσματα για προβλήματα μεγάλης κλίμακας και άλλα για μικρά σύνολα δεδομένων. Η προτελευταία στήλη αλλά εξίσου σημαντική, δείχνει τον αριθμό των νευρώνων που χρησιμοποιούνται στην πρώτη φάση του αλγορίθμου που είναι συνήθως η φάση αρχικοποίησης. Συνήθως, ο αριθμός των νευρώνων που χρησιμοποιούνται είναι ίσος με τον αριθμό των πόλεων του TSP, αλλά μερικές φορές είναι είτε 1,5 είτε 2 φορές είτε περισσότερες. Η επιλογή του αριθμού των κόμβων που είναι ίση με τον αριθμό των πόλεων καθιστά τη διαδικασία διαχωρισμού κόμβων πιο λεπτή και έχει ως αποτέλεσμα την ευαισθησία του αλγορίθμου στην αρχική διαμόρφωση. Όλα τα μοντέλα έχουν ένα κοινό. Έχουν 2 στρώματα που είναι το επίπεδο εισόδου και εξόδου. Στο επίπεδο εισόδου, λαμβάνουν ως είσοδο τις καρτεσιανές συντεταγμένες του σημείου (πόλη). Οι πίνακες 6.2, 6.3, 6.5 και 6.7 παρουσιάζουν τα μοντέλα που συζητήσαμε πριν και τα μοντέλα ή τους αλγόριθμους που ανταγωνίζονταν κατά τη διάρκεια των πειραμάτων τους. Με το σύμβολο ***X***, μπορούμε να αντιληφθούμε τους ανταγωνιστές τους. Κατά τις συγκρίσεις οι έννοιες σημαίνουν AVL-Angeniol, Vaubois and LeTexier, GN-GuiltyNet, KL-KNIES, SA-Simulated annealing, KG-KNIES global, CEN-convex elastic net, K-means-Firefly(combination). Ο πίνακας 6.8 έχει συγκεντρωτικά τις κατηγορίες των μοντέλων και τα μοντέλα σε αυτά.

Τύπος	Όνομα μοντέλου	Date	Σετ δεδομένων & Επαναλήψεις	Απόδοση	Το νούμερο των νευρώνων σε σχέση με τις πόλεις	Πληροφορίες
Greedy	Self-Organizing Iterative	2018	51 έως 200 πόλεις TSPLIB instances Δεν βρέθηκαν επαναλήψεις	Μικρά δεδομένα	Νευρώνες = Πόλεις	5-10% ποιότητα διαλύματος (% από βέλτιστη).
Greedy	A different Hybrid approach	2015	52 έως 561 πόλεις Welch Two Sample t-test Επαναλήψεις > 2000	Δεν αναφέρει	Νευρώνες = Πόλεις	Η υβριδική προσέγγιση υπερτερεί των ACO και SOM σε επίπεδο 5%.
Evolutionary	CVOA + SOM	2021	14 έως 99 πόλεις TSPLIB instances 2500 ή 3000 επαναλήψεις	Μεγάλα δεδομένα	Νευρώνες = Πόλεις	Ανταγωνίστηκε το κλασικό SOM.
Evolutionary	Hybrid	2018	51 έως 442 πόλεις TSPLIB95 instances 500000 επαναλήψεις	Μεγάλα δεδομένα	Νευρώνες = 2 × Πόλεις	Λιγότερος χρόνος επεξεργασίας αλλά όχι καλύτερη ποιότητα λύσης στα περισσότερα.
Evolutionary	2opt + SOM	2007	51 έως 10000 πόλεις TSPLIB instances 25000 επαναλήψεις	Μεγάλα δεδομένα	Νευρώνες = Πόλεις	Η ταχύτητά του μπορεί να είναι εντυπωσιακή, αλλά εξακολουθεί να είναι αργή.
Evolutionary	SETSP	2003	70 έως 442 πόλεις TSPLIB instances Επαναλήψεις > 30	Μεγάλα δεδομένα	Νευρώνες > Πόλεις	3.69 Μέση απόκλιση από τη βέλτιστη διάρκεια περιοδείας σε σύγκριση με τα KL, KG.
Genetic	Clustered SOM	2021	100 έως 10000 πόλεις TSPLIB instances Επαναλήψεις > 50	Μεγάλα δεδομένα	Νευρώνες = Πόλεις	Σε σύγκριση με toSOM-Firefly [72], K-means-GA, και toK-means-Firefly.
Genetic	EvolvedISOM (eISOM)	2003	30 έως 2393 πόλεις TSPLIB instances 160 × n επαναλήψεις	Μεγάλα δεδομένα	Νευρώνες = Πόλεις	3.24% ποιότητα λύσης στο βέλτιστο μήκος.
Genetic	ART/SOFM	2001	1000 έως 14000 πόλεις TSPLIB instances Δεν βρέθηκαν επαναλήψεις	Μεγάλα δεδομένα	Νευρώνες = Πόλεις	Ποσοστιαία υπέρβαση σε σχέση με την περιοδεία Lin Kernighan.
Memetic	MEMETIC	2008	29 έως 85900 πόλεις TSPLIB instances Δεν βρέθηκαν επαναλήψεις	Μικρά δεδομένα	Νευρώνες = 2 × Πόλεις	Καλύτερα %PDM και %PDB στους περισσότερους ανταγωνιστές της, ιδίως σε μικρή κλίμακα.
Memetic	PMSOM	2015	29 έως 10639 πόλεις TSPLIB instances 60 έως 480 επαναλήψεις	Μεγάλα δεδομένα	Νευρώνες = 5 × Πόλεις	Αυτή η μεθοδολογία είναι περισσότερο από 4 φορές ταχύτερη από εκείνες των μη παράλληλων συστημάτων κατά μέσο όρο.
Chaotic	CHAOSOM	2006	48 πόλεις TSPLIB instances 100 επαναλήψεις	Δεν αναφέρει	Νευρώνες = 2 × Πόλεις	2.02% σφάλμα από τη βέλτιστη απόσταση.
Chaotic	Chaotic maps	2015	50 έως 10000 πόλεις OpenOpal framework 1000 έως 1000000 επαναλήψεις	Εξαρτάται από τον χάρτη	Νευρώνες = Πόλεις	Burgers, Delayed Logistics, ή Tinkerbell Map είναι για μεγάλα προβλήματα και Henon Map είναι για τα μικρά.
Fuzzy	Fuzzy	2008	51 έως 1200 πόλεις TSPLIB instances 25000 επαναλήψεις	Μεγάλα δεδομένα	Νευρώνες = Πόλεις	Έχει καλύτερα μέσα ποσοστιαία αποτελέσματα από τον Lin Kernighan.
Simulated annealing	SOM + SA	2010	10 έως 2000 πόλεις TSPLIB instances Επαναλήψεις = πόλεις × 5	Μεγάλα δεδομένα	Νευρώνες = Πόλεις	Η υπολογιστική του πολυπλοκότητα είναι περίπου $O(k * n^{1.8})$.
Elastic Net	EN + SOM	2007	1000 έως 2000 πόλεις Δεν βρέθηκε βιβλιοθήκη 10 εκατομμύρια επαναλήψεις	Δεν αναφέρει	Νευρώνες = 2.5 × Πόλεις	Η ενσωμάτωση των SOM και EN έχει αποδειχθεί ότι είναι ένας πολύ αποτελεσματικός τρόπος για τον βέλτιστο σχεδιασμό διαδρομών.

