ΕΛΛΗΝΙΚΟ ΑΝΟΙΧΤΟ ΠΑΝΕΠΙΣΤΗΜΙΟ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ ΚΑΙ ΤΕΧΝΟΛΟΓΙΑΣ

ΜΕΤΑΠΤΥΧΙΑΚΗ ΕΞΕΙΔΙΚΕΥΣΗ ΣΤΑ
ΠΛΗΡΟΦΟΡΙΑΚΑ ΣΥΣΤΗΜΑΤΑ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΑΝΑΚΑΛΥΨΗ ΠΛΗΡΟΦΟΡΙΑΣ ΠΛΑΙΣΙΟΥ: ΒΕΛΤΙΣΤΟΠΟΙΗΣΗ ΣΜΗΝΟΥΣ ΣΩΜΑΤΙΔΙΩΝ ΚΑΙ ΘΕΩΡΙΑ ΒΕΛΤΙΣΤΗΣ ΠΑΥΣΗΣ

ΝΙΚΟΣ ΚΟΥΤΣΟΥΛΗΣ
Α.Μ. 58411

ΕΠΙΒΛΕΠΩΝ: ΕΥΣΤΑΘΙΟΣ ΧΑΤΖΗΕΥΘΥΜΙΑΔΗΣ

ΠΑΤΡΑ
ΜΑΙΟΣ 2012

Διπλωματική Εργασία

Ανακάλυψη Πληροφορίας Πλαισίου: Βελτιστοποίηση Σμήνους Σωματιδίων και Θεωρία Βέλτιστης Παύσης

Κουτσούλης Νίκος

Ημερομηνία
31/03/2012

Ανακάλυψη Πληροφορίας Πλαισίου: Βελτιστοποίηση Σμήνους Σωματιδίων και Θεωρία Βέλτιστης Παύσης

Κουτσούλης Νίκος

Ονοματεπώνυμο Επιβλέποντα	**Ονοματεπώνυμο Μέλους 1**	**Ονοματεπώνυμο Μέλους 2**
Χατζηευθυμιάδης Ευστάθειος	Σκάρλας Λάμπρος	Καψάλης Βασίλειος

Περίληψη

Ο βασικός στόχος της εργασίας αυτής είναι η αξιοποίηση της Βελτιστοποίησης Σμήνους Σωματιδίων (Particle Swarm Optimization – PSO) και της Θεωρίας Βέλτιστης Παύσης (Optimal Stopping Theory – OST) για την αντιμετώπιση του Προβλήματος Ανακάλυψης Πληροφορίας Πλαισίου (Context Discovery Problem – CDP).

Ο όρος «Ανακάλυψη Πληροφορίας Πλαισίου» αναφέρεται στο μηχανισμό που υιοθετούν κινητοί κόμβοι σε μια καθορισμένη περιοχή ώστε να αναζητούν πηγές πληροφορίας.

Ο αλγόριθμος Βελτιστοποίησης Σμήνους Σωματιδίων είναι ένας πληθυσμιακός αλγόριθμος αναζήτησης που βασίζεται στην προσομοίωση της κοινωνικής συμπεριφοράς των πουλιών μέσα σε ένα σμήνος.

Η Θεωρία Βέλτιστης Παύσης μελετάει το πρόβλημα της επιλογής της χρονικής στιγμής εκτέλεσης μιας συγκεκριμένης ενέργειας, βασισμένη σε μια ακολουθία παρατηρούμενων τυχαίων μεταβλητών, προκειμένου να μεγιστοποιηθεί κάποια αναμενόμενη ανταμοιβή ή να ελαχιστοποιηθεί κάποιο αναμενόμενο κόστος.

Στην εργασία αυτή μελετάται η Ανακάλυψη Πληροφορίας Πλαισίου σε ένα σύστημα κινητών (ρομποτικών) κόμβων αισθητήρων, οι οποίοι καθορίζουν δυναμικά την κίνησή τους στο χώρο με στόχο την απόκτηση καλύτερης ποιότητας πληροφορίας πλαισίου. Το σύστημα αυτό προσομοιώνεται από ένα σμήνος σωματιδίων και το Πρόβλημα Ανακάλυψης Πληροφορίας Πλαισίου μετασχηματίζεται σε ένα πρόβλημα Βελτιστοποίησης Σμήνους Σωματιδίων. Ο εντοπισμός της πληροφορίας πλαισίου επεκτείνεται με τον κατάλληλο χρονοπρογραμματισμό των μετακινήσεων των κόμβων. Ενδεχόμενη καθυστέρηση στην αλλαγή θέσης ενός κόμβου μπορεί να συντελέσει στη σύλληψη πληροφορίας πλαισίου υψηλότερης ποιότητας. Ο χρονοπρογραμματισμός αυτός βασίζεται στη Θεωρία Βέλτιστης Παύσης και ειδικότερα σε μια παραλλαγή του Κλασικού Προβλήματος της Γραμματέως.

Στα πλαίσια αυτά, υλοποιήθηκαν δύο βασικοί αλγόριθμοι: ο αλγόριθμος Βελτιστοποίησης Σμήνους Σωματιδίων (PSO) και ο αλγόριθμος Βελτιστοποίησης Σμήνους Σωματιδίων με εφαρμογή της Θεωρίας Βέλτιστης Παύσης (OSPSO). Κατά την πειραματική μελέτη του συστήματος εξετάστηκαν σημαντικά χαρακτηριστικά του, όπως είναι η μέση ποιότητα πληροφορίας πλαισίου και το μέσο ενεργειακό κόστος των σωματιδίων του σμήνους, σε συνάρτηση με τις βασικές παραμέτρους του (π.χ. αριθμός αισθητήρων κόμβων, συχνότητα ανίχνευσης πληροφορίας, διάστημα παρατήρησης θεωρίας βέλτιστης παύσης, κ.ά.). Έγινε σύγκριση της απόδοσης των δύο αλγόριθμων και εξήχθησαν χρήσιμα συμπεράσματα για την αποτελεσματικότητά τους.

Λέξεις-Κλειδιά: Ανακάλυψη Πληροφορίας Πλαισίου, Ασύρματα Δίκτυα Αισθητήρων, Αλγόριθμος Βελτιστοποίησης Σμήνους Σωματιδίων, Θεωρία Βέλτιστης Παύσης.

Context Discovery: Particle Swarm Optimization and Optimal Stopping Theory

Koutsoulis Nikos

Supervisor	1st Member	2nd Member
Name	Name	Name
Hadjiefthymiades Stathes	Skarlas Labros	Capsalis Vasilios

Abstract

The basic purpose of this research is the combined use of Particle Swarm Optimization (PSO) and Optimal Stopping Theory (OST) in order to confront the Context Discovery Problem (CDP).

Context Discovery refers to the mechanism adopted by mobile nodes moving in a specified area in order to seek information sources.

The Particle Swarm Optimization algorithm is a population-based search algorithm based on the simulation of the social behavior of birds within a flock.

The Optimal Stopping Theory is concerned with the problem of choosing a time to take a given action based on sequentially observed random variables in order to maximize an expected payoff or to minimize an expected cost.

This research deals with the Context Discovery Problem in a system of mobile (robotic) sensor nodes that dynamically adjust their motion in order to obtain context of better quality. This system is simulated by a swarm of particles and the CDP maps to a OST problem. Context tracking is facilitated by suitable time-scheduling of particle movements. A potential delay to the movement time of a particle may result to the discovery of better quality context. This type of time-scheduling is based on Optimal Stopping Theory and specifically on a variation of the Classical Secretary Problem.

In this framework, two main algorithms were developed: the Particle Swarm Optimization (PSO) algorithm and the Particle Swarm Optimization with use of Optimal Stopping Theory (OSPSO) algorithm. Experimental study of this system was carried out and important aspects, such as mean context quality and mean energy cost, were measured to basic system parameters (e.g. number of sensor nodes, sensing rate, OST observation interval etc.). The two algorithms were compared and useful conclusions were made.

Key-Words: Context Discovery, Mobile Sensor Nodes, Particle Swarm Optimization Algorithm, Optimal Stopping Theory.

στη μητέρα μου που έφυγε πρόσφατα

Ευχαριστίες

Θα ήθελα να εκφράσω τις ευχαριστίες μου στον επιβλέποντα της εργασίας Επίκουρο Καθηγητή κ. Ευστάθιο Χατζηευθυμιάδη καθώς και στον Επίκουρο Καθηγητή κ. Χρήστο Αναγνωστόπουλο για τη σημαντική καθοδήγηση και την καθοριστική βοήθειά τους.

Επίσης, θα ήθελα να ευχαριστήσω την Ευγενία για την υποστήριξη και την υπομονή της.

Περιεχόμενα

Κεφάλαιο 1: Βελτιστοποίηση Σμήνους Σωματιδίων (Particle Swarm Optimization – PSO)....17

1.1. Νοημοσύνη Σμήνους....17

1.2. Εισαγωγή στη Βελτιστοποίηση Σμήνους Σωματιδίων....17

1.3. Βασικός Αλγόριθμος Βελτιστοποίησης Σμήνους Σωματιδίων (Basic PSO)....18

1.3.1. Global Best PSO....19

1.3.2. Local Best PSO....21

1.3.3. Σύγκριση αλγόριθμων *gbest PSO* και *lbest PSO*....23

1.4. Δομές Κοινωνικής Δικτύωσης....23

1.5. Παραλλαγές του Αλγόριθμου Βελτιστοποίησης Σμήνους Σωματιδίων....26

1.5.1. Αποκοπή ταχύτητας (velocity clamping)....26

1.5.2. Βάρος Αδράνειας (Inertia Weight)....27

1.5.3. Συντελεστής Περιορισμού (Constriction Coefficient)....32

1.5.4. Μοντέλα ταχύτητας....33

1.6. Στοιχεία του Αλγόριθμου Βελτιστοποίησης Σμήνους Σωματιδίων....34

1.6.1. Μέγεθος παραμέτρων....34

1.6.2. Αρχικοποίηση αλγόριθμου....36

1.6.3. Τερματισμός αλγόριθμου....37

1.7. Εφαρμογή του αλγόριθμου PSO σε δυναμικά περιβάλλοντα....40

1.7.1. Προβλήματα εφαρμογής....40

1.7.2. Επίδραση παραμέτρων....41

1.7.3. Μέθοδοι αντιμετώπισης....42

Κεφάλαιο 2: Θεωρία Βέλτιστης Παύσης (Optimal Stopping Theory – OST)....45

2.1. Εισαγωγή....45

2.2. Ορισμός του προβλήματος....45

2.3. Προβλήματα πεπερασμένου ορίζοντα....48

2.3.1. Το Πρόβλημα της Γραμματέως (Secretary Problem) 49
2.3.2. Παραλλαγές του Προβλήματος της Γραμματέως 52
2.3.3. Μια νέα προσέγγιση του Προβλήματος της Γραμματέως 53
2.3.4. Το Πρόβλημα του Παρκαρίσματος (Parking Problem) 55
Κεφάλαιο 3: Πληροφορία Πλαισίου 57
3.1. Πληροφορία Πλαισίου 57
3.1.1. Ορισμός και περιγραφή 57
3.1.2. Κατηγορίες πλαισίου 59
3.2. Επίγνωση Πληροφορίας Πλαισίου 59
3.3. Θέματα Επίγνωσης Πληροφορίας Πλαισίου 62
3.4. Αρχιτεκτονική Συστήματος Επίγνωσης Πληροφορίας Πλαισίου 65
3.5. Μοντελοποίηση Πληροφορίας Πλαισίου 66
3.6. Συνεργατική Επίγνωση Πληροφορίας Πλαισίου 67
3.6.1. Ορισμός 67
3.6.2. Κινητικότητα στη ΣΕΠΠ 68
3.7. Ανακάλυψη Πληροφορίας Πλαισίου (Context Discovery Problem – CDP) 69
Κεφάλαιο 4: Ανακάλυψη Πληροφορίας Πλαισίου σε Κινητά Περιβάλλοντα με χρήση PSO και OST… 73
4.1. Εισαγωγή 73
4.2. Αναπαράσταση Πληροφορίας Πλαισίου – Ποιότητα Πληροφορίας Πλαισίου 75
4.3. Ανακάλυψη Πληροφορίας Πλαισίου με χρήση Βελτιστοποίησης Σμήνους Σωματιδίων 76
4.4. Εφαρμογή της Θεωρίας Βέλτιστης Παύσης 79
Κεφάλαιο 5: Αλγόριθμοι υλοποίησης 81
5.1. Εισαγωγή 81
5.2. Βασικός αλγόριθμος Βελτιστοποίησης Σμήνους Σωματιδίων (PSO) 81
5.2.1. Παράμετροι 81
5.2.2. Κίνηση σωματιδίων και πηγών 84

5.2.3. Βάρος αδράνειας....86
5.2.4. Αποκοπή ταχύτητας και αποκοπή θέσης....87
5.3. Μέθοδοι μεταβολής του βάρους αδράνειας....87
5.4. Παραλλαγή του βασικού αλγόριθμου PSO....94
5.5. Αλγόριθμοι υλοποίησης της Θεωρίας Βέλτιστης Παύσης....94
Κεφάλαιο 6: Πειραματική Μελέτη....99
6.1. Εισαγωγή....99
6.2. Αλγόριθμος Βελτιστοποίησης Σμήνους Σωματιδίων (PSO)....103
6.2.1. Μέθοδοι μεταβολής βάρους αδράνειας....103
6.2.2. Επίδραση του αριθμού των πηγών....106
6.2.3. Επίδραση της συχνότητας ανανέωσης πληροφορίας πλαισίου....109
6.2.4. Συμπεράσματα – Παρατηρήσεις....112
6.3. Αλγόριθμος Βελτιστοποίησης Σμήνους Σωματιδίων με χρήση Θεωρίας Βέλτιστης Παύσης (OSPSO)....114
6.3.1. Μέθοδοι μεταβολής βάρους αδράνειας – διάστημα παρατήρησης OST....114
6.3.2. Επίδραση του αριθμού των πηγών....117
6.3.3. Συμπεράσματα – Παρατηρήσεις....123
Κεφάλαιο 7: Γενικά Συμπεράσματα – Προοπτικές....125
7.1. Συμπεράσματα....125
7.2. Προοπτικές....127
Αναφορές....129
Παράρτημα....135

Κατάλογος Σχημάτων

Σχήμα 1: Μεταβολές θέσης και ταχύτητας ενός σωματιδίου για την 2-D περίπτωση.....20
Σχήμα 2: Δομές κοινωνικής δικτύωσης.....25
Σχήμα 3: Εννοιολογικός χώρος Επίγνωσης Πλαισίου 64
Σχήμα 4: Αρχιτεκτονική Συστήματος Επίγνωσης Πλαισίου.....65
Σχήμα 5: Δίκτυο κινητών κόμβων 69
Σχήμα 6: PSO. Στιγμιότυπο της κίνησης σωματιδίων και πηγών στο χώρο 85
Σχήμα 7: Μέθοδος μεταβολής βάρους αδράνειας “non-linear decreasing” 89
Σχήμα 8: Μέθοδος μεταβολής βάρους αδράνειας “non-linear decreasing 2” 89
Σχήμα 9: Μέθοδος μεταβολής βάρους αδράνειας “chaotic” 91
Σχήμα 10: Μέθοδος μεταβολής βάρους αδράνειας “chaotic random” 91
Σχήμα 11: Αλγόριθμος PSO. $\overline{g(t)} = f(t)$ 101
Σχήμα 12: Αλγόριθμος PSO. $\overline{d_t(t)} = f(t)$ 102
Σχήμα 13: Αλγόριθμος PSO. $\overline{d(t)} = f(t)$ 102
Σχήμα 14: PSO. Συγκλίνουσα μέση ποιότητα πληροφορίας πλαισίου $\overline{g(t)}$, $N = 100$ 106
Σχήμα 15: PSO. Συγκλίνουσα μέση διανυόμενη απόσταση $\overline{d_t}$, $N = 100$ 107
Σχήμα 16: PSO. Συγκλίνουσα μέση ποιότητα πληροφορίας πλαισίου $\overline{g(t)}$ για $N = 500$ 108
Σχήμα 17: PSO. Συγκλίνουσα μέση διανυόμενη απόσταση $\overline{d_t}$ για $N = 500$ 108
Σχήμα 18: PSO. Συγκλίνουσα μέση απόσταση $\overline{d}$ για $N = 500$ 109
Σχήμα 19: PSO. Συγκλίνουσα μέση ποιότητα πληροφορίας πλαισίου $\overline{g(t)}$ σε συνάρτηση με τη συχνότητα ανανέωσης q 111
Σχήμα 20: PSO. Συγκλίνουσα μέση διανυόμενη απόσταση $\overline{d_t}$ σε συνάρτηση με τη συχνότητα q 111
Σχήμα 21: OSPSO. Μεταβολή του $\overline{g(t)}$ για τις διάφορες μεθόδους μεταβολής του w 117
Σχήμα 22: OSPSO. Μέθοδος non-linear. $\overline{g(t)} = f(n_0)$ για διάφορες τιμές του M 121
Σχήμα 23: OSPSO. Μέθοδος non-linear. $\overline{d_t} = f(n_0)$ για διάφορες τιμές του M 121
Σχήμα 24: OSPSO. Μέθοδος chaotic. $\overline{g(t)} = f(n_0)$ για διάφορες τιμές του M 122
Σχήμα 25: OSPSO. Μέθοδος chaotic. $\overline{d_t} = f(n_0)$ για διάφορες τιμές του M 122

Κατάλογος Πινάκων

Πίνακας 1: Πρόβλημα Γραμματέως. Ενδεικτικές τιμές παραμέτρων. 51
Πίνακας 2: Νέα προσέγγιση Προβλήματος Γραμματέως. Ενδεικτικές τιμές παραμέτρων. 54
Πίνακας 3: Αντιστοιχία PSO - CDP 79
Πίνακας 4: PSO, standard *pBest* calculation, $N_r = 100$ 104
Πίνακας 5: PSO, standard *pBest* calculation, $N_r = 1000$ 104
Πίνακας 6: PSO, memoryless *pBest* calculation, $N_r = 100$ 105
Πίνακας 7: PSO, memoryless *pBest* calculation, $N_r = 1000$ 105
Πίνακας 8: PSO, μοντέλο *pBest* κλασικό, μέθοδος non-linear, $q = 0,02$ Hz 106
Πίνακας 9: PSO, μοντέλο *pBest* κλασικό, μέθοδος non-linear, $N = 500$, $q = 0,02$ Hz 107
Πίνακας 10: Επίδραση της συχνότητας ανανέωσης q 110
Πίνακας 11: OSPSO, n_0, $q = 0,02$ Hz 114
Πίνακας 12: OSPSO, $n_0, \theta_{threshold}, q = 0,02$ Hz 115
Πίνακας 13: OSPSO. Μέθοδος non-linear, $q = 0,02$ Hz 118
Πίνακας 14: OSPSO. Μέθοδος chaotic, $q = 0,02$ Hz 119

Κεφάλαιο 1: Βελτιστοποίηση Σμήνους Σωματιδίων (Particle Swarm Optimization – PSO)

1.1. Νοημοσύνη Σμήνους

Η Νοημοσύνη Σμήνους (Swarm Intelligence), η οποία είναι ένας κλάδος Τεχνητής Ευφυΐας (Artificial Intelligence), ασχολείται με τη σχεδίαση ευφυών συστημάτων πολλαπλών πρακτόρων (multi-agent systems), εμπνεόμενη τόσο από τη συλλογική συμπεριφορά κοινωνικών εντόμων, όπως είναι τα μυρμήγκια, οι τερμίτες, οι μέλισσες και οι σφήκες, όσο και από άλλες κοινωνίες του ζωικού βασιλείου, όπως είναι τα σμήνη πουλιών ή τα κοπάδια ψαριών. Οι αποικίες των κοινωνικών εντόμων αποτελούν αντικείμενο έρευνας εδώ και πολλά χρόνια και οι μηχανισμοί που διέπουν τη συμπεριφορά τους παρέμεναν για πολύ καιρό αδιευκρίνιστοι. Παρά το γεγονός ότι τα μέλη μιας τέτοιας αποικίας έχουν περιορισμένες δυνατότητες ως άτομα, μπορούν να επιτυγχάνουν σύνθετες εργασίες ως σύνολο. Η συντεταγμένη συμπεριφορά μιας αποικίας προκύπτει από τις σχετικά απλές ενέργειες ή αλληλεπιδράσεις μεταξύ των μελών της. Πολλές πλευρές από τις συλλογικές δραστηριότητες των κοινωνικών εντόμων είναι αυτο-οργανούμενες και χωρίς την ύπαρξη κάποιου κεντρικού ελέγχου [1], [2].

Ο όρος «Νοημοσύνη Σμήνους» χρησιμοποιήθηκε για πρώτη φορά στο πεδίο των κυτταρικών ρομποτικών συστημάτων (cellular robotic systems), όπου απλοί πράκτορες αυτο-οργανώνονται μέσω της αλληλεπίδρασης πλησιέστερου γείτονα [3]. Ακόλουθα, ο όρος χρησιμοποιείται σε ένα πολύ ευρύτερο ερευνητικό πλαίσιο. Οι μέθοδοι νοημοσύνης σμήνους έχουν αποδειχτεί ιδιαίτερα επιτυχημένες στον τομέα της βελτιστοποίησης. Παραδείγματα τέτοιων εφαρμογών αποτελούν προβλήματα χρονοπρογραμματισμού, σχεδίασης τηλεπικοινωνιακών δικτύων, εύρεσης βέλτιστης διαδρομής, υπολογιστικής βιολογίας κ.λπ..

1.2. Εισαγωγή στη Βελτιστοποίηση Σμήνους Σωματιδίων

Ο αλγόριθμος Βελτιστοποίησης Σμήνους Σωματιδίων (PSO) είναι ένας πληθυσμιακός αλγόριθμος αναζήτησης, που βασίζεται στην προσομοίωση της κοινωνικής συμπεριφοράς των πουλιών μέσα σε ένα σμήνος. Ο αρχικός στόχος της ιδέας του πλήθους σωματιδίων ήταν η γραφική προσομοίωση της χαριτωμένης και απρόβλεπτης χορογραφίας ενός σμήνους

πουλιών [4]. Η ιδέα αυτή εξελίχθηκε σε έναν απλό και αποτελεσματικό αλγόριθμο βελτιστοποίησης [2], [5].

Στον αλγόριθμο Βελτιστοποίησης Σμήνους Σωματιδίων, τα άτομα, τα οποία αναφέρονται ως σωματίδια, κινούνται μέσα σε ένα χώρο αναζήτησης πολλών διαστάσεων. Οι μεταβολές των θέσεων των σωματιδίων μέσα στο χώρο αναζήτησης βασίζονται στην κοινωνικο-ψυχολογική τάση των ατόμων να μιμούνται την επιτυχία των άλλων ατόμων. Έτσι, η κίνηση ενός σωματιδίου μέσα στο σμήνος επηρεάζεται από την εμπειρία ή τη γνώση των γειτονικών του σωματιδίων.

Από την εισαγωγή του το 1995 [4], ο αλγόριθμος Βελτιστοποίησης Σμήνους Σωματιδίων έχει υποστεί πολλές βελτιώσεις και έχει βρει διάφορα πεδία εφαρμογής. Στη συνέχεια παραθέτουμε το βασικό αλγόριθμο PSO και μερικές από τις κυριότερες παραλλαγές του.

1.3. Βασικός Αλγόριθμος Βελτιστοποίησης Σμήνους Σωματιδίων (Basic PSO)

Ένας αλγόριθμος PSO διατηρεί ένα σμήνος από σωματίδια καθένα από τα οποία αντιπροσωπεύει μια πιθανή λύση [2], [5].

Έστω $\boldsymbol{x}_i(t)$ η θέση ενός σωματιδίου i στο χώρο αναζήτησης τη διακριτή χρονική στιγμή t. Η θέση του σωματιδίου μεταβάλλεται με την πρόσθεση μιας ταχύτητας $\boldsymbol{v}_i(t)$ στην τρέχουσα θέση:

$$\boldsymbol{x}_i(t+1) = \boldsymbol{x}_i(t) + \boldsymbol{v}_i(t+1) \tag{1.1}$$

με $\boldsymbol{x}_i(0) \sim U\left(\boldsymbol{x}_{\min}, \boldsymbol{x}_{\max}\right)$.

Το διάνυσμα της ταχύτητας οδηγεί τη διαδικασία βελτιστοποίησης και αντικατοπτρίζει τόσο την εμπειρική γνώση του σωματιδίου όσο και την κοινωνικά ανταλλασσόμενη πληροφορία από τη γειτονιά του σωματιδίου. Η εμπειρική γνώση του σωματιδίου αναφέρεται ως η γνωσιακή συνιστώσα (cognitive component), η οποία είναι ανάλογη ως προς την απόσταση του σωματιδίου από τη δική του βέλτιστη θέση (personal best position). Η κοινωνικά ανταλλασσόμενη πληροφορία αναφέρεται ως η κοινωνική συνιστώσα (social component) της εξίσωσης ταχύτητας.

Αρχικά είχαν αναπτυχθεί δύο αλγόριθμοι PSO, οι οποίοι διαφέρουν ως προς το μέγεθος της γειτονιάς τους: ο *global best PSO* (*gbest PSO*) και ο *local best PSO* (*lbest PSO*).

1.3.1. Global Best PSO

Για τον αλγόριθμο Βελτιστοποίησης Σμήνους Σωματιδίων καθολικού βέλτιστου – *global best PSO*, ή *gbest PSO*, η γειτονιά κάθε σωματιδίου είναι ολόκληρο το σμήνος. Το κοινωνικό δίκτυο που υλοποιείται από τον *global best PSO* απεικονίζεται από την τοπολογία αστέρα (βλέπε εδάφιο 1.4). Στην περίπτωση αυτή, η κοινωνική συνιστώσα της εξίσωσης μεταβολής της ταχύτητας του σωματιδίου αντικατοπτρίζει την πληροφορία από όλα τα σωματίδια του σμήνους. Η πληροφορία αυτή αντιστοιχεί στη βέλτιστη θέση του σμήνους $\hat{\boldsymbol{y}}(t)$.

Για τον αλγόριθμο *gbest PSO*, η ταχύτητα ενός σωματιδίου i δίνεται από τη σχέση:

$$v_{ij}(t+1) = v_{ij}(t) + c_1 r_{1j}(t)\left[y_{ij}(t) - x_{ij}(t)\right] + c_2 r_{2j}(t)\left[\hat{y}_j(t) - x_{ij}(t)\right] \tag{1.2}$$

όπου $v_{ij}(t)$ είναι η ταχύτητα του σωματιδίου i στη διάσταση $j = 1,\ldots,n_x$ τη χρονική στιγμή t, $x_{ij}(t)$ είναι η θέση του σωματιδίου i στη διάσταση j τη χρονική στιγμή t, c_1 και c_2 θετικές σταθερές επιτάχυνσης, οι οποίες χρησιμοποιούνται για να ρυθμίσουν τις συνεισφορές της γνωσιακής και της κοινωνικής συνιστώσας αντίστοιχα, και $r_{1j}(t),\ r_{2j}(t) \sim U(0,1)$ τυχαίες τιμές στο διάστημα $[0,1]$, δειγματοληπτημένες από μια ομοιόμορφη κατανομή. Αυτές οι τυχαίες τιμές εισάγουν στοχαστικότητα στον αλγόριθμο.

Οι μεταβολές θέσης και ταχύτητας της γνωσιακής και της κοινωνικής συνιστώσας ενός σωματιδίου του σμήνους για δύο διαδοχικές χρονικές στιγμές απεικονίζονται στο Σχήμα 1.

Η ατομική βέλτιστη θέση (personal best position – *pbest*) ενός σωματιδίου i $\boldsymbol{y}_i$ είναι η καλύτερη θέση που έχει επισκεφθεί το σωματίδιο από την πρώτη χρονική στιγμή. Αν θεωρήσουμε πρόβλημα ελαχιστοποίησης, η θέση *pbest* την επόμενη χρονική στιγμή $t+1$ δίνεται από τη σχέση:

$$\boldsymbol{y}_i(t+1) = \begin{cases} \boldsymbol{y}_i(t), & f\left(\boldsymbol{x}_i(t+1)\right) \geq f\left(\boldsymbol{y}_i(t)\right) \\ \boldsymbol{x}_i(t+1), & f\left(\boldsymbol{x}_i(t+1)\right) < f\left(\boldsymbol{y}_i(t)\right) \end{cases} \tag{1.3}$$

όπου $f : \mathbb{R}^{n_x} \to \mathbb{R}$ είναι η συνάρτηση βελτιστοποίησης (fitness function). Η συνάρτηση βελτιστοποίησης μετρά πόσο κοντά στη βέλτιστη τιμή βρίσκεται μια λύση.

Η καθολική βέλτιστη θέση (global best position – *gbest*) $\hat{\boldsymbol{y}}(t)$ τη χρονική στιγμή t ορίζεται ως εξής:

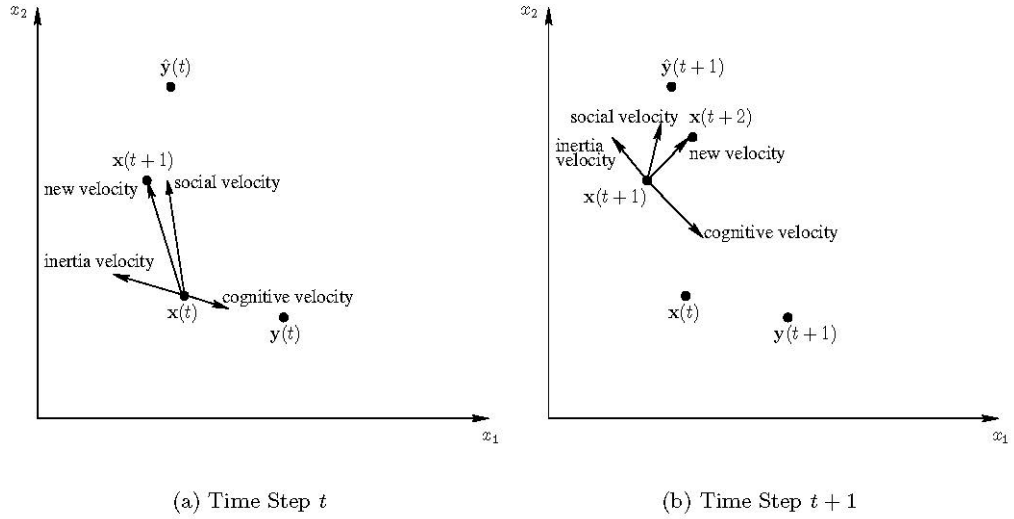
$$\hat{\boldsymbol{y}}(t) \in \left\{ \boldsymbol{y}_1(t), \ldots, \boldsymbol{y}_{n_s}(t) \mid f\left(\hat{\boldsymbol{y}}(t)\right) \right\} = \arg\min \left\{ f\left(\boldsymbol{y}_1(t)\right), \ldots, f\left(\boldsymbol{y}_{n_s}(t)\right) \right\} \tag{1.4}$$

όπου n_s ο συνολικός αριθμός σωματιδίων στο σμήνος. Η τιμή $\hat{\boldsymbol{y}}$ είναι η βέλτιστη θέση που έχει βρεθεί από οποιαδήποτε σωματίδιο μέχρι τώρα.

Η καθολική βέλτιστη θέση μπορεί να υπολογιστεί και από τα σωματίδια του τρέχοντος σμήνους:

$$\hat{\boldsymbol{y}}(t) = \arg\min \left\{ f\left(\boldsymbol{x}_1(t)\right), \ldots, f\left(\boldsymbol{x}_{n_s}(t)\right) \right\} \tag{1.5}$$

Ο αλγόριθμος *gbest PSO* συνοψίζεται στον Αλγόριθμο 1.



(a) Time Step t (b) Time Step $t+1$

Σχήμα 1: Μεταβολές θέσης και ταχύτητας ενός σωματιδίου για την 2-D περίπτωση

Αλγόριθμος 1: ***gbest PSO***

Δημιουργία και αρχικοποίηση σμήνους σωματιδίων n_x διαστάσεων
repeat
  for σωματίδιο $i = 1,\ldots,n_s$ **do**
    // υπολογισμός ατομικής βέλτιστης θέσης
    if $f\left(\boldsymbol{x}_i\right) < f\left(\boldsymbol{y}_i\right)$ **then**
      $\boldsymbol{y}_i = \boldsymbol{x}_i$
    end
    // υπολογισμός καθολικής βέλτιστης θέσης
    if $f\left(\boldsymbol{y}_i\right) < f\left(\hat{\boldsymbol{y}}\right)$ **then**
      $\hat{\boldsymbol{y}} = \boldsymbol{y}_i$
    end
  end // for
  for σωματίδιο $i = 1,\ldots,n_s$ **do**
    ενημέρωση της ταχύτητας με χρήση της εξίσωσης (1.2)
    ενημέρωση της θέσης με χρήση της εξίσωσης (1.1)
  end // for
until (συνθήκη τερματισμού αληθής)

1.3.2. Local Best PSO

Ο αλγόριθμος Βελτιστοποίησης Σμήνους Σωματιδίων τοπικού βέλτιστου – *local best PSO*, ή *lbest PSO*, χρησιμοποιεί ένα κοινωνικό δίκτυο με τοπολογία δακτυλίου (βλέπε εδάφιο 1.4). Η κοινωνική συνιστώσα της εξίσωσης μεταβολής της ταχύτητας απεικονίζει την πληροφορία που ανταλλάσσεται μέσα στη γειτονιά του σωματιδίου, η οποία αντιστοιχεί σε τοπική γνώση του περιβάλλοντος.

Για τον αλγόριθμο *lbest PSO*, η ταχύτητα ενός σωματιδίου i δίνεται από τη σχέση:

$$v_{ij}(t+1) = v_{ij}(t) + c_1 r_{1j}(t)\left[y_{ij}(t) - x_{ij}(t)\right] + c_2 r_{2j}(t)\left[\hat{y}_{ij}(t) - x_{ij}(t)\right] \tag{1.6}$$

όπου $\hat{y}_{ij}(t)$ είναι η βέλτιστη θέση που βρίσκεται από τη γειτονιά ενός σωματιδίου i για τη διάσταση j. Η τοπική βέλτιστη θέση (local best position – *lbest*) $\hat{\boldsymbol{y}}_i(t)$, δηλαδή η βέλτιστη θέση που βρίσκεται στη γειτονιά N_i τη χρονική στιγμή t, ορίζεται ως εξής:

$$\hat{\boldsymbol{y}}_i(t) \in \left\{N_i \mid f\left(\hat{\boldsymbol{y}}_i(t)\right)\right\} = \arg\min\left\{f\left(\boldsymbol{x}(t)\right)\right\}, \quad \forall \boldsymbol{x}(t) \in N_i \tag{1.7}$$

όπου η γειτονιά ορίζεται ως

$$N_i = \left\{ \boldsymbol{y}_{i-n_{N_i}}(t), \boldsymbol{y}_{i-n_{N_i}+1}(t), \ldots, \boldsymbol{y}_{i-1}(t), \boldsymbol{y}_i(t), \boldsymbol{y}_{i+1}(t), \ldots, \boldsymbol{y}_{i+n_{N_i}}(t) \right\} \quad (1.8)$$

για γειτονιά μεγέθους n_{N_i}.

Η επιλογή της γειτονιάς βασίζεται συνήθως στους δείκτες των σωματιδίων του σμήνους. Εναλλακτικά, έχουν αναπτυχθεί μέθοδοι που βασίζονται στη χωρική ομοιότητα μεταξύ των σωματιδίων, π.χ. με χρήση της Ευκλείδειας απόστασης μεταξύ αυτών [6].

Επίσης, οι γειτονιές μπορεί και να αλληλοεπικαλύπτονται. Ένα σωματίδιο μπορεί να ανήκει σε περισσότερες από μία γειτονιές. Αυτή η αλληλοσύνδεση χρησιμεύει στο μοίρασμα της πληροφορίας μεταξύ των γειτονιών και διασφαλίζει ότι το σμήνος θα συγκλίνει σε ένα μοναδικό σημείο, το καθολικά βέλτιστο σωματίδιο. Ο *gbest PSO* είναι μια ειδική περίπτωση του *lbest PSO* για $n_{N_i} = n_s$. Ο αλγόριθμος *lbest PSO* συνοψίζεται στον Αλγόριθμο 2.

Αλγόριθμος 2: ***lbest PSO***

Δημιουργία και αρχικοποίηση σμήνους σωματιδίων n_x διαστάσεων
repeat
  for σωματίδιο $i = 1, \ldots, n_s$ **do**
    // υπολογισμός ατομικής βέλτιστης θέσης
    if $f(\boldsymbol{x}_i) < f(\boldsymbol{y}_i)$ **then**
      $\boldsymbol{y}_i = \boldsymbol{x}_i$
    end
    // υπολογισμός τοπικής βέλτιστης θέσης
    if $f(\boldsymbol{y}_i) < f(\hat{\boldsymbol{y}}_i)$ **then**
      $\hat{\boldsymbol{y}}_i = \boldsymbol{y}_i$
    end
  end // for
  for σωματίδιο $i = 1, \ldots, n_s$ **do**
    ενημέρωση της ταχύτητας με χρήση της εξίσωσης (1.6)
    ενημέρωση της θέσης με χρήση της εξίσωσης (1.1)
  end // for
until (συνθήκη τερματισμού αληθής)

1.3.3. Σύγκριση αλγόριθμων *gbest PSO* και *lbest PSO*

Οι δύο αλγόριθμοι είναι παρόμοιοι με την έννοια ότι η κοινωνική συνιστώσα της εξίσωσης μεταβολής ταχύτητας ωθεί και τους δύο να συγκλίνουν προς το καθολικά βέλτιστο σωματίδιο. Αυτό είναι δυνατό για τον *lbest PSO* εξαιτίας της αλληλοεπικάλυψης των γειτονιών των σωματιδίων.

Οι βασικές διαφορές των δύο προσεγγίσεων οφείλονται στα χαρακτηριστικά σύγκλισής τους και είναι οι εξής:

- Ο αλγόριθμος *gbest PSO* συγκλίνει ταχύτερα από τον *lbest PSO*, εξαιτίας της μεγαλύτερης διασύνδεσης μεταξύ των σωματιδίων που επιτυγχάνει. Όμως, η ταχύτερη σύγκλιση έχει ως κόστος τη μικρότερη ποικιλομορφία.
- Ο αλγόριθμος *lbest PSO* είναι λιγότερο επιρρεπής στο να οδηγείται σε τοπικά ελάχιστα, εξαιτίας ακριβώς της μεγαλύτερης ποικιλομορφίας που επιδεικνύει, η οποία έχει ως αποτέλεσμα την κάλυψη μεγαλύτερου μέρους του χώρου αναζήτησης.

1.4. Δομές Κοινωνικής Δικτύωσης

Το χαρακτηριστικό που οδηγεί τη Βελτιστοποίηση Σμήνους Σωματιδίων είναι η κοινωνική αλληλεπίδραση. Τα σωματίδια του σμήνους μαθαίνουν το ένα από το άλλο και με βάση τη γνώση αυτή κινούνται για να μοιάσουν στους «καλύτερους» γείτονές τους. Η κοινωνική δομή της Βελτιστοποίησης Σμήνους Σωματιδίων καθορίζεται από το σχηματισμό αλληλοεπικαλυπτόμενων γειτονιών, στις οποίες τα σωματίδια επηρεάζουν το ένα το άλλο.

Η απόδοση του PSO εξαρτάται από τη δομή του κοινωνικού δικτύου. Η ροή της πληροφορίας διά μέσου του κοινωνικού δικτύου καθορίζεται από τους εξής παράγοντες: α) το βαθμό συνδεσιμότητας μεταξύ των κόμβων (μελών) του δικτύου, β) το βαθμό συσσώρευσης (clustering) των κόμβων και γ) τη μέση μικρότερη απόσταση μεταξύ των κόμβων.

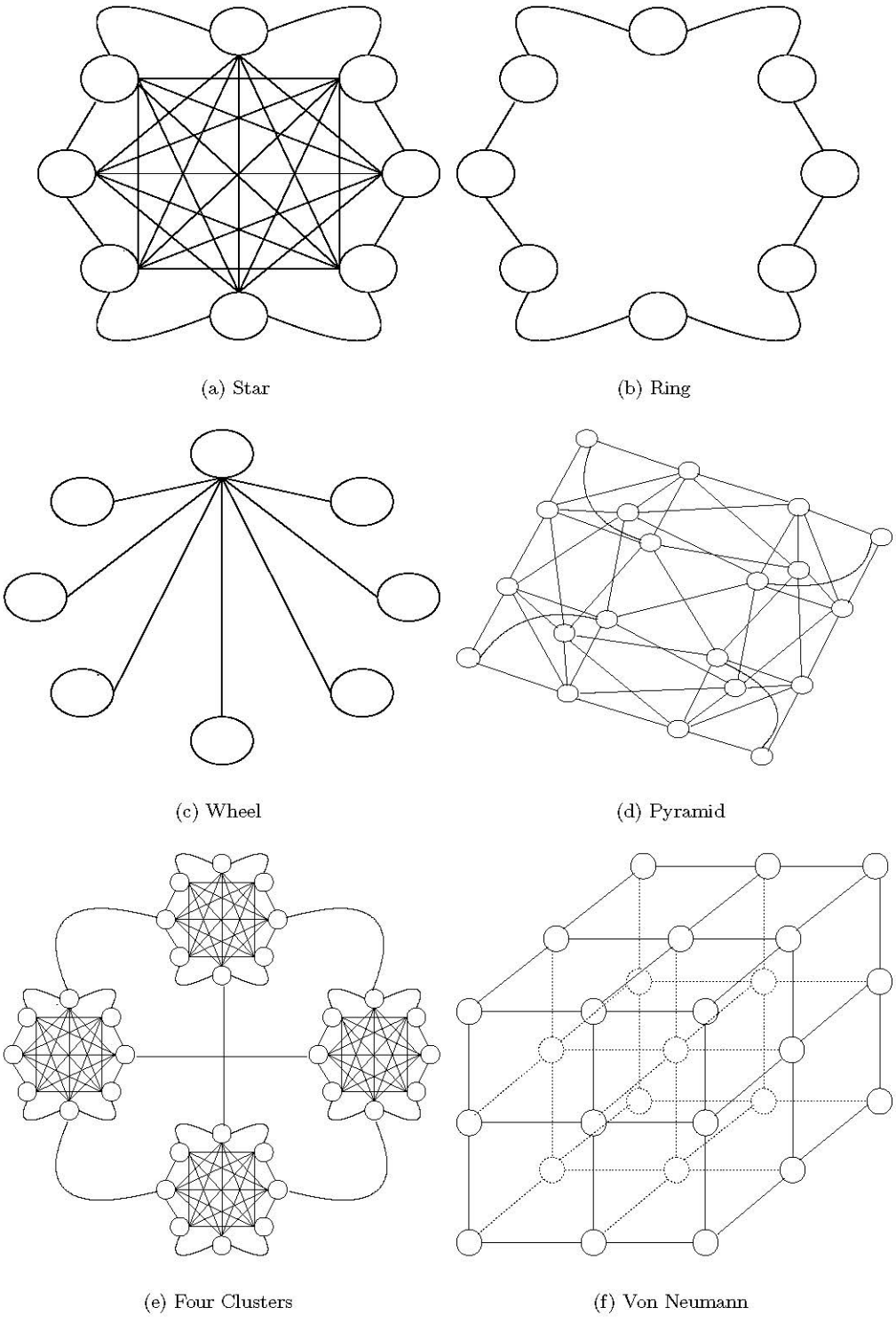
Σε ένα κοινωνικό δίκτυο υψηλής συνδεσιμότητας, τα περισσότερα μέλη του δικτύου επικοινωνούν μεταξύ τους, με αποτέλεσμα τη γρήγορη διάχυση της πληροφορίας. Αυτή οδηγεί σε ταχύτερη σύγκλιση σε μια λύση σε σχέση με ένα δίκτυο μικρότερης συνδεσιμότητας. Η ταχύτερη σύγκλιση έχει ως αντίτιμο την αυξημένη πιθανότητα εγκλωβισμού σε τοπικά ελάχιστα του χώρου αναζήτησης.

Σε ένα κοινωνικό δίκτυο χαμηλής συνδεσιμότητας, με υψηλό βαθμό συσσώρευσης των σωματιδίων στις γειτονιές, ο χώρος αναζήτησης δεν καλύπτεται επαρκώς, με αποτέλεσμα την αδυναμία εύρεσης των βέλτιστων λύσεων.

Οι σημαντικότερες δομές κοινωνικής δικτύωσης είναι οι εξής [5], [2]:

- Η κοινωνική δομή *αστέρα* (*star*), στην οποία τα σωματίδια συνδέονται όπως απεικονίζεται στο Σχήμα 2(a). Στην περίπτωση αυτή, κάθε σωματίδιο επικοινωνεί με όλα τα άλλα σωματίδια και έλκεται προς τη βέλτιστη λύση που υπολογίζεται από το σύνολο του σμήνους.
- Η κοινωνική δομή *δακτυλίου* (*ring*), όπου κάθε σωματίδιο επικοινωνεί με τους n_N άμεσους γείτονές του. Η περίπτωση $n_N = 2$, όπου κάθε σωματίδιο επικοινωνεί με τους πλησιέστερους γείτονές του, απεικονίζεται στο Σχήμα 2(b). Κάθε σωματίδιο επιχειρεί να μοιάσει στο βέλτιστο γείτονά του, κινούμενο προς τη βέλτιστη λύση που υπολογίζεται στη γειτονιά του. Στην περίπτωση της δομής δακτυλίου, οι γειτονιές αλληλοεπικαλύπτονται, το οποίο επιτρέπει την ανταλλαγή πληροφορίας μεταξύ τους και τελικά τη σύγκλιση σε μια μοναδική λύση. Επειδή η ροή της πληροφορίας γίνεται με μικρότερο ρυθμό, η σύγκλιση είναι πιο αργή, αλλά καλύπτονται μεγαλύτερα τμήματα του χώρου αναζήτησης σε σύγκριση με τη δομή αστέρα.
- Η κοινωνική δομή *τροχού* (*wheel*), στην οποία τα σωματίδια μιας γειτονιάς είναι απομονωμένα το ένα από το άλλο. Ένα σωματίδιο λειτουργεί ως κεντρικό σημείο και η πληροφορία μεταδίδεται διά μέσου του σωματιδίου αυτού (βλέπε Σχήμα 2(c)). Το κεντρικό σωματίδιο συγκρίνει την απόδοση όλων των σωματιδίων της γειτονιάς και ρυθμίζει τη θέση του προς το βέλτιστο γείτονα. Αν η νέα θέση του κεντρικού σωματιδίου βελτιώνει την απόδοση, τότε η μεταβολή μεταδίδεται σε όλα τα σωματίδια της γειτονιάς. Η κοινωνική δομή τροχού επιβραδύνει τη διάδοση των καλών λύσεων μέσα στο σμήνος.
- Η κοινωνική δομή *πυραμίδας* (*pyramid*), η οποία σχηματίζει ένα τρισδιάστατο πλαίσιο, όπως απεικονίζεται στο Σχήμα 2(d).
- Η κοινωνική δομή *τεσσάρων συσσωρεύσεων* (*four clusters*), η οποία απεικονίζεται στο Σχήμα 2(e). Σε αυτήν, σχηματίζονται τέσσερις συσσωρεύσεις, με δύο συνδέσεις μεταξύ τους. Τα σωματίδια κάθε συσσώρευσης έχουν πέντε γείτονες.

- Η κοινωνική δομή *Von Neumann*, στην οποία τα σωματίδια συνδέονται σε μια δομή πλέγματος, όπως απεικονίζεται στο Σχήμα 2(f).



Σχήμα 2: Δομές κοινωνικής δικτύωσης

Οι γειτονιές συνήθως καθορίζονται με βάση τους δείκτες των σωματιδίων. Για παράδειγμα, για τον αλγόριθμο *lbest* PSO με $n_N = 2$, η γειτονιά ενός σωματιδίου i αποτελείται από τα σωματίδια με δείκτες $i-1$, i και $i+1$. Επίσης, μπορεί να καθοριστεί με βάση την Ευκλείδεια απόσταση μεταξύ των σωματιδίων.

1.5. Παραλλαγές του Αλγόριθμου Βελτιστοποίησης Σμήνους Σωματιδίων

Ένας αριθμός από τροποποιήσεις στο βασικό αλγόριθμο Βελτιστοποίησης Σμήνους Σωματιδίων έχουν αναπτυχθεί με στόχο τη βελτίωση τόσο της ταχύτητας σύγκλισής του όσο και της ποιότητας των αποτελεσμάτων που παράγει. Οι κυριότερες από τις τροποποιήσεις αυτές παρουσιάζονται στη συνέχεια.

1.5.1. Αποκοπή ταχύτητας (velocity clamping)

Οι εξισώσεις μεταβολής της ταχύτητας (1.2) και (1.6) αποτελούνται από τρεις όρους οι οποίοι συμβάλλουν στο μέγεθος βήματος των σωματιδίων. Από τις πρώτες εφαρμογές του βασικού αλγόριθμου PSO έχει παρατηρηθεί ότι η ταχύτητα γρήγορα λαμβάνει μεγάλες τιμές, ιδιαίτερα για σωματίδια που βρίσκονται μακριά από τη βέλτιστη θέση γειτονιάς τους (lbest position) και τη βέλτιστη ατομική τους θέση (pbest position). Αυτό έχει ως συνέπεια τα σωματίδια να αποκτούν μεγάλες μεταβολές θέσης και να αποκλίνουν από το χώρο αναζήτησης. Για να αποφευχθεί αυτό, οι ταχύτητες αποκόπτονται για να παραμένουν μέσα σε συγκεκριμένα χωρικά όρια [7]. Έτσι, αν η ταχύτητα ενός σωματιδίου υπερβεί μια καθορισμένη μέγιστη ταχύτητα, ως ταχύτητα του σωματιδίου τίθεται η μέγιστη ταχύτητα. Έστω $V_{\max,j}$ η μέγιστη επιτρεπτή ταχύτητα στη διάσταση j. Τότε, η ταχύτητα του σωματιδίου καθορίζεται ως εξής:

$$v_{ij}(t+1) = \begin{cases} v'_{ij}(t+1), & v'_{ij}(t+1) < V_{\max,j} \\ V_{\max,j}, & v'_{ij}(t+1) \geq V_{\max,j} \end{cases} \tag{1.9}$$

όπου η ταχύτητα v'_{ij} υπολογίζεται με βάση την εξίσωση (1.2) ή (1.6).

Η τιμή $V_{\max,j}$ είναι πολύ σημαντική γιατί καθορίζει την αναλυτικότητα του χώρου αναζήτησης. Μεγάλες τιμές της $V_{\max,j}$ ευνοούν την καθολική εξερεύνηση (global

exploration), ενώ μικρές τιμές της ενθαρρύνουν την τοπική εκμετάλλευση (local exploitation) του χώρου αναζήτησης.

Οι τιμές $V_{\max,j}$ συνήθως επιλέγονται να είναι ένα κλάσμα του πεδίου τιμών της κάθε διάστασης του χώρου αναζήτησης. Δηλαδή,

$$V_{\max,j} = \delta\left(x_{\max,j} - x_{\min,j}\right) \tag{1.10}$$

όπου $x_{\max,j}$, $x_{\min,j}$ είναι αντίστοιχα η μέγιστη και η ελάχιστη τιμή του πεδίου ορισμού του $\boldsymbol{x}$ στη διάσταση j και $\delta \in (0,1]$. Η επιλογή του δ εξαρτάται από το πρόβλημα.

1.5.2. Βάρος Αδράνειας (Inertia Weight)

Το βάρος αδράνειας έχει εισαχθεί ως ένας μηχανισμός ελέγχου των δυνατοτήτων εξερεύνησης και εκμετάλλευσης ενός σμήνους σωματιδίων, καθώς και για την εξάλειψη της ανάγκης χρησιμοποίησης της μεθόδου αποκοπής ταχύτητας [8], [9]. Το βάρος αδράνειας w ελέγχει την ορμή ενός σωματιδίου με την προσθήκη ενός συντελεστή βαρύτητας στη συνεισφορά της προηγούμενης ταχύτητάς του. Για τον αλγόριθμο *gbest PSO*, η εξίσωση μεταβολής της ταχύτητας (1.2) τροποποιείται ως εξής:

$$v_{ij}(t+1) = wv_{ij}(t) + c_1 r_{1j}(t)\left[y_{ij}(t) - x_{ij}(t)\right] + c_2 r_{2j}(t)\left[\hat{y}_j(t) - x_{ij}(t)\right] \tag{1.11}$$

Παρόμοια τροποποίηση γίνεται και στην εξίσωση μεταβολής της ταχύτητας του αλγόριθμου *lbest PSO*.