Πίνακας 6.1: Υβριδικά μοντέλα

Σχολιασμός πινάκων

Σύγκριση	Greedy	ACO	Kohonen	Peker	Yongquan Zhou	AVL	GN	KL	MEMETIC
Self-Organizing Iterative	***X***								
A different Hybrid approach		***X***	***X***						
CVOA + SOM			***X***						
Hybrid				***X***	***X***				
SETSP			***X***			***X***	***X***	***X***	
SOM + SA			***X***						
PMSOM									***X***

Πίνακας 6.2: Ανταγωνιστές υβριδικών μοντέλων 1

Σύγκριση	ESOM	SA	Budinich's SOM	CEN	Lin Kernighan	CAN	eISOM	Kohonen	Evolutionary	K-means-Firefly (combination)
Clustered SOM										***X***
EvolvedISOM (eISOM)	***X***	***X***	***X***	***X***						
ART/SOFM					***X***					
MEMETIC	***X***					***X***	***X***			
CHAOSOM					***X***			***X***		
Fuzzy					***X***				***X***	***X***
2opt + SOM					***X***				***X***	

Πίνακας 6.3: Ανταγωνιστές υβριδικών μοντέλων 2

Τύπος	Όνομα μοντέλου	Date	Σετ δεδομένων & Επαναλήψεις	Απόδοση	Το νούμερο των νευρώνων σε σχέση με τις πόλεις	Πληροφορίες
Convex Hull	TOPSOM	2021	52 έως 16862 πόλεις TSPLIB instances National TSP's instances 30000 επαναλήψεις	Μεγάλα δεδομένα	Νευρώνες = 8 × Πόλεις	Σε σύγκριση με το παραδοσιακό SOM, το TOPSOM έχει καλύτερες διαδρομές με μείωση έως και 7.67% όσον αφορά το PDM και μείωση 3.19% κατά μέσο όρο όσον αφορά το PDB.
Convex Hull	ORC-SOM	2011	51 έως 2392 πόλεις TSPLIB instances 200 επαναλήψεις	Μεγάλα δεδομένα	Νευρώνες = 2 ή 3 × Πόλεις	Το ORC-SOM είναι ανώτερο σε μέσο ποσοστό από τα αντίστοιχά του στην ποιότητα του διαλύματος.
Convex Hull	CHSOM	2008	70 έως 532 πόλεις TSPLIB instances 200 επαναλήψεις	Δεν αναφέρει	Νευρώνες = Πόλεις	Αποκτήθηκε καλύτερη ποιότητα λύσης (% από τη βέλτιστη) από τον αλγόριθμο του Leung σε ορισμένα TSPs, αλλά όχι σε όλα.
Convex Hull	Expanding Property SOM	2005	2400 πόλεις Δεν βρέθηκε βιβλιοθήκη 100 × n επαναλήψεις	Μεγάλα δεδομένα	Νευρώνες = Πόλεις	Το πλάτος μειώνεται γραμμικά στο 1 στο πρώτο 65% των επαναλήψεων.
Convex Hull	ESOM	2004	50 έως 2400 πόλεις TSPLIB instances 100 × n επαναλήψεις	Μικρά δεδομένα	Νευρώνες = Πόλεις	4.18% ποιότητα λύσης κατά μέσο όρο σε 20 TSPs. Σε σύγκριση με KNIES-global, KNIES-local, SA και Budinich.
Growing Ring	MGSOM	2005	6 έως 442 πόλεις TSPLIB instances 20 επαναλήψεις	Μεγάλα δεδομένα	Νευρώνες = 2 × Πόλεις	2.3215 μέσες αποκλίσεις από το βέλτιστο μήκος περιοδείας σε σύγκριση με τα KL, KG και SETSP.
Growing Ring	EGSOM	2006	51 έως 106 πόλεις TSPLIB instances Δεν βρέθηκαν επαναλήψεις	Μεγάλα δεδομένα	Νευρώνες = 1.5 × Πόλεις	2.4925% διαφορά από τη βέλτιστη περιήγηση και 233.8 μέσος χρόνος επεξεργασίας (δευτ.)
Growing Ring	MSTSP	2006	51 έως 442 πόλεις TSPLIB instances 20 επαναλήψεις	Μεγάλα δεδομένα	Νευρώνες = 2 × Πόλεις	Η μέση απόκλιση 2.4372% από το βέλτιστο μήκος περιοδείας.
Heuristic	Enhanced SOM	2021	50 έως 200 πόλεις Δεν βρέθηκε βιβλιοθήκη 100000 επαναλήψεις	Δεν αναφέρει	Νευρώνες = Πόλεις	0,07891 της βαθμολογίας F1 που επιτεύχθηκε.
Heuristic	SDP	2005	100000 πόλεις TSPLIB instances 50 έως 500 επαναλήψεις	Μεγάλα δεδομένα	Νευρώνες = 5 × Πόλεις	Τα αποτελέσματα δείχνουν μια πολύ ωραία συμπεριφορά επιτάχυνσης και καθιστούν βιώσιμη τη λύση πολύ μεγάλων TSPs, έως και εκατοντάδων χιλιάδων πόλεων.
Heuristic	CAN	2003	51 έως 85900 πόλεις TSPLIB instances Πόλεις / 2 επαναλήψεις	Μεγάλα δεδομένα	Νευρώνες = 2.5 × Πόλεις	Το CAN και το ORCSOM φαίνεται να είναι πιο ισχυρά.
Heuristic	Modified CAN	2011	6 έως 574 πόλεις TSPLIB instances Πόλεις / 2 επαναλήψεις	Μικρά δεδομένα	Νευρώνες = 2.5 × Πόλεις	Το σφάλμα στον CAN αλγόριθμο είναι αμελητέο για μικρά προβλήματα.
Heuristic	GSOA	2018	51 έως 2392 πόλεις TSPLIB instances 150 επαναλήψεις	Μεγάλα δεδομένα	Νευρώνες = 2.5 × Πόλεις	Ταχύτερος χρόνος υπολογισμού και καλύτερα αποτελέσματα από το ORC-SOM.
Heuristic	KNIES	1999	51 έως 532 πόλεις TSPLIB instances Δεν βρέθηκαν επαναλήψεις	Μικρά δεδομένα	Νευρώνες = Πόλεις	2.73% σχετική απόκλιση από το βέλτιστο μήκος περιοδείας. Καλύτερα από PKN, GN, AVL, KG.
Heuristic	KNIES DECOMPOSE	2003	51 έως 2392 πόλεις TSPLIB instances Δεν βρέθηκαν επαναλήψεις	Μεγάλα δεδομένα	Νευρώνες = Clusters	Δεν είναι ο αριθμός των clusters, αλλά η ποιότητα των clustering που είναι ζωτικής σημασίας. Αυτό έχει ως αποτέλεσμα υψηλότερη ταχύτητα και ακρίβεια ταυτόχρονα.
Heuristic	Modares, Somhom & Enkawa	1997	51 έως 1400 πόλεις TSPLIB instances Δεν βρέθηκαν επαναλήψεις	Μεγάλα δεδομένα	Νευρώνες = 2 × Πόλεις	Η απόκλιση από τις πιο γνωστές λύσεις είναι 2.17% στη χειρότερη περίπτωση και 1.22% κατά μέσο όρο.
Heuristic	Bernard Angeniol,	1988	1000 πόλεις Δεν βρέθηκε βιβλιοθήκη Δεν βρέθηκαν επαναλήψεις	Μεγάλα δεδομένα	Νευρώνες = Πόλεις	Σε σύγκριση με το EA μέθοδο που παρουσίασαν οι Durbin και Willshaw.

Πίνακας 6.4: Εμπλουτισμένα μοντέλα

Σχολιασμός πινάκων

Σύγκριση	Kohonen	CAN	eISOM	ESOM	Elastic net	AVL	GN	KL	ORC-SOM	Somhom's	Matsuyama
TOPSOM	***X***	***X***	***X***								
ORC-SOM	***X***			***X***							
CHSOM				***X***		***X***					
Expanding Property SOM				***X***							
ESOM					***X***						
MGSOM	***X***					***X***	***X***	***X***			
MSTSP						***X***	***X***	***X***			
CAN									***X***		
KNIES				***X***					***X***		
Bernard Angeniol,					***X***						
GSOA		***X***							***X***		
Modified CAN		***X***								***X***	
KNIES DECOMPOSE								***X***			
Modares, Somhom & Enkawa					***X***	***X***	***X***				***X***

Πίνακας 6.5: Ανταγωνιστές εμπλουτισμένων μοντέλων

Όνομα μοντέλου	**Date**	**Σετ δεδομένων & Επαναλήψεις**	**Απόδοση**	**Το νούμερο των νευρώνων σε σχέση με τις πόλεις**	**Πληροφορίες**
MinMax Multiple-TSP	2019	50 έως 99 πόλεις TSPLIB instances 5000 επαναλήψεις	Δεν είναι συγκεκριμένο	Νευρώνες = 3 × Πόλεις	Η βέλτιστη λύση επιτυγχάνεται σε ορισμένες περιπτώσεις προβλημάτων
MTSP	2003	29 έως 561 πόλεις TSPLIB instances 100 επαναλήψεις	Δεν αναφέρει	Νευρώνες = Πόλεις	Καλύτερη διάρκεια της μέσης λύσης και χρόνος λειτουργίας από το συμβατικό SOM.