Η τιμή του w είναι εξαιρετικά σημαντική τόσο για τη διασφάλιση της σύγκλισης του αλγόριθμου, όσο και για τη βέλτιστη ισορροπία μεταξύ των δυνατοτήτων του για εξερεύνηση και εκμετάλλευση. Για τιμές $w \geq 1$, οι ταχύτητες αυξάνονται με το χρόνο και ο αλγόριθμος αποκλίνει. Για τιμές $w < 1$, τα σωματίδια επιβραδύνονται μέχρι οι ταχύτητές τους να φτάσουν την τιμή μηδέν. Στην περίπτωση αυτή, μεγάλες τιμές του w διευκολύνουν την εξερεύνηση, ενώ μικρές τιμές προωθούν την τοπική εκμετάλλευση. Όσο μικρότερη είναι η τιμή του w, τόσο περισσότερο η γνωσιακή και η κοινωνική συνιστώσα ελέγχουν τις μεταβολές θέσης ενός σωματιδίου.

Οι αρχικές υλοποιήσεις του βάρους αδράνειας χρησιμοποιούσαν μια στατική τιμή, για όλα τα σωματίδια και σε όλες τις διαστάσεις, για το συνολικό χρόνο αναζήτησης. Οι επόμενες υλοποιήσεις έκαναν χρήση δυναμικά μεταβαλλόμενων τιμών αδράνειας. Συνήθως, οι προσεγγίσεις αυτές ξεκινούν με μεγάλες τιμές του βάρους αδράνειας που ελαττώνονται με το χρόνο. Έτσι, αρχικά διευκολύνουν τη φάση εξερεύνησης, ενώ στη συνέχεια προωθούν τη φάση εκμετάλλευσης του χώρου αναζήτησης.

Οι μέθοδοι που χρησιμοποιούν δυναμικά μεταβαλλόμενες τιμές αδράνειας μπορούν να κατηγοριοποιηθούν ως εξής [2], [5], [10]:

- **Τυχαίες μεταβολές.** Το βάρος αδράνειας κάθε σωματιδίου επιλέγεται τυχαία σε κάθε επανάληψη του αλγόριθμου. Εναλλακτικές προσεγγίσεις αποτελούν:
  - Η δειγματοληψία από μια Γκαουσιανή κατανομή (Gaussian distribution) [11],

$$w \sim N\left(0.72, \sigma\right) \tag{1.12}$$

    όπου σ αρκετά μικρό έτσι ώστε $w < 1$.

  - Η μεταβολή του w σύμφωνα με τη σχέση:

$$w = \left(c_1 r_1 + c_2 r_2\right) \tag{1.13}$$

    χωρίς στοχαστική μεταβολή της γνωσιακής και κοινωνικής συνιστώσας.

  - Ο υπολογισμός του w από τη σχέση [10]:

$$w = 0.5 + \frac{1}{2}\text{rand}(w_{\min}, w_{\max}) \tag{1.14}$$

    όπου $\text{rand}(a,b)$ τυχαίος αριθμός στο διάστημα $[a,b]$, ενώ για τις ακραίες τιμές $w_{\min}$, $w_{\max}$ ισχύει: $0 \le w_{\min} < w_{\max} < 1$.

- **Γραμμική μείωση (linear decreasing).** Στην περίπτωση αυτή, μια μεγάλη αρχική τιμή του βάρους αδράνειας (συνήθως 0.9) ελαττώνεται γραμμικά με το χρόνο σε μια μικρή τελική τιμή (συνήθως 0.4) [12]. Ισχύει:

$$w(t) = \big(w(0) - w(n_t)\big)\frac{n_t - t}{n_t} + w(n_t) \tag{1.15}$$

όπου n_t είναι ο μέγιστος αριθμός επαναλήψεων του αλγόριθμου, $w(0)$ το αρχικό βάρος αδράνειας, $w(n_t)$ το τελικό βάρος αδράνειας και $w(t)$ το βάρος αδράνειας στο χρονικό βήμα t. Επίσης, $w(0) > w(n_t)$.

- **Μη γραμμική μείωση (nonlinear decreasing).** Εδώ, μια μεγάλη αρχική τιμή ελαττώνεται μη γραμμικά με το χρόνο σε μια μικρή τελική τιμή. Οι μη γραμμικές μέθοδοι επιτρέπουν μικρότερο χρόνο εξερεύνησης από ότι οι γραμμικές μέθοδοι, δίνοντας έμφαση στη φάση εκμετάλλευσης. Μερικές από τις πιο σημαντικές μη γραμμικές μεθόδους είναι οι ακόλουθες:

  - Το βάρος αδράνειας μεταβάλλεται σύμφωνα με τη σχέση [13]:

$$w\big(t+1\big) = \frac{\big(w(0) - 0.4\big)\big(n_t - t\big)}{n_t + 0.4} \tag{1.16}$$

όπου $w(0) = 0.9$.

  - Το w μεταβάλλεται ως εξής [14]:

$$w\big(t+1\big) = \alpha w(t') \tag{1.17}$$

όπου $\alpha = 0.975$ και t' το χρονικό βήμα που συνέβη η τελευταία μεταβολή του βάρους αδράνειας. Το βάρος αδράνειας τροποποιείται όταν δεν υπάρχει σημαντική μεταβολή της συνάρτησης βελτιστοποίησης του σμήνους.

– Το μέγεθος μεταβολής του βάρους αδράνειας είναι ανάλογο με τη σχετική βελτίωση του σμήνους [15],

$$w_i(t+1) = w(0) + \left(w(n_t) - w(0)\right)\frac{e^{m_i(t)} - 1}{e^{m_i(t)} + 1} \tag{1.18}$$

όπου η σχετική βελτίωση $m_i(t)$ υπολογίζεται από τη σχέση

$$m_i(t) = \frac{f\left(\hat{\boldsymbol{y}}_i(t)\right) - f\left(\boldsymbol{x}_i(t)\right)}{f\left(\hat{\boldsymbol{y}}_i(t)\right) + f\left(\boldsymbol{x}_i(t)\right)} \tag{1.19}$$

με $w(n_t) \approx 0.5$ και $w(0) < 1$. Η μέθοδος αυτή αναπτύχθηκε για μεταβολές της ταχύτητας δίχως τη γνωσιακή συνιστώσα.

- **Προσαρμοστικό βάρος αδράνειας.** Το βάρος αδράνειας δεν καθορίζεται από το χρόνο αλλά από μεταβλητές που απεικονίζουν το δυναμικό περιβάλλον του σμήνους.

  Μια τέτοια προσέγγιση είναι η χρήση του ελαττούμενου βάρους αδράνειας σιγμοειδούς συνάρτησης (sigmoid function decreasing inertia weight) [16]. Στη μέθοδο αυτή, για κάθε σωματίδιο του σμήνους η μέση απόστασή του από τα υπόλοιπα d_i σε κάθε χρονική στιγμή (επανάληψη του αλγόριθμου) δίνεται από τη σχέση:

$$d_i = \frac{1}{n_s - 1}\sum_{j=1, j\neq i}^{n_s}\sqrt{\sum_{k=1}^{n_x}\left(x_i^k - x_j^k\right)^2} \tag{1.20}$$

όπου n_s το πλήθος των σωματιδίων και n_x ο αριθμός των διαστάσεων του προβλήματος.

Η απόσταση d_i του καθολικά βέλτιστου σωματιδίου συμβολίζεται με d_g. Επίσης, οι μεταβλητές $d_{\max}$, $d_{\min}$ αντιστοιχούν στη μέγιστη και στην ελάχιστη απόσταση μεταξύ των σωματιδίων.

Ο «παράγοντας εξέλιξης» ("evolutionary factor") f_e ορίζεται ως εξής:

$$f_e = \frac{d_g - d_{\min}}{d_{\max} - d_{\min}} \tag{1.21}$$

όπου $f_e \in [0,1]$. Το βάρος αδράνειας w δίνεται από τη σχέση:

$$w(f_e) = \frac{1}{1+1.5e^{-2.6 f_e}} \tag{1.22}$$

όπου $w(f_e) \in [0.4, 0.9]$ $\forall f_e \in [0,1]$. Το w δεν είναι μονοτονική συνάρτηση του χρόνου αλλά του παράγοντα f_e. Ο παράγοντας f_e ρυθμίζει τις φάσεις εξερεύνησης και τοπικής εκμετάλλευσης του σμήνους.

- **Χαοτικό βάρος αδράνειας.** Η τιμή του βάρους αδράνειας καθορίζεται από μια χαοτική απεικόνιση. Υπάρχουν δύο εναλλακτικές στρατηγικές υλοποίησης [17]:
  - Το χαοτικό ελαττούμενο βάρος αδράνειας (chaotic decreasing inertia weight), όπου το $w(t)$ υπολογίζεται από τη σχέση:

$$w(t) = \left(w(0) - w(n_t)\right)\frac{n_t - t}{n_t} + w(n_t)\cdot z \tag{1.23}$$

όπου z τυχαίος αριθμός στο διάστημα $(0,1)$, $z = \text{rand}(0,1)$, για τον οποίο ισχύει η λογιστική απεικόνιση

$$z = 4 \cdot z \cdot (1 - z) \tag{1.24}$$

– Το χαοτικό τυχαίο βάρος αδράνειας (chaotic random inertia weight), όπου το w υπολογίζεται ως εξής:

$$w = 0.5 \cdot \text{rand}\left(w(0), w(n_t)\right) + 0.5z \tag{1.25}$$

και για το z ισχύουν τα παραπάνω.

- **Fuzzy βάρος αδράνειας.** Στην περίπτωση αυτή, το βάρος αδράνειας ρυθμίζεται δυναμικά με βάση fuzzy σύνολα και κανόνες [18].

- **Γραμμική αύξηση.** Το βάρος αδράνειας αυξάνεται γραμμικά μεταξύ δύο ακραίων τιμών.

1.5.3. Συντελεστής Περιορισμού (Constriction Coefficient)

Στην προσέγγιση αυτή οι ταχύτητες περιορίζονται από μια σταθερά χ, η οποία καλείται συντελεστής περιορισμού (constriction coefficient) [19]. Η εξίσωση μεταβολής της ταχύτητας για τον αλγόριθμο *gbest PSO* τροποποιείται ως εξής:

$$v_{ij}(t+1) = \chi\left[v_{ij}(t) + \phi_1\left(y_{ij}(t) - x_{ij}(t)\right) + \phi_2\left(\hat{y}_j(t) - x_{ij}(t)\right)\right] \tag{1.26}$$

όπου

$$\chi = \frac{2\kappa}{\left|2 - \phi - \sqrt{\phi(\phi-4)}\right|} \tag{1.27}$$

με $\phi = \phi_1 + \phi_2$, $\phi_1 = c_1 r_1$ και $\phi_2 = c_2 r_2$. Για την εξίσωση (1.27) ισχύουν οι περιορισμοί $\phi \geq 4$ και $\kappa \in [0,1]$.

Η προσέγγιση αυτή αναπτύχθηκε ως ένας φυσικός, δυναμικός τρόπος για τη διασφάλιση της σύγκλισης του αλγόριθμου σε ένα ευσταθές σημείο, δίχως την ανάγκη για χρήση του περιορισμού ταχύτητας. Η σύγκλιση του σμήνους είναι εγγυημένη κάτω από τις συνθήκες

$\phi \geq 4$ και $\kappa \in [0,1]$. Ο συντελεστής περιορισμού χ λαμβάνει τιμές στο διάστημα $[0,1]$, το οποίο υποδηλώνει ότι η ταχύτητα ελαττώνεται σε κάθε χρονικό βήμα.

Η παράμετρος κ στην εξίσωση (1.27) ελέγχει τις δυνατότητες εξερεύνησης και εκμετάλλευσης του σμήνους. Για $\kappa \approx 0$ επιτυγχάνεται γρήγορη σύγκλιση με τοπική εκμετάλλευση. Για $\kappa \approx 1$ η σύγκλιση του αλγόριθμου είναι αργή με υψηλό βαθμό δυνατότητας εξερεύνησης. Συνήθως, η παράμετρος κ λαμβάνει μια σταθερή τιμή. Όμως, κατά την εκτέλεση του αλγόριθμου μπορεί να επιτευχθεί αρχικά υψηλός βαθμός εξερεύνησης και στη συνέχεια ενίσχυση της φάσης τοπικής εκμετάλλευσης, χρησιμοποιώντας μια αρχική τιμή για το κ κοντά στη μονάδα και ελαττώνοντάς την προς το μηδέν.

Η μέθοδος του συντελεστή περιορισμού είναι ισοδύναμη με τη μέθοδο του βάρους αδράνειας. Για συγκεκριμένη τιμή του συντελεστή χ, το ισοδύναμο μοντέλο βάρους αδράνειας μπορεί να καθοριστεί θέτοντας $w = \chi$, $\phi_1 = \chi c_1 r_1$ και $\phi_2 = \chi c_2 r_2$. Οι διαφορές των δύο μεθόδων είναι οι εξής:

- Ο περιορισμός ταχύτητας δεν είναι απαραίτητος στη μέθοδο συντελεστή περιορισμού.
- Η μέθοδος συντελεστή περιορισμού εγγυάται σύγκλιση κάτω από συγκεκριμένες συνθήκες.
- Στο μοντέλο περιορισμού ο καθορισμός της αλλαγής διεύθυνσης των σωματιδίων γίνεται μέσω των παραμέτρων ϕ_1 και ϕ_2.

1.5.4. Μοντέλα ταχύτητας

Γνωσιακό μοντέλο (Cognition-only model)

Το γνωσιακό μοντέλο εξαιρεί την κοινωνική συνιστώσα από την εξίσωση μεταβολής της ταχύτητας (1.2), η οποία λαμβάνει τη μορφή

$$v_{ij}(t+1) = v_{ij}(t) + c_1 r_{1j}(t)\left[y_{ij}(t) - x_{ij}(t)\right] \tag{1.28}$$

Στην παραπάνω εξίσωση μπορεί να προστεθεί και το βάρος αδράνειας w.

Η συμπεριφορά των σωματιδίων στο γνωσιακό μοντέλο επιδεικνύει μια στοχαστική τάση για την επιστροφή τους στις προηγούμενες βέλτιστες θέσεις τους. Το γνωσιακό μοντέλο είναι

λίγο πιο επιρρεπές σε αποτυχία από το πλήρες μοντέλο [20]. Επίσης, έχει μικρή απόδοση σε δυναμικά μεταβαλλόμενα περιβάλλοντα [21].

Κοινωνικό μοντέλο (Social-only model)

Το κοινωνικό μοντέλο εξαιρεί τη γνωσιακή συνιστώσα από την εξίσωση μεταβολής της ταχύτητας:

$$v_{ij}(t+1) = v_{ij}(t) + c_2 r_{2j}(t)\left[\hat{y}_j(t) - x_{ij}(t)\right] \tag{1.29}$$

Η παραπάνω εξίσωση ισχύει για τον αλγόριθμο *gbest PSO*. Για τον αλγόριθμο *lbest PSO*, η παράμετρος $\hat{y}_j(t)$ αντικαθίσταται από την $\hat{y}_{ij}(t)$.

Στο κοινωνικό μοντέλο, τα σωματίδια δεν έχουν την τάση να επιστρέφουν στις προηγούμενες βέλτιστες θέσεις τους. Στην περίπτωση αυτή, όλα τα σωματίδια έλκονται από τη βέλτιστη θέση της γειτονιάς τους (*lbest position*).

Το μοντέλο αυτό είναι ταχύτερο και αποδοτικότερο από το γνωσιακό και το πλήρες μοντέλο [20]. Αυτό ισχύει και σε δυναμικά περιβάλλοντα [21].

Αλτρουιστικό μοντέλο (Selfless model)

Το αλτρουιστικό μοντέλο είναι βασικά το κοινωνικό μοντέλο, με τη διαφορά ότι η βέλτιστη θέση της γειτονιάς ενός σωματιδίου υπολογίζεται μόνο από τους γείτονές του, δηλαδή το ίδιο το σωματίδιο δεν μπορεί να γίνει το βέλτιστο της γειτονιάς του. Το αλτρουιστικό μοντέλο είναι ταχύτερο από το κοινωνικό μοντέλο σε ορισμένες εφαρμογές [20], ενώ έχει μικρή απόδοση σε δυναμικά μεταβαλλόμενα περιβάλλοντα [21].

1.6. Στοιχεία του Αλγόριθμου Βελτιστοποίησης Σμήνους Σωματιδίων

1.6.1. Μέγεθος παραμέτρων

Ο βασικός αλγόριθμος PSO επηρεάζεται από ένα πλήθος παραμέτρων ελέγχου. Οι σημαντικότερες είναι οι εξής:

- **Μέγεθος σμήνους**, n_s, δηλαδή το πλήθος των σωματιδίων του σμήνους. Όσο μεγαλύτερο είναι το πλήθος των σωματιδίων, τόσο μεγαλύτερη είναι η αρχική ποικιλομορφία του σμήνους. Ένα μεγάλο σμήνος καλύπτει μεγαλύτερες περιοχές του χώρου αναζήτησης ανά επανάληψη του αλγόριθμου. Όμως, όσο μεγαλύτερος είναι ο αριθμός των σωματιδίων, τόσο αυξάνεται η υπολογιστική πολυπλοκότητα του αλγόριθμου ανά επανάληψη. Τυπικές τιμές του αριθμού σωματιδίων είναι από 10 έως 30, $n_s \in [10,30]$. Γενικότερα, το βέλτιστο μέγεθος σμήνους εξαρτάται από το πεδίο εφαρμογής του αλγόριθμου.

- **Μέγεθος γειτονιάς**. Το μέγεθος της γειτονιάς καθορίζει την έκταση της κοινωνικής αλληλεπίδρασης μέσα στο σμήνος. Όσο μικρότερο είναι το μέγεθος, τόσο μικρότερη είναι η αλληλεπίδραση. Μικρό μέγεθος γειτονιάς οδηγεί σε πιο αργή σύγκλιση του αλγόριθμου, αλλά ταυτόχρονα επιτυγχάνει πιο αξιόπιστη σύγκλιση σε βέλτιστη λύση. Τα σμήνη με μικρό μέγεθος γειτονιάς είναι λιγότερο επιρρεπή σε τοπικά ελάχιστα.

- **Αριθμός επαναλήψεων.** Ο αριθμός των επαναλήψεων που απαιτείται για την επίτευξη βέλτιστης λύσης εξαρτάται από τη φύση του προβλήματος. Μικρός αριθμός επαναλήψεων οδηγεί σε πρόωρο τερματισμό του αλγόριθμου, ενώ μεγάλος αριθμός επαναλήψεων μπορεί να προσθέσει περιττή υπολογιστική πολυπλοκότητα.

- **Συντελεστές επιτάχυνσης (acceleration coefficients).** Οι συντελεστές επιτάχυνσης c_1 και c_2 μαζί με τα τυχαία διανύσματα r_1, r_2, ελέγχουν τη στοχαστική συνεισφορά της γνωσιακής και της κοινωνικής συνιστώσας στη συνολική ταχύτητα του σωματιδίου. Ο συντελεστής c_1 εκφράζει την εμπιστοσύνη ενός σωματιδίου στον εαυτό του, ενώ ο c_2 εκφράζει πόσο εμπιστεύεται ένα σωματίδιο τους γείτονές του.

  Μικρές τιμές για τους συντελεστές c_1 και c_2 έχουν ως αποτέλεσμα ομαλές τροχιές σωματιδίων, επιτρέποντας στα σωματίδια να περιπλανηθούν μακριά από καλές περιοχές για εξερεύνηση πριν επιστρέψουν ξανά σε αυτές. Μεγάλες τιμές προκαλούν μεγαλύτερη επιτάχυνση, με πιο απότομες κινήσεις προς ή μακριά από καλές περιοχές.

  Συνήθως, οι συντελεστές c_1 και c_2 λαμβάνουν σταθερές τιμές, με τις βέλτιστες τιμές τους να υπολογίζονται εμπειρικά. Όμως, υπάρχουν και προσεγγίσεις όπου

χρησιμοποιούνται προσαρμοστικοί συντελεστές επιτάχυνσης, μερικές από τις οποίες είναι και οι ακόλουθες:

– Ο συντελεστής c_2, υιοθετώντας το κοινωνικό μοντέλο, δίνεται από τη σχέση [15]:

$$c_2(t) = \frac{c_{2,\min} + c_{2,\max}}{2} + \frac{c_{2,\max} - c_{2,\min}}{2} + \frac{e^{-m_i(t)} - 1}{e^{-m_i(t)} + 1} \tag{1.30}$$

όπου το m_i δίνεται από τη σχέση (1.19).

– Ο συντελεστής c_1 ελαττώνεται γραμμικά με το χρόνο, ενώ ο c_2 αυξάνεται γραμμικά [12]:

$$c_1(t) = \left(c_{1,\min} - c_{1,\max}\right)\frac{t}{n_t} + c_{1,\max} \tag{1.31}$$

$$c_2(t) = \left(c_{2,\max} - c_{2,\min}\right)\frac{t}{n_t} + c_{2,\min} \tag{1.32}$$

όπου $c_{1,\max} = c_{2,\max} = 2.5$ και $c_{1,\min} = c_{2,\min} = 0.5$. Η προσέγγιση αυτή εστιάζει στην εξερεύνηση στα αρχικά στάδια της βελτιστοποίησης, ενώ ενθαρρύνει τη σύγκλιση σε βέλτιστο προς το τέλος της διαδικασίας βελτιστοποίησης.

1.6.2. Αρχικοποίηση αλγόριθμου.

Το πρώτο βήμα κατά την εκτέλεση του αλγόριθμου PSO είναι η αρχικοποίηση του σμήνους και των παραμέτρων ελέγχου. Επομένως, θα πρέπει να καθοριστούν οι συντελεστές επιτάχυνσης c_1 και c_2, οι αρχικές ταχύτητες και θέσεις των σωματιδίων του σμήνους, καθώς και οι αρχικές ατομικές βέλτιστες θέσεις τους (*pbest positions*). Επίσης, θα πρέπει να καθοριστεί το μέγεθος σμήνους και στην περίπτωση του αλγόριθμου *lbest PSO*, το μέγεθος γειτονιάς.

Συνήθως, οι θέσεις των σωματιδίων αρχικοποιούνται έτσι ώστε να καλύπτουν ομοιόμορφα το χώρο αναζήτησης. Η αποδοτικότητα του αλγόριθμου PSO επηρεάζεται τόσο από την αρχική διασπορά του σμήνους, όσο και από την ομοιόμορφη κατανομή των σωματιδίων στο χώρο αναζήτησης.

Έστω ότι ο χώρος αναζήτησης ορίζεται από τα διανύσματα $\boldsymbol{x}_{\min}$, $\boldsymbol{x}_{\max}$, τα οποία αντιστοιχούν στην ελάχιστη και μέγιστη απόσταση σε κάθε διάσταση. Τότε, μια αποτελεσματική μέθοδος αρχικοποίησης των θέσεων των σωματιδίων του σμήνους είναι η ακόλουθη:

$$x_{ij}(0) = x_{\min,j} + r_j \left(x_{\max,j} - x_{\min,j} \right), \qquad \forall\, j = 1,\ldots,n_x, \;\; \forall\, i = 1,\ldots,n_s \tag{1.33}$$

όπου $r_j \sim U(0,1)$.

Οι αρχικές ταχύτητες των σωματιδίων μπορούν να τεθούν ίσες με το μηδέν:

$$\boldsymbol{v}_i(0) = \mathbf{0} \tag{1.34}$$

Η ατομική βέλτιστη θέση κάθε σωματιδίου αρχικοποιείται ως η θέση του σωματιδίου το χρονικό βήμα $t = 0$:

$$\boldsymbol{y}_i(0) = \boldsymbol{x}_i(0) \tag{1.35}$$

Επίσης, άλλα σχήματα αρχικοποίησης των θέσεων των σωματιδίων του σμήνους αποτελούν οι ακολουθίες Sobol, οι ακολουθίες Faure και η μη γραμμική μέθοδος Simplex.

1.6.3. Τερματισμός αλγόριθμου

Η επιλογή της συνθήκης τερματισμού του αλγόριθμου PSO θα πρέπει να πληροί τα εξής κριτήρια:

- Η συνθήκη τερματισμού δεν θα πρέπει να οδηγεί σε πρόωρη σύγκλιση του αλγόριθμου, καθώς τότε οι λύσεις που προκύπτουν είναι μη βέλτιστες.

- Η συνθήκη τερματισμού δεν θα πρέπει να απαιτεί συχνό υπολογισμό της συνάρτησης βελτιστοποίησης, ο οποίος αυξάνει σημαντικά την υπολογιστική πολυπλοκότητα της διαδικασίας αναζήτησης.

Στα πλαίσια αυτά, οι ακόλουθες συνθήκες τερματισμού μπορούν να χρησιμοποιηθούν:

- **Μέγιστος αριθμός επαναλήψεων ή υλοποιήσεων της συνάρτησης βελτιστοποίησης.** Το κριτήριο αυτό χρησιμοποιείται συνήθως σε συνάρτηση με κριτήρια σύγκλισης, έτσι ώστε να επιβάλλεται ο τερματισμός του, όταν ο αλγόριθμος αποτυγχάνει να συγκλίνει. Το κριτήριο αυτό είναι χρήσιμο σε εφαρμογές όπου ζητείται η εύρεση της βέλτιστης λύσης σε μια περιορισμένη χρονική περίοδο.

- **Εύρεση αποδεκτής λύσης.** Έστω $\boldsymbol{x}^*$ το βέλτιστο της αντικειμενικής συνάρτησης f. Τότε, η διαδικασία αναζήτησης τερματίζεται όταν βρεθεί σωματίδιο $\boldsymbol{x}_i$ τέτοιο ώστε να ισχύει: $f\left(\boldsymbol{x}_i\right) \le \left| f\left(\boldsymbol{x}^*\right) - \varepsilon \right|$, δηλαδή όταν επιτευχθεί αποδεκτό μέγεθος σφάλματος. Η επιλογή του κατωφλίου ε είναι ιδιαίτερα σημαντική. Αν το ε είναι πολύ μεγάλο, η αναζήτηση καταλήγει σε μια υποβέλτιστη λύση. Αν το ε είναι πολύ μικρό, η αναζήτηση μπορεί να μην τερματίσει.

- **Μη ύπαρξη βελτίωσης για καθορισμένο αριθμό επαναλήψεων.** Η βελτίωση μπορεί να υπολογιστεί με διάφορους τρόπους. Για παράδειγμα, όταν η μέση μεταβολή των θέσεων των σωματιδίων είναι πολύ μικρή ή η μέση ταχύτητά τους για ορισμένο αριθμό επαναλήψεων είναι περίπου ίση με το μηδέν, τότε η αναζήτηση μπορεί να τερματιστεί.

- **Η κανονικοποιημένη ακτίνα του σμήνους βρίσκεται κοντά στο μηδέν.** Η κανονικοποιημένη ακτίνα σμήνους ορίζεται ως εξής [22]:

$$R_{norm} = \frac{R_{\max}}{\text{diameter}(S)} \tag{1.36}$$

όπου $\text{diameter}(S)$ είναι η διάμετρος του αρχικού σμήνους και η μέγιστη διάμετρος $R_{\max}$ είναι:

$$R_{\max} = \left\| \boldsymbol{x}_m - \hat{\boldsymbol{y}} \right\|, \quad m = 1, \ldots, n_s \tag{1.37}$$

με

$$\left\| \boldsymbol{x}_m - \hat{\boldsymbol{y}} \right\| \geq \left\| \boldsymbol{x}_i - \hat{\boldsymbol{y}} \right\|, \quad \forall\ i = 1, \ldots, n_s \tag{1.38}$$

Στις παραπάνω εξισώσεις $\| \ \|$ είναι μια κατάλληλη νόρμα απόστασης, π.χ. η Ευκλείδεια απόσταση.

Ο αλγόριθμος τερματίζεται όταν $R_{norm} < \varepsilon$. Αν το ε είναι πολύ μεγάλο, η διαδικασία αναζήτησης μπορεί να σταματήσει πρόωρα πριν την εύρεση μιας καλής λύσης. Αν το ε είναι πολύ μικρό, η αναζήτηση μπορεί να χρειαστεί πολύ περισσότερες επαναλήψεις ώσπου τα σωματίδια να σχηματίσουν ένα συμπαγές σμήνος γύρω από την καθολική βέλτιστη θέση.

- **Η κλίση της αντικειμενικής συνάρτησης είναι κατά προσέγγιση μηδενική.** Η κλίση της αντικειμενικής συνάρτησης δίνεται από τη σχέση [22]:

$$f'(t) = \frac{f\left(\hat{\boldsymbol{y}}(t)\right) - f\left(\hat{\boldsymbol{y}}(t-1)\right)}{f\left(\hat{\boldsymbol{y}}(t)\right)} \tag{1.39}$$

Αν ισχύει $f'(t) < \varepsilon$ για έναν αριθμό διαδοχικών επαναλήψεων, θεωρούμε ότι το σμήνος έχει συγκλίνει.

1.7. Εφαρμογή του αλγόριθμου PSO σε δυναμικά περιβάλλοντα

1.7.1. Προβλήματα εφαρμογής

Οι αρχικές εφαρμογές του έδειξαν ότι ο αλγόριθμος PSO έχει μια εγγενή δυνατότητα παρακολούθησης μεταβολών, για δυναμικά περιβάλλοντα τύπου I με μικρή χωρική μεταβλητότητα [21], [11], [23].

Κατά την εκτέλεση του αλγόριθμου PSO, κάθε σωματίδιο σταδιακά συγκλίνει σε ένα σημείο της νοητής ευθείας που συνδέει την ατομική βέλτιστη θέση του (*pbest position*) με την καθολική βέλτιστη θέση (*gbest position*). Η τροχιά ενός σωματιδίου μπορεί να περιγραφεί ως ένα ημιτονοειδές κύμα φθίνοντος πλάτους γύρω από την καθολική βέλτιστη θέση [24].

Στις περιπτώσεις όπου υπάρχει μικρή μεταβολή της θέσης του βέλτιστου, είναι πιθανό κάποιο από τα ταλαντευόμενα σωματίδια να ανακαλύψει το νέο κοντινό βέλτιστο και να οδηγήσει το σμήνος γύρω από αυτό. Όμως, εάν η χωρική μεταβλητότητα είναι μεγάλη, ωθώντας το βέλτιστο έξω από την ακτίνα του σμήνους, ο αλγόριθμος PSO θα αποτύχει να εντοπίσει το καινούργιο βέλτιστο. Σε τέτοιες περιπτώσεις απαιτούνται μηχανισμοί για την αύξηση της ποικιλομορφίας του σμήνους.

Επίσης, στις περιπτώσεις χωρικών μεταβολών όπου η τιμή του βέλτιστου παραμένει αναλλοίωτη ή είναι μεγαλύτερη από την προηγούμενη, θεωρώντας πρόβλημα ελαχιστοποίησης, δηλαδή όταν ισχύει $f\left(\boldsymbol{x}^*(t)\right) \le f\left(\boldsymbol{x}^*\left(t+1\right)\right)$ για $\boldsymbol{x}^*(t) \ne \boldsymbol{x}^*\left(t+1\right)$, η καθολική βέλτιστη θέση δεν μεταβάλλεται. Αυτό έχει ως αποτέλεσμα ο αλγόριθμος PSO να αποτυγχάνει να παρακολουθήσει το μεταβαλλόμενο ελάχιστο. Τέτοιου είδους προβλήματα αντιμετωπίζονται με την επανεκτίμηση της συνάρτησης βελτιστοποίησης όλων των σωματιδίων τη χρονική στιγμή $t+1$ και την ενημέρωση των θέσεων ατομικού και καθολικού βέλτιστου.

Η προσαρμοστική ικανότητα του αλγόριθμου PSO να ακολουθεί μεταβαλλόμενα βέλτιστα προϋποθέτει ότι ο αλγόριθμος δεν έχει ήδη συγκλίνει σε μια κατάσταση ισορροπίας. Στην περίπτωση αυτή, οι ταχύτητες των σωματιδίων είναι ίσες με μηδέν. Επομένως, τα σωματίδια θα παραμείνουν στις ίδιες θέσεις ακόμα και όταν το βέλτιστο αλλάξει.

1.7.2. Επίδραση παραμέτρων

Η επίδραση των ακόλουθων παραμέτρων στην προσαρμοστική ικανότητα του αλγόριθμου PSO είναι ιδιαίτερα σημαντική:

- **Μνήμη σωματιδίων.** Κάθε σωματίδιο του σμήνους έχει τη μνήμη της καλύτερης θέσης που έχει επισκεφθεί μέχρι τώρα και η ταχύτητά του τείνει να κατευθυνθεί προς τη θέση αυτή. Επίσης, κατέχει πληροφορία για την καθολική βέλτιστη θέση (ή τοπική βέλτιστη θέση) του σμήνους. Όταν το περιβάλλον του σμήνους μεταβληθεί, οι πληροφορίες αυτές καθίστανται ξεπερασμένες. Εάν μετά από μια τέτοια μεταβολή, τα σωματίδια κάνουν χρήση αυτών των πληροφοριών, θα οδηγηθούν στο παλιό βέλτιστο, το οποίο μπορεί να μην υπάρχει στο νέο περιβάλλον. Μια λύση στο πρόβλημα αυτό είναι η εκ νέου αρχικοποίηση ή επανεκτίμηση των ατομικών βέλτιστων θέσεων των σωματιδίων.

- **Βάρος αδράνειας.** Το βάρος αδράνειας ισορροπεί τις δυνατότητες εξερεύνησης και τοπικής εκμετάλλευσης του σμήνους. Συνήθως, το w μεταβάλλεται από μια μεγάλη αρχική τιμή προς μια μικρή τελική τιμή. Αν τη στιγμή που συμβεί μια μεταβολή, η τιμή του βάρους αδράνειας είναι πολύ μικρή, τότε η δυνατότητα του σμήνους για εξερεύνηση θα είναι πολύ περιορισμένη και δεν θα μπορέσει να ακολουθήσει το νέο βέλτιστο. Ένας τρόπος αντιμετώπισης αυτού του προβλήματος είναι η επαναφορά του w στην αρχική του τιμή όταν εντοπιστεί μια αλλαγή στο περιβάλλον του σμήνους. Ανάλογα ισχύουν και για τους συντελεστές επιτάχυνσης.

- **Αποκοπή ταχύτητας.** Η αποκοπή ταχύτητας έχει μεγάλη επίδραση στην ικανότητα εξερεύνησης του σμήνους. Μεγάλη αποκοπή ταχύτητας (δηλαδή μικρές τιμές του $V_{\max, j}$) οδηγεί σε μικρές μεταβολές της ταχύτητας και επομένως περιορίζει την ικανότητα εξερεύνησης του σμήνους.

- **Μοντέλα ταχύτητας.** Το κοινωνικό μοντέλο (βλέπε εδάφιο 1.5.4) είναι ταχύτερο στην παρακολούθηση αλλαγών από το πλήρες μοντέλο [21]. Όμως, όσο μεγαλύτερη είναι η συχνότητα των μεταβολών, τόσο μικρότερη είναι η αξιοπιστία του. Τα μοντέλα αλτρουιστικό και γνωσιακό δεν έχουν καλή απόδοση σε δυναμικά περιβάλλοντα.

1.7.3. Μέθοδοι αντιμετώπισης

Ανίχνευση μεταβολών περιβάλλοντος. Ο αλγόριθμος PSO, για να μπορεί να παρακολουθεί τις μεταβολές ενός δυναμικού περιβάλλοντος, θα πρέπει να τις ανιχνεύει γρήγορα και αποτελεσματικά. Μια μέθοδος που έχει προταθεί είναι η χρήση ενός ή περισσοτέρων σωματιδίων ανίχνευσης (sentry particles) [25].

Ένα σωματίδιο ανίχνευσης κρατά αποθηκευμένη την πιο πρόσφατη τιμή της συνάρτησης βελτιστοποίησης. Στην αρχή κάθε επανάληψης του αλγόριθμου, η συνάρτηση βελτιστοποίησης υπολογίζεται ξανά και συγκρίνεται με την προηγούμενη τιμή. Αν υπάρχει διαφορά, μια μεταβολή έχει συμβεί.

Όσο μεγαλύτερος είναι ο αριθμός των ανιχνευτών, τόσο ταχύτερη και πιο αξιόπιστη είναι η ανίχνευση των μεταβολών του περιβάλλοντος. Όμως, αυτό οδηγεί και σε αύξηση της υπολογιστικής πολυπλοκότητας του αλγόριθμου αναζήτησης.

Μια εναλλακτική πρόταση είναι η παρακολούθηση των μεταβολών της συνάρτησης βελτιστοποίησης του σωματιδίου καθολικής βέλτιστης θέσης (ή και της δεύτερης καθολικής βέλτιστης θέσης) [23], [26].

Δυναμικό βάρος αδράνειας. Διάφορες μέθοδοι έχουν προταθεί με χρήση δυναμικά μεταβαλλόμενου βάρους αδράνειας [10].

Μια προσέγγιση υιοθετεί την τυχαία μεταβολή του w σε κάθε χρονικό βήμα με χρήση της εξίσωσης (1.12) [11].

Άλλες μέθοδοι χρησιμοποιούν τη γραμμική (εξίσωση (1.15)) ή μη γραμμική (εξισώσεις (1.16)-(1.18)) μείωση του βάρους αδράνειας από μια μεγάλη αρχική τιμή σε μια μικρή τελική τιμή [12], [13], [14], [15].

Εναλλακτικές προσεγγίσεις κάνουν χρήση του χαοτικού βάρους αδράνειας (εξισώσεις (1.23), (1.25)) [27], ή του ελαττούμενου βάρους αδράνειας σιγμοειδούς συνάρτησης (εξίσωση (1.22)) [16].

Επαναρχικοποίηση σμήνους. Μια απλή μέθοδος για την αύξηση της ποικιλομορφίας του σμήνους είναι η επαναρχικοποίησή του, δηλαδή όλα τα σωματίδια αποκτούν νέες τυχαίες θέσεις και οι βέλτιστες θέσεις, ατομικές ή γειτονιάς, επανεκτιμώνται. Έχουν προταθεί οι εξής προσεγγίσεις [11]:

- *Η μη επαναρχικοποίηση του σμήνους*. Η προσέγγιση αυτή δουλεύει μόνο για μικρές μεταβολές και όταν το σμήνος δεν έχει φτάσει σε κατάσταση ισορροπίας.
- *Επαναρχικοποίηση ολόκληρου του σμήνους*. Η προσέγγιση αυτή διαγράφει τη μνήμη του σμήνους και απομακρύνει το σμήνος από το παλιό βέλτιστο. Όμως, έχει το μειονέκτημα ότι η διαδικασία αναζήτησης θα πρέπει να ξεκινήσει από την αρχή. Η επαναρχικοποίηση ολόκληρου του σμήνους είναι πιο αποτελεσματική σε μεγάλες μεταβολές του περιβάλλοντος, ενώ όταν οι μεταβολές είναι μικρές αυξάνει αναίτια την υπολογιστική πολυπλοκότητα και μπορεί να οδηγήσει σε χειρότερες λύσεις.
- *Επαναρχικοποίηση ενός μέρους του σμήνους, με διατήρηση της καθολικής βέλτιστης θέσης*. Η προσέγγιση αυτή αυξάνει την ποικιλομορφία και ταυτόχρονα διατηρεί τη μνήμη της διαδικασίας αναζήτησης. Το ποσοστό επαναρχικοποίησης βρίσκεται σε αναλογία με το εύρος των μεταβολών του περιβάλλοντος [26]. Η μερική επαναρχικοποίηση του σμήνους είναι πιο αποτελεσματική σε σχέση με την ολική.

Επανεκτίμηση βέλτιστων θέσεων. Μια άλλη μέθοδος είναι η επαναφορά των ατομικών βέλτιστων θέσεων των σωματιδίων στις τρέχουσες θέσεις τους όταν ανιχνευτεί μια μεταβολή [21], [23]. Η μέθοδος αυτή διαγράφει τη μνήμη των σωματιδίων και είναι αποτελεσματική όταν το σμήνος δεν έχει φτάσει σε κατάσταση ισορροπίας [26]. Η μέθοδος αυτή μπορεί να συνδυαστεί με τη μερική επαναρχικοποίηση του σμήνους.

Εναλλακτικά, οι ατομικές βέλτιστες θέσεις των σωματιδίων επανεκτιμώνται μετά από μια μεταβολή του περιβάλλοντος και, αν είναι χειρότερες από τις τρέχουσες θέσεις, τότε μόνο λαμβάνουν τις τιμές αυτών [28].

Ανάλογες τεχνικές μπορούν να εφαρμοστούν τόσο στην καθολική βέλτιστη θέση του σμήνους όσο και στις βέλτιστες θέσεις γειτονιάς.

Κεφάλαιο 2: Θεωρία Βέλτιστης Παύσης (Optimal Stopping Theory – OST)

2.1. Εισαγωγή

Η Θεωρία Βέλτιστης Παύσης (Optimal Stopping Theory) μελετάει το πρόβλημα της επιλογής της χρονικής στιγμής εκτέλεσης μιας συγκεκριμένης ενέργειας, βασισμένη σε μια ακολουθία παρατηρούμενων τυχαίων μεταβλητών, προκειμένου να μεγιστοποιηθεί κάποια αναμενόμενη ανταμοιβή ή να ελαχιστοποιηθεί κάποιο αναμενόμενο κόστος. Προβλήματα αυτού του είδους συναντώνται στον τομέα της στατιστικής, όπου η εκτελούμενη ενέργεια μπορεί να αντιστοιχεί στον έλεγχο μιας υπόθεσης ή στην εκτίμηση μιας παραμέτρου, και στον τομέα της επιχειρησιακής έρευνας, όπου η ενέργεια μπορεί να είναι η αντικατάσταση μιας μηχανής, η πρόσληψη μιας γραμματέως, η αγορά μετοχών, κ.λπ. [29].

Ιστορικά, το πρόβλημα προέκυψε στην ακολουθιακή ανάλυση στατιστικών παρατηρήσεων με τη θεωρία του κριτηρίου λόγου ακολουθιακής πιθανότητας (Wald, 1945) [30]. Η γενίκευση της ακολουθιακής ανάλυσης σε προβλήματα παύσης χωρίς στατιστική δομή έγινε από τον Snell το 1952 [31]. Στη δεκαετία του 1960, η έρευνα για το πρόβλημα αυτό γνώρισε μεγάλη ώθηση [32].

2.2. Ορισμός του προβλήματος

Τα προβλήματα βέλτιστης παύσης καθορίζονται από δύο αντικείμενα [29]:

(i) μια ακολουθία τυχαίων μεταβλητών $X_1, X_2, \ldots$, της οποίας η κοινή κατανομή (joint distribution) θεωρείται γνωστή, και

(ii) μια ακολουθία από συναρτήσεις απολαβής, οι οποίες λαμβάνουν πραγματικές τιμές:

$$y_0, y_1(x_1), y_2(x_1, x_2), \ldots, y_\infty(x_1, x_2, \ldots)$$

Με δεδομένα τα δύο αυτά αντικείμενα, το αντίστοιχο πρόβλημα βέλτιστης παύσης διατυπώνεται ως εξής: Παρατηρούμε την ακολουθία $X_1, X_2, \ldots$ για όσο χρόνο θέλουμε. Για κάθε $n = 1, 2, \ldots$, μετά τις παρατηρήσεις $X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n$, μπορούμε να σταματήσουμε και να λάβουμε την προκαθορισμένη ανταμοιβή $y_n(x_1, \ldots, x_n)$, ή να

συνεχίσουμε με την παρατήρηση της X_{n+1}. Αν επιλέξουμε να μην κάνουμε παρατηρήσεις, λαμβάνουμε ως ανταμοιβή τη σταθερή ποσότητα y_0. Αν δεν σταματήσουμε ποτέ τις παρατηρήσεις, λαμβάνουμε την $y_\infty(x_1, x_2, \ldots)$.

Το πρόβλημα είναι να επιλέξουμε μια χρονική στιγμή για να σταματήσουμε, έτσι ώστε να βελτιστοποιήσουμε την αναμενόμενη ανταμοιβή. Για να το πετύχουμε αυτό, μπορούμε να κάνουμε χρήση τυχαίων αποφάσεων, δηλαδή όταν φτάσουμε στο στάδιο n, έχοντας πραγματοποιήσει τις παρατηρήσεις $X_1 = x_1, \ldots, X_n = x_n$, μπορούμε να επιλέξουμε μια πιθανότητα παύσης που να εξαρτάται από τις παρατηρήσεις αυτές. Συμβολίζουμε την πιθανότητα αυτή με $\phi_n(x_1, \ldots, x_n)$. Ένας κανόνας παύσης τυχαίας κατανομής (randomized) αποτελείται από την ακολουθία αυτών των συναρτήσεων

$$\boldsymbol{\phi} = (\phi_0, \phi_1(x_1), \phi_2(x_1, x_2), \ldots) \tag{2.1}$$

όπου $0 \le \phi_n(x_1, \ldots, x_n) \le 1$ για όλα τα n και $x_1, \ldots, x_n$. Ο κανόνας παύσης δεν ακολουθεί τυχαία κατανομή (non-randomized) αν κάθε συνάρτηση $\phi_n(x_1, \ldots, x_n)$ είναι 0 ή 1.

Έτσι, η συνάρτηση ϕ_0 αντιπροσωπεύει την πιθανότητα να μην κάνουμε παρατηρήσεις, ενώ η $\phi_1(x_1)$ να σταματήσουμε μετά την πρώτη παρατήρηση κ.λπ.. Ο κανόνας παύσης $\boldsymbol{\phi}$ και η ακολουθία των παρατηρήσεων $\boldsymbol{X} = \boldsymbol{x} = (x_1, x_2, \ldots)$ καθορίζουν την τυχαία χρονική στιγμή N στην οποία γίνεται η παύση, όπου $0 \le N \le \infty$. Στην περίπτωση που η παύση δεν γίνει ποτέ, $N = \infty$. Η αθροιστική συνάρτηση κατανομής του N για $\boldsymbol{X} = \boldsymbol{x} = (x_1, x_2, \ldots)$ συμβολίζεται με $\boldsymbol{\psi} = (\psi_0, \psi_1, \psi_2, \ldots, \psi_\infty)$, όπου

$$\begin{aligned} \psi_n(x_1, \ldots, x_n) &= P(N = n \mid \boldsymbol{X} = \boldsymbol{x}), \qquad n = 0, 1, 2, \ldots \\ \psi_\infty(x_1, x_2, \ldots) &= P(N = \infty \mid \boldsymbol{X} = \boldsymbol{x}) \end{aligned} \tag{2.2}$$

Η αθροιστική συνάρτηση κατανομής $\boldsymbol{\psi}$ και ο κανόνας παύσης $\boldsymbol{\phi}$ συνδέονται ως εξής:

$$
\begin{aligned}
\psi_0 &= \phi_0 \\
\psi_1(x_1) &= (1-\phi_0)\phi_1(x_1) \\
&\vdots \\
\psi_n(x_1,\ldots,x_n) &= \left[(1-\phi_0)\prod_{j=1}^{n-1}\left(1-\phi_j\left(x_1,\ldots,x_j\right)\right)\right]\phi_n\left(x_1,\ldots,x_n\right) \\
&\vdots \\
\psi_\infty(x_1,x_2,\ldots) &= 1-\sum_{j=0}^{\infty}\psi_j\left(x_1,\ldots,x_j\right)
\end{aligned}
\tag{2.3}
$$

όπου η ποσότητα $\psi_\infty(x_1,x_2,\ldots)$ αντιπροσωπεύει την πιθανότητα η παύση να μην γίνει ποτέ, με δεδομένες όλες τις παρατηρήσεις.

Το πρόβλημα λοιπόν είναι η επιλογή ενός κανόνα παύσης $\boldsymbol{\phi}$ έτσι ώστε να μεγιστοποιείται η μέση αναμενόμενη ανταμοιβή $V(\boldsymbol{\phi})$, η οποία ορίζεται ως

$$
\begin{aligned}
V(\boldsymbol{\phi}) &= E\left[y_N\left(X_1,\ldots,X_N\right)\right] \\
&= E\left[\sum_{j=0}^{=\infty}\psi_j\left(X_1,\ldots,X_j\right)y_j\left(X_1,\ldots,X_j\right)\right]
\end{aligned}
\tag{2.4}
$$

όπου ο συμβολισμός "$=\infty$" δηλώνει ότι η άθροιση είναι για τις τιμές του j από 0 μέχρι ∞, συμπεριλαμβανομένου του ∞. Ο κανόνας παύσης $\boldsymbol{\phi}$ μπορεί να εκφραστεί σε σχέση με τον τυχαίο χρόνο παύσης N ως εξής:

$$
\phi_n\left(X_1,\ldots,X_n\right) = P\left(N=n \mid N\geq n, \boldsymbol{X}=\boldsymbol{x}\right), \qquad n=0,1,2,\ldots \tag{2.5}
$$

Παρατηρήσεις

- Συχνά, η δομή του προβλήματος υπαγορεύει να θεωρήσουμε απώλεια ή κόστος αντί για ανταμοιβή. Στην περίπτωση αυτή, η συνάρτηση y_n δηλώνει το αναμενόμενο κόστος σταματώντας τη χρονική στιγμή n, και το πρόβλημα μετατρέπεται στην επιλογή ενός κανόνα παύσης για την ελαχιστοποίηση της $V(\boldsymbol{\phi})$.
- Σε μερικές εφαρμογές, η ακολουθία ανταμοιβής περιγράφεται πιο ρεαλιστικά ως μια ακολουθία τυχαίων μεταβλητών $Y_0, Y_1, \ldots, Y_\infty$, της οποίας η κοινή συνάρτηση

κατανομής με τις παρατηρήσεις $X_1, X_2, \ldots$ είναι γνωστή. Έτσι, η πραγματική τιμή της Y_n δεν είναι επακριβώς καθορισμένη τη χρονική στιγμή n όταν λαμβάνεται η απόφαση για τη συνέχιση ή την παύση των παρατηρήσεων.

2.3. Προβλήματα πεπερασμένου ορίζοντα

Ένα πρόβλημα κανόνα παύσης έχει πεπερασμένο ορίζοντα αν υπάρχει ένα γνωστό άνω όριο στον αριθμό των σταδίων που μπορεί κανείς να σταματήσει. Λέμε ότι ένα πρόβλημα έχει ορίζοντα T αν η παύση είναι υποχρεωτική μετά από τις παρατηρήσεις $X_1, \ldots, X_T$. Ένα πρόβλημα πεπερασμένου ορίζοντα μπορεί να θεωρηθεί ως ειδική περίπτωση του γενικού προβλήματος βέλτιστης παύσης θέτοντας $y_{T+1} = \cdots = y_\infty = -\infty$ [29].

Γενικά, ένα τέτοιο πρόβλημα μπορεί να επιλυθεί με τη μέθοδο της ανάδρομης επαγωγής (backward induction). Δεδομένου ότι πρέπει να σταματήσουμε στο στάδιο T, βρίσκουμε αρχικά το βέλτιστο κανόνα στο στάδιο $T-1$. Στη συνέχεια, γνωρίζοντας το βέλτιστο κανόνα στο στάδιο $T-1$, υπολογίζουμε το βέλτιστο κανόνα στο στάδιο $T-2$, και συνεχίζουμε έτσι μέχρι το αρχικό στάδιο (στάδιο 0). Ορίζουμε την ποσότητα $V_T^{(T)}(x_1, \ldots, x_T) = y_T(x_1, \ldots, x_T)$. Στη συνέχεια, με επαγωγή από την τιμή $j = T-1$ μέχρι την τιμή $j = 0$ λαμβάνουμε:

$$V_j^{(T)}(x_1, \ldots, x_j) = \max\left\{ y_j(x_1, \ldots, x_j), E\left[V_{j+1}^{(T)}(x_1, \ldots, x_j, X_{j+1}) \mid X_1 = x_1, \ldots, X_j = x_j \right] \right\} \quad (2.6)$$

Επαγωγικά, η ποσότητα $V_j^{(T)}(x_1, \ldots, x_j)$ αντιπροσωπεύει τη μέγιστη απολαβή που μπορεί να αποκομίσει κανείς ξεκινώντας από το στάδιο j και έχοντας κάνει τις παρατηρήσεις $X_1 = x_1, \ldots, X_j = x_j$. Στο στάδιο j, συγκρίνουμε την απολαβή για παύση, $y_j(x_1, \ldots, x_j)$, με την απολαβή που αναμένουμε να αποκομίσουμε συνεχίζοντας και χρησιμοποιώντας το βέλτιστο κανόνα για τα στάδια $j+1, j+2, \ldots, T$, η οποία είναι ίση με $E\left[V_{j+1}^{(T)}(x_1, \ldots, x_j, X_{j+1}) \mid X_1 = x_1, \ldots, X_j = x_j \right]$. Επομένως, η βέλτιστη ανταμοιβή μας είναι η μεγαλύτερη από τις δύο αυτές ποσότητες, και είναι βέλτιστο να σταματήσουμε στο στάδιο j αν ισχύει $V_j^{(T)}(x_1, \ldots, x_j) = y_j(x_1, \ldots, x_j)$, ενώ συμφέρει να συνεχίσουμε στην αντίθετη περίπτωση. Τότε, η αξία του προβλήματος κανόνα παύσης θα είναι $V_0^{(T)}$.