Πίνακας 6.6: Multiple TSP μοντέλα

Σύγκριση	ACO	SOM	SOM-ACO	EA	SOM-EA	SOM-EA-2opt	Modares, Somhom, Enkawa
MinMax Multiple-TSP	***X***	***X***	***X***	***X***	***X***	***X***	
MTSP							***X***

Πίνακας 6.7: Ανταγωνιστές Multiple TSP μοντέλων

Heuristic	Enhanced SOM	SDP	CAN	Modified CAN	KNIES	KNIES DECOMPOSE	Modares, Somhom & Enkawa	GSOA	Bernard Angeniol,
Convex-Hull	ORCSOM	ESOM	TOPSOM	CHSOM	Expanding Property SOM				
Evolutionary	CVOA + SOM	SETSP	Hybrid	2opt + SOM					
Genetic	Clustered SOM	eISOM	ART/SOFM						
Multiple Salesman	MTSP	MinMax Multiple-TSP							
Greedy	Iterative	A different Hybrid approach							
Chaotic	Chaotic Maps	CHAOSOM							
Memetic	MEMETIC	PMSOM							
Fuzzy	Fuzzy								
Simulated Annealing	SOM + SA								
Elastic Net	EN + SOM								

Πίνακας 6.8: Κατηγορίες μοντέλων

Κεφάλαιο 7ο: Επίλογος

Όπως έχει ήδη αναφερθεί σκοπός της παρούσας Πτυχιακής εργασίας ήταν η περιγραφή του προβλήματος του περιοδεύοντος πωλητή και πιο συγκεκριμένα η επίλυση του με Χάρτες Αυτό-οργάνωσης. Αναλύθηκαν τα διαφορετικά μοντέλα των χαρτών αυτών, και έγινε μια συγκεντρωτική κατηγοροποίηση των μοντέλων σε υβριδικά και μη. Κάποια μοντέλα είχανε ομοιότητες λόγο ότι τα περισσότερα βασίζονταν σε προηγούμενες μελέτες. Επίσης έγιναν αναφορές Χαρτών Αυτό-οργάνωσης σε άλλες εφαρμογές. Τα συμπεράσματα που βγήκαν μετά από συγκρίσεις είναι ότι το ORC-SOM έχει τα καλύτερα μέσα αποτελέσματα από τη βέλτιστη διάρκεια περιοδείας λαμβάνοντας υπόψη όλα τα άλλα εμπλουτισμένα μοντέλα SOM για προβλήματα μικρής κλίμακας. Όταν πρόκειται για μεγάλα σύνολα δεδομένων, το TOPSOM παράγει τα ταχύτερα αποτελέσματα με καλή απόδοση. Στα υβριδικά μοντέλα, για μικρά σύνολα δεδομένων, το μοντέλο eISOM παράγει τα καλύτερα αποτελέσματα και το Fuzzy φαίνεται να παράγει τον ταχύτερο χρόνο αλλά όχι τόσο αξιοπρεπή αποτελέσματα όσο το eISOM. Στη μεγάλη κλίμακα προβλημάτων όπως ένα 10000 ενός TSP κατά μέσο όρο, PMSOM και το MEMETIC SOM έχουν τις καλύτερες βαθμολογίες. Το PMSOM έχει την καλύτερη βαθμολογία το οποίο απαιτεί παραλληλοποίηση. Το MinMax Multiple-TSP παράγει τα καλύτερα αποτελέσματα στην κατηγορία των πολλαπλών πωλητών.

Συνοψίζοντας, αυτό το συγκεντρωτικό έργο είναι κατάλληλο για κάποιον που θέλει να ερευνήσει τα συγκεκριμένα νευρωνικά δίκτυα σε προβλήματα βελτιστοποίησης.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Travelling salesman problem. Wikipedia. Available: https://en.wikipedia.org/wiki/Travelling_salesman_problem.

[2] Rashid Ahmad and Dohyeun Kim, “An Extended Self-Organizing Map based on 2-opt algorithm for solving symmetrical Traveling Salesperson Problem,” *Neural Comput.* Applications, vol. 26, issue 4, pp. 987–994, May 2015, doi: 10.1007/s00521-014-1773-z.

[3] Danijel MARKOVIĆ, Miloš MADIĆ, Vojislav TOMIĆ, Sonja STOJKOVIĆ, “SOLVING TRAVELLING SALESMAN PROBLEM BY USE OF KOHONEN SELF-ORGANIZING MAPS,” JOURNAL OF ACTA TECHNICA CORVINIENSIS – Bulletin of Engineering, Tome V(2012), ISSN 2067-3809, pp. 21-24.

[4] Q. Guan, X. Hong, W. Ke, L. Zhang, G. Sun and Y. Gong, “Kohonen Self-Organizing Map based Route Planning: A Revisit,” 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2021, pp. 7969-7976, doi: 10.1109/IROS51168.2021.9636025.