Υπάρχουν διάφορα προβλήματα που μπορούν να επιλυθούν αποτελεσματικά με τη χρήση της μεθόδου ανάδρομης επαγωγής. Το πιο διάσημο από αυτά είναι το πρόβλημα της γραμματέως. Άλλα προβλήματα πεπερασμένου ορίζοντα είναι το πρόβλημα Cayley-Moser και το πρόβλημα παρκαρίσματος των MacQueen-Miller.

2.3.1. Το Πρόβλημα της Γραμματέως (Secretary Problem)

Το Πρόβλημα της Γραμματέως και οι παραλλαγές του αποτελούν μια σημαντική κατηγορία προβλημάτων πεπερασμένου ορίζοντα. Το πρόβλημα συνήθως περιγράφεται ως αυτό της επιλογής της καλύτερης γραμματέως ή του καλύτερου γραμματέα (το Πρόβλημα της Γραμματέως), αλλά μερικές φορές περιγράφεται ως το πρόβλημα επιλογής της καλύτερης συζύγου (το Πρόβλημα του Γάμου) ή του μεγαλύτερου αριθμού από έναν άγνωστο σύνολο αριθμών (Πρόβλημα Googol).

Το κλασικό Πρόβλημα της Γραμματέως περιγράφεται ως εξής [29]:

1. Υπάρχει μόνο μία θέση γραμματέα διαθέσιμη.
2. Υπάρχουν n *αιτούντες* (*applicants*) για τη θέση αυτή, όπου n γνωστός αριθμός.
3. Θεωρούμε ότι είναι δυνατή η γραμμική ταξινόμηση των αιτούντων από τον καλύτερο στον χειρότερο χωρίς ισοβαθμίες.
4. Οι αιτούντες περνούν διαδοχικά από συνέντευξη με τυχαία διάταξη και οι $n!$ δυνατές διατάξεις είναι ισοπίθανες.
5. Κατά τη διάρκεια της συνέντευξης, ο αιτών, είτε γίνεται δεκτός για τη θέση και το πρόβλημα επιλογής τερματίζεται, είτε απορρίπτεται και η διαδικασία συνεχίζεται με τον επόμενο, αν υπάρχει.
6. Η απόφαση για την αποδοχή ή όχι ενός αιτούντος βασίζεται μόνο στη σχετική διάταξη των αιτούντων που έχουν περάσει από συνέντευξη μέχρι εκείνη τη στιγμή.
7. Ένας υποψήφιος που έχει απορριφθεί δεν μπορεί να ξανακληθεί.
8. Ο στόχος είναι η επιλογή του καλύτερου αιτούντος. Αν επιλεγεί ο καλύτερος, η απολαβή είναι 1, διαφορετικά είναι 0.

Το παραπάνω πρόβλημα μπορεί να αντιμετωπιστεί ως ένα πρόβλημα κανόνα παύσης, αντιστοιχώντας την επιλογή ενός από τους αιτούντες με τη παύση σε μια ακολουθία παρατηρήσεων. Επίσης, οι παρατηρήσεις ενός προβλήματος βέλτιστης παύσης μπορούν να

αντιστοιχηθούν στις σχετικές διατάξεις $X_1, X_2, \ldots, X_n$, όπου X_j είναι η διάταξη του αιτούντος j ανάμεσα στους πρώτους j αιτούντες, με τη διάταξη 1 να θεωρείται η καλύτερη. Με βάση την παραδοχή 4, αυτές οι τυχαίες μεταβλητές είναι ανεξάρτητες και η διάταξη X_j ακολουθεί ομοιόμορφη κατανομή στο ακέραιο διάστημα από 1 έως j. Επομένως, $X_1 \equiv 1$, $P(X_2 = 1) = P(X_2 = 2) = 1/2$, κ.λπ..

Ένας αιτών θα πρέπει να επιλέγεται μόνο εάν είναι ο σχετικά καλύτερος από αυτούς που έχουν ήδη εξεταστεί. Ο σχετικά καλύτερος αιτών καλείται *υποψήφιος* (*candidate*). Έτσι, ο αιτών j είναι υποψήφιος αν και μόνο αν $X_j = 1$. Αν ο υποψήφιος επιλεγεί στο στάδιο j, η πιθανότητα βέλτιστης επιλογής είναι ίση με την πιθανότητα ο καλύτερος ανάμεσα στους πρώτους j αιτούντες να είναι ο καλύτερος όλων. Η πιθανότητα ο βέλτιστος υποψήφιος να βρίσκεται ανάμεσα στους πρώτους j αιτούντες είναι j/n. Έτσι,

$$y_j(x_1, \ldots, x_j) = \begin{cases} j/n, & \text{if applicant } j \text{ is a candidate} \\ 0, & \text{otherwise} \end{cases} \qquad (2.7)$$

Παρατηρούμε ότι $y_0 = 0$ και η ανταμοιβή y_j εξαρτάται μόνο από τις τιμές x_j για $j \geq 1$.

Το βασικό πρόβλημα έχει μια απλή λύση, η οποία μπορεί να υπολογιστεί χωρίς να κάνουμε χρήση της (2.6). Έστω W_j η πιθανότητα της βέλτιστης επιλογής κάνοντας χρήση ενός βέλτιστου κανόνα μεταξύ των πρώτων j αιτούντων. Τότε ισχύει: $W_j \geq W_{j+1}$, γιατί ο βέλτιστος κανόνας μεταξύ των πρώτων $j+1$ αιτούντων βρίσκεται μεταξύ των κανόνων που ενεργούν στους πρώτους j αιτούντες. Η παύση σε έναν υποψήφιο στο στάδιο j είναι βέλτιστη αν ισχύει $j/n \geq W_j$. Αν είναι βέλτιστη η παύση σε έναν υποψήφιο στο στάδιο j, τότε είναι βέλτιστη και στο στάδιο $j+1$, καθώς ισχύει: $(j+1)/n \geq j/n \geq W_j \geq W_{j+1}$. Επομένως, ένας βέλτιστος κανόνας μπορεί να βρεθεί μεταξύ κανόνων της ακόλουθης μορφής N_r, όπου $r \geq 1$:

N_r: Απόρριψε τους πρώτους $r-1$ αιτούντες και στη συνέχεια διάλεξε τον σχετικά καλύτερο από τους επόμενους αιτούντες, αν υπάρχει.

Ένας τέτοιος κανόνας καλείται *κανόνας κατωφλίου* (*threshold rule*) με κατώφλι r. Η πιθανότητα βέλτιστης επιλογής χρησιμοποιώντας τον κανόνα N_r είναι:

$$\begin{aligned} P_r &= \sum_{k=r}^{n} P\left(\text{ο αιτών } k \text{ είναι ο βέλτιστος και έχει επιλεγεί}\right) \\ &= \frac{r-1}{n}\sum_{k=r}^{n}\frac{1}{k-1} \end{aligned} \tag{2.8}$$

Έστω r_1 είναι η τιμή του r που μεγιστοποιεί την πιθανότητα P_r. Ισχύει:

$$P_{r+1} \le P_r \Leftrightarrow \frac{r}{n}\sum_{r+1}^{n}\frac{1}{k-1} \le \frac{r-1}{n}\sum_{r}^{n}\frac{1}{k-1} \Leftrightarrow \sum_{r+1}^{n}\frac{1}{k-1} \le 1$$

Επομένως, ο βέλτιστος κανόνας είναι η επιλογή του πρώτου υποψηφίου που εμφανίζεται μεταξύ των αιτούντων από το στάδιο r_1 και μετά, όπου

$$r_1 = \min\left\{ r \ge 1 :\ \sum_{r+1}^{n}\frac{1}{k-1} \le 1 \right\} \tag{2.9}$$

Ο επόμενος πίνακας είναι ενδεικτικός για τις τιμές που λαμβάνουν τα r_1, P_{r_1}.

Πίνακας 1: Πρόβλημα Γραμματέως. Ενδεικτικές τιμές παραμέτρων.

n	1	2	3	4	5	6	7	8
r_1	1	1	2	2	3	3	3	4
P_{r_1}	1.0	0.500	0.500	0.458	0.433	0.428	0.414	0.410

Για μεγάλες τιμές του n ισχύει: $\sum_{r+1}^{n} 1/(k-1) \approx \log\left(n/r\right)$. Επομένως, κατά προσέγγιση είναι $\log\left(n/r_1\right) = 1$ ή $r_1/n = e^{-1}$. Άρα, για μεγάλες τιμές του n είναι κατά προσέγγιση βέλτιστη η απόρριψη ποσοστού $e^{-1} = 36.8\%$ των αιτούντων και στη συνέχεια η επιλογή του σχετικά καλύτερου υποψηφίου. Η πιθανότητα βέλτιστης επιλογής είναι κατά προσέγγιση e^{-1}.

2.3.2. Παραλλαγές του Προβλήματος της Γραμματέως

Οι παραλλαγές του Προβλήματος της Γραμματέως μπορούν να ταξινομηθούν σε τρεις γενικές κατηγορίες [29]:

- Στα *προβλήματα μη ύπαρξης πληροφορίας* (*no-information problems*), στα οποία η κατανομή των παρατηρήσεων $X_1, X_2, \ldots$ είναι τελείως άγνωστη.

  Το κλασικό Πρόβλημα της Γραμματέως περιλαμβάνεται στην κατηγορία αυτή. Μια γνωστή παραλλαγή του αποτελεί το παιχνίδι με το όνομα "Googol". Σε αυτό, ένας παίκτης Ι επιλέγει n αριθμούς $X_1, \ldots, X_n$, τους αναγράφει σε κομμάτια χαρτιού και τα τοποθετεί σε ένα καπέλο. Στη συνέχεια, ένας παίκτης ΙΙ, μη γνωρίζοντας τους αριθμούς, τραβά τα χαρτάκια από το καπέλο, ένα τη φορά. Ο παίκτης ΙΙ μπορεί να σταματήσει οποιαδήποτε χρονική στιγμή και κερδίζει όταν ο τελευταίος αριθμός που έχει τραβήξει είναι ο μεγαλύτερος από τους n αριθμούς. Ο παίκτης ΙΙ έχει πιθανότητα επιτυχίας τουλάχιστον P_{r_1} αν χρησιμοποιήσει τη σχετική διάταξη των αριθμών που τραβάει και τον κανόνα βέλτιστης παύσης του κλασικού Προβλήματος της Γραμματέως [33].

- Στα *προβλήματα πλήρους πληροφορίας* (*full-information problems*), στα οποία οι παρατηρήσεις $X_1, X_2, \ldots$ είναι ανεξάρτητες και ταυτόσημα κατανεμημένες (i.i.d.) τυχαίες μεταβλητές με γνωστή κατανομή.

  Ένα τέτοιο πρόβλημα είναι όταν έχουμε μια ακολουθία από ανεξάρτητες και ταυτόσημα κατανεμημένες συνεχείς τυχαίες μεταβλητές $X_1, \ldots, X_n$, οι οποίες παρουσιάζονται μία κάθε χρονική στιγμή. Η διαδικασία τερματίζεται με την επιλογή ενός εμφανιζόμενου αριθμού. Η ανταμοιβή αποτελεί συνάρτηση της αληθινής διάταξης του επιλεγόμενου αριθμού και είναι 1 αν επιλεγεί ο μεγαλύτερος αριθμός και 0 διαφορετικά. Όταν η κατανομή των X_i είναι πλήρως γνωστή, η πιθανότητα επιλογής του βέλτιστου συγκλίνει στην τιμή $v^* = 0.58016$ όταν $n \to \infty$.

- Στα *προβλήματα μερικής πληροφορίας* (*partial information problems*), όπου κάποια μορφή πληροφορίας για τα X_i είναι διαθέσιμη.

  Μια τέτοια περίπτωση είναι όταν γνωρίζουμε ότι τα X_i ακολουθούν την κανονική κατανομή με άγνωστη μέση τιμή. Τότε, ασυμπτωτικά η πιθανότητα επιλογής του βέλτιστου είναι και πάλι v^*. Επίσης, στην περίπτωση που η κατανομή είναι

ομοιόμορφη στο διάστημα (a,b) με άγνωστα τα a, b, τότε ο κανόνας τερματισμού βασίζεται μόνο στη σχετική διάταξη των παρατηρήσεων X_i και η οριακή πιθανότητα επιτυχίας είναι e^{-1}.

Μια άλλη παραλλαγή είναι όταν παραβιάζεται η συνθήκη 2 του κλασικού Προβλήματος της Γραμματέως και ο αριθμός των παρατηρήσεων είναι άγνωστος. Στην περίπτωση αυτή, θεωρούμε ότι ο αριθμός των παρατηρήσεων είναι τυχαίος σύμφωνα με κάποια γνωστή κατανομή.

Μια ακόμη παραλλαγή επιτρέπει την επιλογή ενός αντικειμένου που έχει ήδη περάσει. Αυτό καλείται *αναδρομικό αίτημα* (*backward solicitation*). Εδώ, η πιθανότητα επιτυχίας εξαρτάται από τη χρονική στιγμή παρατήρησης του αντικειμένου αυτού.

2.3.3. Μια νέα προσέγγιση του Προβλήματος της Γραμματέως

Η προσέγγιση αυτή αποτελεί επέκταση του κλασικού Προβλήματος της Γραμματέως, στην οποία η ανταμοιβή δεν είναι πλέον 1 στην περίπτωση εύρεσης του βέλτιστου υποψηφίου και 0 διαφορετικά, αλλά ισούται με την «πραγματική» τιμή του υποψηφίου [34], [35]. Το κίνητρο για τη θεμελίωση αυτής της προσέγγισης είναι ότι, σε μερικές περιπτώσεις ακολουθιακής αναζήτησης, ο κριτής μπορεί να εξετάζει την πληροφορία διάταξης, αλλά κάνει χρήση της πραγματικής τιμής της επιλεγμένης παρατήρησης και όχι της διάταξής της. Π.χ. ένας χρηματιστής που θέλει να πουλήσει μετοχές στη μέγιστη τιμή τους σε κάποιο καθορισμένο χρονικό διάστημα. Το πρόβλημα διατυπώνεται ως εξής:

Έστω ότι ένας κριτής παρατηρεί μια ακολουθία από n αιτούντες, των οποίων οι τιμές είναι ανεξάρτητες και ταυτόσημα κατανεμημένες τυχαίες μεταβλητές $X_1,\ldots,X_n$ με ομοιόμορφη κατανομή στο διάστημα $[0,1]$. Όπως και στο κλασικό Πρόβλημα της Γραμματέως, ο κριτής έχει δύο επιλογές για κάθε αιτούντα: έγκριση ή απόρριψη. Η ανταμοιβή του κριτή για την επιλογή ενός αιτούντα με $X_t = x_t$ είναι x_t. Ένας υποψήφιος που απορρίπτεται δεν μπορεί να επανακληθεί. Η επιλογή ενός αιτούντα οδηγεί στον τερματισμό του προβλήματος. Αν έρθει η σειρά του, ο τελευταίος υποψήφιος επιλέγεται υποχρεωτικά. Σε κάθε στάδιο t ο κριτής παρατηρεί μόνο ένα δείκτη I_t του X_t, όπου $I_t = 1$ αν και μόνο αν $x_t = \max\{x_1, x_2, \ldots, x_t\}$, διαφορετικά $I_t = 0$. Δηλαδή, ο κριτής γνωρίζει αν ο παρατηρούμενος αιτών είναι ο καλύτερος μέχρι τώρα.

Η αναμενόμενη τιμή για τον t αιτούντα, δεδομένου ότι ισχύει $x_t = \max\{x_1, x_2, \ldots, x_t\}$, δίνεται από τη σχέση:

$$E_t = E\left[X_t \mid I_t = 1\right] = \frac{t}{t+1} \tag{2.10}$$

Έστω c_n^* το μικρότερο t για το οποίο είναι βέλτιστο να επιλέξουμε έναν αιτούντα με $I_t = 1$, για ένα πρόβλημα n αιτούντων. Ο αριθμός c_n^* καλείται *βέλτιστη αποκοπή* (*optimal cutoff*). Επίσης, συμβολίζουμε με $V_n(c)$ την αναμενόμενη απολαβή για τον επιλεγμένο αιτούντα, όταν εφαρμόσουμε αποκοπή $1 \le c \le n$ από το στάδιο c στο στάδιο n. Τότε ισχύει:

$$c_n^* = \arg \max_{c \in \{1,2,\ldots,n\}} V_n(c) \tag{2.11}$$

Η τιμή του βέλτιστου κανόνα για πρόβλημα μεγέθους n είναι $V_n^* \equiv V_n(c_n^*)$.

Αποδεικνύεται ότι [34], [35]:

$$c_n^* \in \left\{ \left\lfloor n^{\frac{1}{2}} \right\rfloor, \left\lceil n^{\frac{1}{2}} \right\rceil \right\} \tag{2.12}$$

Για τιμές του n που είναι τέλεια τετράγωνα $c_n^* = n^{\frac{1}{2}}$. Επίσης, το V_n^* αυξάνεται με το n και

$$\lim_{n \to \infty} V_n^* = 1 \tag{2.13}$$

Κάποιες ενδεικτικές τιμές των c_n^*, V_n^* παρουσιάζονται στον Πίνακα 2.

Πίνακας 2: Νέα προσέγγιση Προβλήματος Γραμματέως. Ενδεικτικές τιμές παραμέτρων.

n	10	100	500	1000	5000	10000	50000	100000
c_n^*	3	10	22	32	71	100	224	316
$n^{\frac{1}{2}}$	3.16	10.00	22.36	31.62	70.71	100.00	223.61	316.23
V_n^*	0.7333	0.9050	0.9563	0.9689	0.9860	0.9901	0.9955	0.9968

Συμπερασματικά, η λύση που προτείνει η προσέγγιση αυτή είναι: Απόρριψε τους πρώτους $c_n^* - 1$ αιτούντες και στη συνέχεια επίλεξε τον επόμενο αιτούντα με διάταξη 1, όπου $c_n^* = n^{\frac{1}{2}}$.

2.3.4. Το Πρόβλημα του Παρκαρίσματος (Parking Problem)

Το πρόβλημα διατυπώνεται ως εξής: Οδηγούμε σε ένα δρόμο άπειρου μήκους προς τον προορισμό μας, π.χ. το θέατρο. Κατά μήκος του δρόμου υπάρχουν θέσεις παρκαρίσματος, αλλά οι περισσότερες από αυτές είναι κατειλημμένες. Θέλουμε να παρκάρουμε όσο το δυνατόν πιο κοντά στο θέατρο, χωρίς να γυρίσουμε πίσω. Αν εντοπίσουμε μια θέση παρκαρίσματος σε απόσταση d από το θέατρο, είναι σκόπιμο να την εκμεταλλευτούμε;

Η μοντελοποίηση του προβλήματος γίνεται σε διακριτό πλαίσιο. Υποθέτουμε ότι ξεκινάμε από την αρχή και ότι οι θέσεις παρκαρίσματος βρίσκονται σε διακριτά σημεία της πραγματικής ευθείας. Έστω $X_0, X_1, X_2, \ldots$ ανεξάρτητες τυχαίες μεταβλητές με κοινή κατανομή Bernoulli και κοινή πιθανότητα επιτυχίας p, όπου $X_j = 1$ σημαίνει ότι η θέση παρκαρίσματος j είναι κατειλημμένη και $X_j = 0$ σημαίνει ότι είναι κενή. Έστω $T > 0$ η θέση στην οποία στοχεύουμε. Αν σταματήσουμε στη θέση j, με $X_j = 0$, έχουμε απώλεια ίση με $|T - j|$. Δεν μπορούμε να δούμε τη θέση παρκαρίσματος $j+1$ όταν βρισκόμαστε στη j και δεν μπορούμε να επιστρέψουμε σε μια θέση που έχουμε προσπεράσει. Αν φτάσουμε στο στόχο T, θα πρέπει να επιλέξουμε την επόμενη διαθέσιμη θέση. Αν ο στόχος T είναι κατειλημμένος, η αναμενόμενη απώλεια είναι ίση με

$$(1-p) + 2p(1-p) + 3p^2(1-p) + \cdots = \frac{1}{1-p}$$

Επομένως, μπορούμε να θεωρήσουμε το πρόβλημα αυτό ως πρόβλημα κανόνα παύσης με πεπερασμένο ορίζοντα T και απώλεια

$$y_T = \begin{cases} 0, & X_T = 0 \\ \dfrac{1}{1-p}, & X_T = 1 \end{cases}$$

και για $j = 0, \ldots, T-1$ ισχύει:

$$y_j = \begin{cases} T - j, & X_j = 0 \\ \infty, & X_j = 1 \end{cases}$$

Η τιμή $y_j = \infty$ μας υποχρεώνει να συνεχίσουμε αν φτάσουμε σε μια θέση παρκαρίσματος j και αυτή είναι κατειλημμένη.

Αναζητάμε έναν κανόνα παύσης, $N \le T$, έτσι ώστε η αναμενόμενη τιμή $E[Y_N]$ να ελαχιστοποιείται. Αποδεικνύεται ότι υπάρχει ο εξής βέλτιστος κανόνας N_r, όπου $r \ge 0$ [29]:

N_r: Συνεχίζουμε μέχρι r θέσεις από τον προορισμό μας και παρκάρουμε στην πρώτη διαθέσιμη θέση από εκεί και πέρα.

Έστω P_r το αναμενόμενο κόστος που έχουμε κάνοντας χρήση του παραπάνω κανόνα. Τότε, ισχύουν:

$$\begin{aligned} P_0 &= \frac{p}{1-p} \\ P_r &= r + 1 + \frac{2p^{r+1} - 1}{1-p}, \quad r \ge 1 \end{aligned} \tag{2.14}$$

Η τιμή του r που ελαχιστοποιεί το αναμενόμενο κόστος P_r είναι:

$$r_1 = \min\left\{ r \ge 0 : \; p^{r+1} \le \frac{1}{2} \right\} \tag{2.15}$$

Για παράδειγμα, αν $p \le 1/2$, θα φτάσουμε στον προορισμό μας πριν κοιτάξουμε για θέση παρκαρίσματος. Όμως, αν $p = 0.9$, θα πρέπει να κοιτάμε για να παρκάρουμε $r = 6$ θέσεις πριν από τον προορισμό μας.

Υπάρχουν πολλές παραλλαγές αυτού του προβλήματος. Σε μερικές από αυτές, το κόστος χρόνου ή βενζίνης συμπεριλαμβάνεται κατά τον υπολογισμό του αναμενόμενου κόστους. Σε κάποιες άλλες, υπάρχει δυνατότητα αναστροφής με κάποιο επιπλέον κόστος.

Κεφάλαιο 3: Πληροφορία Πλαισίου

3.1. Πληροφορία Πλαισίου

3.1.1. Ορισμός και περιγραφή

Γενικά το «πλαίσιο» (“context”) ορίζεται ως το «σύνολο των περιστάσεων ή δεδομένων που περιβάλλουν ένα συγκεκριμένο γεγονός ή κατάσταση» [36]. Η έννοια αυτή σχετίζεται με την τρέχουσα / περιρρέουσα κατάσταση που γίνεται αντιληπτή από τον άνθρωπο λόγω των παρατηρούμενων συμβάντων. Η λέξη context προέρχεται από τη λατινική λέξη *contextus*, που σημαίνει *σύνθεση, ενοποίηση*. Συνεπώς, ένα πλαίσιο κατάστασης αποτελείται από τη σύνθεση των παρατηρούμενων συμβάντων.

Στην επιστήμη της Πληροφορικής η έννοια της «πληροφορίας πλαισίου» (“contextual information”) αποτελεί αντικείμενο εκτεταμένης έρευνας κυρίως στα πλαίσια ανάπτυξης εφαρμογών «κινητού υπολογισμού» (“mobile computing”) και γενικότερα δικτυακών εφαρμογών. Συγκεκριμένα, στην Πληροφορική, οι έννοιες «πληροφορία πλαισίου» και «πλαίσιο» είναι ισοδύναμες, με την προϋπόθεση ότι το ενδιαφέρον εστιάζεται στην πληροφορία που περιγράφει την τρέχουσα κατάσταση. Παράλληλα εμφανίζεται και η ανάγκη για τον κατάλληλο προσδιορισμό του πλαισίου σε σχέση με τις ανάγκες κάθε εφαρμογής [37].

Για την έννοια του «πλαισίου» έχουν δοθεί διάφοροι ορισμοί. Σύμφωνα με τους Schilit και Theimer το πλαίσιο είναι η θέση μιας οντότητας, οι ταυτότητες των γειτονικών της αντικειμένων ή προσώπων και οι μεταβολές τους [38]. Παραπλήσια, αλλού το πλαίσιο ορίζεται ως η θέση ενός χρήστη, οι ταυτότητες των ανθρώπων γύρω από αυτόν, η ώρα της ημέρας, η εποχή, η θερμοκρασία κ.λπ. [39]. Τέτοιου είδους ορισμοί που χρησιμοποιούν παραδείγματα για τον προσδιορισμό του πλαισίου είναι δύσκολο να εφαρμοστούν στην πράξη.

Άλλοι ορισμοί παρέχουν συνώνυμα για το πλαίσιο, π.χ. αναφέρονται στο πλαίσιο ως περιβάλλον ή κατάσταση. Μερικοί από αυτούς θεωρούν ως πλαίσιο τα στοιχεία του περιβάλλοντος του χρήστη [40], ενώ άλλοι το περιβάλλον μιας εφαρμογής [41]. Η πρακτική εφαρμογή των ορισμών αυτών είναι επίσης δύσκολη.

Ένας πιο ολοκληρωμένος και εύχρηστος ορισμός είναι ο ακόλουθος: «*Πλαίσιο είναι κάθε πληροφορία που μπορεί να χρησιμοποιηθεί για το χαρακτηρισμό της κατάστασης μιας*

οντότητας. Μια οντότητα είναι ένας άνθρωπος, τόπος ή αντικείμενο που μπορεί να συσχετιστεί με την αλληλεπίδραση μεταξύ ενός χρήστη και μιας εφαρμογής, του χρήστη και της εφαρμογής συμπεριλαμβανομένων.» [42]. Ο ορισμός αυτός διευκολύνει τον καθορισμό του πλαισίου μιας εφαρμογής κατά την ανάπτυξή της.

Γενικότερα, μπορούμε να θεωρήσουμε ως «πλαίσιο» το σύνολο των συστατικών του περιβάλλοντός μας. Έτσι, ο όρος αυτός χρησιμοποιείται κυρίως σε σχέση με το φυσικό κόσμο που περιβάλλει μια κινητή συσκευή, μια εφαρμογή ή ένα ολόκληρο σύστημα. Αρχικά, η έννοια του πλαισίου βασιζόταν σε τρία στοιχεία: τη θέση του χρήστη, τις ταυτότητες των χρηστών που βρίσκονται γύρω του καθώς και στη γνώση των γειτονικών πηγών πληροφορίας του και των γειτονικών του αντικειμένων. Αν και η πληροφορία θέσης είναι βασικό συστατικό για την περιγραφή της περιρρέουσας κατάστασης του χρήστη, δεν μπορεί να εντοπίσει τυχόν αλλαγές του περιβάλλοντος του χρήστη ή δυναμικές ανακατατάξεις των γειτονικών του αντικειμένων. Επομένως, η έννοια του πλαισίου επεκτείνεται σε μια πιο γενική θεώρηση της πληροφορίας που μπορεί να περιγράψει την κατάσταση του χρήστη, η οποία συμπεριλαμβάνει εκτός από την πληροφορία θέσης και πληροφορίες όπως, για παράδειγμα, επίπεδο φωτεινότητας, επίπεδο θορύβου, διαθεσιμότητα πρόσβασης σε δίκτυο και κοινωνικά γεγονότα.

Παρ' όλα αυτά, διάφορα είδη πλαισίου μπορούν να προσδιορισθούν, όπως για παράδειγμα οι «δραστηριότητες του χρήστη» ("activity context"), το «πλαίσιο ενός συστήματος» ("system context") με το οποίο αλληλεπιδρά ο χρήστης, το «τηλεπικοινωνιακό πλαίσιο» ("network context") κ.λπ.. Ένα ποσοστό αοριστίας υπεισέρχεται όταν ο όρος πλαίσιο χρησιμοποιείται σε διαφορετικά επίπεδα αφαίρεσης και ειδών πλαισίου. Το πλαίσιο π.χ. μπορεί να αναφέρεται:

- στην πραγματική κατάσταση του γύρω κόσμου που περιβάλλει μια συσκευή / σύστημα, ή,
- σε μια οπτική γωνία από την οποία αντιλαμβανόμαστε ή προσδιορίζουμε μια κατάσταση, ή,
- σε ένα συγκεκριμένο στιγμιότυπο κάποιας παραμέτρου, όπως μια τοποθεσία.

Τέλος, η ανάγκη για εκμετάλλευση της πληροφορίας πλαισίου στις κινητές συσκευές αποκτά ολοένα και αυξανόμενο ενδιαφέρον από διάφορα ερευνητικά πεδία, όπως τα πεδία Κινητού Υπολογισμού (Mobile Computing), Φορητής Υπολογιστικής Ικανότητας (Wearable Computing), Επαυξημένης Πραγματικότητας (Augmented Reality), Πανταχού Παρόντα Υπολογισμού (Ubiquitous Computing) ή Διάχυτου Υπολογισμού (Pervasive Computing) και Επικοινωνίας Ανθρώπου-Μηχανής (Human-Computer Interaction).

3.1.2. Κατηγορίες πλαισίου

Η ανάγκη προσδιορισμού ενός κατάλληλου πλαισίου κατά την ανάπτυξη μιας εφαρμογής οδηγεί σε κατηγοριοποίηση των τύπων πλαισίου. Η τοποθεσία, η ταυτότητα, η δραστηριότητα και ο χρόνος αποτελούν τους κυριότερους τύπους πλαισίου για το χαρακτηρισμό της κατάστασης μιας οντότητας. Αυτοί οι τύποι πλαισίου δεν απαντούν μόνο στα ερωτήματα *ποιος*, *τι*, *πότε* και *πού*, αλλά ενεργούν και ως δείκτες άλλων πηγών πληροφορίας πλαισίου. Για παράδειγμα, γνωρίζοντας την ταυτότητα ενός ατόμου, μπορούμε να έχουμε πρόσβαση σε σχετικές πληροφορίες, όπως αριθμός τηλεφώνου, διεύθυνση, e-mail, ημερομηνία γέννησης, λίστα φίλων, σχέσεις με άλλους ανθρώπους του περιβάλλοντός του, κ.λπ. [42].

Στην κατηγοριοποίηση αυτή, οι τέσσερις κύριοι τύποι πλαισίου αποτελούν το πρώτο επίπεδο. Όλοι οι άλλοι τύποι πλαισίου βρίσκονται στο δεύτερο επίπεδο. Κοινό χαρακτηριστικό των τύπων του δεύτερου επιπέδου είναι ότι μπορούν να ταξινομηθούν από τους τύπους πλαισίου του πρώτου επιπέδου, καθώς αποτελούν χαρακτηριστικά τους. Έτσι, ο αριθμός τηλεφώνου ενός ατόμου είναι τμήμα πλαισίου δευτέρου επιπέδου, γιατί μπορεί να προσδιοριστεί χρησιμοποιώντας ως δείκτη την ταυτότητα του ατόμου σε ένα χώρο πληροφορίας όπως αυτόν του τηλεφωνικού καταλόγου. Επίσης, υπάρχουν περιπτώσεις όπου πολλοί κύριοι τύποι πλαισίου απαιτούνται για την ταξινόμηση ενός χώρου πληροφορίας, όπως η περίπτωση ενός δελτίου πρόβλεψης καιρού, ο καθορισμός του οποίου απαιτεί τόσο τη γεωγραφική θέση όσο και την ημερομηνία της πρόγνωσης.

3.2. Επίγνωση Πληροφορίας Πλαισίου

Οι δυνατότητες ενός συστήματος που έχει πλήρη «επίγνωση» του τι συμβαίνει γύρω του είναι πολύ μεγάλες. Η πληροφορία που το επηρεάζει είναι αχανής και περιλαμβάνει από χωρικές και χρονικές παραμέτρους ως και παράγοντες που δεν είναι πάντα τόσο προφανείς, αλλά μπορεί να αποδειχθούν κρίσιμοι. Αν ένα σύστημα μπορέσει να φτάσει στο σημείο να «προσαρμόζεται» κάθε φορά στο περιβάλλον του, ακόμη και στους ίδιους του τους χρήστες, με αξιοπιστία και προνοητικότητα, τότε θα μπορούμε να μιλάμε για ένα πραγματικά εύχρηστο, χρήσιμο, κατά το δυνατόν διαθέσιμο και ανθρωποκεντρικό προϊόν τεχνολογίας [37].

Μέγιστη σημασία, λοιπόν, δίνεται στην έννοια της «επίγνωσης πλαισίου» (“context awareness”). Η επίγνωση πλαισίου μπορεί να οριστεί ως:

«*Η ικανότητα ενός συστήματος να ανακαλύπτει, να ερμηνεύει, να συμπεραίνει, να αξιοποιεί και να συλλογίζεται βάσει της περιρρέουσας πληροφορίας ώστε να λαμβάνει αποφάσεις, να προβαίνει σε προκαθορισμένες ενέργειες και να προσαρμόζεται σε διάφορες καταστάσεις*.» [43].

Με μια προσεκτική και δομημένη εισαγωγή της «επίγνωσης πλαισίου» στην υπολογιστική ικανότητα που διέπει την καθημερινότητά μας, πολλές δραστηριότητες μπορούν να γίνουν πολύ πιο απλές και αποδοτικές, με ελάχιστη συμβολή και διαμεσολάβηση του χρήστη. Η δυνατότητα για επικοινωνία μπορεί όχι μόνο να διευκολυνθεί, αλλά και να ενσωματωθεί με φυσικό τρόπο στη ζωή μας. Αυτά τα δύο συνδέονται με συνεχή και διάφανο τρόπο, έχοντας ως συνδετικό κρίκο το πλαίσιο, την αναπαράστασή του, την επίγνωσή του, το συμπερασμό του, την προσαρμογή του στις καταστάσεις του χρήστη / δικτύου, καθώς και την έγκαιρη πρόβλεψή του.

Η ενσωμάτωση της επίγνωσης πλαισίου σε μια «εφαρμογή επίγνωσης πλαισίου» (“context-aware application”) αποτελεί αντικείμενο επιστημονικής έρευνας. Για παράδειγμα, τα προβλήματα που σχετίζονται με την αλληλεπίδραση ανθρώπου και υπολογιστή είναι ιδιαίτερα σημαντικά. Μέσω του πλαισίου, η σχέση άνθρωπος-υπολογιστής ορίζεται σχεδόν εκ νέου. Το ερώτημα που τίθεται είναι πώς θα μπορούσε ο χρήστης να αισθάνεται ασφαλής και ικανοποιημένος απέναντι στο τρέχον σύστημά του, ενώ ταυτόχρονα το τελευταίο να παίρνει όσο το δυνατόν περισσότερες και όσο το δυνατόν πιο πρώιμες αποφάσεις;

Η επιστημονική έρευνα επικεντρώνεται στην αρχιτεκτονική των συστημάτων που βασίζονται στο πλαίσιο, εστιάζοντας στην απαραίτητη υποδομή σε υλικό και στις αντίστοιχες ενδιάμεσες υπηρεσίες που είναι υπεύθυνες για τη συλλογή, επεξεργασία και προώθηση της πληροφορίας, σε συνδυασμό όχι μόνο με την αλληλεπίδραση με το χρήστη αλλά και με το υπόλοιπο κατανεμημένο σύστημα.

Επίσης, αντικείμενο έρευνας αποτελεί και το ζήτημα της «αναπαράστασης» (“representation”) και «ερμηνείας» (“interpretation”) της πληροφορίας πλαισίου. Η πληροφορία που συλλέγεται από πολλές (διαφορετικές) πηγές ερμηνεύεται διαφορετικά από κάθε εφαρμογή επίγνωσης πλαισίου. Έτσι, αναπτύσσονται μηχανισμοί μέσω των οποίων το πλαίσιο ελέγχεται και ερμηνεύεται κατάλληλα με βάση τη μελλοντική του χρήση και αξιοποίηση.

Η επίγνωση πληροφορίας πλαισίου έχει γίνει ιδιαίτερα σημαντική στο πεδίο του κινητού και κατανεμημένου υπολογισμού (mobile and distributed computing). Οι σημερινές φορητές

συσκευές που έχουν ικανότητα επεξεργασίας γίνονται ολοένα και πιο δυνατές τεχνολογικά, διαθέτοντας ενσωματωμένη δυνατότητα αποθήκευσης, αυξανόμενη υπολογιστική ισχύ, δυνατότητα τηλεπικοινωνίας και υποστήριξη πολλών και διαφορετικών εφαρμογών. Η επίγνωση πλαισίου για τέτοιου είδους συσκευές και για τα συστήματα στα οποία περικλείονται θεωρείται σημείο-κλειδί για τη μελλοντική τους εξέλιξη. Τα συστήματα που μπορούν να εκμεταλλευτούν τέτοιου είδους πληροφορία είναι είτε καθεαυτό κινητά, όπως φορητοί υπολογιστές (laptops), υπολογιστές χειρός (palmtops), tablets, κινητά τηλέφωνα, είτε σταθερά, όπως καλωδιακές τηλεοράσεις, συστήματα οικιακής ψυχαγωγίας, ακόμη και ολόκληρα δωμάτια ή κτήρια εξοπλισμένα με διαδραστικές συσκευές και ειδικούς ανιχνευτές.

Η τεχνολογία των κινητών συσκευών με υπολογιστική ικανότητα έχει δώσει ιδιαίτερη ώθηση στη χρήση των υπολογιστών σε ποικίλα και συνεχώς μεταβαλλόμενα περιβάλλοντα. Οι κινητές συσκευές μπορούν να αυξήσουν τις ικανότητές τους αν τους δοθεί η δυνατότητα να λαμβάνουν υπόψη τους το πλαίσιο στο οποίο βρίσκονται και αλληλεπιδρούν. Για να μπορέσουν να προσλάβουν πληροφορία από το πλαίσιο στο οποίο βρίσκονται (περιβάλλον, περιρρέουσα κατάσταση) έχουν την ανάγκη να εξοπλιστούν με ειδικούς αισθητήρες και πηγές πληροφόρησης που τους επιτρέπουν να έχουν πρόσβαση στην πληροφορία αυτή. Οι αισθητήρες είναι εξειδικευμένα εξαρτήματα, τα οποία δεν έχουν ως πρώτιστο στόχο την υπολογιστική ικανότητα αλλά την πρόσληψη των παραμέτρων του περιβάλλοντος στο μοντέλο του υπολογισμού.

Οι κινητές συσκευές που έχουν ενσωματωμένες δυνατότητες άμεσης επίγνωσης πλαισίου διαθέτουν πολλαπλούς αισθητήρες και χρησιμοποιούν αλγόριθμους για την επεξεργασία των δεδομένων που λαμβάνουν από το περιβάλλον και τη μετατροπή τους σε δεδομένα χρήσιμα (αξιοποιήσιμα) για τις λειτουργίες τους. Μεγάλο κομμάτι της έρευνας ασχολείται με τη χρήση μεμονωμένων αλλά ισχυρότατων αισθητήρων, όπως ανιχνευτών θέσης χρήστη (εσωτερικών ή εξωτερικών χώρων), τοποθεσίας της κινητής συσκευής, φυσικών ή περιβαλλοντολογικών παραμέτρων (π.χ. θερμοκρασίας, ταχύτητας κίνησης, πίεσης, φωτεινότητας, κατεύθυνσης, ήχου, υγρασίας, ταχύτητας ανέμου), καθώς και καμερών.

Οι αισθητήρες θέσης παρέχουν πληροφορία που σχετίζεται με την τοποθεσία ως ανεξάρτητο, αλλά και ιδιαίτερα χρήσιμο δεδομένο. Συχνά, βέβαια, δεν είναι η τοποθεσία που συνιστά μόνη της άμεσα αξιοποιήσιμο δεδομένο, αλλά σε συνδυασμό με επιπρόσθετη πληροφορία που μπορεί να παραχθεί (π.χ. οι πόροι που είναι διαθέσιμοι σε κάποιο σημείο). Επίσης, οι μετρήσεις από διάφορους αισθητήρες μπορούν να ορίσουν μια πιο σύνθετη αναπαράσταση και επίγνωση του πλαισίου (π.χ. η ανίχνευση πυρκαγιάς συμπεραίνεται από μετρήσεις

αισθητήρων καπνού, θερμοκρασίας και υγρασίας). Ακόμη, οι κάμερες παρέχουν δυνητικά πολύ πλούσιο υλικό, από το οποίο είναι δυνατόν να αντληθεί η απαραίτητη πληροφορία. Παρόλο που το χωρικό, το φυσικό και το οπτικό πλαίσιο μπορούν να αποδειχθούν πολύ ισχυρά στη διευκόλυνση των συσκευών όσον αφορά την επίγνωση του πλαισίου, δεν πρέπει να αποτελούν μόνο μια στατική περιγραφή ενός χώρου και χρόνου, η οποία δεν αντανακλά τη δυναμική σκοπιά του πλαισίου. Αντίθετα, περαιτέρω γνώση για το χώρο μπορεί να αποτελέσει επέκταση του χωρικού πλαισίου (π.χ. αν μια αίθουσα διδασκαλίας είναι μέρος ενός πανεπιστημιακού κτηρίου, τότε ένας φοιτητής που βρίσκεται στην αίθουσα αυτή βρίσκεται επίσης και μέσα στο αντίστοιχο κτήριο).

3.3. Θέματα Επίγνωσης Πληροφορίας Πλαισίου

Η επίγνωση πλαισίου επιτρέπει σε ένα σύστημα να προσαρμόζεται στο περιβάλλον του, αποκτώντας έτσι πολλά πλεονεκτήματα και δυνατότητες για νέες εφαρμογές. Μερικά από τα πλεονεκτήματα αυτά είναι [37]:

- «Διεισδυτικότητα» (“pervasiveness”). Ένα προσαρμοζόμενο σύστημα μπορεί να συνεργάζεται πολύ πιο αποτελεσματικά με τους χρήστες του και να δρα αυτόνομα όπου απαιτείται.
- «Προ-δραστικότητα» (“proactiveness”). Το σύστημα μπορεί να προλαβαίνει τις ανάγκες που θα παρουσιαστούν στο κοντινό μέλλον προσαρμοζόμενο κατάλληλα προκειμένου να τις αντιμετωπίσει.
- «Ικανότητα συμπερασμού» (“inference capability”). Ένα σύστημα που μπορεί να συνάγει συμπεράσματα από το πλαίσιό του (περαιτέρω πληροφορία και επαύξηση γνώσης) μπορεί όχι μόνο να συλλογίζεται για την περιρρέουσα κατάσταση, αλλά και να παρέχει πολύ πιο εξειδικευμένες υπηρεσίες στους χρήστες του.

Κατά την υλοποίηση της επίγνωσης πλαισίου ανακύπτουν διάφορα προβλήματα. Ενδεικτικά αναφέρουμε τα εξής:

- *Αναπαράσταση πληροφορίας πλαισίου*. Η πληροφορία πλαισίου πρέπει να αναπαριστάνεται κατάλληλα (context representation) με καθορισμένα «μοντέλα αναπαράστασης πλαισίου» (“context models”), έτσι ώστε το εκάστοτε μοντέλο γνώσης του πλαισίου να αντικατοπτρίζει το τρέχον πλαίσιο και να επιτυγχάνεται ο κατάλληλος «συμπερασμός» (“context inference”) και «συλλογισμός» του (“context reasoning”).

- *Ανίχνευση πληροφορίας*. Οι συσκευές / συστήματα πρέπει να διαθέτουν κατάλληλους ανιχνευτές για την πρόσληψη της αναγκαίας πληροφορίας τόσο από το περιβάλλον όσο και από τους χρήστες, με έξυπνο και διάχυτο τρόπο (δηλαδή με όσο το δυνατό λιγότερη παρέμβαση του χρήστη).
- *Επεξεργασία πληροφορίας*. Η πληροφορία που συλλέγεται από το σύστημα και τους ανιχνευτές του είναι σε μεγάλο βαθμό ετερογενής και έτσι πολλές μέθοδοι και αλγόριθμοι «σύντηξης» (“context fusion”) και «συνάθροισης πλαισίου» (“context aggregation”) θα πρέπει να εφαρμοστούν.
- *Ταξινόμηση και πρόβλεψη πλαισίου*. Η «ταξινόμηση» (“classification”) και η «πρόβλεψη» (“prediction”) του πλαισίου είναι αναγκαίες κυρίως σε συστήματα με υπολογιστικές ικανότητες, πράγμα το οποίο εισάγει ένα σύνολο αλγόριθμων που μπορούν να χρησιμοποιηθούν (π.χ. αλγόριθμοι Μηχανικής Μάθησης).
- *Εκπαίδευση και προσαρμογή συστήματος*. Η «εκπαίδευση» (“training”) και η «προσαρμογή» (“adaptation”) του συστήματος πρέπει να γίνονται σε πραγματικό χρόνο και να προηγούνται της κατάστασης την οποία καλούνται να διευκολύνουν.
- *Αλληλεπίδραση με το χρήστη*. Η «αλληλεπίδραση με το χρήστη» (“user interaction / user intervention”) πρέπει να μένει σε χαμηλό επίπεδο και οπωσδήποτε να διεκπεραιώνεται όσο το δυνατόν πιο διακριτικά.

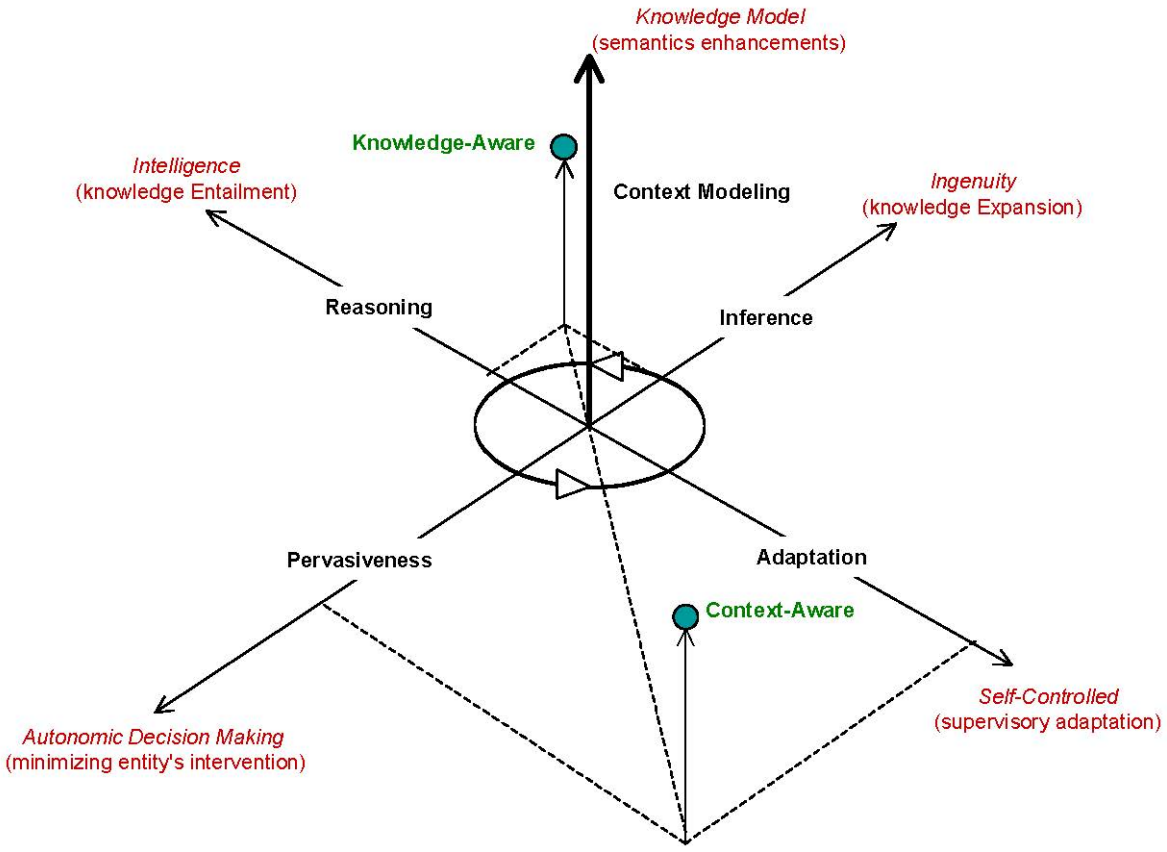
Με βάση τα παραπάνω, ένα Σύστημα Επίγνωσης Πλαισίου – ΣΕΠ (Context-Aware System) μπορεί να απεικονιστεί σε ένα λογικό χώρο (βλέπε Σχήμα 3). Ο χώρος αυτός αποτελείται από το λογικό άξονα Μοντελοποίηση Πλαισίου (Context Modeling) με συνιστώσες την αναπαράσταση, το συμπερασμό, το συλλογισμό, τη διεισδυτικότητα και την προσαρμογή. Η Μοντελοποίηση Πλαισίου είναι μια μεθοδολογία βασισμένη στην αναπαράσταση πληροφορίας που σχετίζεται με ένα πεπερασμένο σύνολο αφηρημένων λογικών οντοτήτων. Οι οντότητες αυτές περιγράφουν ένα φυσικό αντικείμενο ή μια έννοια. Ο προσδιορισμός του μοντέλου πλαισίου μιας εφαρμογής ποικίλλει από αδόμητα μοντέλα δεδομένων (π.χ. θέση, περιβαλλοντολογικές μετρήσεις), έως μοντέλα γνώσης βασισμένα στη λογική (π.χ. χωρο-χρονικά, αντικειμενοστραφή, σχεσιακά μοντέλα, προτασιακή λογική, ασαφής λογική, λογική δεύτερης τάξης, λογισμός καταστάσεων). Τα λογικά αυτά μοντέλα συνήθως σχετίζονται με δραστηριότητες συγκεκριμένων οντοτήτων, όπως για παράδειγμα δραστηριότητες ανθρώπων (συνάντηση, εργασία, κίνηση, συνομιλία).

Επίσης, συστήματα που βασίζονται σε δίκτυα αισθητήρων υιοθετούν απλά μοντέλα δεδομένων, άρα και μοντέλα πλαισίου, για να αναπαραστήσουν την ανακτηθείσα

πληροφορία. Συνεπώς, αφηρημένη και συναγόμενη γνώση δεν μπορούν να παραχθούν από αισθητήρες. Όσο περισσότερο ένα σύστημα μπορεί να συνάγει γνώση και να συμπεραίνει επιπρόσθετη γνώση, τόσο πιο πολύ διεισδυτικό μπορεί να είναι στις ενέργειες και στις αυτόνομες αποφάσεις του. Η έννοια της διείσδυσης ενός ΣΕΠ αναφέρεται στην ικανότητα του συστήματος

(i) να εκπαιδεύεται – είτε με «επιβλεπόμενη μάθηση» (“supervised learning”), είτε με «μη επιβλεπόμενη μάθηση» (“unsupervised learning”) – για τις συνθήκες που πρέπει να ισχύουν όταν αναγνωρίζεται η κατάσταση μιας οντότητας και παράλληλα να αναγνωρίζει την κατάσταση αυτή και να λαμβάνει αποφάσεις (συλλογίζεται)

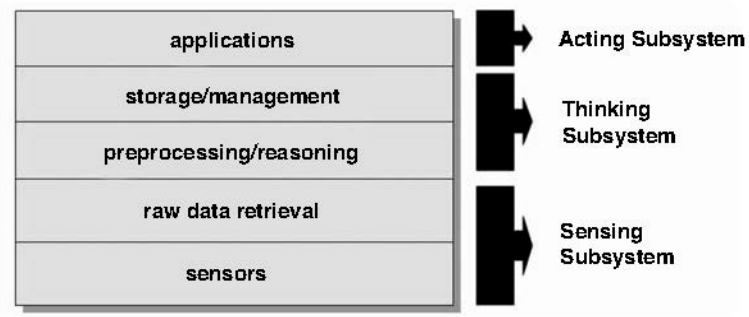
(ii) και να αναγνωρίζει καταστάσεις που ήταν άγνωστες ή δεν είχε την ευκαιρία να τις μάθει και ταυτόχρονα να μαθαίνει τις αποφάσεις που πρέπει να λαμβάνει για τις άγνωστες αυτές καταστάσεις.



Σχήμα 3: Εννοιολογικός χώρος Επίγνωσης Πλαισίου

3.4. Αρχιτεκτονική Συστήματος Επίγνωσης Πληροφορίας Πλαισίου

Ένα Σύστημα Επίγνωσης Πλαισίου (ΣΕΠ) περιλαμβάνει τρεις βασικές λειτουργίες: την αίσθηση (sensing) της περιρρέουσας κατάστασης, τη σκέψη (thinking) και τη δράση (acting). Στο Σχήμα 4 φαίνεται η αρχιτεκτονική ενός ΣΕΠ.



Σχήμα 4: Αρχιτεκτονική Συστήματος Επίγνωσης Πλαισίου

Ένα Σύστημα Επίγνωσης Πλαισίου για να αποκτήσει πληροφορία από το πλαίσιο στο οποίο βρίσκεται (περιβάλλον, περιρρέουσα κατάσταση) θα πρέπει να είναι εξοπλισμένο με ειδικούς αισθητήρες και πηγές πληροφόρησης που να του επιτρέπουν πρόσβαση στην πληροφορία αυτή. Οι αισθητήρες είναι εξειδικευμένα εξαρτήματα, τα οποία δεν έχουν ως πρώτιστο στόχο την υπολογιστική ικανότητα αλλά την πρόσληψη των παραμέτρων του περιβάλλοντος.

Ένα ΣΕΠ για να μπορέσει να λάβει εικόνα της περιρρέουσας κατάστασης μέσα από τις μετρήσεις των αισθητήρων και να αποφασίσει για περαιτέρω ενέργειες θα πρέπει να εξαγάγει τις πληροφορίες αυτές με τη βοήθεια μαθηματικών μοντέλων και μεθόδων. Τέτοια μοντέλα είναι και τα ακόλουθα [44]:

- Μαθηματικά μοντέλα (π.χ. Kalman filtering).
- Αναγνώριση προτύπων (pattern recognition), νευρωνικά δίκτυα (neural nets), αλγόριθμοι clustering.
- Γνωσιακά μοντέλα (cognitive based models), ασαφής λογική (fuzzy logic).

Η τρίτη λειτουργία ενός Συστήματος Επίγνωσης Πλαισίου αφορά την υλοποίηση ενεργειών για την ικανοποίηση των αναγκών της εφαρμογής που εξυπηρετεί. Αυτές οι ενέργειες μπορεί να παραπέμπουν ακόμη και σε επιπρόσθετη συλλογή δεδομένων και θα πρέπει να

πραγματοποιηθούν αρκετά γρήγορα, έτσι ώστε να έχουν νόημα και να συμβαδίζουν με τη περιρρέουσα κατάσταση.

3.5. Μοντελοποίηση Πληροφορίας Πλαισίου

Η Μοντελοποίηση Πλαισίου είναι μια περιοχή έρευνας της Επίγνωσης Πλαισίου που εστιάζει στη μελέτη των εξής προβλημάτων [37]:

- Εύρεση της καταλληλότερης πληροφορίας πλαισίου (π.χ. μεταβλητές, παράμετροι, γνωρίσματα), που μπορεί να αναπαραστήσει όσο το δυνατόν καλύτερα την πληροφορία (π.χ. χρόνος, θέση, περιβαλλοντικά μεγέθη, έννοιες, δραστηριότητες) για ένα συγκεκριμένο ερευνητικό πεδίο (π.χ. Location-based Services). Οι μεταβλητές αυτές καλούνται «συνιστώσες» της πληροφορίας πλαισίου (contextual ingredients), εφόσον φανταστούμε το πλαίσιο ως ένα πολυδιάστατο διάνυσμα (multivariate context vector ή context vector) σε ένα διανυσματικό χώρο καθοριζόμενο από μεταβλητές που αναπαριστάνουν την πληροφορία του εκάστοτε ερευνητικού πεδίου. Αξίζει να σημειωθεί ότι στο χώρο των διανυσμάτων αυτών οι μεταξύ τους συνιστώσες δεν είναι πάντα ανεξάρτητες.
- Εύρεση των εξαρτήσεων – συσχετίσεων μεταξύ των συνιστωσών αυτών (π.χ. χρονικές εξαρτήσεις, σχέσεις εκλέπτυνσης / εξειδίκευσης και γενίκευσης, εξαρτήσεις μέρους-όλου / μερεολογικές, εξαρτήσεις συμβατότητας, εξαρτήσεις περιορισμών, έτερο-συσχετίσεις).
- Εύρεση των μεθόδων εκμάθησης, ταξινόμησης και προσαρμογής του διανύσματος του πλαισίου, εφόσον κάτι τέτοιο κρίνεται αναγκαίο.

Υπάρχουν δύο διακριτές μεταξύ τους προσεγγίσεις μοντελοποίησης του πλαισίου: η «Θεωρητική Μοντελοποίηση» (“Theoretic Context Modeling”) και η «Εννοιολογική Μοντελοποίηση» (“Conceptual Context Modeling”). Η Θεωρητική Μοντελοποίηση βασίζεται σε καταστάσεις και σε ενέργειες που είναι παρατηρούμενες στις καταστάσεις αυτές. Οι ενέργειες αυτές είναι υπεύθυνες και για τη μετάβαση καταστάσεων. Η Εννοιολογική Μοντελοποίηση βασίζεται στην αντιστοίχιση του πλαισίου με έννοιες – αντικείμενα (π.χ. κατηγορήματα Λογικής Πρώτης Τάξης, Ασαφή Σύνολα) και με συσχετίσεις μεταξύ εννοιών (π.χ. ν-αδικές σχέσεις).

3.6. Συνεργατική Επίγνωση Πληροφορίας Πλαισίου

3.6.1. Ορισμός

Στις μέρες μας συχνά αντιμετωπίζουμε καταστάσεις στις οποίες άτομα συγκεντρώνονται κατά ομάδες σε ένα χώρο, π.χ. σε μουσεία, σε αίθουσες διδασκαλίας κ.λπ.. Γενικά, τα άτομα αυτά μοιράζονται (τουλάχιστον μέσα σε ίδια χρονικά πλαίσια) κοινά ενδιαφέροντα και προτιμήσεις (π.χ. άτομα που ενδιαφέρονται για την ίδια έκθεση σε ένα μουσείο). Τα μέλη μιας τέτοιας ομάδας βιώνουν παρόμοιες καταστάσεις με αυτές όταν πολλοί φορητοί υπολογιστές / κόμβοι ανιχνεύουν, εντοπίζουν και επεξεργάζονται πανομοιότυπη πληροφορία πλαισίου [37].

Η συνύπαρξη κόμβων σε κοινούς χώρους εισάγει την ανάγκη για το χειρισμό της επίγνωσης πληροφορίας πλαισίου μέσα από ένα πνεύμα συνεργασίας. Η συνεργασία (collaboration) υποδηλώνει τη συνεργία (synergy) μεταξύ των κόμβων των αντίστοιχων μελών μιας ομάδας όσον αφορά την ανίχνευση, την ερμηνεία και το διαμοιρασμό πληροφορίας πλαισίου, π.χ. η πληροφορία που λείπει από τον κόμβο Α λαμβάνεται και στέλνεται από τον κόμβο Β, ο κόμβος C ερμηνεύει την πληροφορία που διαμοιράζεται από τον κόμβο D κ.ο.κ..

Η Συνεργατική Επίγνωση Πληροφορίας Πλαισίου – ΣΕΠΠ (Collaborative Context-Awareness) υποδηλώνει την κατανόηση της πληροφορίας πλαισίου από όλα τα μέλη μιας ομάδας, παρέχοντας κατά συνέπεια μια πιο αναβαθμισμένη πληροφορία πλαισίου για κάθε μέλος της ομάδας χωριστά. Η συνεργασία μεταξύ των κόμβων αναβαθμίζει την ποιότητα πληροφορίας πλαισίου παρέχοντας σε καθέναν κόμβο μια αναβαθμισμένη επίγνωση του πλαισίου.

Η «Συνεργατική Πληροφορία Πλαισίου» (“Collaborative Context”) είναι η πληροφορία πλαισίου που ανακτάται μέσω της γειτονικής δικτύωσης αισθητήρων ανίχνευσης και κόμβων, έτσι ώστε να βελτιώνεται η κοινή κατανόηση / αντίληψη για τον γύρω χώρο, αυξάνοντας τη διαθεσιμότητα και το βαθμό αξιοπιστίας του πλαισίου (μέσω του διαμοιρασμού επιπρόσθετης και συμπληρωματικής πληροφορίας πλαισίου από γειτονικούς κόμβους).

Ένα Σύστημα Συνεργατικής Επίγνωσης Πληροφορίας Πλαισίου – ΣΣΕΠΠ (Collaborative Context-Aware System) συνίσταται από μια ομάδα κόμβων ικανών να ανιχνεύουν, να συγκεράζουν, να συμπεραίνουν και να επικοινωνούν, έτσι ώστε να προσεγγίσουν μια κοινή ή παρόμοια κατανόηση / αντίληψη για τον γύρω τους χώρο.

Το μοντέλο ΣΕΠΠ αποφέρει εύρωστες Εφαρμογές Επίγνωσης Πλαισίου (ΕΕΠ), δηλαδή ασύδοτες σε παροδικές αποτυχίες αισθητήρων ή στη διατάραξη της πληροφορίας πλαισίου.

Επίσης, είναι δυνατό να επιτευχθούν σημαντικές οικονομίες κλίμακας, καθώς δεν απαιτείται όλοι οι κόμβοι να έχουν τα ίδια ή και επικαλυπτόμενα σύνολα αισθητήρων ή συνιστωσών ανίχνευσης πληροφορίας πλαισίου που να παρέχουν ακρίβεια μετρήσεων και ανοχή σε σφάλματα.

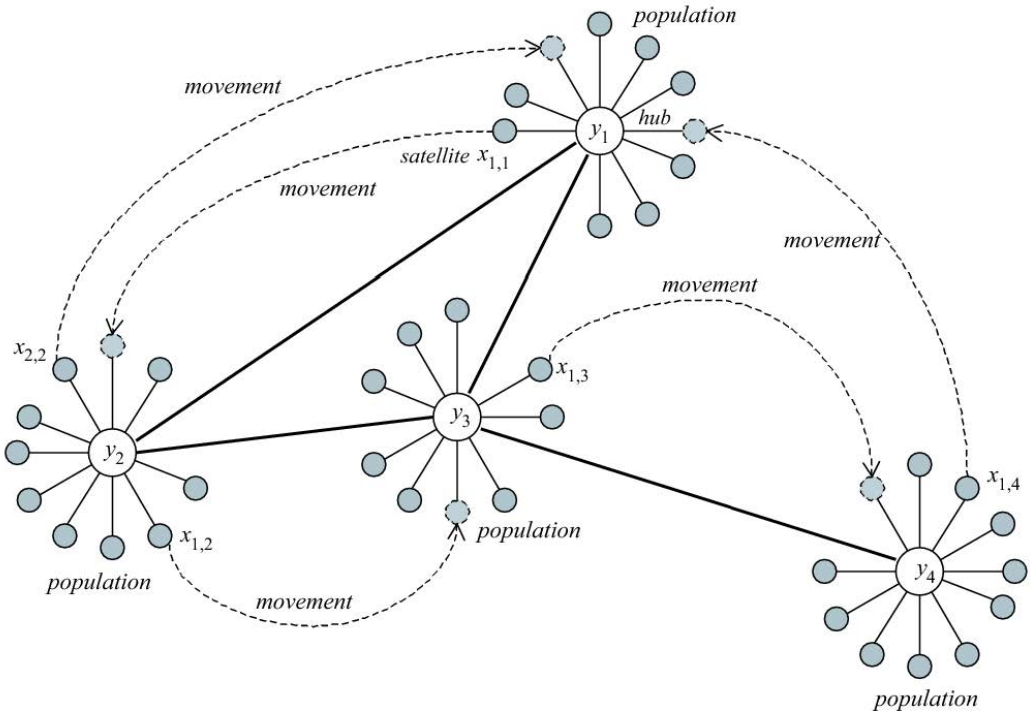
Μια Εφαρμογή Επίγνωσης Πλαισίου βασίζεται σε αλγόριθμους μετάδοσης πληροφορίας για το διαμοιρασμό πλαισίου σε μια ομάδα κόμβων.

3.6.2. Κινητικότητα στη ΣΕΠΠ

Οι κινητοί κόμβοι ενός δικτύου προσομοιάζουν με ένα σύνολο ανθρώπων που έχουν παρόμοια κινητική συμπεριφορά. Τα δίκτυα κινητών κόμβων ποικίλλουν στα χαρακτηριστικά τους λόγω της κινητικότητας των κόμβων τους. Σε τέτοιου είδους δίκτυα εμφανίζεται συνήθως μια μη ομοιογενής κατανομή κόμβων. Η κατανομή αυτή μπορεί να δημιουργεί νομαδικά δίκτυα, στα οποία οι κόμβοι κατανέμονται συγκροτώντας «διασυνδεδεμένες ομάδες» (“clusters”) από κόμβους, ή να δημιουργεί ad-hoc δίκτυα στα οποία οι κόμβοι κινούνται ελεύθερα χωρίς να υπάρχει κάποια συγκεκριμένη δομή (π.χ. η τοπολογία του ασύρματου δικτύου μπορεί να αλλάζει απρόβλεπτα και αυθαίρετα) [37].

Στα δίκτυα αυτά, μερικοί κόμβοι μπορούν να έχουν περισσότερες συνδέσεις επικοινωνίας (να προσελκύουν περισσότερους κόμβους) από άλλους. Αυτοί οι «συγκεντρωτικοί κόμβοι» (“hubs”) μορφοποιούν μη ομοιογενείς κατανομές, όταν ένας «κόμβος επισκέπτης» (“satellite”) σε ένα συγκεντρωτικό κόμβο έχει περισσότερους γειτονικούς κόμβους από έναν άλλο επισκέπτη κόμβο που προσκολλάται σε ένα λιγότερο δημοφιλή συγκεντρωτικό κόμβο, όπως απεικονίζεται στο Σχήμα 5.

Ένα δίκτυο κινητών κόμβων μπορεί να μοντελοποιηθεί από πολλές ομάδες αποτελούμενες από συγκεντρωτικούς και επισκέπτες κόμβους. Κάθε ομάδα μπορεί να θεωρηθεί ως ένας ξεχωριστός «πληθυσμός» (population) και, καθώς οι επισκέπτες μετακινούνται μεταξύ των πληθυσμών, κομίζουν πληροφορία πλαισίου σε αυτούς.



Σχήμα 5: Δίκτυο κινητών κόμβων

3.7. Ανακάλυψη Πληροφορίας Πλαισίου (Context Discovery Problem – CDP)

Η έννοια «Ανακάλυψη Πληροφορίας Πλαισίου» αναφέρεται στο μηχανισμό που υιοθετούν κινητοί κόμβοι σε μια καθορισμένη περιοχή ώστε να αναζητούν πηγές πληροφορίας. Οι πηγές αυτές περιέχουν την πληροφορία πλαισίου που χρειάζονται οι κινητοί κόμβοι για τη συγκεκριμένη εφαρμογή τους. Οι πηγές αυτές μπορεί να είναι κινητές ή στατικές, με ικανότητα όμως ανάκτησης πληροφορίας, π.χ. αισθητήρες. Στόχος των κόμβων είναι να εντοπίσουν τις πηγές αυτές το συντομότερο δυνατό ελαχιστοποιώντας τον επικείμενο φόρτο επικοινωνίας και ανταλλαγής πληροφορίας.

Η πληροφορία πλαισίου που κρίνεται κατάλληλη για έναν κόμβο δεν κρίνεται αναγκαστικά εξίσου κατάλληλη από κάποιο γειτονικό του κόμβο. Ο χαρακτηρισμός «κατάλληλη πληροφορία πλαισίου» βασίζεται στην εκτίμηση της ποιότητας πλαισίου που θέτει κάθε εφαρμογή επίγνωσης πλαισίου για τη διαθέσιμη πληροφορία. Για παράδειγμα, μια εφαρμογή

η οποία απαιτεί διαρκώς πρόσφατη ανακτημένη και ενημερωμένη πληροφορία είναι περισσότερο αυστηρή στην εκτίμηση της πληροφορίας που θα λάβει από ένα γειτονικό κόμβο, σε σχέση με μια άλλη εφαρμογή που δεν ενδιαφέρεται για μια τόσο πρόσφατη ενημερωμένη πληροφορία πλαισίου. Επίσης, η «κατάλληλη πληροφορία» δεν βασίζεται μόνο στη χρονική εγκυρότητα του πλαισίου αλλά, για παράδειγμα, στην ακρίβεια μέτρησης, στην αξιοπιστία της πηγής από όπου ανακτάται, καθώς και σε άλλες μετρικές εγκυρότητας, όπως η χωρική διαθεσιμότητα.

Ας φανταστούμε μια ομάδα M κόμβων με ικανότητα ανάκτησης και αντίληψης του περιβάλλοντα χώρου και της περιρρέουσας κατάστασης, όπου ένα υποσύνολο της ομάδας αυτής αποτελείται από N πηγές πληροφορίας. Οι πηγές δεν πασχίζουν για ανακάλυψη πλαισίου, εφόσον έχουν την ικανότητα να ανακτούν άμεσα το πλαίσιο. Οι υπόλοιποι κόμβοι πασχίζουν να εντοπίσουν τις (κινούμενες) πηγές, έτσι ώστε να ανακτήσουν την απαιτούμενη πληροφορία πλαισίου. Οι κόμβοι αυτοί καλούνται «οπαδοί» (“followers”). Οι οπαδοί ακολουθούν τις πηγές εφόσον τις εντοπίσουν στην ακτίνα δράσης επικοινωνίας τους και τότε, ανάλογα με συγκεκριμένες «πολιτικές ανακάλυψης» (“foraging policies”), ανακτούν το κατάλληλο για αυτούς πλαίσιο.

Στην περίπτωση που οι πηγές δεν κινούνται και έχουμε μια πλήρως επιβλεπόμενη κατάσταση όλων των κόμβων, τότε ο μηχανισμός ανακάλυψης πλαισίου μοιάζει πολύ με τους αλγόριθμους Ant Colony Optimization [45].

Όμως, σε περιβάλλοντα διάχυτου υπολογισμού, ένας μηχανισμός ανακάλυψης πλαισίου θα πρέπει να αντιμετωπίζει τις εξής συνθήκες:

- Πλήρης έλεγχος όλων των κόμβων δεν υφίσταται.
- Η πληροφορία που αναζητείται δεν μπορεί να εκτιμηθεί καθολικά εξίσου από όλους τους κόμβους.
- Η εκτίμηση για την πληροφορία πλαισίου μεταβάλλεται χρονικά καθώς μεταβάλλονται οι συνιστώσες πλαισίου.
- Οι πηγές δεν είναι πάντα ακίνητες και επομένως μπορεί να ανακαλυφθεί μόνο μια προσωρινά βέλτιστη λύση.
- Οι κόμβοι δεν μπορούν να έχουν πάντα πλήρη συνεργατική συμπεριφορά.

Ένας μηχανισμός ανακάλυψης πλαισίου κατάλληλος για περιβάλλοντα διάχυτου υπολογισμού, ο οποίος ανταποκρίνεται στις παραπάνω συνθήκες, υιοθετεί τον αλγόριθμο Βελτιστοποίησης Σμήνους Σωματιδίων (PSO).

Κεφάλαιο 4: Ανακάλυψη Πληροφορίας Πλαισίου σε Κινητά Περιβάλλοντα με χρήση PSO και OST

4.1. Εισαγωγή

Τα συστήματα κινητού και κατανεμημένου υπολογισμού έχουν γίνει ιδιαίτερα δημοφιλή τα τελευταία χρόνια. Πολλές κινητές εφαρμογές επιδεικνύουν αυτοοργάνωση σε δυναμικά περιβάλλοντα, η οποία απορρέει από τη χρήση συστημάτων πολλαπλών πρακτόρων (multi-agent systems) ή σμήνους. Η βασική ιδέα είναι ότι οι εργασίες μπορούν να διεκπεραιωθούν πιο αποτελεσματικά κάνοντας χρήση πολλαπλών απλών αυτόνομων πρακτόρων αντί ενός πολύπλοκου πράκτορα. Τέτοιου είδους συστήματα παρουσιάζουν μεγαλύτερη προσαρμοστικότητα, επεκτασιμότητα και ευστάθεια σε σχέση με συγκεντρωτικά συστήματα [46].

Ένα σύστημα σμήνους μπορεί να θεωρηθεί ως ένα αποκεντρωμένο σύνολο αυτόνομων πρακτόρων (σωματιδίων), οι οποίοι έχουν περιορισμένες υπολογιστικές δυνατότητες. Τα σωματίδια θα πρέπει να συνεργάζονται μεταξύ τους για την επίτευξη κοινών εργασιών.

Το Πρόβλημα της Ανακάλυψης Πληροφορίας Πλαισίου (CDP) μπορεί να αντιμετωπιστεί από συστήματα σμήνους, στα οποία κάθε σωματίδιο (κινητός κόμβος / πράκτορας) θα πρέπει να ανακαλύψει, να εντοπίσει και να παρακολουθήσει την πηγή που παράγει την απαιτούμενη πληροφορία πλαισίου (π.χ. περιβαλλοντικούς παράγοντες όπως θερμοκρασία, φωτεινότητα κ.λπ.), για να τη μεταδώσει στην αντίστοιχη κινητή εφαρμογή.

Η Νοημοσύνη Σμήνους (Swarm Intelligence) επιτρέπει τη δημιουργία πλήρως κατανεμημένων συστημάτων, των οποίων η λειτουργικότητα καθορίζεται από την αλληλεπίδραση των αυτόνομων πρακτόρων μεταξύ τους και με το περιβάλλον τους. Τα συστήματα σμήνους έχουν από τη φύση τους υψηλή παραλληλία και επιδεικνύουν αυξημένη ευστάθεια και αξιοπιστία. Μερικά βασικά χαρακτηριστικά τους είναι τα εξής:

- Η απουσία ιεραρχικής δομής εντολών και ελέγχου. Το σύστημα είναι εγγενώς ανθεκτικό στα σφάλματα (fault-tolerant), καθώς η αποτυχία ενός ή μερικών πρακτόρων δεν επηρεάζει το σύνολο. Οι πράκτορες είναι συνήθως πολύ απλοί, πανομοιότυποι και ανιχνεύουν και μοιράζονται πληροφορία σε παραλληλία.
- Η απλότητα των πρακτόρων, π.χ. πράκτορας μπορεί να είναι ένα κινητό τηλέφωνο με αισθητήρες. Οι πράκτορες έχουν περιορισμένες δυνατότητες μνήμης, ανίχνευσης,

επεξεργασίας και μετάδοσης δεδομένων και συνεργάζονται μεταξύ τους ανταλλάσσοντας πληροφορίες για να ανακτήσουν την απαιτούμενη πληροφορία πλαισίου. Η αύξηση του αριθμού των πρακτόρων δεν οδηγεί απαραίτητα στη βελτίωση της απόδοσης του συστήματος.

- Η συνεχής αναζήτηση έγκυρης πληροφορίας πλαισίου. Η πληροφορία πλαισίου περιοδικά καθίσταται απαρχαιωμένη (μη έγκυρη) και θα πρέπει να ανιχνεύεται και να προσδιορίζεται σε τακτικά χρονικά διαστήματα.

Ο αλγόριθμος Βελτιστοποίησης Σμήνους Σωματιδίων (PSO) μπορεί να χρησιμοποιηθεί για την αναζήτηση πληροφορίας σε ένα δυναμικό περιβάλλον (βλέπε εδάφιο 1.7). Στην εργασία αυτή εξετάζουμε την καταλληλότητά του για την Ανακάλυψη Πληροφορίας Πλαισίου σε ένα σύστημα κινητού και κατανεμημένου υπολογισμού. Στην περίπτωση αυτή, θεωρούμε ότι τα σωματίδια του σμήνους αντιπροσωπεύουν τους μη αισθητήριους κόμβους ενός ασύρματου δικτύου, ενώ οι πιθανές λύσεις του αλγόριθμου αντιστοιχούν στους αισθητήριους κόμβους του δικτύου αυτού. Οι αισθητήριοι κόμβοι ανιχνεύουν πληροφορία και ακολουθούν τυχαία κίνηση στο χώρο. Η πληροφορία ανανεώνεται περιοδικά, καθώς καθίσταται απαρχαιωμένη με το πέρασμα του χρόνου. Τα σωματίδια του σμήνους προσπαθούν να ανακτήσουν την πληροφορία αυτή, συνεργαζόμενα μεταξύ τους και έχοντας περιορισμένες δυνατότητες μνήμης, επεξεργασίας και επικοινωνίας.

Σε αντίθεση με το κλασικό πρόβλημα PSO, υπάρχουν περισσότερα από ένα βέλτιστα σημεία (οι αισθητήριοι κόμβοι του δικτύου), στα οποία προσπαθούν να συγκλίνουν τα σωματίδια του σμήνους. Επίσης, δεν μεταβάλλεται δυναμικά μόνο η θέση των βέλτιστων αλλά και η τιμή τους (σταδιακή υποβάθμιση και περιοδική ανανέωση πληροφορίας). Επομένως, αναφερόμαστε σε δυναμικό περιβάλλον τύπου III [2].

Αναλυτική περιγραφή του τρόπου χρήσης του αλγόριθμου Βελτιστοποίησης Σμήνους Σωματιδίων για την Ανακάλυψη Πληροφορίας Πλαισίου σε ένα σύστημα κινητού και κατανεμημένου υπολογισμού γίνεται στο εδάφιο 4.3. Στο επόμενο εδάφιο ορίζουμε τα μεγέθη Πληροφορία Πλαισίου και Ποιότητα Πληροφορίας Πλαισίου.

4.2. Αναπαράσταση Πληροφορίας Πλαισίου – Ποιότητα Πληροφορίας Πλαισίου

Η πληροφορία πλαισίου αναφέρεται στις τρέχουσες τιμές των παραμέτρων που αναπαριστάνουν τη δραστηριότητα / κατάσταση μιας οντότητας ή περιβαλλοντικής κατάστασης (βλέπε εδάφιο 3.1.1) [43].

Έστω $\boldsymbol{Y} = [Y_1, \ldots, Y_m]$ ένα m διαστάσεων διάνυσμα παραμέτρων, οι οποίες λαμβάνουν τιμές y_l στο πεδίο τιμών $\mathrm{Dom}(Y_l)$, Θεωρούμε ότι μια παράμετρος Y_l αποκτά υπόσταση αν σε κάποια χρονική στιγμή t μια τιμή y_l αντιστοιχίζεται σε αυτή. Έτσι, το πλαίσιο $\boldsymbol{y}$ είναι το υποστασιοποιημένο διάνυσμα $\boldsymbol{Y}$, δηλαδή $\boldsymbol{y} = [y_1, \ldots, y_m]$. Για κάθε υποστασιοποιημένη παράμετρο Y_l, $l = 1, \ldots, m$, ορίζουμε μια συνάρτηση $u: Y_l \times T \to [0, \alpha)$, $\alpha > 0$, $\alpha \in \mathbb{R}$, η οποία καθορίζει αν η τιμή y_l είναι έγκυρη τη χρονική στιγμή t. Η παράμετρος T συμβολίζει το χρόνο. Σε μια εφαρμογή επίγνωσης πλαισίου που εκτελείται στον κόμβο i, η τιμή y_l είναι έγκυρη τη χρονική στιγμή t αν ισχύει: $u(y_l, t) < \theta_{il}$, για κάποιο συγκεκριμένο κατώφλι $\theta_{il} \in (0, \alpha)$ που εξαρτάται από την εφαρμογή. Η συνάρτηση u μπορεί να είναι οποιαδήποτε αύξουσα με το χρόνο συνάρτηση F, $u = F(t)$. Για λόγους απλότητας μπορούμε να τη θεωρήσουμε ως την ταυτοτική συνάρτηση, δηλαδή $u = t$. Η παράμετρος α εξαρτάται από την εφαρμογή. Στην περίπτωσή μας, ορίζουμε ως a το μέγιστο χρόνο για τον οποίο η τιμή που λαμβάνει η παράμετρος Y_l είναι έγκυρη. Η τιμή της συνάρτησης u δηλώνει κατά πόσο η τιμή y_l αντιστοιχεί σε πρόσφατη ή παλιά μέτρηση. Η συνάρτηση u αναφέρεται μόνο στη χρονική εγκυρότητα της τιμής y_l. Επίσης, αξίζει να αναφέρουμε ότι μπορούν να οριστούν και άλλες συναρτήσεις εγκυρότητας, οι οποίες να αντιστοιχούν σε διαφορετικούς δείκτες ποιότητας, αντικειμενικούς ή μη (δείκτες θέσης, αξιοπιστίας μετρήσεων, κ.λπ.) [46], [47].

Ορίζουμε ως δείκτη «ποιότητας πληροφορίας πλαισίου» (“quality of context”) του πλαισίου $\boldsymbol{y}$ τη χρονική στιγμή t τη συνάρτηση $g: \boldsymbol{Y} \times T \to [0, \alpha)$. Η συνάρτηση g καθορίζει την εγκυρότητα των τιμών των παραμέτρων του διανύσματος $\boldsymbol{Y}$ σε σχέση με ένα συγκεκριμένο κατώφλι. Η τιμή της g είναι ο ελάχιστος δείκτης των τιμών αυτών, δηλαδή

$$g(\boldsymbol{y}, t) = \min_{l=1}^{m} \{u(y_l, t)\} \tag{4.1}$$

όπου το κατώφλι $\theta_{iy} = \min_{l=1}^{m} \{\theta_{il}\}$.

Τιμές του θ_{iy} κοντά στην τιμή α δηλώνουν άκυρο περιεχόμενο, δηλαδή απαρχαιωμένο περιεχόμενο (άκυρη ή απαρχαιωμένη πληροφορία πλαισίου), ενώ τιμές του θ_{iy} κοντά στο 0 δηλώνουν έγκυρο (φρέσκο) περιεχόμενο (έγκυρη ή πρόσφατη πληροφορία πλαισίου).

Κάθε κόμβος i προσπαθεί να μεγιστοποιήσει τη χρονική περίοδο Δt για την οποία ισχύει: $g(\mathbf{y}, t+\Delta t) < \theta_{iy}$, για κάποια χρονική στιγμή t. Δηλαδή, κάθε κόμβος προσπαθεί να διατηρήσει φρέσκο περιεχόμενο για όσο το δυνατόν μεγαλύτερο χρονικό διάστημα. Γενικά, ένας κόμβος i αξιολογεί την ποιότητα πλαισίου διαφορετικά από έναν κόμβο j, δηλαδή $g_i(\mathbf{y}, t) \neq g_j(\mathbf{y}, t)$. Στην περίπτωσή μας, δίχως βλάβη της γενικότητας, θεωρούμε ότι όλοι οι κόμβοι αξιολογούν την ποιότητα πλαισίου το ίδιο, δηλαδή με το ίδιο κατώφλι θ_{iy}, $\theta_{iy} = \theta_y$.

4.3. Ανακάλυψη Πληροφορίας Πλαισίου με χρήση Βελτιστοποίησης Σμήνους Σωματιδίων

Στο εδάφιο αυτό περιγράφουμε πώς ο αλγόριθμος Βελτιστοποίησης Σμήνους Σωματιδίων μπορεί να χρησιμοποιηθεί για την Ανακάλυψη Πληροφορίας Πλαισίου (Context Discovery Problem – CDP) [46].

Θεωρούμε ότι εργαζόμαστε σε διακριτό χρόνο και πεπερασμένο δισδιάστατο χώρο. Θεωρούμε ένα σύνολο από N κινητούς κόμβους, οι οποίοι αντιστοιχούν σε ένα σμήνος σωματιδίων, και ένα σύνολο από M κινητές πηγές (δηλαδή αισθητήρες που ανιχνεύουν πληροφορία πλαισίου), οι οποίες αντιστοιχούν στις πιθανές λύσεις του αλγόριθμου PSO. Κάθε πηγή ανιχνεύει πληροφορία πλαισίου με συχνότητα ανίχνευσης q. Κάθε κόμβος επιχειρεί να μετακινηθεί σε μια περιοχή όπου υπάρχει μία τουλάχιστον πηγή που να έχει φρέσκο περιεχόμενο (πρόσφατη πληροφορία πλαισίου). Εναλλακτικά, ένας κόμβος προσπαθεί να εντοπίσει περιοχές όπου υπάρχουν άλλοι κόμβοι που μεταφέρουν πρόσφατη πληροφορία πλαισίου ή πληροφορία πλαισίου καλύτερης ποιότητας από αυτήν που ο ίδιος μεταφέρει. Επίσης, ένας κόμβος δεν γνωρίζει τη θέση μιας πηγής και το σμήνος δεν γνωρίζει τον αριθμό των πηγών. Οι κόμβοι συνεχίζουν την αναζήτηση μέχρι τον εντοπισμό όλων των πηγών ή μέχρι που όλοι να μεταφέρουν έγκυρη πληροφορία πλαισίου.

Το εξεταζόμενο πρόβλημα Ανακάλυψης Πληροφορίας Πλαισίου είναι ένα δισδιάστατο πρόβλημα στη Βελτιστοποίηση Σμήνους Σωματιδίων ($n_x = 2$, βλέπε εδάφιο 1.3). Αφορά τη δισδιάστατη πληροφορία θέσης των πηγών / κόμβων που μεταφέρουν πληροφορία πλαισίου. Θεωρούμε ότι ένας κόμβος ή πηγή μπορεί να ανιχνεύσει ή να μεταδώσει πληροφορία σε οποιοδήποτε γειτονικό κόμβο ή πηγή σε μια περιοχή ακτίνας R. Ένας κόμβος i κινείται προς ένα γειτονικό κόμβο j, αν ο j μεταφέρει πιο πρόσφατη πληροφορία πλαισίου από τον κόμβο i.

Η τιμή της ποιότητας πλαισίου $g_i(\boldsymbol{y},t)$ καθορίζει την επιθυμία του κόμβου i για αναζήτηση πρόσφατης ή τουλάχιστον καλύτερης ποιότητας πληροφορίας πλαισίου. Η ποιότητα πλαισίου $g_i(\boldsymbol{y},t)$ αντιστοιχεί στη συνάρτηση βελτιστοποίησης f της Βελτιστοποίησης Σμήνους Σωματιδίων. Επομένως, ένας κόμβος i επιχειρεί ταυτόχρονα

- να ελαχιστοποιήσει την τιμή της $g_i(\boldsymbol{y},t)$ σε κάθε χρονική στιγμή t και
- να μεγιστοποιήσει το χρονικό διάστημα στο οποίο μεταφέρει πρόσφατη πληροφορία πλαισίου, δηλαδή όταν $g_i(\boldsymbol{y},t) < \theta_y$.

Η τιμή της ποιότητας πληροφορίας πλαισίου $g_i(\boldsymbol{y},t)$ αυξάνεται με την πάροδο του χρόνου. Επομένως, ένας κόμβος i προσπαθεί να ελαχιστοποιήσει την τιμή αυτή, καθορίζοντας σε κάθε βήμα δυναμικά τη θέση του με βάση τις θέσεις *pbest* και *lbest* (βλέπε εδάφιο 1.3).

Για το εξεταζόμενο πρόβλημα, οι εξισώσεις (1.1), (1.2) μπορούν να γραφούν ως εξής:

$$v_{ij}(t+1) = w \cdot v_{ij}(t) + c_1 r_1 \left(x_{ij}^{\#} - x_{ij} \right) + c_2 r_2 \left(x_{ij}^{*} - x_{ij} \right) \tag{4.2}$$

$$x_{ij}(t+1) = x_{ij}(t) + v_{ij}(t) \tag{4.3}$$

όπου $j = 1,2$, $\boldsymbol{x}_i^{\#} = \left[x_{ij}^{\#} \right]$ η θέση *pbest* και $\boldsymbol{x}_i^{*} = \left[x_{ij}^{*} \right]$ η θέση *lbest* του κόμβου i το χρονικό βήμα t.

Έστω N_i οι δείκτες των γειτονικών κόμβων του κόμβου i και $g_i^{\#}(\boldsymbol{y})$ η ατομική βέλτιστη τιμή (*pbest*) της συνάρτησης βελτιστοποίησης για τον κόμβο i, το χρονικό βήμα t. Αν $g_i^{\#}(\boldsymbol{y}) > \min_l \left\{ g_l(\boldsymbol{y},t) \right\}$, $l \in \{N_i\}$, τότε ισχύουν οι σχέσεις:

$$g_i^{\#}(\boldsymbol{y}) = \min_l \left\{ g_l(\boldsymbol{y},t) \right\}, \quad l \in \{N_i\} \cup \{i\} \tag{4.4}$$

$$\boldsymbol{x}_i^{\#} = \boldsymbol{x}_e : \ e = \arg\min_l \left\{ g_l(\boldsymbol{y},t) \wedge \left(g_l(\boldsymbol{y},t) < g_i(\boldsymbol{y},t) \right) \right\}, \ l \in \{N_i\} \tag{4.5}$$

Το διάνυσμα $\left(\boldsymbol{x}_i^{\#} - \boldsymbol{x}_i\right)$ αποτελεί τη γνωσιακή συνιστώσα του κόμβου i, ο οποίος έλκεται από τον κόμβο e. Ο κόμβος e ονομάζεται οδηγός (leader) του κόμβου i.

Επίσης, συμβολίζουμε με $g_{N_i}(\boldsymbol{y},t)$ την τοπική συνάρτηση βελτιστοποίησης (local fitness) της γειτονιάς του κόμβου i, το χρονικό βήμα t:

$$g_{N_i}(\boldsymbol{y},t) = \frac{1}{|N_i|} \sum_{j=1}^{|N_i|} g_j(\boldsymbol{y},t) \tag{4.6}$$

Αν $g_{N_i}(\boldsymbol{y},t) < \theta_{iy}$ ο κόμβος i μπορεί να αποφασίσει να μην απομακρυνθεί από τη γειτονιά του, καθώς σε αυτήν υπάρχουν κόμβοι με φρέσκο περιεχόμενο. Αν $g_i^*(\boldsymbol{y})$ η τοπική βέλτιστη τιμή (*lbest*) της συνάρτησης βελτιστοποίησης για τον κόμβο i, το χρονικό βήμα t, τότε

$$\boldsymbol{x}_i^*(t+1) = \begin{cases} \boldsymbol{x}_i^*(t), & g_{N_i}(\boldsymbol{y},t+1) \geq g_i^*(\boldsymbol{y}) \\ \boldsymbol{x}_i(t+1), & g_{N_i}(\boldsymbol{y},t+1) < g_i^*(\boldsymbol{y}) \end{cases} \tag{4.7}$$

Το διάνυσμα $\left(\boldsymbol{x}_i^* - \boldsymbol{x}_i\right)$ αναφέρεται στην κοινωνική συνιστώσα του κόμβου i και δηλώνει την έλξη του κόμβου i προς τη γειτονιά του. Καθώς η τιμή $g_i^*(\boldsymbol{y})$ αυξάνεται με το πέρασμα του χρόνου, ο κόμβος i θα πρέπει δυναμικά να υπολογίζει την τοπική βέλτιστη θέση $\boldsymbol{x}_i^*$.

Ο Πίνακας 3 απεικονίζει την αντιστοιχία μεταξύ της Βελτιστοποίησης Σμήνους Σωματιδίων (PSO) και του Προβλήματος Ανακάλυψης Πληροφορίας Πλαισίου (CDP). Παρατηρούμε ότι η συνάρτηση βελτιστοποίησης f στην PSO εξαρτάται μόνο από το διάνυσμα τιμών $\boldsymbol{x}_i$ και όχι από το χρόνο. Το ίδιο ισχύει και για τις θέσεις *pbest* και *lbest*. Αντίθετα, στο πρόβλημα ανακάλυψης πλαισίου, η αντίστοιχη συνάρτηση $g_i(\boldsymbol{y},t)$ εξαρτάται από το χρόνο, καθώς η εγκυρότητα του $\boldsymbol{y}$ ελαττώνεται με την πάροδο του χρόνου. Επίσης, οι τιμές $g_i^{\#}(\boldsymbol{y})$, $g_i^*(\boldsymbol{y})$ αυξάνονται με το χρόνο.

Πίνακας 3: Αντιστοιχία PSO - CDP

Έννοιες PSO	Χρονική Μεταβλητότητα		**Έννοιες CDP**
Σμήνος N σωματιδίων	όχι	όχι	Σύνολο N κινητών κόμβων
Σωματίδιο i	-	-	Κόμβος i
Χώρος τιμών προβλήματος	όχι	ναι	Πληροφορία πλαισίου $\boldsymbol{y}$
Τοπική βέλτιστη τιμή x_i^*	όχι	ναι	Πηγή στη θέση x_i^*
Ατομική βέλτιστη τιμή $x_i^\#$	όχι	ναι	Κόμβος με ενημερωμένο πλαίσιο στη θέση $x_i^\#$
Αριθμός βέλτιστων	όχι	όχι	Αριθμός πηγών M
Συνάρτηση βελτιστοποίησης f	όχι	ναι	Ποιότητα πλαισίου $g_i(\boldsymbol{y},t)$
pbest $x_i^\#$	όχι	ναι	Θέση του γειτονικού κόμβου e που μεγιστοποιεί την $g_e(\boldsymbol{y},t)$
lbest x_i^*	όχι	ναι	Θέση στη γειτονιά του κόμβου i που μεγιστοποιεί την $g_{N_i}(\boldsymbol{y},t)$

4.4. Εφαρμογή της Θεωρίας Βέλτιστης Παύσης

Στο πρόβλημα που εξετάζουμε, θεωρούμε ότι οι κόμβοι διαθέτουν περιορισμένες δυνατότητες επεξεργασίας δεδομένων, καθώς και περιορισμένη διαθέσιμη ενέργεια. Αναζητούμε λοιπόν τρόπους για τον περιορισμό του ενεργειακού αποτυπώματος κάθε κόμβου (σωματιδίου) που συμμετέχει στην κίνηση του σμήνους. Στη μελέτη αυτή εφαρμόζουμε τη Θεωρία Βέλτιστης Παύσης ως μέθοδο για τη βελτίωση των ενεργειακών χαρακτηριστικών των κόμβων και εξετάζουμε την επίδρασή της στη μέση ποιότητα πληροφορίας πλαισίου του σμήνους, καθώς και στη μέση απόσταση που διανύει ένας κόμβος.

Θεωρούμε ότι αρχικά οι κόμβοι του σμήνους παραμένουν ακίνητοι, ενώ οι πηγές ακολουθούν τυχαία κίνηση. Κάθε κόμβος έχει τη δυνατότητα απόφασης αν θα πρέπει να «ακολουθήσει» σε κίνηση σμήνους έναν άλλο κόμβο ή πηγή με καλύτερη ποιότητα πληροφορίας πλαισίου μέσα σε διάστημα n_0 διακριτών χρονικών στιγμών. Στο διάστημα αυτό, ο κόμβος θα πρέπει να κρίνει ποια είναι η βέλτιστη χρονική στιγμή για να ξεκινήσει ή να μεταβάλει την κίνησή του. Το πρόβλημα αυτό μπορεί να αντιστοιχηθεί με το γνωστό Πρόβλημα της Γραμματέως (βλέπε εδάφιο 2.3.1) [29]. Εδώ, χρησιμοποιούμε την προσέγγιση του Προβλήματος της Γραμματέως, όπως παρουσιάζεται στο εδάφιο 2.3.3 [34], [35].

Έτσι, ένα σωματίδιο i σε κάθε χρονική στιγμή $k=1,2,\ldots,n_o$ εξετάζει τις τιμές ποιότητας περιεχομένου των σωματιδίων ή και πηγών της γειτονιάς του και υπολογίζει την καλύτερη (μικρότερη) τιμή, έστω $g_i^{\triangle}(k)$. Ο αλγόριθμος λήψης απόφασης είναι ο εξής:

- Για $k=1,\ldots,\sqrt{n_o}-1$ υπολόγισε τη μικρότερη τιμή $g_{i,\min}^{\triangle}=\min\left\{g_i^{\triangle}(1),\ldots,g_i^{\triangle}\left(\sqrt{n_o}-1\right)\right\}$.
- Για $k=\sqrt{n_o},\ldots,n_o-1$, αν $g_i^{\triangle}(k)<g_{i,\min}^{\triangle}$ τότε σταμάτα τη διαδικασία λήψης απόφασης και λάβε ως ελάχιστη τιμή την $g_i^{\triangle}(k)$, διαφορετικά συνέχισε.
- Για $k=n_o$, αφού δεν έχεις βρει μικρότερη τιμή από την $g_{i,\min}^{\triangle}$, δέξου ως τιμή την $g_i^{\triangle}(n_o)$.

Κατά τη διαδικασία λήψης της απόφασης οι τιμές $g_i^{\triangle}(k)$ κανονικοποιούνται στο διάστημα $[0,1]$, καθώς θεωρούμε ότι ακολουθούν ομοιόμορφη κατανομή.

Στον υπολογισμό της κίνησης του σμήνους, η ποσότητα $g_i^{\triangle}(\boldsymbol{y},t)$ αντικαθιστά την $g_i^{\#}(\boldsymbol{y},t)$ στις εξισώσεις (4.4)-(4.5).

Κεφάλαιο 5: Αλγόριθμοι υλοποίησης

5.1. Εισαγωγή

Στα πλαίσια της μελέτης του εξεταζόμενου συστήματος κινητού και κατανεμημένου υπολογισμού (βλέπε εδάφιο 4.3) έχουν αναπτυχθεί δύο βασικοί αλγόριθμοι:

- ο αλγόριθμος Βελτιστοποίησης Σμήνους Σωματιδίων (PSO) και
- ο αλγόριθμος Βελτιστοποίησης Σμήνους Σωματιδίων με εφαρμογή της Θεωρίας Βέλτιστης Παύσης (Optimal Stopping PSO – OSPSO)

Στο κεφάλαιο αυτό περιγράφονται οι δύο αυτοί βασικοί αλγόριθμοι, καθώς και οι παραλλαγές τους. Τα πειραματικά δεδομένα από την προσομοίωση του συστήματος αυτού παρουσιάζονται στο Κεφάλαιο 6.

5.2. Βασικός αλγόριθμος Βελτιστοποίησης Σμήνους Σωματιδίων (PSO)

5.2.1. Παράμετροι

Ο βασικός αλγόριθμος υλοποίησης παρουσιάζεται στον Αλγόριθμο 3. Οι κύριες παράμετροί του (οι οποίες ορίζονται στη σειρά 1) είναι οι εξής:

- N ο αριθμός των σωματιδίων (μη αισθητήριων κόμβων) του σμήνους και M ο αριθμός των πηγών πληροφορίας (αισθητήριων κόμβων).
- $x_{min}, x_{max}, y_{min}, y_{max}$ οι συντεταγμένες του χώρου όπου κινούνται τα σωματίδια και οι πηγές. Οι διαστάσεις του χώρου κίνησης δίνονται από τις σχέσεις:

$$\begin{aligned} L_x &= x_{\max} - x_{\min} + 1 \\ L_y &= y_{\max} - y_{\min} + 1 \end{aligned} \tag{5.1}$$

- $R = 0.001 \cdot L_x \cdot L_y$ η ακτίνα ανίχνευσης / μετάδοσης σωματιδίων και πηγών.
- q η συχνότητα ανίχνευσης (ανανέωσης) πληροφορίας πλαισίου των πηγών.

- $v_{\min}, v_{\max}$ οι οριακές ταχύτητες πηγών και σωματιδίων κατά την εκτέλεση τυχαίας κίνησης.
- c_1, c_2 οι σταθερές επιτάχυνσης της εξίσωσης μεταβολής της ταχύτητας του αλγόριθμου Βελτιστοποίησης Σμήνους Σωματιδίων (βλέπε εδάφιο 1.3).
- $w_{\min}, w_{\max}$ η ελάχιστη και η μέγιστη τιμή του βάρους αδράνειας w.
- Η μέθοδος μεταβολής του βάρους αδράνειας w. Οι εξεταζόμενες μέθοδοι παρουσιάζονται στο εδάφιο 5.3.
- Το μοντέλο μεταβολής του βάρους αδράνειας w, το οποίο αναλύεται στο εδάφιο 5.2.3.
- $\theta_{iy} = \theta_y$ το κατώφλι της ποιότητας πληροφορίας πλαισίου $g_i(\boldsymbol{y}, t)$ ενός σωματιδίου i. Θεωρούμε ότι το κατώφλι είναι κοινό για όλα τα σωματίδια.
- t_0 ο μέγιστος αριθμός επαναλήψεων του αλγόριθμου.