[5] Zhang, Junying & Feng, Xuerong & Zhou, Bin & Ren, Dechang, “An overall-regional competitive self-organizing map neural network for the Euclidean traveling salesman problem,” Neurocomputing, volume 89, 2012, pp. 1–11, doi: 10.1016/j.neucom.2011.11.024.

[6] Xinshun Xu, Zhiping Jia, Jun Ma, and Jiahai Wang, “A Self-Organizing Map Algorithm for the Traveling Salesman Problem,” In Proceedings of the 2008 Fourth International Conference on Natural Computation - Volume 03 (ICNC '08), IEEE Computer Society, USA, 2008, pp.431–435, doi:10.1109/ICNC.2008.569.

[7] Haiqing Yang and Haihong Yang, “An Self-organizing Neural Network with Convex-hull Expanding Property for TSP,” 2005 International Conference on Neural Networks and Brain, 2005, pp. 379-383, doi: 10.1109/ICNNB.2005.1614637.

[8] Kwong-Sak Leung, Hui-Dong Jin, and Zong-Ben Xu, “An expanding self-organizing neural network for the traveling salesman problem,” Neurocomput. 62, C (December 2004), pp. 267–292, doi: 10.1016/j.neucom.2004.02.006.

[9] Yanping Bai & Zhang, Wendong & Jin, Zhen, “An new self-organizing maps strategy for solving the traveling salesman problem,” Chaos, Solutions & Fractals, volume 28, 2006, pp. 1082-1089, doi: 10.1016/j.chaos.2005.08.114.

[10] Yanping Bai, Wendong Zhang, Hongping Hu, “An Efficient Growing Ring SOM and Its Application to TSP,” Proceedings of the 9th WSEAS International Conference on Applied Mathematics, Istanbul, Turkey, May 27-29, 2006, pp. 351-355.

[11] Wen Dong Zhang, Yan Ping Bai, and Hong Ping Hu, “The incorporation of an efficient initialization method and parameter adaptation using self-organizing maps to solve the TSP,” Applied Mathematics and Computation 172, 1 January 2006, pp. 603–623, doi: 10.1016/j.amc.2005.02.025.

[12] Joao P. A. Dantas, Andre N. Costa, Marcos R. O. A. Maximo, Takashi Yoneyama, “Enhanced Self-Organizing Map Solution for the Traveling Salesman Problem,” Neural and Evolutionary Computing (cs.NE), 2021, doi: 10.48550/arXiv.2201.07208.

[13] Zhang D.,Wang J. and Zhao X, “Estimating the uncertainty of average f1 scores,” In Proceedings of the 2015 International Conference on The Theory of Information Retrieval, 2015, pp. 317–320, doi: 10.1145/2808194.2809488.

[14] Somhom, S., Modares, A. & Enkawa, T, ”A self-organising model for the travelling salesman problem,” J Oper Res Soc 48, 1997, pp. 919–928, doi:10.1057/palgrave.jors.2600439.

[15] Y. Matsuyama, "Competitive self-organization and combinatorial optimization: applications to traveling salesman problem," 1990 IJCNN International Joint Conference on Neural Networks, 1990, pp. 819-824 vol.3, doi: 10.1109/IJCNN.1990.137937.

[16] H. Schabauer, E. Schikuta and T. Weishaupl, “Solving Very Large Traveling Salesman Problems by SOM Parallelization on Cluster Architectures,” Sixth International Conference on Parallel and Distributed Computing Applications and Technologies (PDCAT'05), 2005, pp. 954-958, doi: 10.1109/PDCAT.2005.223.
[17] Roman Sushkov & Miroslav Kulich, “Self-Organizing Structures for the Travelling Salesman Problem in a Polygonal Domain,” CZECH TECHNICAL UNIVERSITY IN PRAGUE, Faculty of Electrical Engineering, 2015.
[18] EM Cochrane and JE Beasley, “The co-adaptive neural network approach to the euclidean travelling salesman problem,” Neural Networks Volume 16, Issue 10, December 2003, pp. 1499-1525, doi: 10.1016/S0893-6080(03)00056-X.
[19] Burke, L. I., & Damany, P, “The guilty net for the traveling salesman problem. Computers and Operations Research, 19, pp. 255–265, 1992, doi: 10.1016/0305-0548(92)90047-9.
[20] Durbin, R., & Willshaw, D, “An analogue approach to the travelling salesman problem using an elastic net method,” Nature 326, pp. 689-691. 1987, doi: 10.1038/326689a0.
[21] Jan Faigl, “On the performance of self-organizing maps for the non-Euclidean Traveling Salesman Problem in the polygonal domain,” Information Sciences Volume 181, Issue 19, 2011, pp. 4214-4229, doi: 10.1016/j.ins.2011.05.019.
[22] Kazushi Murakoshi, Yuichi Sato, “Reducing topological defects in self-organizing maps using multiple scale neighborhood functions,” Biosystems 90 (1), (2007), pp. 101–104, doi: 10.1016/j.biosystems.2006.07.004.
[23] N. Aras, B.J. Oommen, I.K. Altınel, “The Kohonen network incorporating explicit statistics and its application to the travelling salesman problem,” Neural Networks Volume 12, Issue 9, 1999, pp. 1273-1284, doi: 10.1016/S0893-6080(99)00063-5.
[24] N. Aras, I. K. Altinel and J. Oommen, “A Kohonen-like decomposition method for the Euclidean traveling salesman problem-KNIES-DECOMPOSE,” in IEEE Transactions on Neural Networks, vol. 14, no. 4, 2003, pp. 869-890, doi: 10.1109/TNN.2003.811562.
[25] Jan Faigl, “GSOA: Growing Self-Organizing Array - Unsupervised learning for the Close-Enough Traveling Salesman Problem and other routing problems,” Neurocomputing, 312, C (Oct 2018), pp. 120–134, doi: 10.1016/j.neucom.2018.05.079.
[26] Faigl, Jan, “An application of the self-organizing map in the non-Euclidean Traveling Salesman Problem,” Neurocomputing 74, 2011, pp. 671-679, doi: 10.1016/j.neucom.2010.08.026.
[27] Urfat Nuriyev, Onur Ugurlu, Fidan Nuriyeva, “Self-Organizing Iterative Algorithm for Travelling Salesman Problem,” IFAC PapersOnLine volume 51, issue 30, 2018, pp. 268–270, doi: 10.1016/j.ifacol.2018.11.299.
[28] Kızılateş G., Nuriyeva F., “On the Nearest Neighbour Algorithms for Travelling Salesman Problem.” Advances in Computational Science. Engineering and Information Technology, Springer (AISC,volume 225), 2013, pp. 111-118, doi: 10.1007/978-3-319-00951-3_11.
[29] Appligate, D.L., Bixby, R.E., Chavatal, V., Cook, W.J., “The Travelling Salesman Problem,” A Computational Study, Princeton University Press, Princeton and Oxford, 2006, 593p. ISBN 978-0-691-12993-8.
[30] Dr. Carsten Mueller, Niklas Kiehne, ”Hybrid Approach for TSP Based on Neural Networks and Ant Colony Optimization,” 2015 IEEE Symposium Series on Computational Intelligence, 2015, pp. 1431-1435, doi: 10.1109/SSCI.2015.203257094.
[31] EL Majdoubi, O., Abdoun, F., Abdoun, O, “A New Optimized Approach to Resolve a Combinatorial Problem: CoronaVirus Optimization Algorithm and Self-organizing Maps,” In: Motahhir, S., Bossoufi, B. (eds) Digital Technologies and Applications, ICDTA 2021, Lecture Notes in Networks and Systems, vol 211, Springer, Cham., 2021, pp. 947–957, doi: 10.1007/978-3-030-73882-2_86.
[32] Pongpinyo Tinarut, Komgrit Leksakul, “Hybrid Self-Organizing Map Approach for Traveling Salesman Problem,” CMU J. Nat. Sci. (2019) Vol. 18(1), 2019, pp. 27-37, doi: 10.12982/CMUJNS.2019.0003.

[33] Brocki, Ł., Koržinek, D, "Kohonen Self-Organizing Map for the Traveling Salesperson Problem," In: Jabłoński, R., Turkowski, M., Szewczyk, R. (eds) Recent Advances in Mechatronics. Springer, Berlin, Heidelberg, 2007, pp. 116-119, doi: 10.1007/978-3-540-73956-2_24.

[34] Frederico Carvalho Vieira, Adriao Duarte Doria Neto and Jose Alfredo Ferreira Costa, "An Efficient Approach to the Travelling Salesman Problem Using Self-Organizing Maps", International Journal of Neural Systems, vol. 13, no. 2, pp. 59-66, 2003, doi: 10.1142/S0129065703001443.

[35] Nourmohammadzadeh, A., Voß, S, "Hybridising Self-Organising Maps with Genetic Algorithms," In: Simos, D.E., Pardalos, P.M., Kotsireas, I.S. (eds) Learning and Intelligent Optimization, LION 2021, Lecture Notes in Computer Science(), vol 12931, Springer, Cham, 2021, pp. 265–282, doi: 10.1007/978-3-030-92121-7_22.

[36] Grice, J.V., Montgomery, D.C.: Design and analysis of experiments. Technometrics 42(2), 208-209 (2000), doi: 10.2307/1271458.

[37] Hui-Dong Jin, Kwong-Sak Leung, Man-Leung Wong and Zong-Ben Xu, "An Efficient Self-Organizing Map Designed by Genetic Algorithms for the Traveling Salesman Problem," IEEE Trans. on Systems, Man, and Cybernetics Part B: Cybernetics, vol. 33, 2003, pp. 877-888, doi: 10.1109/TSMCB.2002.804367.

[38] Narayan Vishwanathan, Donald C. Wunsch, "ART/SOFM: A Hybrid Approach to the TSP," IEEE. IJCNN'01. International Joint Conference on Neural Networks. Proceedings (Cat. No.01CH37222), 2001, pp. 2254-2257, doi: 10.1109/IJCNN.2001.938771.

[39] Teresa Serrano- Gotarredona, Bernabe Linares – Barranco and Andreas G.Andreou, "Adaptive Resonance Theory Microchips," Kluwer Academic Publishers, pp. 737-746, 1999, doi: org/10.1007/BFb0098232.

[40] Jean-Charles Creput, Abderrafiaa Koukam, "A memetic neural network for the Euclidean traveling salesman problem," Neurocomputing 72 (4-6), 2009, pp. 1250–1264, doi: 10.1016/j.neucom.2008.01.023.

[41] Bihter Avşar, Danial Esmaeili Aliabadi, "Parallelized neural network system for solving Euclidean traveling salesman problem," Applied Soft Computing Volume 34, September 2015, pp. 862-873, doi: 10.1016/j.asoc.2015.06.011.