Αλγόριθμος 3: Βασικός αλγόριθμος PSO

1. **set** $N, M, x_{min}, x_{max}, y_{min}, y_{max}, R, q, v_{min}, v_{max}, c_1, c_2, w_{\min}, w_{\max}, \theta_y, t_0$
2. **set** w update method
3. **set** w update model *// (universal or per particle)*
4. $t \leftarrow 0$
5. $t_i \leftarrow 0$ *// set time (iteration) per particle*
6. **for** $i = 1:N$
7. **set** random $\boldsymbol{x}_i(t),\ x_{i1}(t) \in [x_{\min}, x_{\max}],\ x_{i2}(t) \in [y_{\min}, y_{\max}],\ \theta_{iy} = \theta_y$
8. **set** random $v_i(t),\ v_i(t) \sim U(v_{\min}, v_{\max})$
9. **set** $g_i(\boldsymbol{y}) = 2\theta_y,\ g_i^*(\boldsymbol{y}) = 2\theta_y,\ g_i^{\#}(\boldsymbol{y}) = 2\theta_y$
10. **set** $w_i = w_{\max}$
11. **end**
12. **for** $s = 1:M$
13. **set** random $\boldsymbol{x}_s(t),\ x_{s1}(t) \in [x_{\min}, x_{\max}],\ x_{s2}(t) \in [y_{\min}, y_{\max}]$
14. **set** $g_s(\boldsymbol{y}) = 0$
15. **end**
16. **for** $i = 1:N$
17. $g_i^*(\boldsymbol{y}) \leftarrow \frac{1}{|N_i|} \sum_{j=1}^{|N_i|} g_j(\boldsymbol{y}, t)$
18. $\boldsymbol{x}_i^* \leftarrow \boldsymbol{x}_i(t)$
19. $g_i^{\#}(\boldsymbol{y}) \leftarrow \min_l \{g_l(\boldsymbol{y}, t)\},\ l \in \{N_i\} \cup \{i\}$

20. $\boldsymbol{x}_i^{\#} \leftarrow \boldsymbol{x}_e : \; e = \arg\min_l \left\{ g_l\left(\boldsymbol{y},t\right) \right\}, \; l \in \left\{ N_i \right\} \cup \left\{ i \right\}, \; leader_i \leftarrow e$
21. **set** random unary vectors r_1, r_2
22. $\boldsymbol{v}_i\left(t+1\right) \leftarrow w_i \cdot \boldsymbol{v}_i(t) + c_1 r_1 \left(\boldsymbol{x}_i^{\#} - \boldsymbol{x}_i \right) + c_2 r_2 \left(\boldsymbol{x}_i^{*} - \boldsymbol{x}_i \right)$
23. $\boldsymbol{x}_i\left(t+1\right) \leftarrow \boldsymbol{x}_i(t) + \boldsymbol{v}_i(t)$
24. **end**
25. **set** $t_g \leftarrow 0$ *// set context update time*
26. **while** $(t < t_0)$ **do**
27. $t \leftarrow t+1, \; t_g \leftarrow t_g + 1$
28. **for** $s = 1 : M$
29. calculate random $\boldsymbol{x}_s(t)$
30. **if** $0 \le g_s(\boldsymbol{y}) < \theta_y$ then $g_s(\boldsymbol{y}) \leftarrow g_s(\boldsymbol{y}) + 1$
31. **if** $t_g = \dfrac{1}{q}$ then $g_s(\boldsymbol{y}) \leftarrow 0, \; t_g \leftarrow 0$
32. **end**
33. **for** $i = 1 : N$
34. **if** $0 \le g_i(\boldsymbol{y}) < \theta_y$ **then** $g_i(\boldsymbol{y}) \leftarrow g_i(\boldsymbol{y}) + 1$
35. $g_{N_i}(\boldsymbol{y},t) \leftarrow \dfrac{1}{\left|N_i\right|} \sum_{j=1}^{\left|N_i\right|} g_j\left(\boldsymbol{y},t\right)$
36. **if** $g_{N_i}(\boldsymbol{y},t) < g_i^{*}(\boldsymbol{y})$ **then**
37. $\boldsymbol{x}_i^{*} \leftarrow \boldsymbol{x}_i(t)$
38. $g_i^{*}(\boldsymbol{y}) \leftarrow g_{N_i}(\boldsymbol{y},t)$
39. **end**
40. **if** $g_i^{\#}(\boldsymbol{y}) > \min_l \left\{ g_l\left(\boldsymbol{y},t\right) \right\}, \; l \in \left\{ N_i \right\}$ **then**
41. $\boldsymbol{x}_i^{\#} \leftarrow \boldsymbol{x}_e : \; e = \arg\min_l \left\{ g_l\left(\boldsymbol{y},t\right) \wedge \left(\; g_l\left(\boldsymbol{y},t\right) < g_i\left(\boldsymbol{y},t\right) \right) \right\}, \; l \in \left\{ N_i \right\}, \; leader_i \leftarrow e$
42. $g_i^{\#}(\boldsymbol{y}) \leftarrow \min_l \left\{ g_l\left(\boldsymbol{y},t\right) \right\}, \; l \in \left\{ N_i \right\}$
43. **end**
44. **if** $(g_i(\boldsymbol{y},t) \ge \theta_y)$ **and** $(g_i(\boldsymbol{y},t-1) \ge \theta_y)$ **then** *// continue random movement*
45. calculate random $x_i(t)$
46. **end**
47. **if** $(g_i(\boldsymbol{y},t) \ge \theta_y)$ **and** $(g_i(\boldsymbol{y},t-1) < \theta_y)$ **then** *// switch from swarm to random movement*
48. calculate random $x_i(t)$
49. $t_i \leftarrow 0$
50. **end**
51. **if** $(g_i(\boldsymbol{y},t) < \theta_y)$ **and** $(g_i(\boldsymbol{y},t-1) \ge \theta_y)$ **then** *// switch from random to swarm movement*
52. $\boldsymbol{v}_i(t) \leftarrow \boldsymbol{x}_i(t) - \boldsymbol{x}_i(t-1)$
53. $t_i \leftarrow t_i + 1$
54. **if** ((w update model) == "per particle") **then** $t \leftarrow t_i$ *// per particle calculation*
55. calculate $w_i(t)$

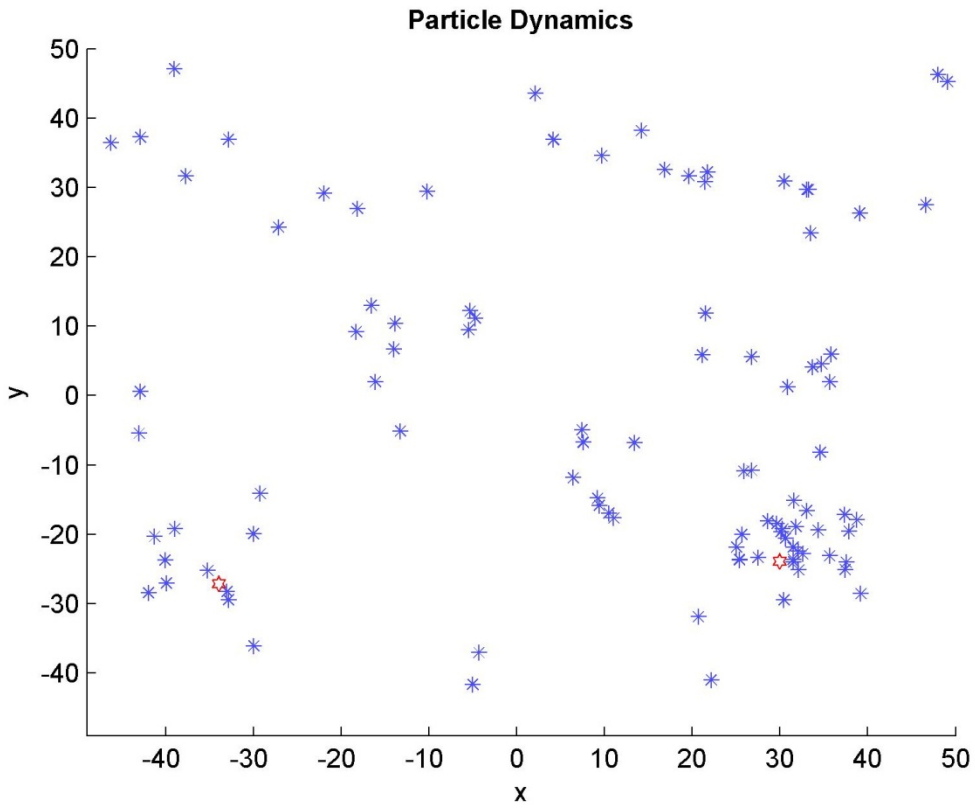
56. $\boldsymbol{v}_i(t+1) \leftarrow w_i(t)\cdot\boldsymbol{v}_i(t) + c_1 r_1\left(\boldsymbol{x}_i^{\#} - \boldsymbol{x}_i\right) + c_2 r_2\left(\boldsymbol{x}_i^{*} - \boldsymbol{x}_i\right)$
57. $\boldsymbol{x}_i(t+1) \leftarrow \boldsymbol{x}_i(t) + \boldsymbol{v}_i(t)$
58. **end**
59. **if** ($g_i(\boldsymbol{y},t) < \theta_y$) **and** ($g_i(\boldsymbol{y},t-1) < \theta_y$) **then** *// continue swarm movement*
60. $t_i \leftarrow t_i + 1$
61. **if** ((w update model) == "per particle") **then** $t \leftarrow t_i$ *// per particle calculation*
62. calculate $w_i(t)$
63. $\boldsymbol{v}_i(t+1) \leftarrow w_i(t)\cdot\boldsymbol{v}_i(t) + c_1 r_1\left(\boldsymbol{x}_i^{\#} - \boldsymbol{x}_i\right) + c_2 r_2\left(\boldsymbol{x}_i^{*} - \boldsymbol{x}_i\right)$
64. $\boldsymbol{x}_i(t+1) \leftarrow \boldsymbol{x}_i(t) + \boldsymbol{v}_i(t)$
65. **end**
66. **end** *// for*
67. $\bar{g}(t) \leftarrow \frac{1}{N}\sum_{i=1}^{N} g_i(t)$
68. **end** *// while*
69. $\bar{g}(t_0) \leftarrow \frac{1}{N}\sum_{i=1}^{N} g_i(t_0)$
70. $\bar{g} \leftarrow \frac{1}{t_0}\sum_{j=1}^{t_0} \bar{g}(t_j)$

5.2.2. Κίνηση σωματιδίων και πηγών

Οι θέσεις της τυχαίας κίνησης πηγών και σωματιδίων υπολογίζονται από το μοντέλο τυχαίας κίνησης *Random Waypoint Mobility Model* [48], [49]. Το μοντέλο αυτό χρησιμοποιεί ως παραμέτρους τις μεταβλητές $x_{min}, x_{max}, y_{min}, y_{max}, v_{\min}, v_{\max}, N, M, t_0$. Οι πηγές ακολουθούν τυχαία κίνηση σε όλη τη διάρκεια της εκτέλεσης του αλγόριθμου. Ο υπολογισμός της τυχαίας κίνησης των πηγών γίνεται μια φορά στην αρχή της εκτέλεσης του αλγόριθμου (γραμμές 12-15). Ο υπολογισμός της τυχαίας κίνησης των σωματιδίων πραγματοποιείται τόσο στην αρχή (γραμμές 6-11), όσο και κατά τη διάρκεια της εκτέλεσης του αλγόριθμου (γραμμές 45, 48), όταν αυτό απαιτείται.

Η κίνηση των σωματιδίων του σμήνους περιγράφεται ως εξής: Τα σωματίδια κινούνται αρχικά τυχαία μέχρι να συναντήσουν πηγές ή άλλα σωματίδια με καλύτερη ποιότητα πληροφορίας πλαισίου, οπότε και μεταπίπτουν σε κίνηση σμήνους (γραμμές 51-58). Αντίστροφα, ένα σωματίδιο μεταπίπτει από κίνηση σμήνους σε τυχαία κίνηση, όταν η ποιότητα πληροφορίας πλαισίου του είναι απαρχαιωμένη και ταυτόχρονα δεν υπάρχουν γειτονικά σωματίδια ή πηγές με καλύτερο περιεχόμενο. Όταν ένα σωματίδιο μεταπίπτει από κίνηση σμήνους σε τυχαία κίνηση (γραμμές 47-50), οι θέσεις τυχαίας κίνησής του

υπολογίζονται με βάση την τρέχουσα θέση του στο χώρο. Στο Σχήμα 6 απεικονίζεται ένα στιγμιότυπο εκτέλεσης του αλγόριθμου που δείχνει την κίνηση στο χώρο ενός σμήνους 100 σωματιδίων και 2 πηγών.



Σχήμα 6: PSO. Στιγμιότυπο της κίνησης σωματιδίων και πηγών στο χώρο

Η κίνηση του σμήνους υπολογίζεται στις γραμμές 59-65 του Αλγόριθμου 3. Οι μεταπτώσεις κίνησης κάθε σωματιδίου i καθορίζονται από την τιμή της συνάρτησης περιεχομένου του $g_i(\boldsymbol{y},t)$ σε συνάρτηση με το κατώφλι $\theta_{iy} = \theta_y$.

Οι θέσεις *lBest* ($\boldsymbol{x}_i^*$) και *pBest* ($\boldsymbol{x}_i^\#$) για κάθε σωματίδιο i για χρονικές στιγμές $t > 1$ (εξισώσεις (4.5), (4.7)) υπολογίζονται στις γραμμές 34-39 και 40-43, αντίστοιχα. Για $t = 1$, ο υπολογισμός τους γίνεται στις γραμμές 17-18 και 19-20, αντίστοιχα.

Συνοπτικά, σε κάθε επανάληψη του αλγόριθμου οι δυνατοί τρόποι κίνησης ενός σωματιδίου i είναι οι εξής:

1. Κίνηση σμήνους:

1.1. Μετάπτωση από τυχαία κίνηση σε κίνηση σμήνους ($g_i(\boldsymbol{y},t) < \theta_y$ και $g_i(\boldsymbol{y},t-1) \geq \theta_y$).

1.2. Συνέχιση κίνησης σμήνους ($g_i(\boldsymbol{y},t) < \theta_y$ και $g_i(\boldsymbol{y},t-1) < \theta_y$).

2. Τυχαία κίνηση:

2.1. Μετάπτωση από κίνηση σμήνους σε τυχαία κίνηση ($g_i(\boldsymbol{y},t) \geq \theta_y$ και $g_i(\boldsymbol{y},t-1) < \theta_y$).

2.2. Συνέχιση τυχαίας κίνησης ($g_i(\boldsymbol{y},t) \geq \theta_y$ και $g_i(\boldsymbol{y},t-1) \geq \theta_y$).

5.2.3. Βάρος αδράνειας

Το βάρος αδράνειας w στην εξίσωση μεταβολής της ταχύτητας ενός σωματιδίου του σμήνους (εξίσωση (4.2) ή (1.6)) μεταβάλλεται δυναμικά σε κάθε επανάληψη του αλγόριθμου, ξεκινώντας από μια μέγιστη αρχική τιμή $w_{\max}$ και καταλήγοντας σε μια ελάχιστη τελική τιμή $w_{\min}$ (βλέπε εδάφιο 1.5.2). Η μεταβολή αυτή καθορίζεται από δύο παραμέτρους: το μοντέλο μεταβολής και τη μέθοδο μεταβολής.

Θεωρούμε ότι υπάρχουν δύο βασικά μοντέλα μεταβολής του βάρους αδράνειας:

- το *καθολικό μοντέλο* (*universal model*), στο οποίο το βάρος αδράνειας $w(t)$ είναι κοινό για όλα τα σωματίδια του σμήνους σε κάθε επανάληψη του αλγόριθμου t.
- το *ατομικό μοντέλο* (*per particle model*), όπου κάθε σωματίδιο έχει το δικό του βάρος αδράνειας $w_i(t_i)$, το οποίο μεταβάλλεται με βάση το χρονικό βήμα συμμετοχής του στην κίνηση του σμήνους t_i.

Στο *καθολικό μοντέλο*, το βάρος αδράνειας w ενός σωματιδίου εξαρτάται από το βήμα του αλγόριθμου t. Όταν ένα σωματίδιο μεταπίπτει από τυχαία κίνηση σε κίνηση σμήνους, οι εξισώσεις κίνησής του υιοθετούν την τρέχουσα τιμή του βάρους αδράνειας του σμήνους.

Στο *ατομικό μοντέλο*, το ατομικό βάρος αδράνειας w_i μεταβάλλεται μόνο όταν το σωματίδιο i συμμετέχει στην κίνηση του σμήνους και δεν εκτελεί τυχαία κίνηση (ή παραμένει σε αδράνεια, όπως συμβαίνει στην περίπτωση της εφαρμογής της Θεωρίας Βέλτιστης Παύσης). Όταν ένα σωματίδιο i μεταπίπτει από τυχαία κίνηση σε κίνηση σμήνους, το βάρος αδράνειας του w_i λαμβάνει την αρχική τιμή, δηλαδή $w_i = w_{\max}$.

Στο εξεταζόμενο πρόβλημα, όπου θεωρούμε ότι κάθε σωματίδιο (κόμβος) είναι αυτόνομο, με δικές του δυνατότητες κίνησης, ανίχνευσης και λήψης αποφάσεων, η επιλογή του ατομικού μοντέλου μεταβολής του βάρους αδράνειας κρίθηκε ότι είναι η πιο κατάλληλη.

Για το δυναμικό υπολογισμό της τιμής του βάρους αδράνειας υλοποιήθηκαν διάφορες μέθοδοι μεταβολής, οι οποίες παρουσιάζονται αναλυτικά στην ενότητα 5.3.

5.2.4. Αποκοπή ταχύτητας και αποκοπή θέσης

Στο βασικό αλγόριθμο PSO έχουν ενσωματωθεί συναρτήσεις για την εφαρμογή των μεθόδων αποκοπής ταχύτητας (velocity clamping) και αποκοπής θέσης (position clamping).

Η αποκοπή ταχύτητας καθορίζει την αναλυτικότητα του χώρου αναζήτησης (βλέπε εδάφιο 1.5.1) και μπορεί να χρησιμοποιηθεί στον αλγόριθμο PSO παράλληλα με την ενσωμάτωση του βάρους αδράνειας στις εξισώσεις κίνησης.

Επίσης, η αποκοπή θέσης περιορίζει την κίνηση των σωματιδίων σε ένα καθορισμένο χωρικό πλαίσιο και είναι χρήσιμη σε εφαρμογές που προϋποθέτουν κάτι τέτοιο.

Στο εξεταζόμενο σύστημα, δεν κρίθηκε σκόπιμη η εφαρμογή των μεθόδων αυτών, αλλά υπάρχει ως δυνατότητα στους υλοποιημένους αλγόριθμους.

5.3. Μέθοδοι μεταβολής του βάρους αδράνειας

Οι μέθοδοι που έχουν υλοποιηθεί είναι οι εξής (βλέπε εδάφιο 1.5.2):

(1) Γραμμικά ελαττούμενο βάρος αδράνειας (Linear decreasing inertia weight).

Το βάρος αδράνειας δίνεται από τον τύπο (καθολικό μοντέλο):

$$w = w_{\max} - \frac{w_{\max} - w_{\min}}{t_0} \cdot t \tag{5.2}$$

όπου t_0 το μέγιστο πλήθος επαναλήψεων και $w_{\max} = 0.9$, $w_{\min} = 0.4$.

Στην περίπτωση του ατομικού μοντέλου ισχύει:

$$w_i = w_{\max} - \frac{w_{\max} - w_{\min}}{t_0} \cdot t_i \tag{5.3}$$

όπου t_i ο χρόνος συμμετοχής του σωματιδίου i στο σμήνος.

(2) Μη γραμμικά ελαττούμενο βάρος αδράνειας (Non-linear decreasing inertia weight).

Το βάρος αδράνειας δίνεται από τον τύπο (καθολικό μοντέλο):

$$w(t+1) = \frac{\left(w(0) - w_{\min}\right)\left(t_0 - t\right)}{t_0 + w_{\min}} \tag{5.4}$$

όπου $w(0) = w_{\max}$ και το w ουσιαστικά μεταβάλλεται μεταξύ $w_{\max} - w_{\min}$ και 0 (βλέπε Σχήμα 7).

Στην περίπτωση του ατομικού μοντέλου ισχύει:

$$w_i\left(t_i + 1\right) = \frac{\left(w_i\left(0\right) - w_{\min}\right)\left(t_0 - t_i\right)}{t_0 + w_{\min}} \tag{5.5}$$

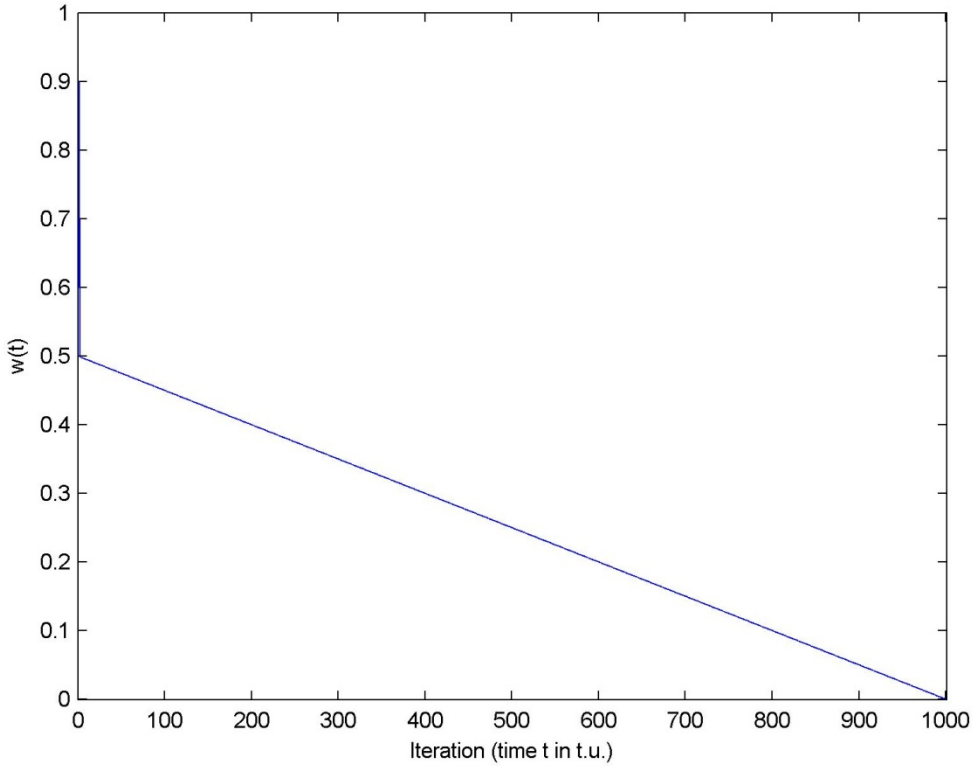
όπου $w_i\left(0\right) = w_{\max}$.

(3) Μη γραμμικά ελαττούμενο βάρος αδράνειας, μέθοδος 2 (Non-linear decreasing inertia weight 2).

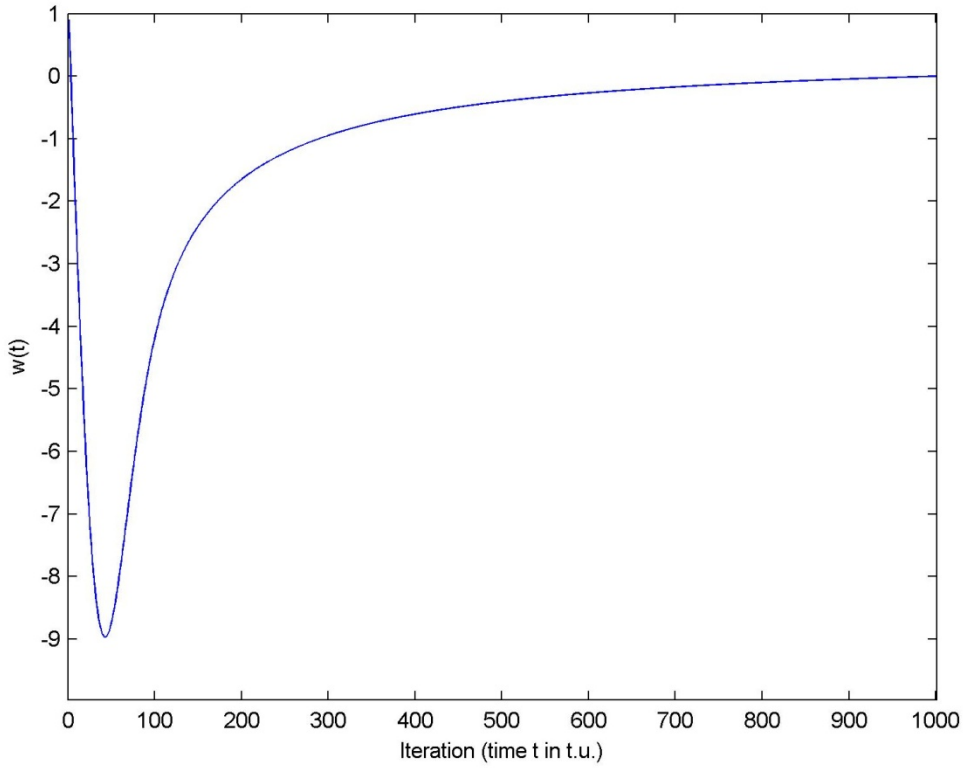
Το w υπολογίζεται από τη σχέση:

$$w(t+1) = \frac{\left(w(t) - w_{\min}\right)\left(t_0 - t\right)}{t_0 + w_{\min}} \tag{5.6}$$

Η χρήση του τύπου (5.6) για τον υπολογισμό του βάρους αδράνειας οδηγεί σε λάθος αποτελέσματα (ο αλγόριθμος δεν συγκλίνει), καθώς τα w που υπολογίζονται υπερβαίνουν το $\left|w_{\max}\right|$ (βλέπε Σχήμα 8), με αποτέλεσμα τα σωματίδια του σμήνους να απομακρύνονται από το γήπεδο. Η μέθοδος αυτή για να λειτουργήσει θα πρέπει να χρησιμοποιηθεί σε συνδυασμό με αποκοπή ταχύτητας (velocity clamping).



Σχήμα 7: Μέθοδος μεταβολής βάρους αδράνειας "non-linear decreasing"



Σχήμα 8: Μέθοδος μεταβολής βάρους αδράνειας "non-linear decreasing 2"

(4) Τυχαίο βάρος αδράνειας (Random inertia weight).

Και για τα δύο μοντέλα μεταβολής το βάρος αδράνειας υπολογίζεται από τη σχέση [10]:

$$w = 0.5 + \frac{1}{2}\text{rand}(w_{\min}, w_{\max}) \tag{5.7}$$

(5) Χαοτικό ελαττούμενο βάρος αδράνειας (Chaotic decresing inertia weight).

Για το καθολικό μοντέλο το w υπολογίζεται από τη σχέση (Σχήμα 9) [17]:

$$w = \left(w_{\max} - w_{\min}\right)\frac{t_0 - t}{t_0} + w_{\min} \cdot z \tag{5.8}$$

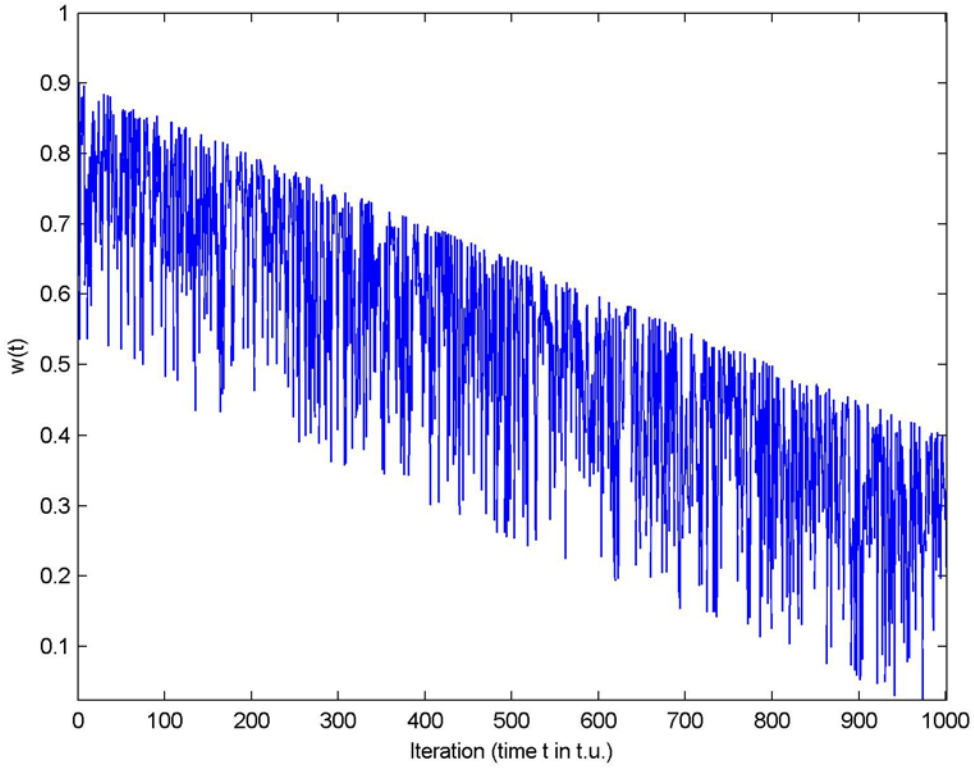
όπου z η λογιστική απεικόνιση:

$$z = 4 \cdot z \cdot \left(1 - z\right) \tag{5.9}$$

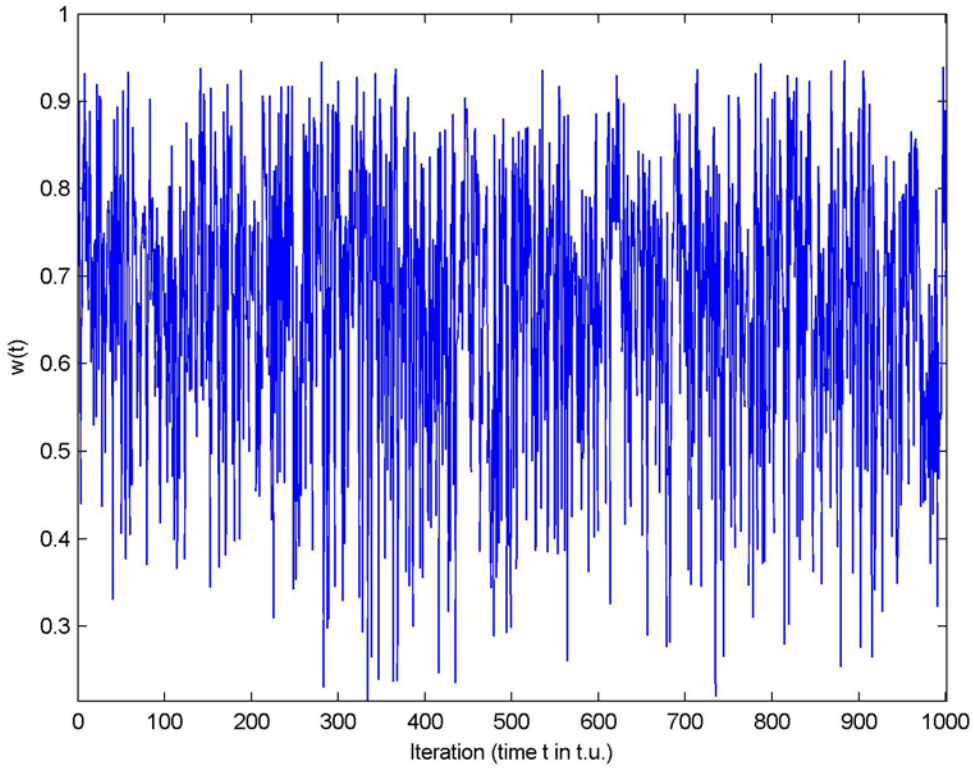
με $z = \text{rand}\left(0,1\right)$.

Η αντίστοιχη σχέση για το ατομικό μοντέλο είναι:

$$w_i = \left(w_{\max} - w_{\min}\right)\frac{t_0 - t_i}{t_0} + w_{\min} \cdot z \tag{5.10}$$



Σχήμα 9: Μέθοδος μεταβολής βάρους αδράνειας "chaotic"



Σχήμα 10: Μέθοδος μεταβολής βάρους αδράνειας "chaotic random"

(6) Χαοτικό τυχαίο βάρος αδράνειας (Chaotic random inertia weight).

Το w υπολογίζεται από τη σχέση [17]:

$$w = 0.5 \cdot \text{rand}\left(w_{\min}, w_{\max}\right) + 0.5z \tag{5.11}$$

όπου το z δίνεται από τη λογιστική απεικόνιση της σχέσης (5.9). Η απεικόνιση μιας ενδεικτικής μεταβολής των τιμών του βάρους αδράνειας για τη μέθοδο αυτή δίνεται στο Σχήμα 10.

(7) Ελαττούμενο βάρος αδράνειας σιγμοειδούς συνάρτησης. Καθολικό μοντέλο. (Sigmoid function decreasing inertia weight – universal).

Για κάθε σωματίδιο του σμήνους η μέση απόστασή του από τα υπόλοιπα d_i σε κάθε χρονική στιγμή (επανάληψη του αλγόριθμου) δίνεται από τη σχέση [16]:

$$d_i = \frac{1}{N-1} \sum_{j=1, j\neq i}^{N} \sqrt{\sum_{k=1}^{D} \left(x_i^k - x_j^k\right)^2} \tag{5.12}$$

όπου N το πλήθος των σωματιδίων και D ο αριθμός των διαστάσεων του προβλήματος.

Η απόσταση d_i του global best σωματιδίου συμβολίζεται με d_g. Επίσης, οι μεταβλητές $d_{\max}$, $d_{\min}$ αντιστοιχούν στη μέγιστη και στην ελάχιστη μέση απόσταση μεταξύ των σωματιδίων.

Ο «παράγοντας εξέλιξης» (“evolutionary factor”) f ορίζεται ως εξής:

$$f = \frac{d_g - d_{\min}}{d_{\max} - d_{\min}} \tag{5.13}$$

όπου $f \in [0,1]$.

Το βάρος αδράνειας w δίνεται από τη σχέση:

$$w(f) = \frac{1}{1+1.5e^{-2.6f}} \tag{5.14}$$

και $w(f) \in [0.4, 0.9], \ \forall f \in [0,1]$.

(8) Ελαττούμενο βάρος αδράνειας σιγμοειδούς συνάρτησης. Ατομικό μοντέλο. (Sigmoid function decreasing inertia weight – per particle).

Αποτελεί παραλλαγή της μεθόδου (7) που αναπτύχθηκε στα πλαίσια της διπλωματικής αυτής εργασίας. Στην περίπτωση αυτή οι ποσότητες d_i, d_g, $d_{\max}$, $d_{\min}$ υπολογίζονται στη γειτονιά του κάθε σωματιδίου (στην περίπτωσή μας ένας κύκλος ακτίνας R) και το N είναι το πλήθος των γειτόνων κάθε σωματιδίου. Οι σχέσεις (5.13), (5.14) μετασχηματίζονται στις ακόλουθες:

$$f_i = \frac{d_g^i - d_{\min}^i}{d_{\max}^i - d_{\min}^i} \tag{5.15}$$

και

$$w_i(f_i) = \frac{1}{1+1.5e^{-2.6f_i}} \tag{5.16}$$

Στην περίπτωση που το πλήθος των γειτόνων ενός σωματιδίου είναι 0 ή 1, δηλαδή δεν μπορούν να χρησιμοποιηθούν οι σχέσεις (5.15), (5.16), δοκιμάστηκαν δύο παραλλαγές:

i) $w_i = w_{\max}$

ii) $w_i = \text{rand}(w_{\min}, w_{\max})$

Η παραλλαγή (ii) έδωσε καλύτερα πειραματικά αποτελέσματα.

5.4. Παραλλαγή του βασικού αλγόριθμου PSO

Στα πλαίσια της μελέτης του εξεταζόμενου συστήματος αναπτύχθηκε μια παραλλαγή του βασικού αλγόριθμου Βελτιστοποίησης Σμήνους Σωματιδίων (Αλγόριθμος 3). Στην παραλλαγή αυτή, η ατομική βέλτιστη θέση ενός σωματιδίου *pBest* υπολογίζεται λαμβάνοντας υπόψη μόνο το γειτονικό σωματίδιο με την καλύτερη πληροφορία πλαισίου κατά την τρέχουσα χρονική στιγμή, δηλαδή απουσιάζει ο έλεγχος της γραμμής 40 στον Αλγόριθμο 3. Έτσι, ένα σωματίδιο δεν έχει γνώση της προηγούμενης ιστορίας του (των θέσεων που έχει επισκεφθεί) και δυναμικά λαμβάνει αποφάσεις με τα δεδομένα του τρέχοντος χρονικού βήματος.

Επομένως, ο υπολογισμός της θέσης *pBest* ενός σωματιδίου του σμήνους γίνεται με δύο διαφορετικούς τρόπους:

(i) με τον κλασικό τρόπο (*pBest calculation model: standard*) [46], όπως αναφέρεται στις γραμμές 40-43 του Αλγόριθμου 3.

(ii) με έναν εναλλακτικό τρόπο δίχως μνήμη (*pBest calculation model: memoryless*).

Ο εναλλακτικός τρόπος υπολογισμού της θέσης *pBest* ενός σωματιδίου δίνει ικανοποιητικά αποτελέσματα για ορισμένες μεθόδους μεταβολής του βάρους αδράνειας, όπως φαίνεται στην πειραματική μελέτη που παρουσιάζεται στο Κεφάλαιο 6.

5.5. Αλγόριθμοι υλοποίησης της Θεωρίας Βέλτιστης Παύσης

Το τμήμα του αλγόριθμου που ενσωματώνει τη Θεωρία Βέλτιστης Παύσης (OST), και ειδικότερα τη νέα προσέγγιση του Προβλήματος της Γραμματέως (βλέπε εδάφια 2.3.3, 4.4), παρουσιάζεται στον Αλγόριθμο 4. Συγκεκριμένα, για την υλοποίηση OST, οι γραμμές κώδικα 2-39 του αλγόριθμου OS (Αλγόριθμος 4) αντικαθιστούν τις γραμμές 44-65 του βασικού αλγόριθμου PSO (Αλγόριθμος 3). Οι κυριότερες νέες παράμετροι είναι οι ακόλουθες:

- n_0 το διάστημα παρατήρησης (observation interval).
- $\sqrt{n_0}$ η βέλτιστη αποκοπή (optimal cutoff).

Συμβολίζουμε με $g_i^{\triangle}(t)$ την καλύτερη (μικρότερη) τιμή πληροφορίας πλαισίου της γειτονιάς N_i ενός σωματιδίου i το χρονικό βήμα t. Αρχικά $(t=0)$, όλα τα σωματίδια βρίσκονται σε

κατάσταση αδράνειας. Ένα σωματίδιο ακολουθεί άμεσα την κίνηση του σμήνους και δεν εφαρμόζει διάστημα παρατήρησης όταν ισχύει $g_i^{\triangle}(t) = 0$.

Αλγόριθμος 4: Υλοποίηση OS του αλγόριθμου OSPSO

1. **for** $i = 1 : N$
2. $g_i^{\triangle}(\boldsymbol{y}) \leftarrow \min_l \left\{ g_l\left(\boldsymbol{y}, t\right) \right\}, \ \ l \in \left\{ N_i \right\}$
3. **if** $g_i^{\triangle}(\boldsymbol{y}) < g_i$ **then**
4. **if** $g_i^{\triangle}(\boldsymbol{y}) == 0$ **then**
5. set status: swarm motion
6. **else if** $0 < g_i^{\triangle}(\boldsymbol{y}) < \theta_y$ **then**
7. **if** wait_counter(i) < optimal_cutoff **then**
8. set status: wait
9. $G_i = \left[G_i \quad g_i^{\triangle}(\boldsymbol{y}) \right]$
10. wait_counter(i) = wait_counter(i) + 1
11. **else if** optimal_cutoff ≤ wait_counter(i) < observation_interval **then**
12. $g_{i,\min}^{\triangle} = \min\left(G_i\right)$
13. **if** $g_i^{\triangle} < g_{i,\min}^{\triangle}$ **then**
14. wait_counter(i) = 0
15. set status: swarm motion
16. **else**
17. set status: wait
18. wait_counter(i) = wait_counter(i) + 1
19. **end**
20. **else**
21. wait_counter(i) = 0
22. set status: swarm motion
23. **end**
24. **else if** $\theta_y \le g_i^{\triangle}(\boldsymbol{y})$ **then**
25. set status: wait
26. **end**
27. **else**
28. continue motion // motion status unchanged
29. **end** // if
30. **if** (status is swarm motion) **then**
31. $t_i \leftarrow t_i + 1$
32. **if** ((w update model) == “per particle”) **then** $t \leftarrow t_i$ *// per particle calculation*
33. calculate $w_i(t)$
34. $\boldsymbol{v}_i\left(t+1\right) \leftarrow w_i(t) \cdot \boldsymbol{v}_i(t) + c_1 r_1 \left(\boldsymbol{x}_i^{\#} - \boldsymbol{x}_i \right) + c_2 r_2 \left(\boldsymbol{x}_i^{*} - \boldsymbol{x}_i \right)$
35. $\boldsymbol{x}_i\left(t+1\right) \leftarrow \boldsymbol{x}_i(t) + \boldsymbol{v}_i(t)$
36. **end** *// swarm motion*
37. **if** (status is wait) **then**
38. *// do nothing*

```
39.  end   // wait
40. end  // for
```

Μια παραλλαγή της παραπάνω υλοποίησης OST παρουσιάζεται στον Αλγόριθμο 5. Εδώ, ένα σωματίδιο i ακολουθεί άμεσα κίνηση σμήνους αν στη γειτονιά του βρίσκεται σωματίδιο ή πηγή με καλύτερης ποιότητας πληροφορία πλαισίου, δηλαδή όταν ισχύει $0 \le g_i^{\triangle}(t) \le \theta_{threshold}$, όπου $\theta_{threshold}$ ένα κατώφλι ποιότητας πλαισίου. Και πάλι, οι γραμμές κώδικα 2-39 του Αλγόριθμου 5 αντικαθιστούν τις γραμμές 44-65 του βασικού αλγόριθμου PSO.

Αλγόριθμος 5: Παραλλαγή υλοποίησης OS του αλγόριθμου OSPSO

```
1.  for i = 1: N
2.      g_i^△(y) ← min_l {g_l(y,t)},  l ∈ {N_i}
3.     if g_i^△(y) < g_i then
4.       if  0 ≤ g_i^△(y) ≤ θ_threshold  then
5.          set status: swarm motion
6.       else if θ_threshold < g_i^△(y) < θ_y  then
7.          if wait_counter(i)  < optimal_cutoff  then
8.            set status: wait
9.            G_i = [G_i   g_i^△(y)]
10.           wait_counter(i) = wait_counter(i) + 1
11.         else if optimal_cutoff ≤ wait_counter(i)  < observation_interval  then
12.           g_{i,min}^△ = min(G_i)
13.           if  g_i^△ < g_{i,min}^△  then
14.             wait_counter(i) = 0
15.             set status: swarm motion
16.           else
17.             set status: wait
18.             wait_counter(i) = wait_counter(i) + 1
19.           end
20.         else
21.           wait_counter(i) = 0
22.           set status: swarm motion
23.         end
24.      else if θ_y ≤ g_i^△(y)  then
25.         set status: wait
26.      end
27.    else
28.      continue motion     // motion status unchanged
29.    end // if
30.    if (status is swarm motion) then
```

31. $t_i \leftarrow t_i + 1$
32. **if** ((w update model) == "per particle") **then** $t \leftarrow t_i$ *// per particle calculation*
33. calculate $w_i(t)$
34. $\boldsymbol{v}_i(t+1) \leftarrow w_i(t) \cdot \boldsymbol{v}_i(t) + c_1 r_1 \left(\boldsymbol{x}_i^{\#} - \boldsymbol{x}_i \right) + c_2 r_2 \left(\boldsymbol{x}_i^{*} - \boldsymbol{x}_i \right)$
35. $\boldsymbol{x}_i(t+1) \leftarrow \boldsymbol{x}_i(t) + \boldsymbol{v}_i(t)$
36. **end** *// swarm motion*
37. **if** (status is wait) **then**
38. // do nothing
39. **end** *// wait*
40. **end** *// for*

Κεφάλαιο 6: Πειραματική Μελέτη

6.1. Εισαγωγή

Στο κεφάλαιο αυτό παρουσιάζουμε τα αποτελέσματα των πειραμάτων που έχουν πραγματοποιηθεί για την προσομοίωση του εξεταζόμενου συστήματος.

Οι βασικές μεταβλητές που εξετάζουμε είναι οι ακόλουθες:

- Η μέση τιμή της ποιότητας πληροφορίας πλαισίου των σωματιδίων του σμήνους κατά τη διάρκεια εκτέλεσης του αλγόριθμου $\overline{g(t)}$.
- Η τυπική απόκλιση της μέσης τιμής της ποιότητας πληροφορίας πλαισίου των σωματιδίων του σμήνους $\text{std}\left(\overline{g(t)}\right)$.
- Η μέση τελική τιμή της ποιότητας πληροφορίας πλαισίου των σωματιδίων του σμήνους το χρονικό βήμα t_0, $\overline{g(t_0)}$.
- Η μέση απόσταση που διανύει ένα σωματίδιο μέχρι το χρονικό βήμα t, $\overline{d_t}$ (σχέση (5.17)).
- Η μέση απόσταση που διανύει ένα σωματίδιο σε κάθε χρονικό βήμα (επανάληψη) του αλγόριθμου $\overline{d}$.

Η μέση απόσταση που διανύει ένα σωματίδιο μέχρι το χρονικό βήμα t, $\overline{d_t}$ ορίζεται ως εξής:

$$\overline{d_t} = \frac{1}{N}\sum_{i \in N} d_i(t) \tag{5.17}$$

όπου $d_i(t)$ είναι η απόσταση που διανύει ένα σωματίδιο i μέχρι το χρονικό βήμα t:

$$d_i(t) = \sum_{\tau=1}^{t} \left\| l_i(\tau) - l_i(\tau-1) \right\| \tag{5.18}$$

και $l_i(0),\ldots,l_i(t)$ οι διαδοχικές θέσεις της τροχιάς του σωματιδίου i μέχρι το χρονικό βήμα t.

Οι τιμές που λαμβάνουν οι ποσότητες $\overline{d_t}$, $\overline{d}$ εξαρτώνται από τις διαστάσεις του χώρου κίνησης των σωματιδίων του σμήνους.

Στα αποτελέσματα αναφέρονται οι συγκλίνουσες τιμές των παραπάνω μεταβλητών για διαδοχικές εκτελέσεις N_r των αλγόριθμων που υλοποιήθηκαν. Ο συμβολισμός δεν αλλάζει για λόγους απλότητας.

Οι τιμές των παραμέτρων του συστήματος που χρησιμοποιήθηκαν στα πειράματα (εκτός και αν αναφέρεται διαφορετικά) είναι οι εξής:

- $N = 100$ ο αριθμός σωματιδίων (κόμβων) του σμήνους και $M = 2$ ο αριθμός των πηγών (αισθητήριων κόμβων).
- $L_x = 100$, $L_y = 100$ οι διαστάσεις του χώρου κίνησης.
- $R = 0,001 \cdot L_x \cdot L_y$ η ακτίνα μετάδοσης πληροφορίας σωματιδίων και πηγών.
- $[v_{\min}, v_{\max}] = [0.1, 2]$ οι οριακές ταχύτητες πηγών και σωματιδίων κατά την εκτέλεση τυχαίας κίνησης (random waypoint mobility model).
- $c_1 = c_2 = 2$ οι σταθερές επιτάχυνσης.
- $w_{\min} = 0,4$, $w_{\max} = 0,9$ οι οριακές τιμές του βάρους αδράνειας w.
- $\theta_{iy} = \theta_y = 100$ το κατώφλι της ποιότητας πληροφορίας πλαισίου.
- $t_0 = 1000$ ο μέγιστος αριθμός βημάτων των αλγόριθμων.
- $N_r = 100$ ο αριθμός εκτελέσεων των αλγόριθμων.

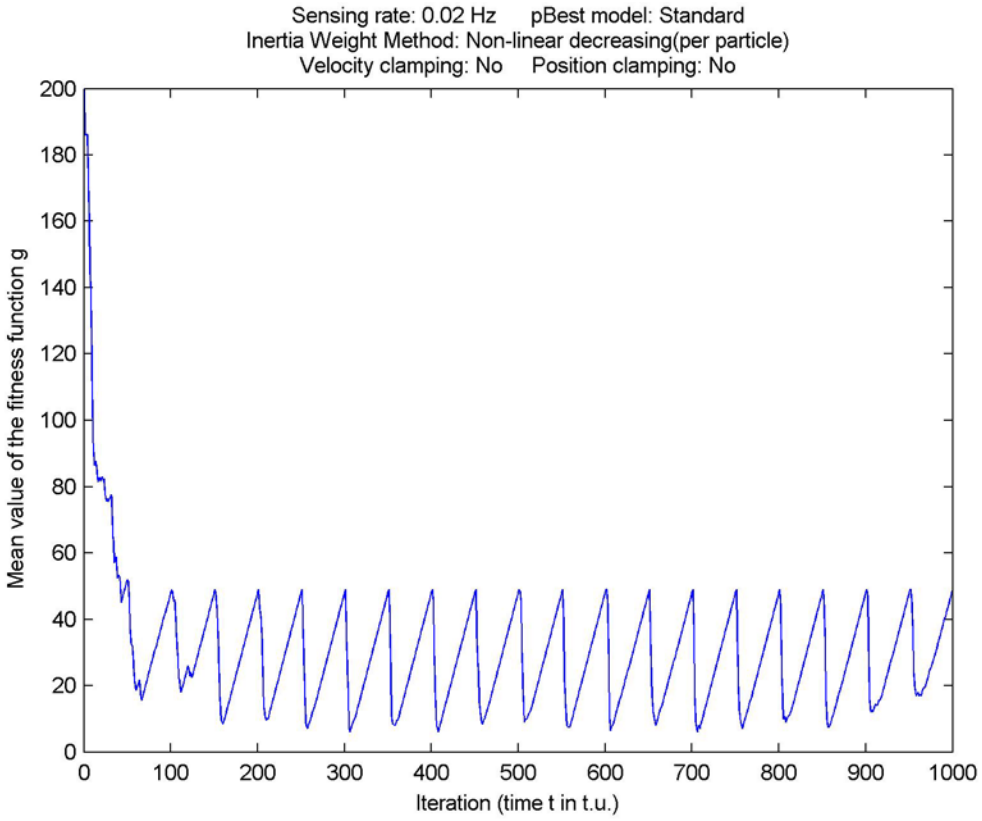
Αρχικά ($t = 0$), η ποιότητα πληροφορίας πλαισίου των σωματιδίων του σμήνους τίθεται ίση με 200 (μέγιστη δυνατή τιμή), ενώ των πηγών με 0 (ελάχιστη δυνατή τιμή).

Η ποιότητα πληροφορίας πλαισίου απαξιώνεται κατά μία μονάδα ανά βήμα εκτέλεσης του αλγόριθμου, τόσο για τις πηγές όσο και για τα σωματίδια του σμήνους.

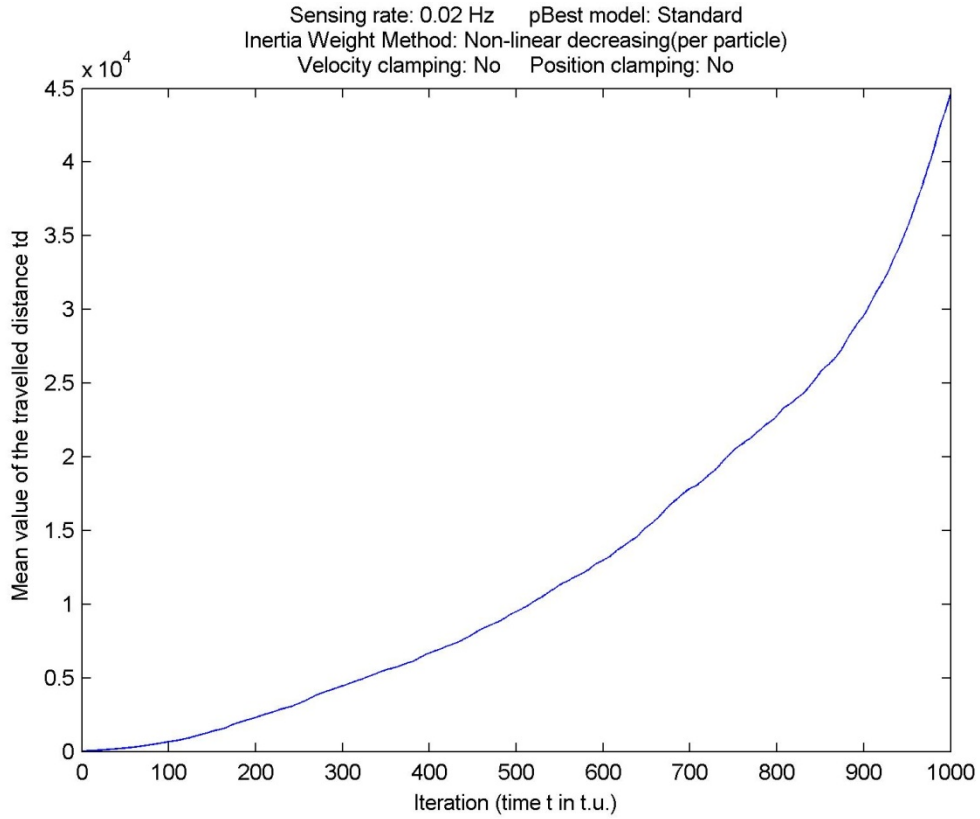
Στα πειράματα χρησιμοποιείται το ατομικό μοντέλο (per particle model) μεταβολής του βάρους αδράνειας w. Επίσης, δεν εφαρμόζονται οι μέθοδοι αποκοπής ταχύτητας και αποκοπής θέσης (βλέπε εδάφια 5.2.3, 5.2.4).