[42] Haruna Matsushita and Yoshifumi Nishio, "CHAOSOM: Collaboration between Chaos and Self-Organizing Map," 2006 RISP International Workshop on Nonlinear Circuits and Signal Processing (NCSP'06), pp. 305-308.

[43] A. L. Hodgkin and A. F. Huxley, "A Quantitative Description of Membrane Current and its Application to Conduction and Excitation in Nerve," Journal of Physiology, vol. 117, pp. 500-544, 1952, doi: 10.1113/jphysiol.1952.sp004764.

[44] Remo Ryter, Michael Stauffer, Thomas Hanne, Rolf Dornberger, "Analysis of Chaotic Maps Applied to Self-Organizing Maps for the Traveling Salesman Problem," 2015 IEEE Congr. Evol. Comput. CEC 2015 - Proc., pp. 1717–1724, 2015, doi: 10.1109/CEC.2015.7257094.

[45] M. Henon, "A two-dimensional mapping with a strange attractor," Communications in Mathematical Physics, vol. 50, pp. 69–77, feb 1976, doi: 10.1007/BF01608556.

[46] E. ELabbasy, H. Agiza, H. El-Metwally, and A. Elsadany, "Bifurcation analysis, chaos and control in the burgers mapping," International Journal of Nonlinear Science, vol. 4, no. 3, pp. 171–185, 2007.

[47] J. Burgers, "Mathematical examples illustrating relations occurring in the theory of turbulent fluid motion," in Selected Papers of J. M. Burgers, F. Nieuwstadt and J. Steketee, Eds. Springer Netherlands, 1995, pp. 281–334. http://dx.doi.org/10.1007/ 978-94-011-0195-0 10.

[48] A. Chaudhuri, K. De and D. Chatterjee, "A Study of the Traveling Salesman Problem Using Fuzzy Self Organizing Map," 2008 IEEE Region 10 and the Third international Conference on Industrial and Information Systems, 2008, pp. 1-5, doi: 10.1109/ICIINFS.2008.4798469.

[49] Applegate D., Bixby R., Chvatal V., and Cook W., On the solution of traveling salesman problems. Doc.Math.J.DMV Extra Volume ICM III, pp. 645-656, 1998. crpc-tr98744.

[50] Hiromi Takano, Yutaka Shirai, Naofumi Matsumoto, “PERFORMANCE EVALUATION OF HYBRID PROCEDURE OF SELF-ORGANIZING MAP AND SA FOR TSP,” International Journal of Innovative Computing, Information and Control Volume 7, Number 5(B), pp. 2931-2944, May 2011.
[51] J. Chen, J. Chen and X. Zhang, “Solving the Traveling Salesman Problem Using Elastic Net Integrate with SOM,” 2007 IEEE International Conference on Automation and Logistics, 2007, pp. 2234-2237, doi:10.1109/ICAL.2007.4338947.
[52] V. -I. Lupoaie, I. -A. Chili, M. E. Breaban and M. Raschip, “SOM-Guided Evolutionary Search for Solving MinMax Multiple-TSP,” 2019 IEEE Congress on Evolutionary Computation (CEC), 2019, pp. 73-80, doi: 10.1109/CEC.2019.8790276.
[53] A. Kiraly, J. Abonyi., “Optimization of multiple traveling salesmen problem by a novel representation based evolutionary algorithm.” In Intelligent Computational Optimization in Engineering, pp. 241-269, Springer, 2011, doi: 10.1007/978-3-642-21705-0_9.
[54] Anmin Zhu and S. X. Yang, “An improved self-organizing map approach to traveling salesman problem,” IEEE International Conference on Robotics, Intelligent Systems and Signal Processing, 2003. Proceedings. 2003, pp. 674-679, vol.1, doi: 10.1109/RISSP.2003.1285655.
[55] K. Kitaori, H. Murakoshi and N. Funakubo, “A new approach to solve the traveling salesman problem by using the improved Kohonen's self-organizing feature map,” Proceedings of IECON '95 - 21st Annual Conference on IEEE Industrial Electronics, 1995, pp. 1384-1388 vol.2, doi: 10.1109/IECON.1995.484152.
[56] BERNARD ANGENIOL, GAEL DE LA CROIX VAUBOIS AND JEAN-YVES LE TEXIER, “Self-Organizing Feature Maps and the Travelling Salesman Problem,” Neural Networks Volume 1, Issue 4, 1988, pp. 289-293, doi: 10.1016/0893-6080(88)90002-0.
[57] S. . -J. Kim, J. . -H. Kim and H. . -M. Choi, “An efficient algorithm for traveling salesman problems based on self-organizing feature maps,” [Proceedings 1993] Second IEEE International Conference on Fuzzy Systems, 1993, pp. 1085-1090 vol.2, doi: 10.1109/FUZZY.1993.327363.
[58] M. Budinich, “A Self-Organizing Neural Network for the Traveling Salesman Problem That Is Competitive with Simulated Annealing,” in Neural Computation, vol. 8, no. 2, pp. 416-424, 15 Feb. 1996, doi: 10.1162/neco.1996.8.2.416.
[59] Tairi, Mohamed, “A Cost Minimization Approach to the Traveling Salesman Problem with Drones Using a Self Organizing Map,” State University of New York at Binghamton ProQuest Dissertations Publishing, 2020. 27993460. [Online]. Available: https://www.proquest.com/openview/8b5256814e3fefbaaf2e6b85f90046ca/1?pq-origsite=gscholar&cbl=18750&diss=y.
[60] Jan Faigl and Libor Přeučil, “Self-organizing map for the multi-goal path planning with polygonal goals,” In Proceedings of the 21th international conference on Artificial neural networks - Volume Part I (ICANN'11), Springer-Verlag, Berlin, Heidelberg, 2011, pp.85–92, doi: 10.1007/978-3-642-21735-7_11.
[61] V. Lobo, “One dimensional Self-Organizing Maps to optimize marine patrol activities," Europe Oceans 2005, pp. 569-572, Vol. 1, doi: 10.1109/OCEANSE.2005.1511777.
[62] J. Faigl, “On self-organizing maps for orienteering problems,” 2017 International Joint Conference on Neural Networks (IJCNN), 2017, pp. 2611-2620, doi: 10.1109/IJCNN.2017.7966175
[63] J. Faigl, R. Pěnička and G. Best, “Self-organizing map-based solution for the Orienteering problem with neighborhoods,” 2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC), 2016, pp. 001315-001321, doi: 10.1109/SMC.2016.7844421.
[64] Faigl, J., Váňa, P, “Self-Organizing Map for the Curvature-Constrained Traveling Salesman Problem,” In: Villa, A., Masulli, P., Pons Rivero, A. (eds) Artificial Neural Networks and Machine Learning – ICANN 2016. ICANN 2016, Lecture Notes in Computer Science(), vol 9887. Springer, Cham, pp. 497-505, doi: 10.1007/978-3-319-44781-0_59.