Στα Σχήματα 11, 12, 13 απεικονίζονται οι μεταβολές των ποσοτήτων $\overline{g(t)}$, $\overline{d_t}$ και $\overline{d}$, αντίστοιχα, για μία εκτέλεση του αλγόριθμου PSO, με εφαρμογή της μεθόδου μεταβολής του

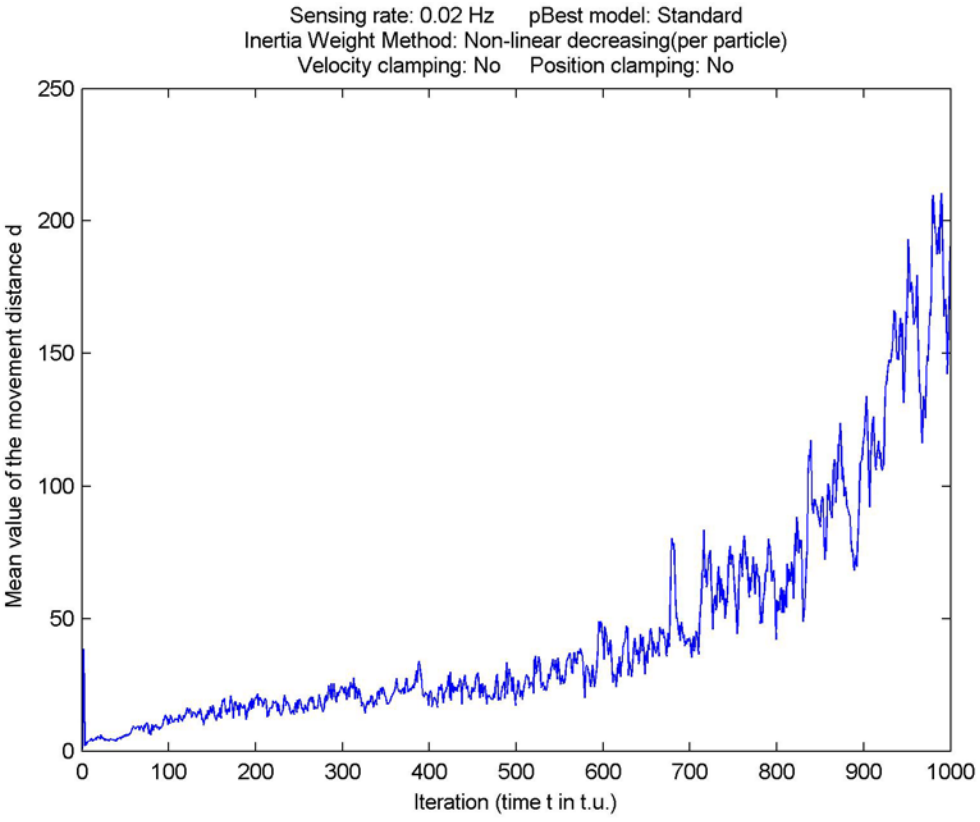
βάρους αδράνειας non-linear (decreasing) και για συχνότητα ανανέωσης πληροφορίας πλαισίου των πηγών $q = 0,02$ Hz σε συνάρτηση με το χρόνο t.



Σχήμα 11: Αλγόριθμος PSO. $\overline{g(t)} = f(t)$



Σχήμα 12: Αλγόριθμος PSO. $\overline{d_t(t)} = f(t)$



Σχήμα 13: Αλγόριθμος PSO. $\overline{d(t)} = f(t)$

6.2. Αλγόριθμος Βελτιστοποίησης Σμήνους Σωματιδίων (PSO)

6.2.1. Μέθοδοι μεταβολής βάρους αδράνειας

Στους Πίνακες 4, 5 παρουσιάζονται τα αποτελέσματα της εφαρμογής του βασικού αλγόριθμου Βελτιστοποίησης Σμήνους Σωματιδίων με χρήση του κλασικού τρόπου υπολογισμού της θέσης *pBest*, για διάφορες μεθόδους μεταβολής του βάρους αδράνειας w και διάφορες τιμές της συχνότητας ανανέωσης πληροφορίας πλαισίου των πηγών q, για $N_r = 100$ και $N_r = 1000$ εκτελέσεις του αλγόριθμου, αντίστοιχα. Οι βέλτιστες τιμές κάθε στήλης εμφανίζονται με έντονη γραφή.[1]

Στους Πίνακες 6, 7 εμφανίζονται τα αποτελέσματα της εφαρμογής του βασικού αλγόριθμου PSO με χρήση του δίχως μνήμη τρόπου υπολογισμού της θέσης *pBest*, για $N_r = 100$ και $N_r = 1000$ εκτελέσεις του αλγόριθμου, αντίστοιχα.

[1] Ο χρόνος διεξαγωγής κάθε πειράματος είναι περίπου 35-45 λεπτά ανάλογα με τη μέθοδο μεταβολής του w, για $N_r = 100$ (Desktop PC, Intel Core 2 Quad Q9550, 2,8GHz overclocked to 3,4GHz, 8GB RAM, Windows 7 64-bit, MATLAB v7.11 64-bit).

Πίνακας 4: PSO, standard *pBest* calculation, $N_r = 100$

Μέθοδος μεταβολής w	$q = 1$ Hz			$q = 0,05$ Hz			$q = 0,02$ Hz		
	$\overline{g(t_0)}$	$\overline{g(t)}$	$\text{std}\left(\overline{g(t)}\right)$	$\overline{g(t_0)}$	$\overline{g(t)}$	$\text{std}\left(\overline{g(t)}\right)$	$\overline{g(t_0)}$	$\overline{g(t)}$	$\text{std}\left(\overline{g(t)}\right)$
Sigmoid	10,009	10,110	16,455	33,286	28,569	**14,867**	58,180	41,254	**16,059**
Non-linear	2,590	5,598	19,679	19,258	**15,167**	18,526	**48,180**	30,453	20,011
Random	33,662	30,143	15,931	49,496	42,889	15,574	68,745	51,528	17,588
Chaotic	**1,735**	**4,019**	**15,735**	**18,848**	15,960	15,906	48,295	**30,424**	17,576
Chaotic Random	2,697	4,881	16,953	19,756	17,155	16,459	48,665	31,476	17,273

Πίνακας 5: PSO, standard *pBest* calculation, $N_r = 1000$

Μέθοδος μεταβολής w	$q = 1$ Hz			$q = 0,05$ Hz			$q = 0,02$ Hz		
	$\overline{g(t_0)}$	$\overline{g(t)}$	$\text{std}\left(\overline{g(t)}\right)$	$\overline{g(t_0)}$	$\overline{g(t)}$	$\text{std}\left(\overline{g(t)}\right)$	$\overline{g(t_0)}$	$\overline{g(t)}$	$\text{std}\left(\overline{g(t)}\right)$
Sigmoid	10,025	10,072	15,991	33,190	28,613	**15,024**	57,785	41,376	**16,161**
Non-linear	2,619	5,315	18,860	19,180	**15,255**	18,373	**48,170**	**30,425**	20,116
Random	34,522	30,303	15,939	49,647	43,175	15,975	69,807	51,700	17,683
Chaotic	**2,010**	**4,095**	**15,890**	**18,729**	15,825	15,464	48,303	30,511	17,660
Chaotic Random	2,716	4,753	16,544	19,828	17,012	15,957	48,639	31,733	17,574

Πίνακας 6: PSO, memoryless *pBest* calculation, $N_r = 100$

Μέθοδος μεταβολής w	$q = 1$ Hz			$q = 0,05$ Hz			$q = 0,02$ Hz		
	$\overline{g(t_0)}$	$\overline{g(t)}$	$\text{std}\left(\overline{g(t)}\right)$	$\overline{g(t_0)}$	$\overline{g(t)}$	$\text{std}\left(\overline{g(t)}\right)$	$\overline{g(t_0)}$	$\overline{g(t)}$	$\text{std}\left(\overline{g(t)}\right)$
Sigmoid	2,769	4,480	15,526	19,854	18,415	18,088	53,080	36,648	21,179
Non-linear	5,644	8,086	17,490	36,585	36,084	28,141	56,348	48,008	28,829
Random	6,268	7,112	**14,297**	19,710	16,641	15,713	**48,655**	**31,573**	**17,482**
Chaotic	2,682	**4,298**	14,853	21,100	16,708	**14,045**	52,540	37,131	20,315
Chaotic Random	**2,528**	4,736	16,303	**18,516**	**14,669**	14,524	52,110	39,806	23,339

Πίνακας 7: PSO, memoryless *pBest* calculation, $N_r = 1000$

Μέθοδος μεταβολής w	$q = 1$ Hz			$q = 0,05$ Hz			$q = 0,02$ Hz		
	$\overline{g(t_0)}$	$\overline{g(t)}$	$\text{std}\left(\overline{g(t)}\right)$	$\overline{g(t_0)}$	$\overline{g(t)}$	$\text{std}\left(\overline{g(t)}\right)$	$\overline{g(t_0)}$	$\overline{g(t)}$	$\text{std}\left(\overline{g(t)}\right)$
Sigmoid	2,842	4,501	15,478	20,007	18,436	17,993	52,896	36,564	20,991
Non-linear	5,707	7,977	16,882	36,437	36,193	27,909	56,336	47,964	28,780
Random	6,428	7,150	**14,304**	19,770	16,752	16,118	**48,425**	**31,868**	**17,683**
Chaotic	3,107	**4,379**	14,799	21,167	16,567	**13,855**	52,549	37,237	20,212
Chaotic Random	**2,546**	4,612	15,910	**18,573**	**14,683**	14,081	52,082	40,131	23,746

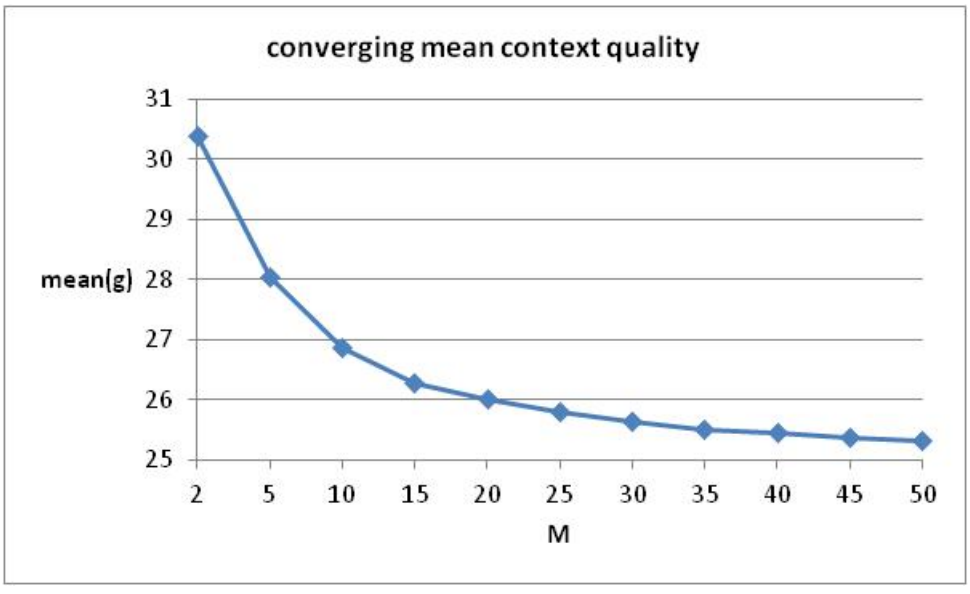
6.2.2. Επίδραση του αριθμού των πηγών

Οι μεταβολές των βασικών μεταβλητών του συστήματος σε συνάρτηση με τον αριθμό των πηγών M, για τον κλασικό τρόπο υπολογισμού του *pBest*, για τη μέθοδο μεταβολής του βάρους αδράνειας non-linear decreasing και για συχνότητα $q = 0,02$ Hz παρουσιάζονται στον Πίνακα 8.

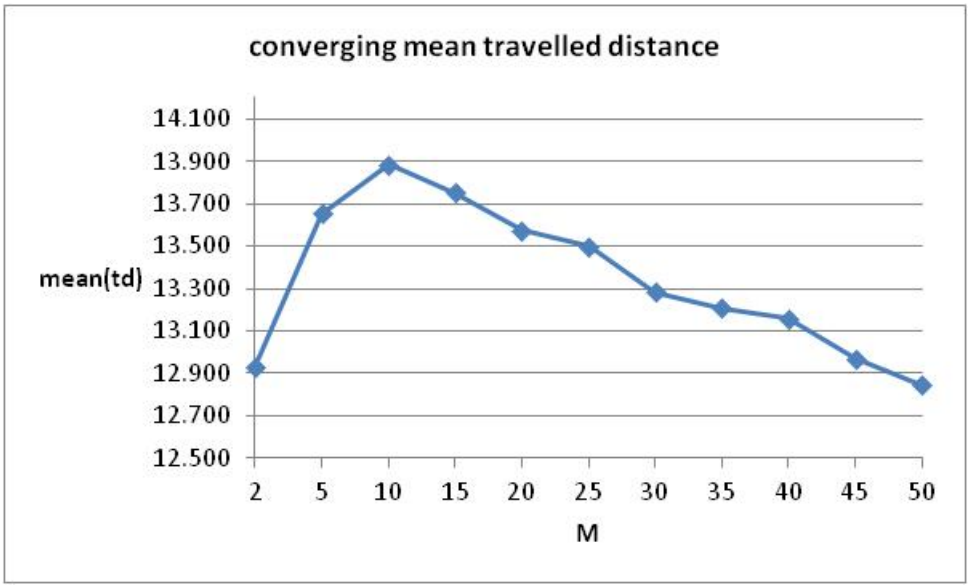
Επίσης, στα Σχήματα 14, 15 απεικονίζεται η μεταβολή της συγκλίνουσας μέσης τιμής της ποιότητας πληροφορίας πλαισίου των σωματιδίων του σμήνους $\overline{g(t)}$ και της συγκλίνουσας μέσης απόστασης που διανύει ένα σωματίδιο του σμήνους $\overline{d_t}$ συναρτήσει του M.

Πίνακας 8: PSO, μοντέλο *pBest* κλασικό, μέθοδος non-linear, $q = 0,02$ Hz

M	$\overline{g(t_0)}$	$\overline{g(t)}$	$\text{std}\left(\overline{g(t)}\right)$	$\overline{d_t}$	$\overline{d}$
2	48,285	30,396	19,686	12.929,9	44,55
5	48,155	28,036	15,919	13.652,3	46,17
10	48,175	26,856	14,777	13.883,8	46,53
15	48,170	26,277	14,515	13.753,3	46,38
20	48,190	26,013	14,481	13.575,5	45,96
25	48,185	25,799	14,501	13.499,0	45,48
30	48,170	25,631	14,559	13.283,4	45,13
35	48,180	25,510	14,629	13.209,5	44,62
40	48,200	25,443	14,667	13.156,9	44,85
45	48,200	25,363	14,728	12.968,2	44,33
50	48,240	25,321	14,768	12.844,0	44,12



Σχήμα 14: PSO. Συγκλίνουσα μέση ποιότητα πληροφορίας πλαισίου $\overline{g(t)}$**,** $N = 100$



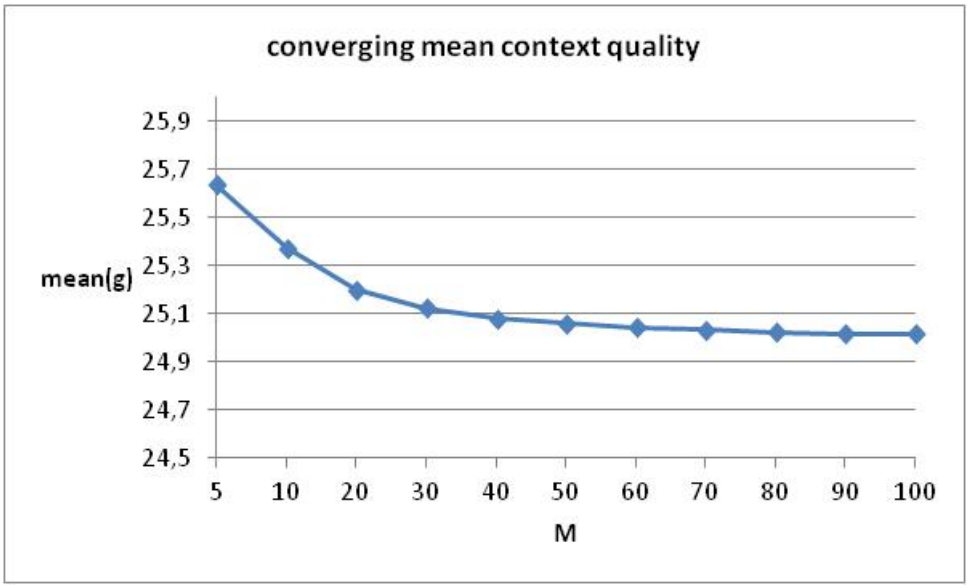
Σχήμα 15: PSO. Συγκλίνουσα μέση διανυόμενη απόσταση $\overline{d_t}$, $N = 100$

Στον Πίνακα 9 καθώς και στα Σχήματα 16-18 παρουσιάζονται οι μεταβολές των βασικών μεταβλητών του εξεταζόμενου συστήματος για σμήνος πλήθους $N = 500$ σωματιδίων και για συχνότητα $q = 0,02$ Hz . Τα χωρικά πλαίσια κίνησης παραμένουν αμετάβλητα.[2]

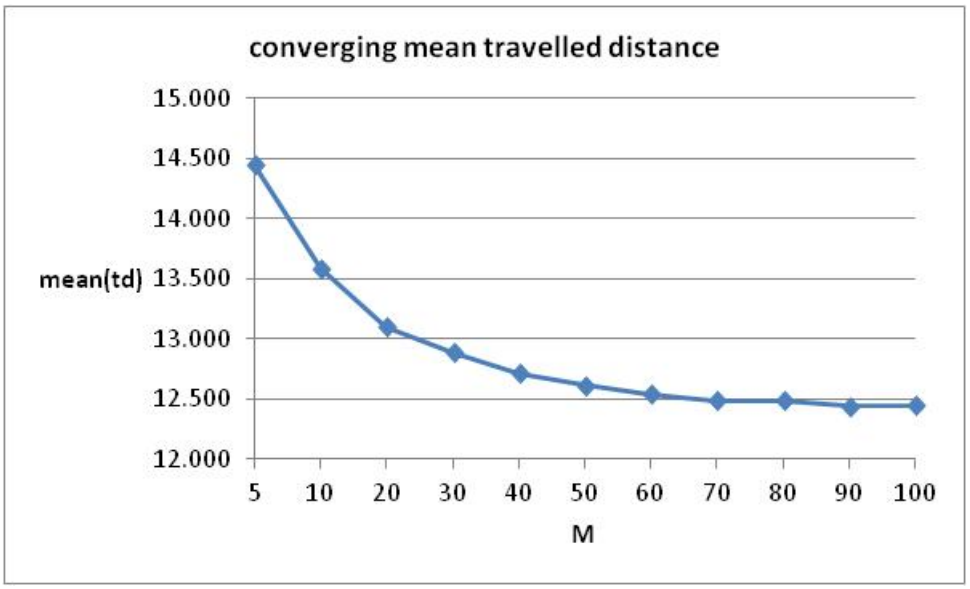
Πίνακας 9: PSO, μοντέλο *pBest* κλασικό, μέθοδος non-linear, $N = 500$, $q = 0,02$ Hz

M	$\overline{g(t_0)}$	$\overline{g(t)}$	$\text{std}\left(\overline{g(t)}\right)$	$\overline{d_t}$	$\overline{d}$
5	48,144	25,638	14,757	14.452,1	49,12
10	48,122	25,375	14,780	13.592,1	47,02
20	48,107	25,201	14,877	13.102,2	45,52
30	48,117	25,126	14,945	12.891,6	44,89
40	48,098	25,082	14,988	12.714,1	44,33
50	48,102	25,060	15,012	12.615,6	44,08
60	48,095	25,044	15,032	12.540,2	43,82
70	48,122	25,033	15,047	12.490,8	43,69
80	48,105	25,026	15,057	12.488,0	43,66
90	48,102	25,017	15,065	12.444,2	43,49
100	48,111	25,017	15,068	12.450,3	43,55

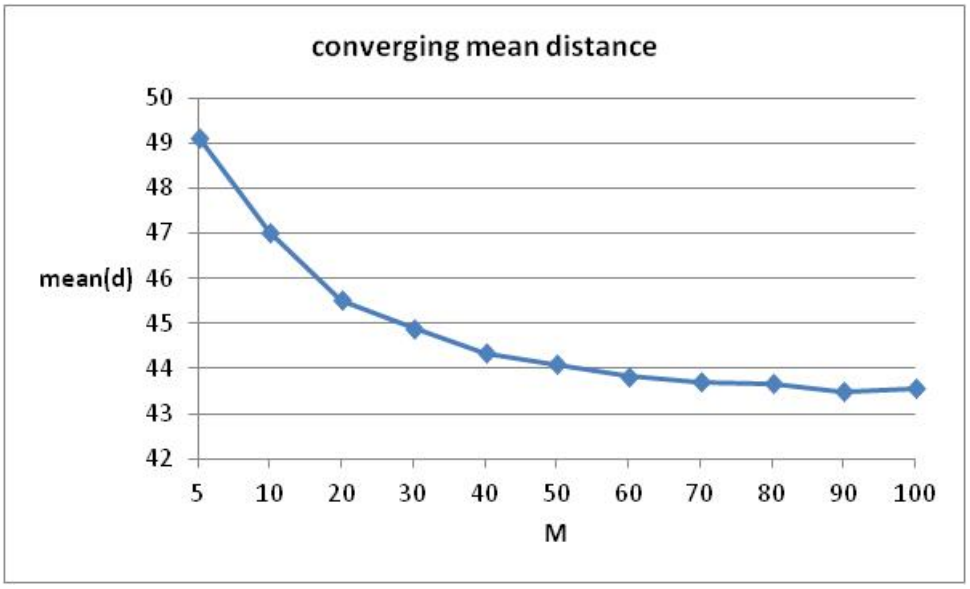
[2] Μέσος χρόνος εκτέλεσης πειραμάτων 570 λεπτά (9,5 ώρες).



Σχήμα 16: PSO. Συγκλίνουσα μέση ποιότητα πληροφορίας πλαισίου $\overline{g(t)}$ για $N = 500$



Σχήμα 17: PSO. Συγκλίνουσα μέση διανυόμενη απόσταση $\overline{d_t}$ για $N = 500$



Σχήμα 18: PSO. Συγκλίνουσα μέση απόσταση $\overline{d}$ για $N = 500$

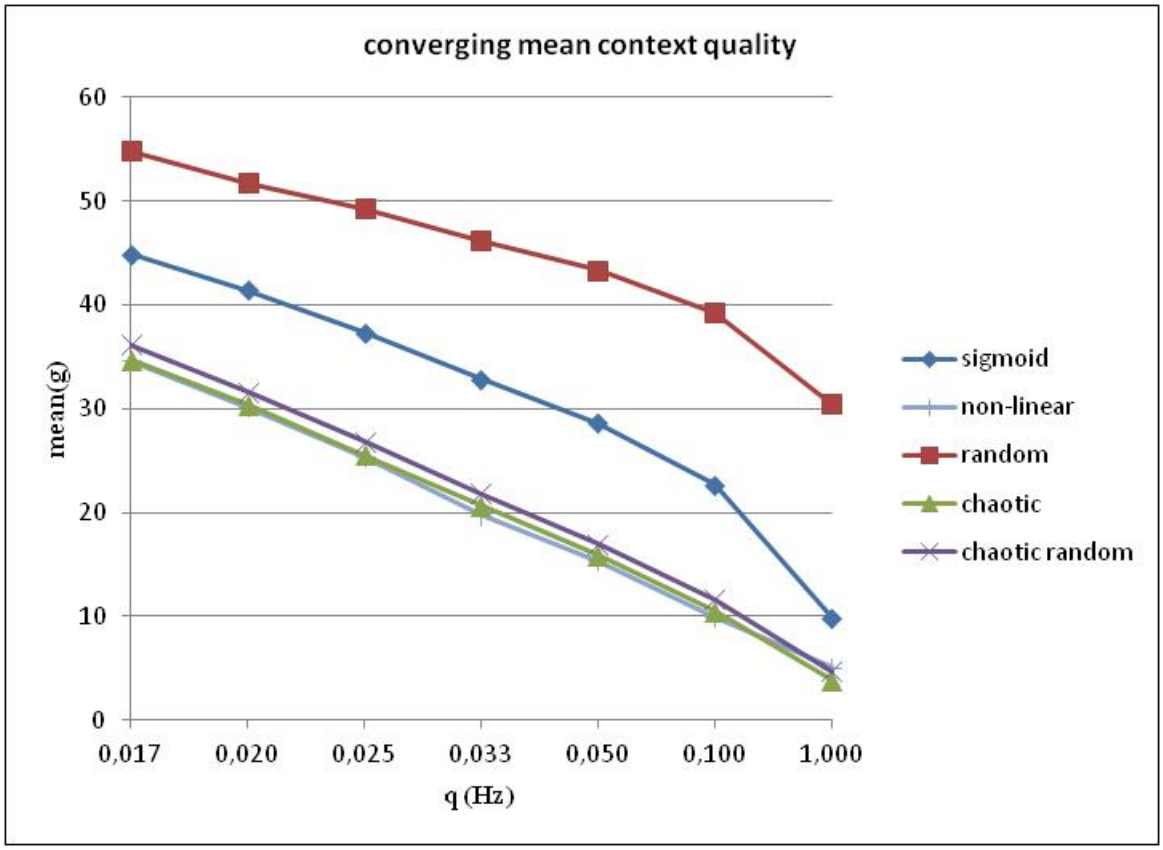
6.2.3. Επίδραση της συχνότητας ανανέωσης πληροφορίας πλαισίου

Η επίδραση της συχνότητας ανανέωσης της πληροφορίας πλαισίου των πηγών $q = \frac{1}{T}$, όπου T η περίοδος ανανέωσης, μελετάται στο διάστημα $[1\text{Hz}, 0,017\text{Hz}]$, δηλαδή για τιμές της T μεταξύ 1 και 60 χρονικών βημάτων, $T \in \{1,10,20,30,40,50,60\}$, για τιμές παραμέτρων $N_r = 100$, $N = 100$, $M = 2$, $t_0 = 1000$ (βλέπε Πίνακα 10).

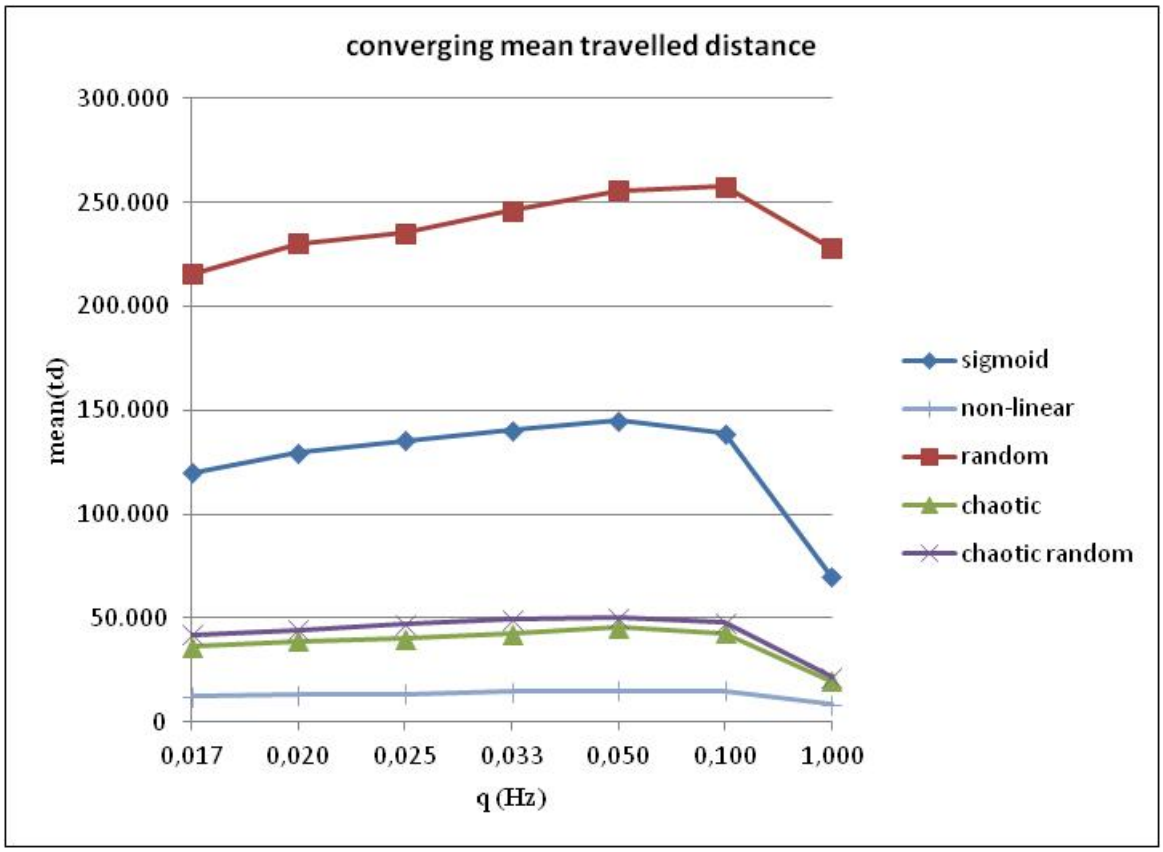
Στα Σχήματα 19, 20 απεικονίζεται η μεταβολή της συγκλίνουσας μέσης τιμής της ποιότητας πληροφορίας πλαισίου των σωματιδίων του σμήνους $\overline{g(t)}$ και της συγκλίνουσας μέσης απόστασης που διανύει ένα σωματίδιο του σμήνους $\overline{d_t}$ συναρτήσει της συχνότητας ανανέωσης q.

Πίνακας 10: Επίδραση της συχνότητας ανανέωσης q

PSO, standard *pBest* calculation, per particle, $N_r = 100$, $N = 100$, $M = 2$, $t_0 = 1.000$						
Μέθοδος μεταβολής w	q (Hz)	$\overline{g(t_0)}$	$\overline{g(t)}$	$\text{std}\left(\overline{g(t)}\right)$	$\overline{d_t}$	$\overline{d}$
Sigmoid	1,000	10,176	9,897	15,130	69.821,4	164,77
	0,100	24,398	22,672	14,773	138.733,0	322,65
	0,050	33,430	28,605	14,199	144.982,5	342,43
	0,033	37,250	32,835	13,761	140.482,0	337,11
	0,025	44,932	37,368	13,933	135.405,5	329,17
	0,020	58,455	41,427	15,553	129.481,8	325,16
	0,017	54,000	44,903	15,326	119.946,1	300,47
Non-linear	1,000	2,577	5,115	18,444	8.607,3	22,65
	0,100	9,956	9,971	18,000	14.801,4	44,08
	0,050	19,188	15,346	18,237	15.074,4	47,37
	0,033	13,562	19,774	16,765	14.689,0	47,96
	0,025	38,376	25,405	18,965	13.460,0	45,19
	0,020	48,195	30,153	19,286	12.972,6	44,62
	0,017	54,000	34,660	20,502	12.399,4	43,46
Random	1,000	34,752	30,434	15,787	227.870,6	523,48
	0,100	43,405	39,283	15,553	257.865,4	576,44
	0,050	51,140	43,331	15,610	255.620,5	576,15
	0,033	53,855	46,220	15,776	246.235,6	550,02
	0,025	51,124	49,290	15,847	235.213,9	529,48
	0,020	70,845	51,775	17,355	230.439,0	529,91
	0,017	54,000	54,856	16,051	215.448,7	486,84
Chaotic	1,000	1,799	3,837	14,971	19.659,7	32,81
	0,100	9,483	10,496	14,604	42.612,3	74,72
	0,050	18,872	15,869	14,762	45.321,7	81,75
	0,033	10,997	20,666	14,769	42.565,0	77,96
	0,025	38,240	25,547	15,675	39.953,8	74,02
	0,020	48,290	30,383	16,779	38.961,1	74,23
	0,017	54,000	34,739	18,597	36.061,5	68,74
Chaotic Random	1,000	3,089	4,710	16,093	21.386,9	46,63
	0,100	10,972	11,663	15,698	47.907,9	106,45
	0,050	19,926	16,952	15,228	49.956,7	113,86
	0,033	14,459	21,856	14,910	49.456,3	114,78
	0,025	38,824	26,866	15,919	47.118,6	112,12
	0,020	48,670	31,652	17,034	44.165,7	107,46
	0,017	54,000	36,158	18,428	42.079,8	103,89



Σχήμα 19: PSO. Συγκλίνουσα μέση ποιότητα πληροφορίας πλαισίου $\overline{g(t)}$ σε συνάρτηση με τη συχνότητα ανανέωσης q



Σχήμα 20: PSO. Συγκλίνουσα μέση διανυόμενη απόσταση $\overline{d_t}$ σε συνάρτηση με τη συχνότητα q

Συμπεράσματα – Παρατηρήσεις

- Οι μέθοδοι μεταβολής του βάρους αδράνειας που δίνουν τα καλύτερα αποτελέσματα ως προς τη συγκλίνουσα μέση τιμή της ποιότητας πληροφορίας πλαισίου των σωματιδίων του σμήνους $\overline{g(t)}$ είναι οι non-linear decreasing και chaotic για τον κλασικό τρόπο υπολογισμού της θέσης *pBest*, και random και chaotic random για τον δίχως μνήμη τρόπο υπολογισμού της θέσης *pBest* (Πίνακες 4-7).
- Η μέθοδος μεταβολής του βάρους αδράνειας που δίνει τα καλύτερα αποτελέσματα ως προς τη συγκλίνουσα μέση απόσταση που διανύει ένα σωματίδιο μέχρι το χρονικό βήμα t, $\overline{d_t}$ είναι η non-linear decreasing για τον κλασικό τρόπο υπολογισμού της θέσης *pBest* (Πίνακας 10 και Σχήμα 20).
- Η στατιστικά ασήμαντη διαφοροποίηση των αποτελεσμάτων για $N_r = 100$ και $N_r = 1000$ εκτελέσεις του αλγόριθμου αποτελεί ένδειξη της ευστάθειάς του, καθώς η εικόνα μεταβολής των μεταβλητών του συστήματος δεν εξαρτάται από τον αριθμό επαναλήψεων του αλγόριθμου.
- Η συγκλίνουσα μέση ποιότητα πληροφορίας πλαισίου του σμήνους $\overline{g(t)}$ βελτιώνεται (μειώνεται) με την αύξηση του αριθμού των πηγών M (Σχήμα 14). Η μείωση είναι πιο σημαντική (περίπου $15,7\%$ για τη μέθοδο non-linear) για τιμές του $2 \le M \le 30$.
- Η συγκλίνουσα μέση ποιότητα πληροφορίας πλαισίου του σμήνους $\overline{g(t)}$ βελτιώνεται (μειώνεται) με την αύξηση της συχνότητας ανανέωσης της πληροφορίας πλαισίου των πηγών q (Πίνακας 10 και Σχήμα 19). Η βελτίωση αυτή κυμαίνεται από περίπου 89% για τη μέθοδο μεταβολής του βάρους αδράνειας chaotic, μέχρι $44,5\%$ για τη μέθοδο random.
- Η συγκλίνουσα μέση απόσταση που διανύει ένα σωματίδιο του σμήνους μέχρι το χρονικό βήμα t, $\overline{d_t}$ ελαττώνεται με την αύξηση του αριθμού των πηγών M, όπως απεικονίζεται στο Σχήμα 17. Η μεταβολή αυτή δεν είναι ξεκάθαρη για μικρότερα πλήθη σωματιδίων (βλέπε Σχήμα 15), εξαιτίας της υιοθέτησης σχετικά μικρής ακτίνας μετάδοσης πηγών και σωματιδίων σε σχέση με το εμβαδόν του χώρου κίνησης ($R = 10^{-3} \cdot V$, όπου V το εμβαδόν του δισδιάστατου χώρου).

- Η συγκλίνουσα μέση απόσταση $\overline{d_t}$ ελαττώνεται με την αύξηση της συχνότητας ανανέωσης της πληροφορίας πλαισίου των πηγών q (Πίνακας 10 και Σχήμα 20) για όλες τις μεθόδους μεταβολής του βάρους αδράνειας, εκτός από τη μέθοδο random. Η μείωση αυτή είναι κατά μέσο όρο περίπου $41{,}8\%$. Οι μέθοδοι random και sigmoid απαιτούν σημαντικά μεγαλύτερες μέσες αποστάσεις ανά σωματίδιο από τις υπόλοιπες μεθόδους.
- Η συγκλίνουσα μέση απόσταση που διανύει ένα σωματίδιο σε κάθε χρονικό βήμα του αλγόριθμου $\overline{d}$ ακολουθεί την ίδια κατανομή με την ποσότητα $\overline{d_t}$ (Σχήμα 18).
- Η μέση τελική τιμή της ποιότητας πληροφορίας πλαισίου των σωματιδίων του σμήνους το χρονικό βήμα t_0, $\overline{g(t_0)}$ εξαρτάται από τη συχνότητα ανανέωσης της πληροφορίας πλαισίου των πηγών q.

6.3. Αλγόριθμος Βελτιστοποίησης Σμήνους Σωματιδίων με χρήση Θεωρίας Βέλτιστης Παύσης (OSPSO)

6.3.1. Μέθοδοι μεταβολής βάρους αδράνειας – διάστημα παρατήρησης OST

Στα παρουσιαζόμενα αποτελέσματα της εφαρμογής του αλγόριθμου OSPSO χρησιμοποιείται ο κλασικός τρόπος υπολογισμού της θέσης *pBest* (*pBest* calculation model: standard). Αυτό γίνεται για να είναι δυνατή η σύγκριση με τα αντίστοιχα αποτελέσματα της βιβλιογραφίας.

Στον Πίνακα 11 παρουσιάζεται η επίδραση της τιμής του διαστήματος παρατήρησης n_0 στη συγκλίνουσα μέση ποιότητα πληροφορίας πλαισίου του σμήνους, για τις διάφορες μεθόδους μεταβολής του βάρους αδράνειας w, για συχνότητα ανανέωσης πλαισίου των πηγών $q = 0,02$ Hz.

Στον Πίνακα 12 αναλύεται η μεταβολή της συγκλίνουσας μέσης ποιότητας πληροφορίας πλαισίου του σμήνους για διάφορες τιμές του διαστήματος παρατήρησης n_0 και του κατωφλίου $\theta_{threshold}$, για συχνότητα ανανέωσης πλαισίου των πηγών $q = 0,02$ Hz.

Στο Σχήμα 21 απεικονίζεται η μεταβολή της συγκλίνουσας μέσης ποιότητας πληροφορίας πλαισίου $\overline{g(t)}$ σε συνάρτηση με το διάστημα παρατήρησης n_0, για τις διάφορες μεθόδους μεταβολής του βάρους αδράνειας.

Πίνακας 11: OSPSO, n_0, $q = 0,02$ Hz

Μέθοδος μεταβολής w	n_0	$\sqrt{n_0}$	$\overline{g(t_0)}$	$\overline{g(t)}$	$\text{std}\left(\overline{g(t)}\right)$
Sigmoid	0	-	**58,180**	**41,254**	**16,059**
	5	2	73,847	51,020	17,981
	10	3	71,360	50,432	18,723
	20	4	69,583	49,242	18,586
	30	5	66,040	46,291	19,482
	50	7	65,404	44,156	21,405
	80	9	63,473	43,518	20,500
	100	10	63,453	43,476	21,482
	120	11	63,392	43,472	21,711
	140	12	62,296	42,980	21,534
	160	13	61,627	43,194	21,248
Non-linear	0	-	48,180	30,453	20,011
	5	2	48,050	**29,993**	**19,282**
	10	3	48,020	30,860	20,640
	20	4	**48,000**	32,521	23,237
	30	5	48,065	33,678	24,006
	50	7	48,040	36,269	27,201
	80	9	48,050	37,067	25,809
	100	10	48,250	38,435	26,985

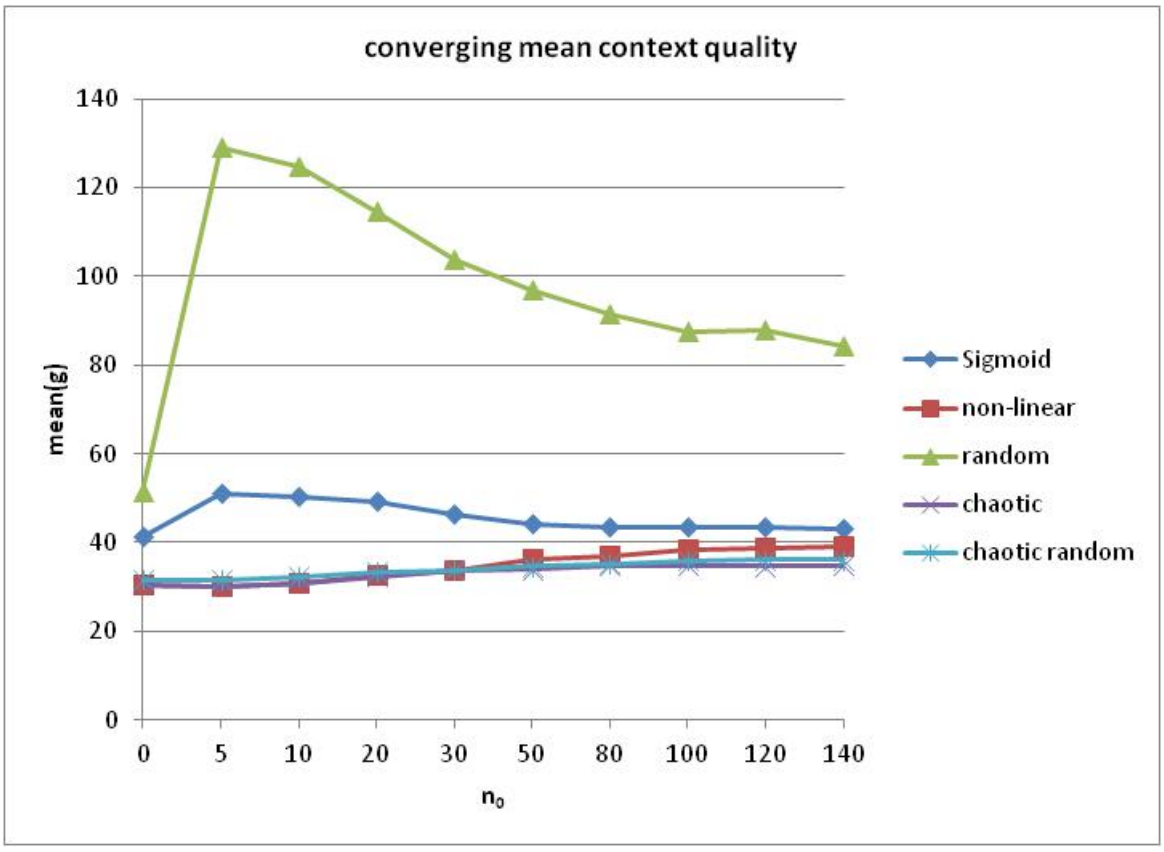
	120	11	48,155	38,748	26,310
	140	12	48,235	39,112	26,550
Random	0	-	**68,745**	**51,528**	**17,588**
	5	2	184,288	129,077	53,121
	10	3	182,708	124,761	52,096
	20	4	174,758	114,763	49,626
	30	5	171,201	103,847	47,635
	50	7	165,978	97,049	47,615
	80	9	156,172	91,435	43,070
	100	10	151,765	87,539	41,605
	120	11	153,800	87,901	41,800
	140	12	145,977	84,368	39,024
	160	13	147,896	86,227	40,189
	180	13	143,002	82,727	38,377
	250	16	138,606	79,062	35,544
	300	17	136,122	78,515	34,519
	350	19	130,868	76,627	33,119
	400	20	127,847	76,274	32,411
	450	21	128,294	76,561	32,482
Chaotic	0	-	48,295	30,424	**17,576**
	5	2	**48,015**	**30,190**	18,555
	10	3	48,025	30,891	18,744
	20	4	48,140	32,196	19,498
	30	5	48,286	33,680	20,190
	50	7	48,450	33,890	22,187
	80	9	48,280	34,808	22,015
	100	10	48,530	34,912	22,207
	120	11	48,520	34,625	20,984
	140	12	48,755	34,852	21,575
Chaotic Random	0	-	48,665	**31,476**	**17,273**
	5	2	48,495	31,510	17,392
	10	3	**48,410**	32,307	18,564
	20	4	48,590	33,473	20,331
	30	5	48,620	33,629	19,883
	50	7	48,621	34,581	23,264
	80	9	48,480	35,114	22,381
	100	10	48,745	35,985	23,950
	120	11	48,800	36,050	23,875
	140	12	48,700	36,073	23,505

Πίνακας 12: OSPSO, $n_0, \theta_{threshold}$**,** $q = 0,02$ Hz

Μέθοδος μεταβολής w	n_0	$\sqrt{n_0}$	$\theta_{threshold}$	$\overline{g(t_0)}$	$\overline{g(t)}$	$\text{std}\left(\overline{g(t)}\right)$
Sigmoid	0	-	-	58.180	41.254	16.059
	5	2	0	73.847	51.020	17.981
	5	2	10	82.650	58.150	20.813
	5	2	50	83.926	58.944	21.770
	5	2	90	83.950	58.353	21.271
	10	3	0	71.360	50.432	18.723
	10	3	10	81.716	56.881	20.402
	10	3	50	82.813	58.344	21.667
	10	3	90	83,863	58,286	21,283
	140	12	0	62.296	42.980	21.534
	140	12	10	78.090	53.713	19.808

	140	12	30	82.849	57.567	21.041
	140	12	50	84.307	58.513	21.719
	140	12	90	84.304	59.050	21.718
Non-linear	0	-	-	48.180	30.453	20.011
	5	2	0	48.050	29.993	19.282
	5	2	10	48.190	30.682	20.719
	5	2	30	48.145	31.406	22.083
	5	2	50	48.125	31.604	22.098
	5	2	70	48.150	31.014	20.820
	5	2	90	48.150	31.513	21.834
	10	3	0	48.020	30.860	20.640
	10	3	10	48.245	30.706	20.486
	10	3	30	48.210	30.777	20.111
	10	3	50	48.190	31.834	22.514
	10	3	70	48.190	30.956	20.549
	10	3	90	48.140	31.214	20.910
	20	4	0	48.140	32.196	19.498
	20	4	10	48.140	31.462	21.620
	20	4	30	48.185	31.329	21.178
	20	4	50	48.150	31.150	20.796
	20	4	90	48.140	31.236	21.400
	50	7	0	48.040	36.269	27.201
	50	7	30	48.165	31.336	21.007
	50	7	50	48.165	31.049	20.693
	50	7	90	48.185	30.888	20.224
	80	9	0	48.050	37.067	25.809
	80	9	30	48.195	31.617	21.809
	80	9	50	48.130	31.247	21.036
	80	9	90	48.205	31.101	20.782
Random	0	-	-	71.795	52.272	17.958
	30	5	0	171.201	103.847	47.635
	30	5	30	185.688	132.387	54.124
	30	5	50	186.265	135.005	55.406
	30	5	90	187.487	136.384	55.613
	450	21	0	128.294	76.561	32.482
	450	21	30	178.417	128.494	51.690
	450	21	50	183.956	134.053	54.357
	450	21	90	187.027	136.375	55.990
Chaotic	0	-	-	48.295	30.424	17.576
	5	2	0	48.015	30.190	18.555
	5	2	10	48.251	30.459	17.316
	5	2	30	48.336	30.816	17.979
	5	2	50	48.321	30.833	18.334
	5	2	70	48.236	30.583	17.693
	5	2	90	48.210	30.577	17.816
	10	3	0	48.025	30.891	18.744
	10	3	10	48.391	31.000	17.925
	10	3	30	48.241	30.739	18.068
	10	3	50	48.290	30.520	17.779
	10	3	70	48.281	30.477	17.497
	10	3	90	48.391	30.525	17.655
	30	5	0	48.286	33.680	20.190
	30	5	50	48.246	30.433	17.399
	30	5	90	48.180	30.446	17.781
	50	7	0	48.450	33.890	22.187
	50	7	50	48.221	30.506	17.579
	50	7	90	48.195	30.585	17.586
	80	9	0	48.280	34.808	22.015

	80	9	50	48.181	30.421	17.525
	80	9	90	48.330	30.529	17.466
Chaotic Random	0	-	-	48.665	31.476	17.273
	5	2	0	48.495	31.510	17.392
	5	2	10	48.856	32.264	17.790
	5	2	30	48.891	32.348	18.053
	5	2	50	48.800	32.163	17.834
	5	2	90	48.876	32.382	18.351
	10	3	0	48.410	32.307	18.564
	10	3	30	48.850	32.150	17.675
	10	3	50	48.731	32.413	18.076
	10	3	90	48.886	32.302	18.084
	80	9	0	48.480	35.114	22.381
	80	9	30	48.936	32.067	17.410
	80	9	50	48.965	32.228	18.014
	80	9	90	48.801	32.180	17.783



Σχήμα 21: OSPSO. Μεταβολή του $\overline{g(t)}$ για τις διάφορες μεθόδους μεταβολής του w

6.3.2. Επίδραση του αριθμού των πηγών

Στους Πίνακες 13, 14 παρουσιάζονται οι μεταβολές των βασικών μεταβλητών του συστήματος σε συνάρτηση με τον αριθμό των πηγών M και την τιμή του διαστήματος

παρατήρησης n_0, για τις μεθόδους μεταβολής του βάρους αδράνειας non-linear decreasing και chaotic, αντίστοιχα, για συχνότητα $q = 0,02$ Hz ($N_r = 100$, $N = 100$, $t_0 = 1000$).

Επίσης, στα Σχήματα 22-25 απεικονίζεται η μεταβολή της συγκλίνουσας μέσης τιμής της ποιότητας πληροφορίας πλαισίου των σωματιδίων του σμήνους $\overline{g(t)}$ και της συγκλίνουσας μέσης απόστασης που διανύει ένα σωματίδιο του σμήνους $\overline{d_t}$ συναρτήσει του n_0, για διαφορετικά πλήθη πηγών M, για τις μεθόδους μεταβολής του βάρους αδράνειας non-linear decreasing και chaotic.