[65] Hongjian Wang, Naiyu Zhang, and Jean-Charles Crput, “A massively parallel neural network approach to large-scale Euclidean traveling salesman problems.” Neurocomputing 240, C (May 2017), pp. 137–151, doi: 10.1016/j.neucom.2017.02.041.
[66] M. Steinhaus, A. N. Shirazi and M. Sodhi, "Modified Self Organizing Neural Network Algorithm for Solving the Vehicle Routing Problem," 2015 IEEE 18th International Conference on Computational Science and Engineering, Porto, Portugal, 2015, pp. 246-252, doi: 10.1109/CSE.2015.56.
[67] H. E. Ghaziri, “Solving routing problems by a self-organizing map,” in Artificial Neural Networks, 1991, International Conference on Artificial Neural Networks. ICANN, 1991, pp. 829–834.
[68] Y. Matsuyama, “Self-organization via competition, cooperation and categorization applied to extended vehicle routing problems,” in Neural Networks, 1991, IJCNN-91-Seattle International Joint Conference on, vol. 1. IEEE, 1991, pp. 385–390, doi: 10.1109/IJCNN.1991.155208.
[69] Ghaziri, H, 1996, “Supervision in the Self-Organizing Feature Map: Application to the Vehicle Routing Problem,” In: Osman, I.H., Kelly, J.P. (eds) Meta-Heuristics. Springer, Boston, MA., doi: 10.1007/978-1-4613-1361-8_39.
[70] A. I. Vakhutinsky and B. Golden, “Solving vehicle routing problems using elastic nets,” in Neural Networks, 1994. IEEE World Congress on Computational Intelligence, 1994 IEEE International Conference on, vol. 7. IEEE, 1994, pp. 4535–4540, doi: 10.1109/ICNN.1994.375004.
[71] M Tairi, Y Wang, and SW Yoon. A Dynamic Approach to the Traveling Salesman Problem with Drones Using a Self- Organizing Map Proceedings of the 9th Annual World Conference of the Society for Industrial and Systems Engineering, 2020 SISE Virtual Conference September 17-18, 2020. ISBN: 97819384961-9-6.
[72] A. Jaradat, B. Matalkeh and W. Diabat, "Solving Traveling Salesman Problem using Firefly algorithm and K-means Clustering," 2019 IEEE Jordan International Joint Conference on Electrical Engineering and Information Technology (JEEIT), Amman, Jordan, 2019, pp. 586-589, doi: 10.1109/JEEIT.2019.8717463.
[73] Eklavya. 2019. [Online]. Available: https://medium.com/towards-data-science/kohonen-self-organizing-maps-a29040d688da.
[74] Kohonen, T., “Self-organized formation of topologically correct feature maps,” Biol. Cybern. 43, pp. 59–69, 1982, doi: 10.1007/BF00337288.

ΠΑΡΑΡΤΗΜΑ Α: ΠΗΓΕΣ ΣΧΗΜΑΤΩΝ

Τα σχήματα έχουν παρθεί από τις παραπάνω πηγές.

- Σχήμα 2.1: [37]
- Σχήμα 2.2: [32]
- Σχήμα 3.1: [4]
- Σχήμα 3.2: [6]
- Σχήμα 3.3: [8]
- Σχήμα 3.4: [21]
- Σχήμα 3.5: [24]
- Σχήμα 3.6: [25]
- Σχήμα 3.7: [26]
- Σχήμα 3.8: [31]
- Σχήμα 3.9: [32]
- Σχήμα 3.10: [32]
- Σχήμα 3.11: [32]
- Σχήμα 3.12: [33]
- Σχήμα 3.13: [34]
- Σχήμα 3.14: [35]
- Σχήμα 3.15: [37]
- Σχήμα 3.16: [37]
- Σχήμα 3.17: [40]
- Σχήμα 3.18: [40]
- Σχήμα 3.19: [41]
- Σχήμα 3.20: [42]
- Σχήμα 3.21: [50]
- Σχήμα 3.22: [52]
- Σχήμα 4.1: [71]