Πίνακας 13: OSPSO. Μέθοδος non-linear, $q = 0,02$ Hz

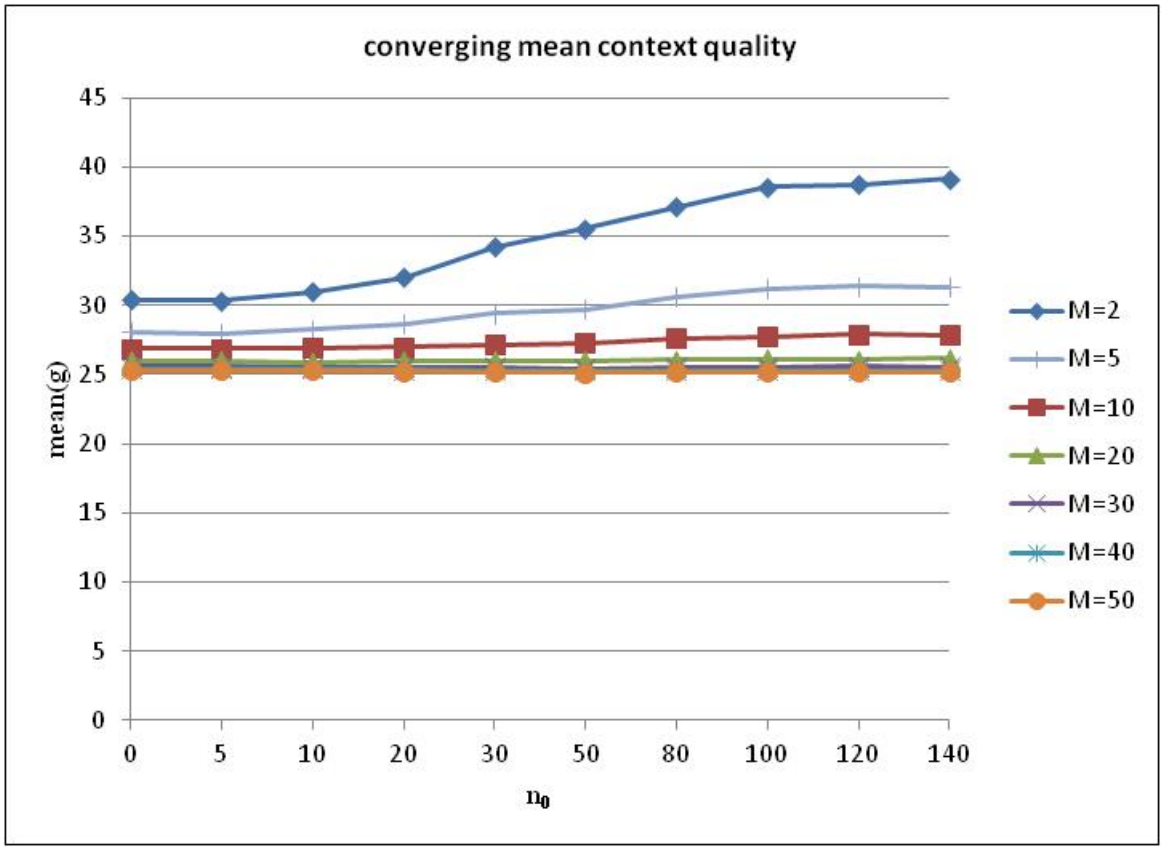
M	n_0	$\sqrt{n_0}$	$\overline{g(t_0)}$	$\overline{g(t)}$	$\mathrm{std}\left(\overline{g(t)}\right)$	$\overline{d_t}$	$\overline{d}$
2	0	0	48,285	30,396	19,686	12.929,9	44,55
2	5	2	48,055	30,336	19,675	10.882,1	33,74
2	10	3	48,005	30,965	20,918	9.232,8	26,42
2	20	4	48,005	32,036	22,433	6.758,7	17,68
2	30	5	48,010	34,179	25,146	4.705,4	11,94
2	50	7	48,025	35,554	25,938	3.987,6	10,48
2	80	9	48,060	37,127	26,318	3.263,9	8,56
2	100	10	48,510	38,563	26,966	3.019,5	8,12
2	120	11	48,145	38,763	26,280	2.906,8	7,78
2	140	12	48,275	39,123	26,690	2.685,8	6,93
5	0	0	48,155	28,036	15,919	13.652,3	46,17
5	5	2	48,070	27,946	15,902	11.828,9	36,28
5	10	3	48,035	28,239	16,276	10.308,2	29,48
5	20	4	48,015	28,633	16,762	7.723,0	20,05
5	30	5	48,010	29,414	17,748	5.658,7	14,04
5	50	7	48,005	29,672	18,104	4.731,2	12,08
5	80	9	48,025	30,594	18,106	4.097,2	10,44
5	100	10	48,125	31,189	18,623	3.719,3	9,59
5	120	11	48,030	31,380	18,519	3.703,3	9,49
5	140	12	48,115	31,280	17,996	3.498,9	8,83
10	0	0	48,175	26,856	14,777	13.883,8	46,53
10	5	2	48,040	26,855	14,889	12.152,6	37,54
10	10	3	48,015	26,894	14,916	10.677,3	30,98
10	20	4	48,000	26,989	15,075	8.423,9	22,13
10	30	5	48,010	27,155	15,181	6.669,5	16,56
10	50	7	48,000	27,276	15,380	5.559,0	14,14
10	80	9	48,000	27,628	15,426	5.023,2	12,75
10	100	10	48,040	27,767	15,420	4.738,7	11,91
10	120	11	48,020	27,914	15,375	4.617,2	11,65
10	140	12	48,010	27,805	15,289	4.443,1	11,13
20	0	0	48,190	26,013	14,481	13.575,5	45,96
20	5	2	48,075	25,966	14,542	12.090,9	38,32
20	10	3	48,030	25,901	14,536	11.123,9	33,23
20	20	4	48,010	25,966	14,634	9.299,9	25,52
20	30	5	48,000	25,949	14,701	7.918,7	20,41

20	50	7	48,005	25,956	14,794	6.809,3	17,76
20	80	9	48,010	26,055	14,725	6.484,2	16,75
20	100	10	48,025	26,008	14,718	6.240,9	16,03
20	120	11	48,020	26,062	14,651	6.229,2	16,06
20	140	12	48,050	26,171	14,669	6.093,2	15,65
30	0	0	48,170	25,631	14,559	13.283,4	45,13
30	5	2	48,120	25,586	14,598	12.112,8	38,83
30	10	3	48,095	25,570	14,616	11.364,9	34,93
30	20	4	48,015	25,538	14,671	9.861,1	28,00
30	30	5	48,025	25,548	14,718	8.681,4	23,24
30	50	7	48,005	25,440	14,778	7.787,4	20,88
30	80	9	48,015	25,532	14,744	7.547,2	20,24
30	100	10	48,035	25,515	14,732	7.381,1	19,56
30	120	11	48,065	25,584	14,728	7.336,5	19,45
30	140	12	48,020	25,540	14,733	7.330,9	19,35
40	0	0	48,200	25,443	14,667	13.156,9	44,85
40	5	2	48,090	25,403	14,693	12.232,1	39,54
40	10	3	48,070	25,387	14,715	11.450,1	35,95
40	30	5	48,035	25,362	14,759	10.252,5	29,85
40	20	4	48,020	25,311	14,786	9.208,2	25,60
40	50	7	48,010	25,260	14,860	8.572,1	23,53
40	80	9	48,030	25,285	14,849	8.302,9	22,75
40	100	10	48,030	25,286	14,838	8.285,3	22,47
40	120	11	48,055	25,313	14,821	8.214,3	22,35
40	120	12	48,020	25,310	14,827	8.119,9	22,03
50	0	0	48,240	25,321	14,768	12.844,7	44,12
50	5	2	48,130	25,308	14,781	12.097,7	39,76
50	10	3	48,110	25,277	14,806	11.592,1	36,44
50	20	4	48,040	25,233	14,841	10.391,6	30,90
50	30	5	48,025	25,193	14,875	9.568,9	27,09
50	50	7	48,025	25,129	14,936	8.999,9	25,13
50	80	9	48,025	25,164	14,917	8.860,4	24,60
50	100	10	48,050	25,157	14,928	8.809,3	24,52
50	120	11	48,055	25,199	14,901	8.755,3	24,25
50	140	12	48,050	25,195	14,901	8.724,2	24,07

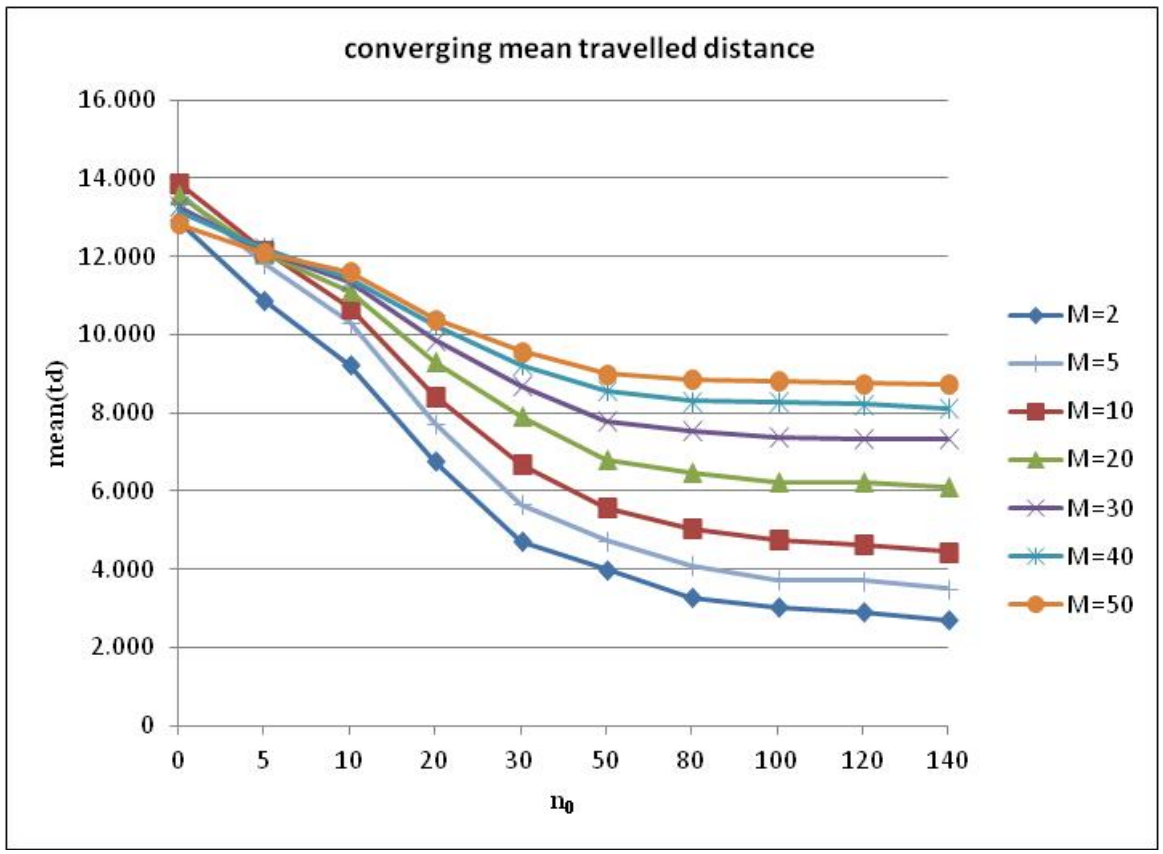
Πίνακας 14: OSPSO. Μέθοδος chaotic, $q = 0,02$ Hz

M	n_0	$\sqrt{n_0}$	$\bar{g}(t_0)$	$\bar{g}(t)$	$\text{std}(\bar{g}(t))$	$\overline{d_t}$	$\overline{d}$
2	0	0	48,285	30,398	17,032	39.162,1	74,14
2	5	2	48,066	30,060	17,987	26.170,7	46,88
2	10	3	48,075	30,980	18,816	24.661,3	45,04
2	20	4	48,100	32,231	19,835	22.729,9	43,27
2	30	5	48,190	33,212	19,703	19.240,1	39,29
2	50	7	48,336	33,240	20,827	19.452,1	42,56
2	80	9	48,275	34,296	21,262	17.939,4	40,21
2	100	10	48,632	34,923	22,207	16.476,2	37,78
2	120	11	48,620	34,514	20,966	16.445,8	37,75
2	140	12	48,966	34,827	21,680	14.696,0	33,79
5	0	0	48,200	28,581	14,751	38.654,0	70,98
5	5	2	48,045	28,397	15,220	29.066,2	51,75
5	10	3	48,045	28,848	15,282	27.721,6	49,86
5	20	4	48,095	29,521	15,269	24.878,8	46,14

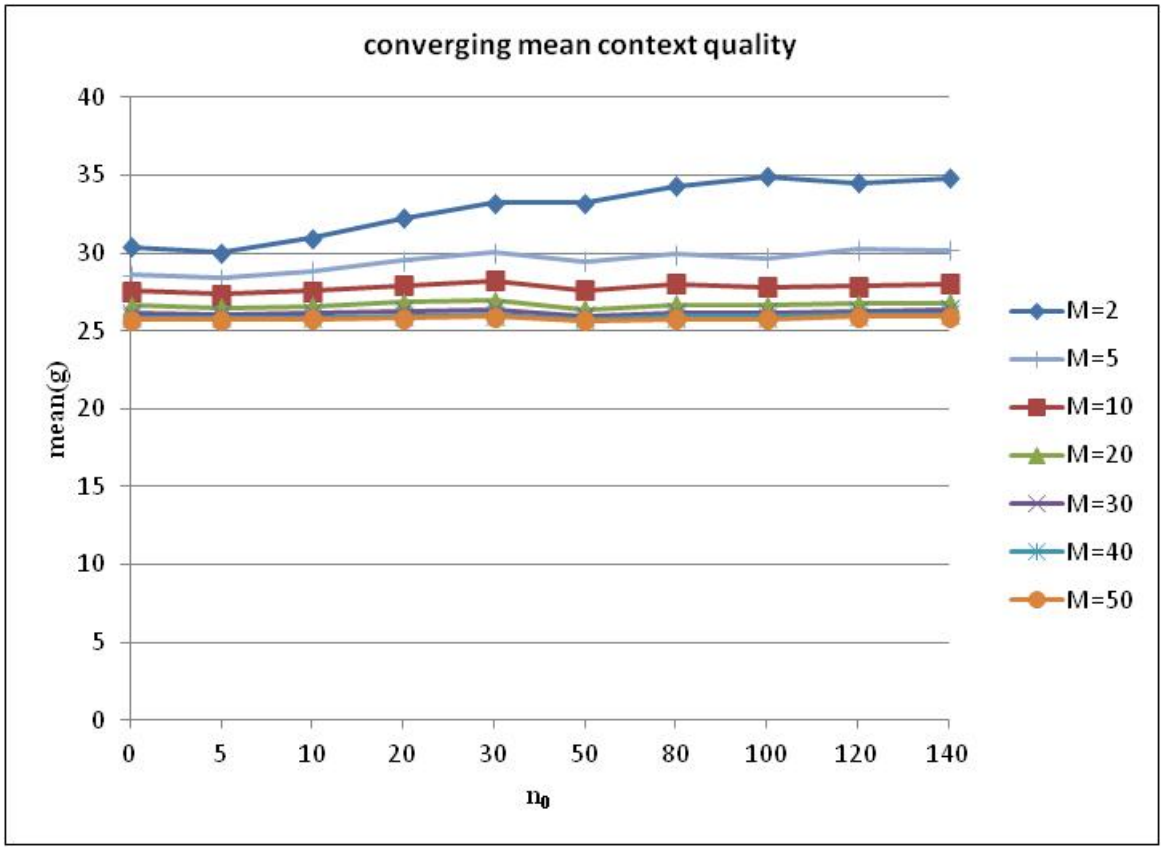
5	30	5	48,141	30,024	15,213	21.644,9	42,29
5	50	7	48,236	29,450	15,844	19.881,9	41,38
5	80	9	48,245	29,946	15,777	18.277,4	39,00
5	100	10	48,445	29,666	15,596	16.857,3	36,12
5	120	11	48,335	30,227	15,873	17.121,1	36,98
5	140	12	48,570	30,211	15,768	15.387,7	33,74
10	0	0	48,145	27,576	14,105	37.945,3	68,17
10	5	2	48,066	27,380	14,236	29.755,4	52,51
10	10	3	48,010	27,574	14,213	28.612,7	50,83
10	20	4	48,105	27,910	13,961	26.025,1	47,26
10	30	5	48,161	28,250	13,876	22.869,2	43,46
10	50	7	48,156	27,639	14,392	20.538,7	40,63
10	80	9	48,160	28,005	14,206	19,766,4	40,06
10	100	10	48,190	27,802	14,347	17.839,2	36,30
10	120	11	48,215	27,879	14,036	18.003,0	36,49
10	140	12	48,335	28,042	14,156	17.078,0	34,78
20	0	0	48,115	26,666	14,031	35.901,0	62,92
20	5	2	48,030	26,476	14,077	29.925,2	51,78
20	10	3	48,035	26,595	14,013	28.809,7	50,11
20	20	4	48,045	26,828	13,899	26.631,5	47,18
20	30	5	48,106	26,965	13,814	23.747,9	42,99
20	50	7	48,115	26,387	14,145	21.547,7	40,19
20	80	9	48,085	26,686	14,012	21.067,5	39,76
20	100	10	48,150	26,621	14,099	19.635,3	37,46
20	120	11	48,125	26,723	13,976	19.830,9	37,69
20	140	12	48,160	26,779	14,026	19.162,1	36,30
30	0	0	48,060	26,149	14,191	33.359,8	57,71
30	5	2	48,005	26,075	14,223	29.167,7	49,90
30	10	3	48,035	26,180	14,169	28.719,5	49,28
30	20	4	48,050	26,257	14,102	26.370,5	45,91
30	30	5	48,050	26,378	14,007	23.976,1	42,57
30	50	7	48,030	25,997	14,291	22.195,3	40,33
30	80	9	48,015	26,189	14,182	22.077,3	40,23
30	100	10	48,075	26,175	14,181	21.258,8	39,03
30	120	11	48,075	26,305	14,126	21.161,2	38,86
30	140	12	48,065	26,372	14,122	20.587,0	38,03
40	0	0	48,050	25,873	14,347	32.113,9	55,02
40	5	2	48,030	25,856	14,356	29.195,4	49,66
40	10	3	48,025	25,888	14,316	27.949,3	47,65
40	20	4	48,015	25,989	14,252	26.093,9	45,22
40	30	5	48,020	26,046	14,212	23.855,5	41,90
40	50	7	48,030	25,787	14,415	22.535,1	40,44
40	80	9	48,020	25,920	14,313	22.617,3	40,57
40	100	10	48,075	25,944	14,324	21.982,0	39,61
40	120	11	48,055	26,049	14,247	21.861,0	39,48
40	120	12	48,070	26,086	14,270	21.479,9	38,68
50	0	0	48,020	25,707	14,464	31.247,3	53,05
50	5	2	48,010	25,704	14,453	28.494,3	48,33
50	10	3	48,015	25,757	14,420	28.040,0	47,54
50	20	4	48,010	25,816	14,388	25.876,1	44,30
50	30	5	48,020	25,895	14,318	24.158,7	42,10
50	50	7	48,005	25,639	14,502	23.102,9	41,03
50	80	9	48,010	25,746	14,439	22.845,8	40,59
50	100	10	48,035	25,781	14,424	22.497,5	40,13
50	120	11	48,045	25,902	14,355	22.385,1	40,00
50	140	12	48,070	25,927	14,380	21.971,1	39,26



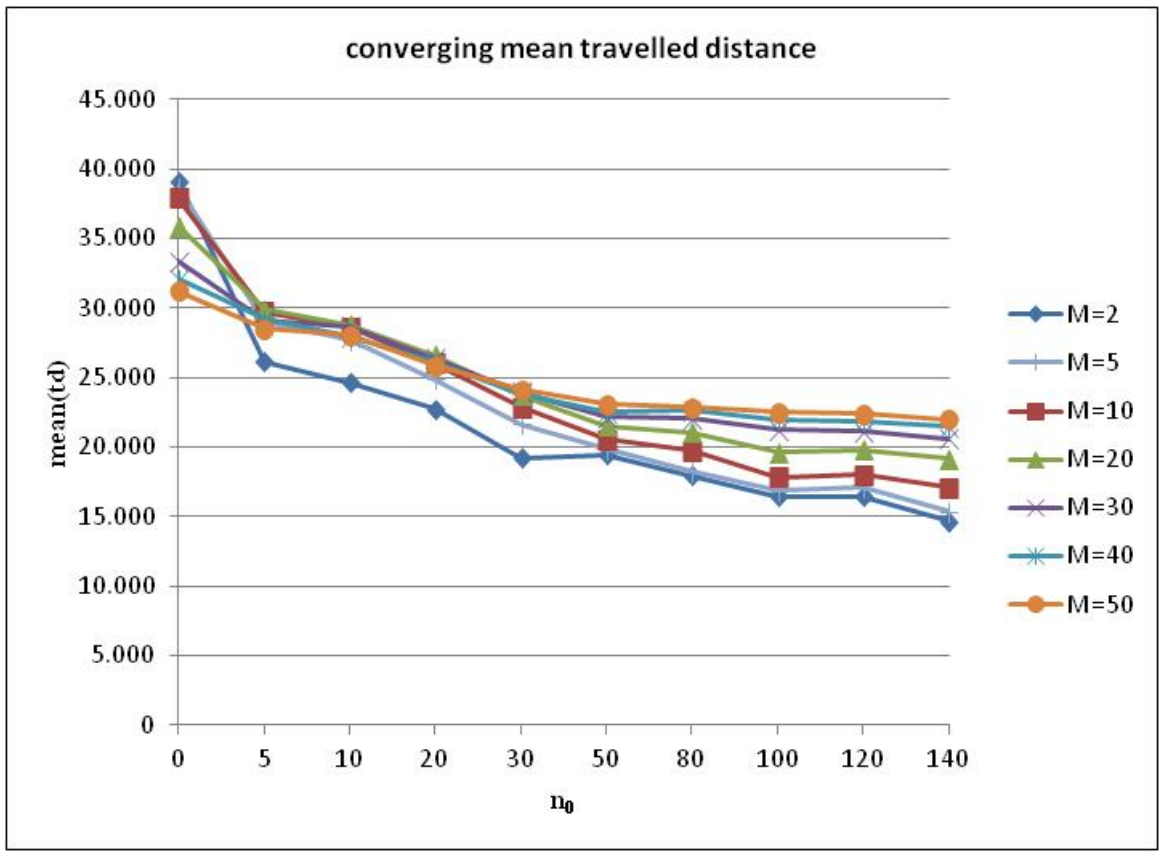
Σχήμα 22: OSPSO. Μέθοδος non-linear. $\overline{g(t)} = f\left(n_0\right)$ **για διάφορες τιμές του** M



Σχήμα 23: OSPSO. Μέθοδος non-linear. $\overline{d_t} = f\left(n_0\right)$ **για διάφορες τιμές του** M



Σχήμα 24: OSPSO. Μέθοδος chaotic. $\overline{g(t)} = f\left(n_0\right)$ **για διάφορες τιμές του** M



Σχήμα 25: OSPSO. Μέθοδος chaotic. $\overline{d_t} = f\left(n_0\right)$ **για διάφορες τιμές του** M

6.3.3. Συμπεράσματα – Παρατηρήσεις

Η εφαρμογή της Θεωρίας Βέλτιστης Παύσης στον αλγόριθμο Βελτιστοποίησης Σμήνους Σωματιδίων (βλέπε Πίνακες 11-14 και Σχήματα 22-25) έχει τα εξής αποτελέσματα:

- Η συγκλίνουσα μέση ποιότητα πλαισίου $\overline{g(t)}$ παραμένει σχεδόν σταθερή με την μεταβολή του διαστήματος παρατήρησης n_0, για τιμές του πλήθους πηγών $M \geq 10$. Για $M < 10$ η ποσότητα $\overline{g(t)}$ αυξάνεται (απαξιώνεται) με την αύξηση του n_0 για $10 \leq n_0 \leq 100$. Η μέγιστη αύξηση είναι περίπου $26,9\%$ ($M = 2$, μέθοδος non-linear). Για $M < 10$ και $n_0 > 100$ ή $n_0 < 10$ η ποσότητα $\overline{g(t)}$ παραμένει σχεδόν σταθερή.

- Η συγκλίνουσα μέση απόσταση που διανύει ένα σωματίδιο του σμήνους μέχρι τη χρονική στιγμή t, $\overline{d_t}$ ελαττώνεται σχεδόν γραμμικά με την αύξηση του διαστήματος παρατήρησης n_0 για τιμές του $n_0 \leq 100$. Η μείωση αυτή κυμαίνεται από $76,7\%$ έως $29,7\%$ για $2 \leq M \leq 50$ και $n_0 \leq 100$, για τις μεθόδους μεταβολής του βάρους αδράνειας που εξετάστηκαν. Αυτό υποδηλώνει ότι το ενεργειακό αποτύπωμα ενός σωματιδίου του σμήνους μικραίνει όσο αυξάνεται το διάστημα παρατήρησης για $n_0 \leq 100$. Μάλιστα, αυτό συμβαίνει δίχως απαξίωση της ποιότητας πληροφορίας πλαισίου για αριθμό πηγών $10 \leq M \leq 50$. Για $n_0 > 100$ η ποσότητα $\overline{d_t}$ παραμένει σχεδόν σταθερή.

- Η συγκλίνουσα μέση απόσταση που διανύει ένα σωματίδιο σε κάθε χρονικό βήμα του αλγόριθμου $\overline{d}$ ακολουθεί την ίδια κατανομή με την ποσότητα $\overline{d_t}$.

- Η εφαρμογή του κατωφλίου $\theta_{threshold}$ σε καμία περίπτωση δεν βελτιώνει την τιμή της συγκλίνουσας μέσης ποιότητας πλαισίου των σωματιδίων του σμήνους $\overline{g(t)}$.

- Η μέθοδος μεταβολής του βάρους αδράνειας random δεν ανταποκρίνεται καλά στην εφαρμογή της Θεωρίας Βέλτιστης Παύσης, όπως αποτυπώνεται στους Πίνακες 11, 12 και στο Σχήμα 21.

Κεφάλαιο 7: Γενικά Συμπεράσματα – Προοπτικές

7.1. Συμπεράσματα

Η Ανακάλυψη Πληροφορίας Πλαισίου αποτελεί ιδιαίτερα κρίσιμη διαδικασία για συστήματα με ικανότητα αυτοοργάνωσης, όπως είναι τα ρομποτικά συστήματα εποπτείας περιβαλλοντικών κινδύνων (π.χ. φωτιά).

Στην εργασία αυτή μελετάται η Ανακάλυψη Πληροφορίας Πλαισίου σε ένα σύστημα αυτόνομων (ρομποτικών) κόμβων, οι οποίοι κινούνται σε ένα δισδιάστατο χωρικό πλαίσιο. Το σύστημα αυτό αποτελείται από αισθητήριους και μη αισθητήριους κόμβους. Κάθε μη αισθητήριος κόμβος καθορίζει δυναμικά την κίνησή του με στόχο την απόκτηση της πιο πρόσφατης πληροφορίας πλαισίου. Οι αισθητήριοι κόμβοι ανιχνεύουν περιοδικά την πληροφορία πλαισίου και ακολουθούν τυχαία κίνηση στο χώρο. Η πληροφορία πλαισίου απαξιώνεται με την πάροδο του χρόνου. Το σύνολο των μη αισθητήριων κόμβων προσομοιώνεται από ένα σμήνος σωματιδίων. Έτσι, το Πρόβλημα Ανακάλυψης Πληροφορίας Πλαισίου (Context Discovery Problem) μετασχηματίζεται σε ένα πρόβλημα Βελτιστοποίησης Σμήνους Σωματιδίων (Particle Swarm Optimization), του οποίου οι λύσεις αντιστοιχούν στους αισθητήριους κόμβους. Το Πρόβλημα Ανακάλυψης Πληροφορίας Πλαισίου επεκτείνεται με τον κατάλληλο χρονοπρογραμματισμό των μετακινήσεων των μη αισθητήριων κόμβων. Ενδεχόμενη καθυστέρηση στην αλλαγή θέσης ενός κόμβου μπορεί να συντελέσει στη σύλληψη πληροφορίας πλαισίου υψηλότερης ποιότητας. Ο χρονοπρογραμματισμός αυτός βασίζεται στη Θεωρία Βέλτιστης Παύσης (Optimal Stopping Theory), της οποίας η επίδραση εξετάζεται στις βασικές παραμέτρους του συστήματος. Η εφαρμογή της OST βασίστηκε σε μια παραλλαγή του Προβλήματος της Γραμματέως.

Στα πλαίσια αυτά, υλοποιήθηκαν δύο βασικοί αλγόριθμοι: ο αλγόριθμος Βελτιστοποίησης Σμήνους Σωματιδίων (PSO) και ο αλγόριθμος Βελτιστοποίησης Σμήνους Σωματιδίων με εφαρμογή της Θεωρίας Βέλτιστης Παύσης (OSPSO).

Ο προτεινόμενος αλγόριθμος Βελτιστοποίησης Σμήνους Σωματιδίων (PSO) ακολουθεί το μοντέλο *lbest*, ενώ η γειτονιά ενός σωματιδίου καθορίζεται χωρικά. Η υλοποίηση της Θεωρίας Βέλτιστης Παύσης στον αλγόριθμο OSPSO αποτελεί προσαρμογή της παραλλαγής του Προβλήματος της Γραμματέως, στην οποία η απολαβή είναι ανάλογη του μεγέθους της επιλεγόμενης παρατήρησης.

Κατά την εφαρμογή των δύο αλγόριθμων εξετάστηκαν δύο τρόποι υπολογισμού της ατομικής βέλτιστης θέσης (*pBest*) ενός σωματιδίου του σμήνους, ο κλασικός (standard) και ο δίχως μνήμη (memoryless), καθώς και πέντε μέθοδοι μεταβολής του βάρους αδράνειας: sigmoid, non-linear decreasing, random, chaotic και chaotic random.

Οι βασικές παράμετροι του συστήματος για τις οποίες ενδιαφερόμαστε είναι η μέση ποιότητα πληροφορίας πλαισίου των σωματιδίων του σμήνους $\overline{g(t)}$ και η μέση απόσταση που διανύει ένα σωματίδιο του σμήνους μέχρι το χρονικό βήμα t, $\overline{d_t}$. Η δεύτερη παράμετρος είναι ανάλογη με τη μέση ενέργεια που καταναλώνει ένα σωματίδιο του σμήνους.

Η πειραματική μελέτη έδειξε ότι οι παράμετροι $\overline{g(t)}$, $\overline{d_t}$ εξαρτώνται τόσο από τον αριθμό των αισθητήριων κόμβων (πηγών) M όσο και από τη συχνότητα ανίχνευσης (ανανέωσης) της πληροφορίας πλαισίου q. Συγκεκριμένα, η μέση ποιότητα πληροφορίας πλαισίου $\overline{g(t)}$ βελτιώνεται (μειώνεται) τόσο με την αύξηση του M για σταθερή συχνότητα q όσο και με την αύξηση του q για σταθερό αριθμό πηγών M. Το ποσοστό βελτίωσης εξαρτάται από τη μέθοδο μεταβολής του βάρους αδράνειας. Η μέση απόσταση που διανύει ένα σωματίδιο $\overline{d_t}$ παρουσιάζει ανάλογη συμπεριφορά.

Η μέθοδος μεταβολής του βάρους αδράνειας non-linear (decreasing) έδωσε τα καλύτερα αποτελέσματα, για τον κλασικό τρόπο υπολογισμού της θέσης *pBest*, ενώ η μέθοδος random για το δίχως μνήμη τρόπο υπολογισμού.

Η εφαρμογή της Θεωρίας Βέλτιστης Παύσης στον αλγόριθμο Βελτιστοποίησης Σμήνους Σωματιδίων βελτιώνει σημαντικά την ενεργειακή συμπεριφορά των σωματιδίων του σμήνους, δίχως να επηρεάζει τη μέση ποιότητα πληροφορίας πλαισίου του σμήνους, όταν ο αριθμός των αισθητήριων κόμβων (πηγών) M αντιστοιχεί τουλάχιστον στο 10% του πλήθους των σωματιδίων του σμήνους N, δηλαδή όταν $M \geq 10$ για $N = 100$. Μάλιστα, για διάστημα παρατήρησης $n_0 = 5$, η παράμετρος $\overline{g(t)}$ παρουσιάζει μικρή βελτίωση. Για διαστήματα παρατήρησης $n_0 > 5$, η παράμετρος $\overline{g(t)}$ παραμένει σχεδόν σταθερή. Στις περιπτώσεις που ο αριθμός των αισθητήριων κόμβων $M < 0{,}1 \cdot N$, η μέση ποιότητα πληροφορίας πλαισίου των σωματιδίων του σμήνους χειροτερεύει με την αύξηση του διαστήματος παρατήρησης.

7.2. Προοπτικές

Ο συνδυασμός Βελτιστοποίησης Σμήνους Σωματιδίων και Θεωρίας Βέλτιστης Παύσης δίνει ενθαρρυντικά αποτελέσματα για την αντιμετώπιση του Προβλήματος Ανακάλυψης Πληροφορίας Πλαισίου σε συστήματα κινητών ρομποτικών κόμβων με ικανότητα αυτοοργάνωσης.

Ο αλγόριθμος OSPSO παρουσιάζει μικρή υπολογιστική πολυπλοκότητα και υλοποιείται εύκολα, οπότε μπορεί να ενσωματωθεί σε τέτοιου είδους συστήματα.

Για την περαιτέρω βελτίωση της απόδοσης του αλγόριθμου OSPSO αξίζει να διερευνηθούν και άλλες προσαρμοστικές μέθοδοι μεταβολής του βάρους αδράνειας, καθώς η μέθοδος sigmoid που υλοποιήθηκε δεν έδωσε τα αναμενόμενα αποτελέσματα. Ιδιαίτερη έμφαση θα πρέπει να δοθεί σε προσαρμοστικά μοντέλα της μεθόδου non-linear decreasing.

Επίσης, θα πρέπει να γίνει πιο ενδελεχής μελέτη του νέου τρόπου υπολογισμού της ατομικής βέλτιστης θέσης ενός σωματιδίου του σμήνους *pBest*, του δίχως μνήμη τρόπου υπολογισμού. Μερικά πρώτα αποτελέσματα δείχνουν ότι ο τρόπος αυτός οδηγεί σε πολύ μεγάλη μείωση της μέσης απόστασης που διανύει ένα σωματίδιο του σμήνους, άρα και σε πολύ μεγάλη μείωση της ενέργειας που απαιτείται για την κίνησή του, αλλά παρουσιάζει χειρότερες επιδόσεις ως προς τη μέση ποιότητα πληροφορίας πλαισίου των σωματιδίων του σμήνους.

Ένα άλλο πεδίο στο οποίο μπορεί να επεκταθεί η χρήση του αλγόριθμου OSPSO είναι η βέλτιστη ρύθμιση της ακτίνας μετάδοσης / ανίχνευσης των σωματιδίων του σμήνους, έτσι ώστε να μειώνεται η ενέργεια που καταναλώνουν χωρίς να επηρεάζεται η ποιότητα πληροφορίας πλαισίου του σμήνους.

Αναφορές

[1] C. Blum and D. Merkle, *Swarm Intelligence - Introduction and Applications*.: Springer-Verlag, 2008.

[2] Andries P. Engelbrecht, *Computational Intelligence, An Introduction*, 2nd ed.: John Wiley & Sons, Ltd, 2007.

[3] G. Beni, "The concept of cellular robotic systems", in *Proceedings of the IEEE International Symposium on Intelligent Systems*, 1988, pp. 57-62.

[4] J. Kennedy and R. C. Eberhart, "Particle Swarm Optimization", in *Proceedings of the IEEE International Joint Conference on Neural Networks*, 1995, pp. 1942-1948.

[5] Andries P. Engelbrecht, *Fundamentals of Computational Swarm Intelligence*.: John Wiley & Sons, Ltd, 2005.

[6] P. N. Suganthan, "Particle Swarm Optimizer with Neighborhood Operator", in *Proceedings of the IEEE Congress on Evolutionary Computation*, 1999, pp. 1985-1962.

[7] R. C. Eberhart, P. K. Simpson, and R. W. Dobbins, *Computational Intelligence PC Tools*, 1st ed.: Academic Press Professional, 1996.

[8] Y. Shi and R. C. Eberhart, "A Modified Particle Swarm Optimizer", in *Proceedings of the IEEE Congress on Evolutionary Computation*, 1998, pp. 69-73.

[9] R. C. Eberhart and Y. Shi, "Particle Swarm Optimization: Developments, Applications and Resources", in *Proceedings of the IEEE Congress on Evolutionary Computation*, vol. 1, 2001, pp. 27-30.

[10] J. C. Bansal et al., "Inertia Weight Strategies in Particle Swarm Optimization", in *Third World Congress on Nature and Biologically Inspired Computing*, 2011, pp. 640-647.

[11] R. C. Eberhart and Shi Y., "Tracking and Optimizing Dynamic Systems with Particle Swarms", in *Proceedings of the IEEE Congress on Evolutionary Computation*, vol. 1, 2001, pp. 94-100.

[12] A. Ratnaweera, S. Halgamuge, and H. Watson, "Particle Swarm Optimization with Self-Adaptive Acceleration Coefficients", in *Proceedings of the First International Conference on Fuzzy Systems and Knowledge Discovery*, 2003, pp. 264-268.

[13] T. Peram, K. Veeramachaneni, and C. K. Mohan, "Fitness-Distance-Ratio based Particle Swarm Optimization", in *Proceedings of the IEEE Swarm Intelligence Symposium*, 2003, pp. 174-181.

[14] G. Venter and J. Sobieszczanski-Sobieski, "Particle Swarm Optimization", *Journal for the American Institute of Aeronautics and Astonautics*, vol. 41, no. 8, pp. 1583-1589, 2003.

[15] M. Clerc, "Think Locally, Act Locally: The Way of Life of Cheap-PSO, An Adaptive PSO", Technical report 2001.

[16] Zhi-Hui Zhan, Jun Zhang, Yun Li, and Henry Shu-Hung Chung, "Adaptive Particle Swarm Optimization", *IEEE Transactions on Systems, Man and Cybernetics - Part B: Cybernetics*, vol. 41, no. 6, pp. 1362 - 1381, September 2011.

[17] G. F. Teng, A. X. Wang, and Y. M. Yao Y. Feng, "Chaotic Inertia Weight in Particle Swarm Optimization", in *Second International Conference on Innovative Computing, Information and Control*, 2007, pp. 475 - 475.

[18] Y. Chi and R. C. Eberhart, "Fuzzy Adaptive Particle Swarm Optimization", in *Proceedings of the IEEE Congress on Evolutionary Computation*, vol. 1, 2001, pp. 101-106.

[19] M. Clerc and J. Kennedy, "The Particle Swarm-Explosion, Stability and Convergence in a Multidimensional Complex Space", *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 1, pp. 58-73, 2002.

[20] J. Kennedy, "The Particle Swarm: Social Adaptation of Knowledge", in *Proceedings of the IEEE International Conference on Evolutionary Computation*, 1997, pp. 303-308.

[21] A. Carlisle and G. Dozier, "Adapting Particle Swarm Optimization to Dynamic Environments", in *Proceedings of the International Conference on Artificial Intelligence*, 2000, pp. 429-434.

[22] F. van den Bergh, "An Analysis of Particle Swarm Optimizers", Department of Computer Science, University of Pretoria, Pretoria, South Africa, PhD thesis 2002.

[23] R. C. Eberhart and X. Hu, "Tracking Dynamic Systems with PSO: Where's the Cheese?", in *Proceedings of the Workshop on Particle Swarm Optimization*, 2001, pp. 80-83.

[24] Ozcan E. and Mohan C. K., "Particle Swarm Optimization: Surfing the Waves", in *Proceedings of the IEEE Congress on Evolutionary Computation*, vol. 3, 1999.

[25] Carlisle A. and Dozier G., "Tracking Changing Extrema with Adaptive Particle Swarm Optimizer", in *Proceedings of the Fifth Biannual Word Automation Congress*, 2002, pp. 265-270.

[26] Xiaohui Hu and R. C. Eberhart, "Adaptive Particle Swarm Optimization: Detection and Response to Dynamic Systems", *Proceedings of the 2002 Congress on Evolutionary*

Computation, CEC 2002, pp. 1666 - 1670, 2002.

[27] Yong Feng, Gui-Fa Teng, Ai-Xin Wang, and Yong-Mei Yao, "Chaotic Inertia Weight in Particle Swarm Optimization", in *Second International Conference on Innovative Computing, Information and Control*, 2007, pp. 475 - 475.

[28] A. Carlisle and G. Dozier, "Tracking Changing Extrema with Adaptive Particle Swarm Optimizer", in *Proceedings of the Fifth Biannual Word Automation Congress*, 2002, pp. 265-270.

[29] Thomas S. Ferguson, *Optimal Stopping and Applications*.: Mathematics Department, UCLA, 2008.

[30] A. Wald, "Sequential Tests of Statistical Hypotheses", *Ann. Math. Statist.*, vol. 16, no. 2, pp. 117-186, 1945.

[31] L. J. Snell, "Applications of martingale system theorems", *Trans. Amer. Math.*, vol. 73, pp. 293-312, 1952.

[32] Y. S. Chow, H. Robbins, and D. Siegmund, *Great Expectations: The Theory of Optimal Stopping*. Boston: Houghton Mifflin Company, 1971.

[33] T. S. Ferguson, "Who solved the secretary problem ?", *Statistical Science*, no. 4, pp. 282-296, 1989.

[34] J. Neil Bearden, "Skip the Square Root of n: A New Secretary Problem", *Journal of Mathematical Psychology*, submitted 9 June 2005.

[35] J. Neil Bearden, "A new secretary problem with rank-based selection and cardinal payoffs", *Journal of Mathematical Psychology*, no. 50, pp. 58-59, 2006.

[36] (2012, Φεβρουάριος) Dictionary.com. [Online]. http://dictionary.reference.com/browse/context

[37] Χρήστος Αναγνωστόπουλος, "Μοντελοποίηση και Αρχιτεκτονική Συστημάτων Κινητού Υπολογισμού", Τμήμα Πληροφορικής και Τηλεπικοινωνιών, Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών, Διδακτορική Διατριβή 2008.

[38] B. N. Schilit and Theimer, "Disseminating active map information to mobile hosts", *IEEE Network*, vol. 8, no. 5, pp. 22-32, 1994.

[39] P. J. Brown, J. D. Bovey, and X. Chen, "Context-Aware Applications: From the Laboratory to the Marketplace", *IEEE Personal Communications*, vol. 4, no. 5, pp. 58-64, 1997.

[40] P. J. Brown, "The Stick-e Document: a Framework for Creating Context-Aware

Application", *Electronic*, pp. 259-272, 1996.

[41] T. Rodden, K. Cheverst, K. Davies, and A. Dix, "Exploiting Context in HCI Design for Mobile Systems", in *Workshop on Human Computer Interaction with Mobile Devices*, 1998.

[42] A. Dey and G. Abowd, "Towards a Better Understanding of Context and Context-Awareness", in *CHI 2000*, The Hague, The Netherlands, 2000.

[43] C. Anagnostopoulos, A. Tsounis, and S. Hadjiefthymiades, "Context Awareness in Mobile Computing Environments", *Wireless Personal Communications*, vol. 42, pp. 445-464, 2006.

[44] S. Loke, *Context Aware Pervasive Systems - Architectures for a new breed of applications*. USA: Auerbach Publications, 2007.

[45] M. Dorigo, V. Maniezzo, and A. Colorni, "Ant System: Optimization by a Colony of Cooperating Agents", *IEEE Transactions on Systems, Man, and Cybernetics – Part B*, vol. 26, no. 1, pp. 29–41, 1996.

[46] C. Anagnostopoulos and S. Hadjiefthymiades, "Context Discovery in Mobile Environments: A Particle Swarm Optimization Approach", *Autonomic Computing and Communications Systems*, vol. 23, pp. 160-175, 2010.

[47] C. Anagnostopoulos, O. Sekkas, and S. Hadjiefthymiades, "Context Fusion: Dealing with Sensor Reliability", in *IEEE Int. Workshop on Information Fusion and Dissemination in Wireless Sensor Networks - IEEE Int. Conference on Mobile Ad-hoc and Sensor Systems*, 2007, pp. 1-7.

[48] T. Camp, J. Boleng, and V. Davies, "A Survey of Mobility Models for Ad Hoc Network Research", *Wireless Communication & Mobile Computing (WCMC): Special issue on Mobile Ad Hoc Networking*, vol. 2, no. 5, pp. 483-502, 2002.

[49] Mathieu Boutin. (2011, April) Matlab Central: Random Waypoint mobility model. [Online]. http://www.mathworks.com/matlabcentral/fileexchange/30939-random-waypoint-mobility-model

[50] Hui Lu and Xiao Chen, "A New Particle Swarm Optimization with a Dynamic Inertia Weight for Solving Constrained Optimization Problems", *Information Technology Journal*, no. 10, pp. 1536-1544, 2011.

[51] G. Wang, Z. Chen, Z. Yu C. Dong, "A Method of Self-Adaptive Inertia Weight for PSO", *International Conference on Computer Science and Software Engineering*, vol. 1, pp. 1195 - 1198, 2008.

[52] S. Cong, X. Y. Feng C. S. Feng, "A new adaptive inertia weight strategy in particle swarm optimization", in *IEEE Congress on Evolutionary Computation, CEC 2007*, 2007, pp. 4186 - 4190.

[53] X. Yang, J. Yuan, and H. Mao, "A modified particle swarm optimizer with dynamic adaptation", *Applied Mathematics and Computation*, vol. 189, pp. 1205–1213, 2007.

[54] R. F. Malik, T. A. Rahman, S. Z. Mohd. Hashim, and R. Ngah, "New Particle Swarm Optimizer with Sigmoid Increasing Inertia Weight", *International Journal of Computer Science and Security (IJCSS)*, vol. 1, no. 2, pp. 35 - 44, 2007.

[55] J. Yuan, J. Yuan, H. Mao X. Yang, "A modified particle swarm optimizer with dynamic adaptation", *Applied Mathematics and Computation*, vol. 189, pp. 1205–1213, 2007.

[56] T. A. Rahman, S. Z. Mohd. Hashim, R. Ngah R. F. Malik, "New Particle Swarm Optimizer with Sigmoid Increasing Inertia Weight", *International Journal of Computer Science and Security (IJCSS)*, vol. 1, no. 2, pp. 35 - 44, 2007.

[57] P. K. Singh, M. Saraswat, A. Verma, S. S. Jadon, A. Abraham J. C. Bansal, "Inertia Weight Strategies in Particle Swarm Optimization", in *Third World Congress on Nature and Biologically Inspired Computing*, 2011, pp. 640-647.

[58] Jun Zhang, Yun Li, Henry Shu-Hung Chung Zhi-Hui Zhan, "Adaptive Particle Swarm Optimization", *IEEE Transactions on Systems, Man and Cybernetics - Part B: Cybernetics*, vol. 41, no. 6, pp. 1362 - 1381, September 2011.

[59] R. C. Eberhart Xiaohui Hu, "Adaptive particle swarm optimization: detection and response to dynamic systems", *Proceedings of the 2002 Congress on Evolutionary Computation, CEC 2002*, pp. 1666 - 1670, 2002.

ΠΑΡΑΡΤΗΜΑ

Περιγραφή Λογισμικού

Ο κώδικας MATLAB που υλοποιεί τους αλγόριθμους PSO και OSPSO αποτελείται από τα εξής αρχεία και συναρτήσεις:

demo_OSPSO_v4_0.m: Αρχείο όπου ορίζονται οι βασικές παράμετροι του συστήματος και από το οποίο καλούνται οι συναρτήσεις που υλοποιούν τους κυρίως αλγόριθμους.

OSPSO_v0_94.m: Η κυρίως συνάρτηση υλοποίησης του αλγόριθμου PSO.

OSPSO_v3_0_n2.m: Η κυρίως συνάρτηση υλοποίησης του αλγόριθμου OSPSO.

l_pBest_standard_v1_x: Συνάρτηση όπου υπολογίζονται οι θέσεις *lBest, pBest* με την κανονική μέθοδο για τον αλγόριθμο PSO.

l_pBest_memoryless_v1_x: Συνάρτηση όπου υπολογίζονται οι θέσεις *lBest, pBest* με τη μέθοδο χωρίς μνήμη για τον αλγόριθμο PSO.

l_pBest_standard_mstatus_v3_0a: Συνάρτηση όπου υπολογίζονται οι θέσεις *lBest, pBest* με την κανονική μέθοδο καθώς και οι καταστάσεις κίνησης των σωματιδίων για τον αλγόριθμο OSPSO.

l_pBest_memoryless_mstatus_v3_0a: Συνάρτηση όπου υπολογίζονται οι θέσεις *lBest, pBest* με τη μέθοδο δίχως μνήμη καθώς και οι καταστάσεις κίνησης των σωματιδίων για τον αλγόριθμο OSPSO.

sigmoid_pp_v2: Συνάρτηση όπου υπολογίζονται παράμετροι της μεθόδου μεταβολής του βάρους αδράνειας w sigmoid per particle.

sigmoid_univ: Συνάρτηση όπου υπολογίζονται παράμετροι της μεθόδου μεταβολής του βάρους αδράνειας w sigmoid universal.

inertia.m: Συνάρτηση όπου υλοποιείται η μέθοδος υπολογισμού του βάρους αδράνειας που επιλέχθηκε.

p_clamping.m: Συνάρτηση που υλοποιεί την αποκοπή θέσης (position clamping).

v_clamping.m: Συνάρτηση που υλοποιεί την αποκοπή ταχύτητας (velocity clamping).

random_waypoint.m: Συνάρτηση όπου υπολογίζεται η τυχαία κίνηση σωματιδίων και πηγών σύμφωνα με το Random Waypoint Mobility Model.

pso_plot.m: Συνάρτηση απεικόνισης της κίνησης των σωματιδίων.

randnum.m: Συνάρτηση υπολογισμού τυχαίου αριθμού μέσα σε ένα διάστημα τιμών.

Κώδικας MATLAB

demo_OSPSO_v4_0.m

```
% demo_OSPSO_v4_0.m
% main file used for parameter input and
% calling of the PSO and OSPSO algorithms
%
% version 4.0
% March 2012

clear all; clc; close all;

D = 2; % problem dimensions
I_max = 1000; % number of iterations

xMin = -49; % 2-D space coordinates
xMax = 50;
yMin = -49;
yMax = 50;

Lx = xMax - xMin + 1; % terrain x dimension
Ly = yMax - yMin + 1; % terrain y dimension
R = 0.001*Lx*Ly; % transmission range;

vMin = 0.1;
vMax = 2;

opt_g = 0; % optimal value of g
theta = 100; % threshold of function g


N = input('Choose the number of particles of the swarm: '); % number of swarm particles
fprintf('\n');

M = input('Choose the number of sources: '); % number of sources
fprintf('\n');

freq = input('Choose source(s) sensing rate (Hz): '); % sensing rate of sources
fprintf('\n');

pB_method = input('Choose pBest calculation model (S)tandard or (M)emoryless (S or M): ', 's');
fprintf('\n');

if isequal(pB_method, 'S') || isequal(pB_method, 's') || isequal(pB_method, 'Standard') || isequal(pB_method,
'standard') || isequal(pB_method, 'STANDARD')
   pBest_method = 1;
elseif isequal(pB_method, 'M') || isequal(pB_method, 'm') || isequal(pB_method, 'Memoryless') ||
isequal(pB_method, 'memoryless') || isequal(pB_method, 'MEMORYLESS')
   pBest_method = 2;
else
   fprintf('\n');
   disp('Okay, please change the settings and try again.');
   break;
end

iws_name = {'Linear decreasing', 'Non-linear decreasing', ...
            'Non-linear decreasing 2', 'Random', 'Chaotic', 'Chaotic Random', ...
            'Sigmoid function (universal)', 'Sigmoid function (per particle)'};
```

```
disp('Inertia Weight Update Strategies:');
for s = 1:length(iws_name)
    fprintf('%d) %s inertia weight\n', s, iws_name{s});
end

w_method = input('Choose inertia weight strategy (1 - 8): ');
fprintf('\n');

if (w_method > 0) && (w_method < 7)
    disp('Inertia Weight Model:');
    disp(' 1) Universal (common inertia weight for all particles per iteration)');
    disp(' 2) Per particle (based on its active participation to the swarm)');
    w_model = input('Choose inertia weight model (1 - 2): ');
    fprintf('\n');
elseif w_method == 7
    w_model = 1;
elseif w_method == 8
    w_model = 2;
else
    fprintf('\n');
    disp('Okay, please change the settings and try again.');
    break;
end

if w_model == 1
    universal_w_update = true;
elseif w_model == 2
    universal_w_update = false;
else
    fprintf('\n');
    disp('Okay, please change the settings and try again.');
    break;
end


ost_choice = input('Enable Optimal Stopping; (Yes or No): ', 's'); % OnOff Optimal Stopping

if isequal(ost_choice, 'Y') || isequal(ost_choice, 'y') || isequal(ost_choice, 'Yes') || isequal(ost_choice, 'yes') ||
isequal(ost_choice, 'YES')
    fprintf('\n');
    ost_enable = true;
    op_st_interval = input('Choose observation interval (Iterations) for optimal stopping: '); % selection of
Optimal Stopping observation interval
    fprintf('\n');
    opt_cutoff = round(sqrt(op_st_interval)); % optimal cutoff
elseif isequal(ost_choice, 'N') || isequal(ost_choice, 'n') || isequal(ost_choice, 'No') || isequal(ost_choice, 'no') ||
isequal(ost_choice, 'NO')
    ost_enable = false;
    op_st_interval = 0;
    opt_cutoff = 0;
    fprintf('\n');
else
    fprintf('\n');
    disp('Okay, please change the settings and try again.');
    break;
end
```

```
p_clamp = input('Position clamping (Yes or No): ', 's'); % OnOff position clamping

if isequal(p_clamp, 'Y') || isequal(p_clamp, 'y') || isequal(p_clamp, 'Yes') || isequal(p_clamp, 'yes') ||
isequal(p_clamp, 'YES')
   position_clamping =true;
elseif isequal(p_clamp, 'N') || isequal(p_clamp, 'n') || isequal(p_clamp, 'No') || isequal(p_clamp, 'no') ||
isequal(p_clamp, 'NO')
   position_clamping = false;
else
   fprintf('\n');
   disp('Okay, please change the settings and try again.');
   break;
end

v_clamp = input('Velocity clamping (Yes or No): ', 's');  % OnOff velocity clamping
fprintf('\n');

if isequal(v_clamp, 'Y') || isequal(v_clamp, 'y') || isequal(v_clamp, 'Yes') || isequal(v_clamp, 'yes') ||
isequal(v_clamp, 'YES')
   velocity_clamping =true;
elseif isequal(v_clamp, 'N') || isequal(v_clamp, 'n') || isequal(v_clamp, 'No') || isequal(v_clamp, 'no') ||
isequal(v_clamp, 'NO')
   velocity_clamping = false;
else
   fprintf('\n');
   disp('Okay, please change the settings and try again.');
   break;
end

p_graphics = input('Display particle movement graphics (Yes or No): ', 's');  % OnOff velocity clamping
fprintf('\n');

if isequal(p_graphics, 'Y') || isequal(p_graphics, 'y') || isequal(p_graphics, 'Yes') || isequal(p_graphics, 'yes') ||
isequal(p_graphics, 'YES')
   particle_graphics_flag =true;
elseif isequal(p_graphics, 'N') || isequal(p_graphics, 'n') || isequal(p_graphics, 'No') || isequal(p_graphics, 'no') ||
isequal(p_graphics, 'NO')
   particle_graphics_flag = false;
else
   fprintf('\n');
   disp('Okay, please change the settings and try again.');
   break;
end

Nr = input('Choose the number of runs of the algorithm: ');  % selection of the runs of the algorithm

decay_flag = true;
debug_mode = false;

pso_flags = [decay_flag velocity_clamping position_clamping universal_w_update particle_graphics_flag
debug_mode];

% Warming up the random number generator
for t=1:1:10000
   rr=randnum(0,1);
end

% naming of the various methods used (for display purposes)
```

```
[ipB_model iw_model vc pc] = method_display(universal_w_update, pBest_method, w_method,
velocity_clamping, position_clamping);

if Nr == 1
   tic;
   tStart = tic;
   if ost_enable == false
      [mg_t0 mg mgstd mtd md] = OSPSO_v0_94(N, M, D, I_max, xMin, xMax, yMin, yMax, R, vMin, vMax,
theta, freq, pBest_method, w_method, pso_flags, Nr, iws_name);
   else
      [mg_t0 mg mgstd mtd md] = OSPSO_v3_0_n2(N, M, D, I_max, xMin, xMax, yMin, yMax, R, vMin,
vMax, theta, freq, op_st_interval, opt_cutoff, pBest_method, w_method, pso_flags, Nr, iws_name);
   end
   tElapsed = toc(tStart);
   fprintf('\nNumber of swarm particles = %d\n', N);
   fprintf('\Number of Source(s) = %d,      Source(s) sensing rate = %3.3f Hz\n', M, freq);
   fprintf('pBest calculation model: %s\n', ipB_model);
   fprintf('w update strategy: %s %s\n', iws_name{w_method}, iw_model);
   fprintf('Optimal stopping: Observation interval = %d,    Optimal cutoff = %d\n', op_st_interval, opt_cutoff);
   fprintf('Velocity clamping: %s     Position clamping: %s\n', vc, pc);
   fprintf('Algorithm runs = %d,     Iterations per run T = %d\n', Nr, I_max);
   fprintf('\nMean value of g(T) = %3.3f     Mean value of g = %3.3f     std of g = %3.3f \n', mg_t0, mg, mgstd);
   fprintf('Mean value of the travelled distance per particle td = %6.1f\n', mtd);
   fprintf('Mean value of the movement distance per particle d = %4.2f\n', md);
   fprintf('Elapsed Time = %3.2f min\n', tElapsed/60);
elseif Nr > 1
   mg = zeros(5,Nr); % initialization of mg matrix (5 x Nr)
   fprintf('Iterations per run T = %d\n', I_max);
   tic;
   tStart = tic;
   for i = 1:1:Nr
      fprintf('Run = %d,   Running...', i);
      if ost_enable == false
         [mg(1,i) mg(2,i) mg(3,i) mg(4,i) mg(5,i)] = OSPSO_v0_94(N, M, D, I_max, xMin, xMax, yMin, yMax,
R, vMin, vMax, theta, freq, pBest_method, w_method, pso_flags, Nr, iws_name);
      else
         [mg(1,i) mg(2,i) mg(3,i) mg(4,i) mg(5,i)] = OSPSO_v3_0_n2(N, M, D, I_max, xMin, xMax, yMin,
yMax, R, vMin, vMax, theta, freq, op_st_interval, opt_cutoff, pBest_method, w_method, pso_flags, Nr,
iws_name);
      end
      fprintf('\b\b\b\b\b\b\b\b\bMean value of g(T) = %3.3f     Mean value of g = %3.3f     std of g = %3.3f
Mean value of  td = %6.1f     Mean value of d = %4.2f\n', mg(1,i), mg(2,i), mg(3,i), mg(4,i), mg(5,i));
   end
   tElapsed = toc(tStart);
   fprintf('\nNumber of swarm particles = %d\n', N);
   fprintf('Number of Source(s) = %d,      Source(s) sensing rate = %3.3f Hz\n', M, freq);
   fprintf('pBest calculation model: %s\n', ipB_model);
   fprintf('w update strategy: %s %s\n', iws_name{w_method}, iw_model);
   fprintf('Optimal stopping: Observation interval = %d,    Optimal cutoff = %d\n', op_st_interval, opt_cutoff);
   fprintf('Velocity clamping: %s     Position clamping: %s\n', vc, pc);
   fprintf('Algorithm runs = %d,     Iterations per run T = %d\n', Nr, I_max);
   fprintf('\nMean converging value of g(T) = %3.3f     Mean converging value of g = %3.3f     converging std of
g = %3.3f \n', mean(mg(1,:)), mean(mg(2,:)), mean(mg(3,:)));
   fprintf('Mean converging value of the travelled distance per particle td = %6.1f\n', mean(mg(4,:)));
   fprintf('Mean converging value of the movement distance per particle d = %4.2f\n', mean(mg(5,:)));
   fprintf('Elapsed Time = %3.2f min\n', tElapsed/60);
else
   fprintf('\n');
   disp('Okay, please change the settings and try again.');
   break;
end
```

OSPSO_v0_94.m

```
% OSPSO_v94.m
% function for the calculation of the PSO algorithm
%
% version 0.9.4
% March 2012

function [mg_t0 mg mgstd mtd md] = OSPSO_v0_94(N, M, D, I_max, xMin, xMax, yMin, yMax, R, vMin,
vMax, theta, freq, pBest_method, w_method, pso_flags, Nr, iws_name)

decay_flag = pso_flags(1);
velocity_clamping = pso_flags(2);
position_clamping = pso_flags(3);
universal_w_update = pso_flags(4);
particle_graphics_flag = pso_flags(5);
debug_mode = pso_flags(6);

% Initializations %%%%%%%%%%%%%%%%%

Iter = 1; % iteration
p_Iter = ones(1,N); % particle random movement iteration

c1 = 2; % definition of acceleration coefficients
c2 = c1;

w_max = 0.9; % definition of inertia weight bounds
w_min = 0.4;

rMax = sqrt((2*xMax)^2 + (2*yMax)^2);

f_w = 0; % w update parameter (universal)
f_w_p = zeros(1,N); % w update parameter (per particle)

w = w_max*ones(1,N); % initial inertia weight values array (per particle w update model)
w_h = zeros(I_max+1, 1, N); % w history record
Iter_p_w = zeros(1,N); % iteration per particle (for w update)

Mean_g_Iter = Inf*ones(1, N); % mean g for the swarm per iteration
Mean_d_Iter = Inf*ones(1, N); % mean travelled distance d for the swarm per iteration

x = zeros(N, D); % initialization of array x
x_h = zeros(I_max+1, N, D); % x history record
r_h1 = zeros(N, D); % temp matrix for the calculation of the r history record
r_h = zeros(I_max+1, N); % r history record
v = zeros(N, D); % initialization of array v

% random waypoint model
particle = random_waypoint(xMin, xMax, yMin, yMax, vMin, vMax, I_max, N);
source = random_waypoint(xMin, xMax, yMin, yMax, vMin, vMax, I_max, M);

g = theta+theta*ones(1,N); % initialization of values of g for particles
di = zeros(1,N); % initialization of values of d for particles
g_h = zeros(I_max+1,1,N); % g history record
g_s = zeros(1,M); % initialization of values of g for sources

% initialization of positions of particles
for i=1:1:N
   x(i,:) = [particle(i).x(p_Iter(i)) particle(i).y(p_Iter(i))];
end

x0 = x;
```

```
%g0 = g;

% initialization of velocities of particles
for i=1:1:N
    for j=1:1:D
        v(i,j) = randnum(vMin, vMax);
    end
end

% initialize local best (lbest)
l_x = x;
l_g = g;

% initialize personal best (pbest)
p_x =  x;
p_g = g;

% particle motion display figure
if particle_graphics_flag == true
    figure(1);
    eval('pso_plot');
end

%%%%%%%%%%%%%%%%%%%%%%%%

% First Iteration %%%%%%%%%%%%%%%

% context quality decay calculations
if decay_flag == true
    g_s_update = 0;
    update_cycle = round(1/freq);
end

% calculate  local best (lbest) & personal best (pbest)
if pBest_method == 1 % standard model
    [l_g l_x p_g p_x g ] = l_pBest_standard_v1_x(x, source, l_x, l_g, p_x, p_g, g, g_s, N, M, R, Iter);
elseif pBest_method == 2 % memoryless model
    [l_g l_x p_g p_x g ] = l_pBest_memoryless_v1_x(x, source, l_x, l_g, p_x, p_g, g, g_s, N, M, R, Iter);
else
    fprintf('Error !\n');
end


for i = 1:1:N
    if g(i) >= theta % continue random movement
        p_Iter(i) = p_Iter(i) + 1;
        x(i,:) = [particle(i).x(p_Iter(i)) particle(i).y(p_Iter(i))];
    else % swarm movement
        v(i,:) = w(i)*v(i,:) + c1*rand(1,D).*(p_x(i,:) - x(i,:)) + c2*rand(1,D).*(l_x(i,:) - x(i,:)); % velocity update
equation
        x(i,:) = x(i,:) + v(i,:); % position update equation
    end
end

r_h1 = (x - x0).^2;
r_h1 = sqrt(r_h1(:,1) + r_h1(:,2));
r_h(Iter,:) = r_h1;
di = r_h(Iter,:);
g_h(Iter,:) = g; % g history record matrix
w_h(Iter,:) = w; % w history record matrix
```

```
Mean_g_Iter(Iter) = mean(g); % calculation of the mean value of g for the first iteration
Mean_d_Iter(Iter) = mean(r_h(Iter,:));
Mean_td_Iter(Iter) = mean(di);

% screen display of the mean value of g for the first iteration
if Nr == 1
   fprintf('\nIteration = %d    mean g = %3.3f     mean td = %6.1f      mean d = %4.2f\n', Iter, Mean_g_Iter(Iter),
Mean_td_Iter(Iter), Mean_d_Iter(Iter));
end

%%%%%%%%%%%%%%%%%%%%%%%%

% Main algorithm %%%%%%%%%%%%%%

stop = 0;
Iter = Iter +1;

% particles motion display figure
if particle_graphics_flag == true
   eval('pso_plot');
end

% context quality decay calculations
if decay_flag == true
   g_s_update = g_s_update +1;
   % g, p_g decay
   for i = 1:1:N
      if (g(i) >= 0) && (g(i) < 2*theta)
         g(i) = g(i) +1;
      end
      if (p_g(i) >= 0) && (p_g(i) < 2*theta)
         p_g(i) = p_g(i) +1;
      end
   end
   % g_s decay
   for i = 1:1:M
      if (g_s(i) >= 0) && (g_s(i) < 2*theta)
         g_s(i) = g_s(i) +1;
      end
   end
end % decay

% debuging mode calculations
if debug_mode == true
   out_counter = 0;
end


while stop < 1

   % calculate  local best (lbest) & personal best (pbest)
   if pBest_method == 1 % standard model
      [l_g l_x p_g p_x g ] = l_pBest_standard_v1_x(x, source, l_x, l_g, p_x, p_g, g, g_s, N, M, R, Iter);
   elseif pBest_method ==2 % memoryless model
      [l_g l_x p_g p_x g ] = l_pBest_memoryless_v1_x(x, source, l_x, l_g, p_x, p_g, g, g_s, N, M, R, Iter);
   else
      fprintf('Error !\n');
   end
```

```
    g_h(Iter,:) = g; % update g history

    if w_method == 7 % Sigmoid function universal
        f_w = sigmoid_univ(x, g, N);
    end % method 7

    if w_method == 8 % Sigmoid function per partilce
        f_w_p = sigmoid_pp_v2(x, source, g, g_s, N, M, R, Iter, w_min, w_max);
    end % method 8

    I_left = I_max - Iter; % parameter for the speedup of the "random_waypoint" function
    if I_left < 200
        I_left = 200;
    end

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Calculation of v, x for all particles
    % Four modes of operation for each particle:
    % 1) Swarm movement.
    %     1.1) Switch from random to swarm movement.
    %     1.2) Continue swarm movement.
    % 2) Random movement
    %     2.1) Switch from swarm to random movement.
    %     2.2) Continue random movement.
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    for i=1:1:N
        if (g(i) >= theta) && (g_h(Iter-1,i) >= theta) % continue random movement
            p_Iter(i) = p_Iter(i) + 1;
            x(i,:) = [particle(i).x(p_Iter(i)) particle(i).y(p_Iter(i))];

            r_h(Iter,i) = r_calculation_v2(x(i,:), x_h(:,i,:), i, Iter, rMax, g(i), g_h(Iter-1,i));

        elseif (g(i) >= theta) && (g_h(Iter-1,i) < theta) % random movement for the first time (transition from 
swarm to random movement)
            Iter_p_w(i) = 0;
            p_Iter(i) = 1;
            if (x(i,1) > xMax) || (x(i,1) < xMin) || (x(i,2) > xMax) || (x(i,2) < xMin) % particle's position is out of 
terrain limits
                particle(i) = random_waypoint(xMin, xMax, yMin, yMax, vMin, vMax, I_left, 1); % random waypoint 
model
                if debug_mode == true
                    fprintf('Out of terrain!\n');
                end
            else % particle's position is inside the terrain
                particle(i) = random_waypoint(xMin-x(i,1), xMax-x(i,1), yMin-x(i,2), yMax-x(i,2), vMin, vMax, 
I_left, 1); % random waypoint model
            end
            particle(i).x = particle(i).x - particle(i).x(1)*ones(1,I_left+1);
            particle(i).y = particle(i).y - particle(i).y(1)*ones(1,I_left+1);
            x(i,:) = x(i,:) + [particle(i).x(p_Iter(i)+1) particle(i).y(p_Iter(i)+1)]; % calculation of random movement 
from the current position

            r_h(Iter,i) = r_calculation_v2(x(i,:), x_h(:,i,:), i, Iter, rMax, g(i), g_h(Iter-1,i));

        elseif (g(i) < theta) && (g_h(Iter-1,i) >= theta) % swarm movement for the first time (transition from 
random to swarm movement)

            % particle velocity calculation from its two previous positions
            if Iter > 2
                v(i,1) = x_h(Iter-1,i,1) - x_h(Iter-2,i,1);
                v(i,2) = x_h(Iter-1,i,2) - x_h(Iter-2,i,2);
```

```
        end

        % selection between universal and per particle  update of w, f_w
        if universal_w_update == true
            Iter_p_w(i) = Iter;
        else
            Iter_p_w(i) = Iter_p_w(i) + 1;
            f_w = f_w_p(i);
        end

        % swarm movement update equations
        w(i) = inertia(w_method, Iter_p_w(i), w_h(Iter-1,i), w_max, w_min, I_max, f_w); % inertia weight
calculation
        v(i,:) = w(i)*v(i,:) + c1*rand(1,D).*(p_x(i,:) - x(i,:)) + c2*rand(1,D).*(l_x(i,:) - x(i,:)); % velocity update
equation
        x(i,:) = x(i,:) + v(i,:); % position update equation

        % velocity clamping
        if velocity_clamping == true
            v(i,:) = v_clamping(v(i,:),D,vMin,vMax);
        end

        % position clamping
        if position_clamping == true
            [x(i,:), v(i,:)] = p_clamping(x(i,:), v(i,:), xMin, xMax, yMin, yMax);
        end

        r_h(Iter,i) = r_calculation_v2(x(i,:), x_h(:,i,:), i, Iter, rMax, g(i), g_h(Iter-1,i));

    elseif (g(i) < theta) && (g_h(Iter-1,i) < theta) % continue swarm movement

        % selection between universal and per particle  update of w, f_w
        if universal_w_update == true
            Iter_p_w(i) = Iter;
        else
            Iter_p_w(i) = Iter_p_w(i) + 1;
            f_w = f_w_p(i);
        end

        % swarm movement update equations
        w(i) = inertia(w_method, Iter_p_w(i), w_h(Iter-1,i), w_max, w_min, I_max, f_w); % inertia weight
calculation
        v(i,:) = w(i)*v(i,:) + c1*rand(1,D).*(p_x(i,:) - x(i,:)) + c2*rand(1,D).*(l_x(i,:) - x(i,:)); % velocity update
equation
        x(i,:) = x(i,:) + v(i,:); % position update equation

        % velocity clamping
        if velocity_clamping == true
            v(i,:) = v_clamping(v(i,:),D,vMin,vMax);
        end

        % position clamping
        if position_clamping == true
            [x(i,:), v(i,:)] = p_clamping(x(i,:), v(i,:), xMin, xMax, yMin, yMax);
        end

        r_h(Iter,i) = r_calculation_v2(x(i,:), x_h(:,i,:), i, Iter, rMax, g(i), g_h(Iter-1,i));

    else
        fprintf('\nError !\n');
    end
```

```
        % debuging mode calculations
        if debug_mode == true
            if (x(i,1) > xMax) && (x_h(Iter-1,i,1) < xMax) || (x(i,1) < xMin) && (x_h(Iter-1,i,1) > xMin)
                out_counter = out_counter+1;
            end

            if (x(i,2) > yMax) && (x_h(Iter-1,i,2) < yMax) || (x(i,2) < yMin) && (x_h(Iter-1,i,2) > yMin)
                out_counter = out_counter+1;
            end
        end

    end
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    x_h(Iter,:,:) = x; % update x history
    %r_h1 = [x(:,1) - x_h(Iter-1,:,1)' x(:,2) - x_h(Iter-1,:,2)'].^2;
    %r_h1 = sqrt(r_h1(:,1) + r_h1(:,2));
    %r_h1(r_h1(:)>rMax)=rMax;
    %r_h(Iter,:) = r_h1;
    di = di + r_h(Iter,:);
    w_h(Iter,:) = w; % update w history

    Mean_g_Iter(Iter) = mean(g); % calculation of the mean value of g per iteration
    Mean_d_Iter(Iter) = mean(r_h(Iter,:)); % calculation of the mean value of d per iteration
    Mean_td_Iter(Iter) = mean(di); % calculation of the mean value of td per iteration

    % screen display of mean value of g per iteration
    if Nr == 1
        fprintf('Iteration = %d    mean g = %3.3f    mean td = %6.1f    mean d = %4.2f\n', Iter, Mean_g_Iter(Iter),
Mean_td_Iter(Iter), Mean_d_Iter(Iter));
    end

    Iter = Iter + 1;

    % particles motion display figure
    if particle_graphics_flag == true
        eval('pso_plot');
    end

    % context quality decay calculations
    if decay_flag == true
        % g, p_g decay
        for i = 1:1:N
            if (g(i) >= 0) && (g(i) < 2*theta)
                g(i) = g(i) +1;
            end
            if (p_g(i) >= 0) && (p_g(i) < 2*theta)
                p_g(i) = p_g(i) +1;
            end
        end
        % g_s decay
        for i = 1:1:M
            if (g_s(i) >= 0) && (g_s(i) < 2*theta)
                g_s(i) = g_s(i) +1;
            end
        end

        % g_s update calculation
        if g_s_update == update_cycle
            for i = 1:1:M
```

```
            g_s(i) = 0;
        end
        g_s_update = 0;
    end

    g_s_update = g_s_update +1;

  end % decay


  if Iter > I_max
     stop=1;
  end

end % while

%%%%%%%%%%%%%%%%%%%%%%%%

% debuging mode calculations
if debug_mode == true
  fprintf('out_counter = %d\n', out_counter);
end

% addition of mean context quality value per iterarion for t=0
Mean_g_Iter = [200 Mean_g_Iter];
Mean_d_Iter = [0 Mean_d_Iter];
Mean_td_Iter = [0 Mean_td_Iter];

% display of mean context quality value per iterarion diagram
if Nr == 1
  [ipB_model iw_model vc pc] = method_display(universal_w_update, pBest_method, w_method,
velocity_clamping, position_clamping);
  figure(2)
  x_axis = linspace(0,I_max,I_max+1);
  plot(x_axis,Mean_g_Iter);
  xlabel('Iteration (time t in t.u.)');
  ylabel('Mean value of the fitness function g');
  title({['Sensing rate: ', num2str(freq), ' Hz      pBest model: ', ipB_model] ; ['Inertia Weight Method: ',
iws_name{w_method}, iw_model] ;  ['Velocity clamping: ', vc, '      Position clamping: ', pc]});
  figure(3)
  x_axis = linspace(0,I_max,I_max+1);
  plot(x_axis,Mean_td_Iter);
  xlabel('Iteration (time t in t.u.)');
  ylabel('Mean value of the travelled distance td');
  title({['Sensing rate: ', num2str(freq), ' Hz      pBest model: ', ipB_model] ; ['Inertia Weight Method: ',
iws_name{w_method}, iw_model] ;  ['Velocity clamping: ', vc, '      Position clamping: ', pc]});
  figure(4)
  x_axis = linspace(0,I_max,I_max+1);
  plot(x_axis,Mean_d_Iter);
  xlabel('Iteration (time t in t.u.)');
  ylabel('Mean value of the movement distance d');
  title({['Sensing rate: ', num2str(freq), ' Hz      pBest model: ', ipB_model] ; ['Inertia Weight Method: ',
iws_name{w_method}, iw_model] ;  ['Velocity clamping: ', vc, '      Position clamping: ', pc]});
end

% mean context quality converging value
mg_t0 = Mean_g_Iter(I_max+1);

% mean context quality value of all iterations
mg = mean(Mean_g_Iter);
md = mean(Mean_d_Iter);
```

```
mtd =  mean(Mean_td_Iter);

% standard deviation of mean context quality value of all iterations
mgstd = std(Mean_g_Iter);

end % function
```

l_pBest_standard_v1_x.m

```
% l_pBest_standard_v1_x.m
% function for the calculation of the local best (lBest) and
% personal best (pBest) positions of the PSO algorithm
% standard pBest calculation model
%
% version 1.2
% March 2012

function [l_g l_x p_g p_x g ] = l_pBest_standard_v1_x(x, source, l_x, l_g, p_x, p_g, g, g_s, N, M, R, Iter)


if Iter == 1 % calculation of lbest, pbest for the first iteration
   for i = 1:1:N
      l_n = 1;
      l_g_sum = g(i);
      MIN_p = Inf;
      for j = 1:1:N
         if (j ~= i) && (norm(x(i,:) - x(j,:)) <= R)
            l_n = l_n + 1;
            l_g_sum = l_g_sum + g(j);
            if g(j) < MIN_p
               MIN_p = g(j);
               p_best = j;
            end
         end
      end
      MIN_s = Inf;
      p_best_s = 0;
      for s = 1:1:M
         if  (norm(x(i,:) - [source(s).x(Iter) source(s).y(Iter)]) <= R)
            l_n = l_n + 1;
            l_g_sum = l_g_sum + g_s(s);
            if g_s(s) < MIN_s
               MIN_s = g_s(s);
               p_best_s = s;
            end
         end
      end

      % lbest calculation
      l_g(i) = l_g_sum/l_n;

      % pbest calculation
      if g(i) <= MIN_p
         MIN = g(i);
         p_best = i;
      else
         MIN = MIN_p;
      end

      if MIN_s < MIN
```

```
        p_x(i,1) = source(p_best_s).x(Iter); p_x(i,2) = source(p_best_s).y(Iter);
        p_g(i) = MIN_s;
        g(i) = MIN_s;
    else
        p_x(i,:) =  x(p_best,:);
        p_g(i) = MIN;
        g(i) = MIN;
    end

  end % for
else % calculation of lbest, pbest for the next iterations
  for i = 1:1:N
    l_n = 1;
    l_g_sum = g(i);
    loc_g = zeros(1,N);
    MIN_p = Inf;
    for j = 1:1:N
      if  (j ~= i) && (norm(x(i,:) - x(j,:)) <= R)
        l_n = l_n + 1;
        l_g_sum = l_g_sum + g(j);
        if g(j) < MIN_p
          MIN_p = g(j);
          p_best = j;
        end
      end
    end
    MIN_s = Inf;
    p_best_s = 0;
    for s = 1:1:M
      if  norm(x(i,:) - [source(s).x(Iter) source(s).y(Iter)]) <= R
        l_n = l_n + 1;
        l_g_sum = l_g_sum + g_s(s);
        if g_s(s) < MIN_s
          MIN_s = g_s(s);
          p_best_s = s;
        end
      end
    end

    % lbest calculation
    loc_g(i) = l_g_sum/l_n;

    if loc_g(i) < l_g(i)
      l_g(i) = loc_g(i);
      l_x(i,:) = x(i,:);
    end

    g(i) = min([g(i), MIN_p, MIN_s]);

    % pbest calculation
    if p_g(i) > min(MIN_p, MIN_s)
      if (MIN_s < MIN_p)
        p_x(i,1) = source(p_best_s).x(Iter); p_x(i,2) = source(p_best_s).y(Iter);
        p_g(i) = MIN_s;
      else
        p_x(i,:) =  x(p_best,:);
        p_g(i) = MIN_p;
      end
    end

  end % for
```

```
end % if


end % function
```

l_pBest_memoryless_v1_x.m

```
% l_pBest_memoryless_v1_x.m
% function for the calculation of the local best (lBest) and
% personal best (pBest) positions of the PSO algorithm
% memoryless pBest calculation model
%
% version 1.2
% March 2012

function [l_g l_x p_g p_x g ] = l_pBest_memoryless_v1_x(x, source, l_x, l_g, p_x, p_g, g, g_s, N, M, R, Iter)


if Iter == 1 % calculation of lbest, pbest for the first iteration
   for i = 1:1:N
      l_n = 1;
      l_g_sum = g(i);
      MIN_p = Inf;
      for j = 1:1:N
         if (j ~= i) && (norm(x(i,:) - x(j,:)) <= R)
            l_n = l_n + 1;
            l_g_sum = l_g_sum + g(j);
            if g(j) < MIN_p
               MIN_p = g(j);
               p_best = j;
            end
         end
      end
      MIN_s = Inf;
      p_best_s = 0;
      for s = 1:1:M
         if  (norm(x(i,:) - [source(s).x(Iter) source(s).y(Iter)]) <= R)
            l_n = l_n + 1;
            l_g_sum = l_g_sum + g_s(s);
            if g_s(s) < MIN_s
               MIN_s = g_s(s);
               p_best_s = s;
            end
         end
      end

      % lbest calculation
      l_g(i) = l_g_sum/l_n;

      % pbest calculation
      if g(i) <= MIN_p
         MIN = g(i);
         p_best = i;
      else
         MIN = MIN_p;
      end

      if MIN_s < MIN
         p_x(i,1) = source(p_best_s).x(Iter); p_x(i,2) = source(p_best_s).y(Iter);
```

```
        p_g(i) = MIN_s;
        g(i) = MIN_s;
    else
        p_x(i,:) =  x(p_best,:);
        p_g(i) = MIN;
        g(i) = MIN;
    end

  end % for
else % calculation of lbest, pbest for the next iterations
  for i = 1:1:N
    l_n = 1;
    l_g_sum = g(i);
    loc_g = zeros(1,N);
    MIN_p = Inf;
    for j = 1:1:N
      if  (j ~= i) && (norm(x(i,:) - x(j,:)) <= R)
        l_n = l_n + 1;
        l_g_sum = l_g_sum + g(j);
        if g(j) < MIN_p
          MIN_p = g(j);
          p_best = j;
        end
      end
    end
    MIN_s = Inf;
    p_best_s = 0;
    for s = 1:1:M
      if  norm(x(i,:) - [source(s).x(Iter) source(s).y(Iter)]) <= R
        l_n = l_n + 1;
        l_g_sum = l_g_sum + g_s(s);
        if g_s(s) < MIN_s
          MIN_s = g_s(s);
          p_best_s = s;
        end
      end
    end

    % lbest calculation
    loc_g(i) = l_g_sum/l_n;

    if loc_g(i) < l_g(i)
      l_g(i) = loc_g(i);
      l_x(i,:) = x(i,:);
    end

    % pbest calculation
    if g(i) <= MIN_p
      MIN = g(i);
      p_best = i;
    else
      MIN = MIN_p;
    end

    if MIN_s < MIN
      p_x(i,1) = source(p_best_s).x(Iter); p_x(i,2) = source(p_best_s).y(Iter);
      p_g(i) = MIN_s;
      g(i) = MIN_s;
    else
      p_x(i,:) =  x(p_best,:);
      p_g(i) = MIN;
```

```
        g(i) = MIN;
      end

    end % for

  end % if


end % function
```

sigmoid_pp_v2.m

```
% sigmoid_pp_v2.m
% calculation of the sigmoid per particle w update method
%
% version 2.0
% March 2012

function f_w_p = sigmoid_pp_v2(x, source, g, g_s, N, M, R, Iter, w_min, w_max)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Sigmoid function per partilce
% Calculation of the following parameters:
% N_i(i): the number of neighbours of each particle
% d(i): mean distance of each particle from its neighbours
% d_max(i): maximum distance between each particle and its neighbours
% d_min(i): minimum distance between each particle and its neighbours
% d_g(i): d(i) of the locally best particle (in the neighborhood)
% f_w_p(i): evolutionary factor of each particle
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


% initialization of parameters
f_w_p = zeros(1,N);
d = zeros(1,N);
d_max = zeros(1,N);
d_min = zeros(1,N);
d_g = zeros(1,N);

x_s = zeros(M, 2);

for s = 1:1:M % search for sources in the neighborhood
  x_s(s,:) = [source(s).x(Iter) source(s).y(Iter)];
end

g_total = [g g_s];
x_total = [x ; x_s];
d_matrix = squareform(pdist(x_total));

d_ij = []; % array of distances between each particle and its neighbours
x_ij = []; % array of particle positions in the neighborhood
g_i = []; % array of g's for all the particles in the neighborhood

for i = 1:1:N

  temp = d_matrix(i,:);
  ind = find(0 < temp & temp <= R);
  if (numel(ind) > 1) % the neighborhood has at least 2 members
    d_ij = temp(ind);
    d(i) = mean(d_ij);
```

```
        d_max(i) = max(d_ij);
        d_min(i) = min(d_ij);
        g_i = g_total(ind);
        x_ij = x_total(ind,:);

        % calculation of d_g(i)
        if g(i) > min(g_i) % if best particle different from itself
           x_ij = [x(i, :) ; x_ij];
           g_i = [g(i), g_i];
           [g_i_best, i_best] = min(g_i); % find best particle in the neighborhood
           d_ij = squareform(pdist(x_ij));
           d_g(i) = mean(d_ij(i_best,:));
        else % best particle is itself
           d_g(i) = d(i);
        end
        f_w_p(i) = (d_g(i) - d_min(i))/(d_max(i) - d_min(i)); % calculation of evolutionary factor
    else
        f_w_p(i) = randnum(w_min, w_max); % calculation of evolutionary factor if the neighborhood has less
than 2 members
        %f_w_p(i) = w_max;
    end % calculation of d_g(i)

end % for N


end % function
```

sigmoid_univ.m

```
% sigmoid_univ.m
% calculation of the sigmoid universal w update method
%
% version 1.0
% March 2012

function f_w = sigmoid_univ(x, g, N)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Sigmoid function universal
% Calculation of the following parameters:
% d(i): mean distance of each particle from all the others (Euclidian metric)
% d_max: maximum distance between all particles
% d_min: minimum distance between all particles
% d_g: mean distance of the globally best particle
% f_w: evolutionary factor
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

[g_best, i_best] = min(g); % find globally best particle


% initialization of parameters
d = zeros(1,N);
d_max = 0;
d_min = 0;

for i = 1:1:N % calculation of d(i)
   for j = 1:1:N
      if j ~= i
         d(i) = d(i) + norm(x(i,:) - x(j,:));
```

```
      end
   end

   d(i) = d(i)/(N-1);
end

d_max = max(d);
d_min = min(d);
d_g = d(i_best);

f_w = (d_g - d_min)/(d_max - d_min); % calculation of evolution factor

end % function
```

pso_plot.m

```
% pso_plot.m
% plot of the movement of swarm particles and sources
% creates a new figure

clf(1)

set(gcf,'Doublebuffer','on');
axis([xMin xMax yMin yMax]);

hold on

if debug_mode == true
   for i = 1:N
      plot(x(i,1), x(i,2), '*', 'color', [0.3 0.3 1]);
      kk = num2str(i);
      text(x(i,1),x(i,2), kk, 'FontSize', 14);
   end
else
   for i = 1:N
      plot(x(i,1), x(i,2), '*', 'color', [0.3 0.3 1]);
   end
end

for s = 1:M
   plot(source(s).x(Iter), source(s).y(Iter),'h','color',[1 0 0]);
end


title('Particle Dynamics','color',[0 0 0],'fontweight','bold');
xlabel('x');
ylabel('y');

hold off

drawnow

% end
```

OSPSO_v3_0_n2.m

```
% OSPSO_v3_0_n2.m
% function for the calculation of the OSPSO algorithm
%
% version 3.0.2
% March 2012

function [mg_t0 mg mgstd mtd md] = OSPSO_v3_0_n2(N, M, D, I_max, xMin, xMax, yMin, yMax, R, vMin,
vMax, theta, freq, op_st_interval, opt_cutoff, pBest_method, w_method, pso_flags, Nr, iws_name)
% Optimal Stopping PSO


decay_flag = pso_flags(1);
velocity_clamping = pso_flags(2);
position_clamping = pso_flags(3);
universal_w_update = pso_flags(4);
particle_graphics_flag = pso_flags(5);
debug_mode = pso_flags(6);


% Initializations %%%%%%%%%%%%%%%%

Iter = 1; % iteration
p_Iter = ones(1,N); % particle random movement iteration

c1 = 2; % definition of acceleration coefficients
c2 = c1;

w_max = 0.9; % definition of inertia weight bounds
w_min = 0.4;

rMax = sqrt((2*xMax)^2 + (2*yMax)^2);

f_w = 0; % w update parameter (universal)
f_w_p = zeros(1,N); % w update parameter (per particle)

w = w_max*ones(1,N);
w_h = zeros(I_max+1, 1, N); % w history record
Iter_p_w = zeros(1,N); % iteration per particle (for w update)

Mean_g_Iter = Inf*ones(1, N); % mean g for the swarm per iteration
Mean_d_Iter = Inf*ones(1, N); % mean movement distance d for the swarm per iteration
Mean_td_Iter = Inf*ones(1, N); % mean travelled distance td for the swarm per iteration

x = zeros(N, D); % initialization of array x
x_h = zeros(I_max+1, N, D); % x history record
r_h1 = zeros(N, D); % temp matrix for the calculation of the r history record
r_h = zeros(I_max+1, N); % r history record
v = zeros(N, D); % initialization of array v

% random waypoint model
particle = random_waypoint(xMin, xMax, yMin, yMax, vMin, vMax, I_max, N);
source = random_waypoint(xMin, xMax, yMin, yMax, vMin, vMax, I_max, M);

g = theta+theta*ones(1,N); % initialization of values of g for particles
di = zeros(1,N); % initialization of values of d for particles
g_h = zeros(I_max+1,1,N); % g history record
g_s = zeros(1,M); % initialization of values of g for sources
g_Ni_MIN = Inf*ones(1,N); % initialization of the array having the smallest g values in each particle
neighborhood
g_Ni_MIN_h = Inf*ones(I_max+1,1,N);
```

```
g_Ni_MIN_h_cutoff = Inf*ones(N, opt_cutoff);
g_Ni_MIN_cutoff = Inf*ones(1,N);

mstatus = 4*ones(1, N); % set movement status for each particle: 1 -> swarm movement, 2 -> random
movement, 3-> monitor (optimal stopping), 4-> wait
mstatus_h = zeros(I_max+1,1,N); % mstatus history record
wait_flag(1, N) = false; % flag to wait on not for better context
wait_counter(1, N) = 0; % initialization of wait counter for each particle

% initialization of positions of particles
for i=1:1:N
   x(i,:) = [particle(i).x(p_Iter(i)) particle(i).y(p_Iter(i))];
end

x0 = x;

% initialization of velocities of particles
for i=1:1:N
   for j=1:1:D
      v(i,j) = randnum(vMin, vMax);
   end
end

% initialize local best (lbest)
l_x = x;
l_g = g;

% initialize personal best (pbest)
p_x =  x;
p_g = g;

% particles motion display figure
if particle_graphics_flag == true
   figure(1);
   eval('pso_plot');
end

%%%%%%%%%%%%%%%%%%%%%%%%

% First Iteration %%%%%%%%%%%%%%%%

% context quality decay calculations
if decay_flag == true
   g_s_update = 0;
   update_cycle = round(1/freq);
end

% local best (lbest), personal best (pbest) and movement status calculation
if pBest_method == 1 % standard model
   [l_g l_x p_g p_x g g_Ni_MIN, g_Ni_MIN_h_cutoff, g_Ni_MIN_cutoff, mstatus, wait_flag, wait_counter] =
l_pBest_standard_mstatus_v3_0a(x, source, l_x, l_g, p_x, p_g, g, g_s, g_Ni_MIN, N, M, R, Iter, mstatus, ...

mstatus_h, wait_flag, wait_counter, op_st_interval, opt_cutoff, g_Ni_MIN_h_cutoff, g_Ni_MIN_cutoff, theta);
elseif pBest_method == 2 % memoryless model
   [l_g l_x p_g p_x g g_Ni_MIN, g_Ni_MIN_h_cutoff, g_Ni_MIN_cutoff, mstatus, wait_flag, wait_counter] =
l_pBest_memoryless_mstatus_v3_0a(x, source, l_x, l_g, p_x, p_g, g, g_s, g_Ni_MIN, N, M, R, Iter, mstatus, ...

mstatus_h, wait_flag, wait_counter, op_st_interval, opt_cutoff, g_Ni_MIN_h_cutoff, g_Ni_MIN_cutoff, theta);
else
   fprintf('Error !\n');
```

```
end


for i = 1:1:N
   if mstatus(i) == 1 % swarm movement
      v(i,:) = w(i)*v(i,:) + c1*rand(1,D).*(p_x(i,:) - x(i,:)) + c2*rand(1,D).*(l_x(i,:) - x(i,:)); % velocity update
equation
      x(i,:) = x(i,:) + v(i,:); % position update equation
   elseif mstatus(i) == 4 % wait
      % do nothing
   else
      fprintf('\nError in status!\n');
   end % if
end % for

x_h(Iter,:,:) = x; % x history record matrix
r_h1 = (x - x0).^2;
r_h1 = sqrt(r_h1(:,1) + r_h1(:,2));
r_h(Iter,:) = r_h1;
di = r_h(Iter,:);
g_h(Iter,:) = g; % g history record matrix
w_h(Iter,:) = w; % w history record matrix
g_Ni_MIN_h(Iter,:) = g_Ni_MIN;
mstatus_h(Iter,:) = mstatus;

Mean_g_Iter(Iter) = mean(g); % calculation of the mean value of g for the first iteration
Mean_d_Iter(Iter) = mean(r_h(Iter,:));
Mean_td_Iter(Iter) = mean(di);

% screen display of the mean value of g for the first iteration
if Nr == 1
   fprintf('\nIteration = %d    mean g = %3.3f     mean td = %6.1f      mean d = %4.2f\n', Iter, Mean_g_Iter(Iter),
Mean_td_Iter(Iter), Mean_d_Iter(Iter));
end

%%%%%%%%%%%%%%%%%%%%%%%%

% Main algorithm %%%%%%%%%%%%%%

stop = 0;
Iter = Iter + 1;

% particles motion display figure
if particle_graphics_flag == true
   eval('pso_plot');
end

% context quality decay calculations
if decay_flag == true
   g_s_update = g_s_update +1;
   % g, p_g decay
   for i = 1:1:N
      if (g(i) >= 0) && (g(i) < 2*theta)
         g(i) = g(i) +1;
      end
      if (p_g(i) >= 0) && (p_g(i) < 2*theta)
         p_g(i) = p_g(i) +1;
      end
   end
   % g_s decay
```

```
    for i = 1:1:M
        if (g_s(i) >= 0) && (g_s(i) < 2*theta)
            g_s(i) = g_s(i) +1;
        end
    end
end % decay

% debuging mode calculations
if debug_mode == true
    out_counter = 0;
end

while stop < 1

    % local best (lbest), personal best (pbest) and movement status calculation
if pBest_method == 1 % standard model
    [l_g l_x p_g p_x g g_Ni_MIN, g_Ni_MIN_h_cutoff, g_Ni_MIN_cutoff, mstatus, wait_flag, wait_counter] =
l_pBest_standard_mstatus_v3_0a(x, source, l_x, l_g, p_x, p_g, g, g_s, g_Ni_MIN, N, M, R, Iter, mstatus, ...

mstatus_h, wait_flag, wait_counter, op_st_interval, opt_cutoff, g_Ni_MIN_h_cutoff, g_Ni_MIN_cutoff, theta);
elseif pBest_method == 2 % memoryless model
    [l_g l_x p_g p_x g g_Ni_MIN, g_Ni_MIN_h_cutoff, g_Ni_MIN_cutoff, mstatus, wait_flag, wait_counter] =
l_pBest_memoryless_mstatus_v3_0a(x, source, l_x, l_g, p_x, p_g, g, g_s, g_Ni_MIN, N, M, R, Iter, mstatus, ...

mstatus_h, wait_flag, wait_counter, op_st_interval, opt_cutoff, g_Ni_MIN_h_cutoff, g_Ni_MIN_cutoff, theta);
else
    fprintf('Error !\n');
end


    g_h(Iter,:) = g; % update g history
    g_Ni_MIN_h(Iter,:) = g_Ni_MIN;
    mstatus_h(Iter,:) = mstatus;

    % Sigmoid function universal (f_w calculation)
    if w_method == 7
        f_w = sigmoid_univ(x, g, N);
    end % method 7

    % Sigmoid function per partilce (f_w_p calculation)
    if w_method == 8
        f_w_p = sigmoid_pp_v2(x, source, g, g_s, N, M, R, Iter, w_min, w_max);
    end % method 8

    I_left = I_max - Iter; % parameter for the speedup of the "random_waypoint" function
    if I_left < 150
        I_left = 150;
    end

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Calculation of v, x for all particles
    % Four modes of operation for each particle:
    % 1) Swarm movement.
    %     1.1) Switch from random to swarm movement.
    %     1.2) Continue swarm movement.
    % 2) Random movement
    %     2.1) Switch from swarch to random movement.
    %     2.2) Continue random movement.
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    for i=1:1:N
        if mstatus(i) == 1 % swarm movement
```

```
        % set w, f_w update models (universal or per particle)
        if universal_w_update == true
            Iter_p_w(i) = Iter;
        else
            Iter_p_w(i) = Iter_p_w(i) + 1;
            f_w = f_w_p(i);
        end

        if (g(i) < theta) && (g_h(Iter-1,i) >= theta) % swarm movement for the first time (transition from random to swarm movement)
            if Iter > 2     % velocity calculation calculation from previous positions
                v(i,1) = x_h(Iter-1,i,1) - x_h(Iter-2,i,1);
                v(i,2) = x_h(Iter-1,i,2) - x_h(Iter-2,i,2);
            end
        end

        w(i) = inertia(w_method, Iter_p_w(i), w_h(Iter-1,i), w_max, w_min, I_max, f_w); % inertia weight calculation
        v(i,:) = w(i)*v(i,:) + c1*rand(1,D).*(p_x(i,:) - x(i,:)) + c2*rand(1,D).*(l_x(i,:) - x(i,:)); % velocity update equation
        x(i,:) = x(i,:) + v(i,:); % position update equation

        % velocity clamping
        if velocity_clamping == true
            v(i,:) = v_clamping(v(i,:),D,vMin,vMax);
        end

        % position clamping
        if position_clamping == true
            [x(i,:), v(i,:)] = p_clamping(x(i,:), v(i,:), xMin, xMax, yMin, yMax);
        end

        r_h(Iter,i) = r_calculation(x(i,:), x_h(:,i,:), i, Iter, rMax, mstatus(i), g(i), g_h(Iter-1,i));

    elseif mstatus(i) == 2  % random movement

        if (g(i) >= theta) && (Iter > 2) && (g_h(Iter-1,i) < theta) % random movement for the first time (transition from swarm to random movement)
            Iter_p_w(i) = 0;
            p_Iter(i) = 1;
            if (x(i,1) > xMax) || (x(i,1) < xMin) || (x(i,2) > xMax) || (x(i,2) < xMin) % particle's position is out of terrain limits
                particle(i) = random_waypoint(xMin, xMax, yMin, yMax, vMin, vMax, I_left, 1); % random waypoint model
                if debug_mode == true
                    fprintf('Out of terrain!\n');
                end
            else % particle's position is inside the terrain
                particle(i) = random_waypoint(xMin-x(i,1), xMax-x(i,1), yMin-x(i,2), yMax-x(i,2), vMin, vMax, I_left, 1); % random waypoint model
            end
            particle(i).x = particle(i).x - particle(i).x(1)*ones(1,I_left+1);
            particle(i).y = particle(i).y - particle(i).y(1)*ones(1,I_left+1);
            x(i,:) = x(i,:) + [particle(i).x(p_Iter(i)+1) particle(i).y(p_Iter(i)+1)]; % calculation of random movement from the current position
            wait_counter(i) = 0;
        else % continue random movement
            p_Iter(i) = p_Iter(i) + 1;
            x(i,:) = [particle(i).x(p_Iter(i)) particle(i).y(p_Iter(i))];
            wait_counter(i) = 0;
```

```
            end

            r_h(Iter,i) = r_calculation(x(i,:), x_h(:,i,:), i, Iter, rMax, mstatus(i));

        elseif mstatus(i) == 3 % monitor

            % no nothing

        elseif mstatus(i) == 4 % wait

            % no nothing

        else

            fprintf('\nError in status!\n');

        end % if status

        if debug_mode == true
            if (x(i,1) > xMax) && (x_h(Iter-1,i,1) < xMax) || (x(i,1) < xMin) && (x_h(Iter-1,i,1) > xMin)
                out_counter = out_counter+1;
            end

            if (x(i,2) > yMax) && (x_h(Iter-1,i,2) < yMax) || (x(i,2) < yMin) && (x_h(Iter-1,i,2) > yMin)
                out_counter = out_counter+1;
            end
        end

    end % for
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    x_h(Iter,:,:) = x; % update x history
    di = di + r_h(Iter,:);
    w_h(Iter,:) = w; % update w history

    Mean_g_Iter(Iter) = mean(g); % calculation of the mean value of g per iteration
    Mean_d_Iter(Iter) = mean(r_h(Iter,:)); % calculation of the mean value of d per iteration
    Mean_td_Iter(Iter) = mean(di); % calculation of the mean value of d per iteration

    % screen display of mean value of g per iteration
    if Nr == 1
        fprintf('\nIteration = %d    mean g = %3.3f     mean td = %6.1f      mean d = %4.2f\n', Iter,
Mean_g_Iter(Iter), Mean_td_Iter(Iter), Mean_d_Iter(Iter));
    end

    Iter = Iter + 1;

    % particles motion display figure
    if particle_graphics_flag == true
        eval('pso_plot');
    end

    % context quality decay calculations
    if decay_flag == true
        % g, p_g decay
        for i = 1:1:N
            if (g(i) >= 0) && (g(i) < 2*theta)
                g(i) = g(i) +1;
            end
            if (p_g(i) >= 0) && (p_g(i) < 2*theta)
                p_g(i) = p_g(i) +1;
```

```
        end
      end
      % g_s decay
      for i = 1:1:M
        if (g_s(i) >= 0) && (g_s(i) < 2*theta)
          g_s(i) = g_s(i) +1;
        end
      end

      if g_s_update == update_cycle
        % g_s update
        for i = 1:1:M
            g_s(i) = 0;
        end
        g_s_update = 0;
      end

      g_s_update = g_s_update +1;

    end % decay


    if Iter > I_max
      stop=1;
    end

end % while

%%%%%%%%%%%%%%%%%%%%%%%%

% debuging mode calculations
if debug_mode == true
    fprintf('out_counter = %d\n', out_counter);
end

% addition of mean context quality value per iterarion for t=0
Mean_g_Iter = [200 Mean_g_Iter];
Mean_d_Iter = [0 Mean_d_Iter];
Mean_td_Iter = [0 Mean_td_Iter];

% display of mean context quality value per iterarion diagram
if Nr == 1
    [ipB_model iw_model vc pc] = method_display(universal_w_update, pBest_method, w_method,
velocity_clamping, position_clamping);
    figure(2)
    x_axis = linspace(0,I_max,I_max+1);
    plot(x_axis,Mean_g_Iter);
    xlabel('Iteration (time t in t.u.)');
    ylabel('Mean value of the fitness function g');
    title({['Sensing rate: ', num2str(freq), ' Hz     Observation interval = ', num2str(op_st_interval)] ; ['pBest
model: ', ipB_model, '      Inertia Weight Method: ', iws_name{w_method}, iw_model] ; ...
        ['Velocity clamping: ', vc, '     Position clamping: ', pc]});
    figure(3)
    x_axis = linspace(0,I_max,I_max+1);
    plot(x_axis,Mean_td_Iter);
    xlabel('Iteration (time t in t.u.)');
    ylabel('Mean value of the travelled distance td');
    title({['Sensing rate: ', num2str(freq), ' Hz     Observation interval = ', num2str(op_st_interval)] ; ['pBest
model: ', ipB_model, '      Inertia Weight Method: ', iws_name{w_method}, iw_model] ; ...
        ['Velocity clamping: ', vc, '     Position clamping: ', pc]});
    figure(4)
```

```
    x_axis = linspace(0,I_max,I_max+1);
    plot(x_axis,Mean_d_Iter);
    xlabel('Iteration (time t in t.u.)');
    ylabel('Mean value of the movement distance d');
    title({['Sensing rate: ', num2str(freq), ' Hz     Observation interval = ', num2str(op_st_interval)] ; ['pBest
model: ', ipB_model, '      Inertia Weight Method: ', iws_name{w_method}, iw_model] ; ...
          ['Velocity clamping: ', vc, '     Position clamping: ', pc]});
end

% mean context quality converging value
mg_t0 = Mean_g_Iter(I_max+1);

% mean context quality value of all iterations
mg = mean(Mean_g_Iter);
md = mean(Mean_d_Iter);
mtd =  mean(Mean_td_Iter);

% standard deviation of mean context quality value of all iterations
mgstd = std(Mean_g_Iter);

end % function
```

l_pBest_standard_mstatus_v3_0a.m

```
% l_pBest_standard_mstatus_v3_0a.m
% function for the calculation of the local best (lBest) and
% personal best (pBest) positions of the OSPSO algorithm
% standard pBest calculation model
% it also calculates the status of movement for each partcle:
% 1 -> swarm movement, 2 -> random movement, 3-> monitor (optimal stopping), 4-> wait
%
% version 3.0.1
% March 2012

 function [l_g l_x p_g p_x g g_Ni_MIN, g_Ni_MIN_h_cutoff, g_Ni_MIN_cutoff, mstatus, wait_flag,
wait_counter] = l_pBest_standard_mstatus_v3_0a(x, source, l_x, l_g, p_x, p_g, g, g_s, g_Ni_MIN, N, M, R, Iter,
mstatus, ...

mstatus_h, wait_flag, wait_counter,  op_st_interval, opt_cutoff, ...

g_Ni_MIN_h_cutoff, g_Ni_MIN_cutoff, theta)


if Iter == 1 % calculation of lbest, pbest for the first iteration
   for i = 1:1:N
      l_n = 1;
      l_g_sum = g(i);
      MIN_p = Inf;
      for j = 1:1:N
         if (j ~= i) && (norm(x(i,:) - x(j,:)) <= R)
            l_n = l_n + 1;
            l_g_sum = l_g_sum + g(j);
            if g(j) < MIN_p
               MIN_p = g(j);
               p_best = j;
            end
         end
      end
      MIN_s = Inf;
      p_best_s = 0;
```

```
    for s = 1:1:M
        if  (norm(x(i,:) - [source(s).x(Iter) source(s).y(Iter)]) <= R)
            l_n = l_n + 1;
            l_g_sum = l_g_sum + g_s(s);
            if g_s(s) < MIN_s
                MIN_s = g_s(s);
                p_best_s = s;
            end
        end
    end


    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % particle movement status calculation
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    g_Ni_MIN(i) = min(MIN_p, MIN_s);

    if g_Ni_MIN(i) == 0 % swarm movement
        wait_flag(i) = false;
        mstatus(i) = 1; % set swarm movement status
        wait_counter(i) = 0; % reset wait counter
        g_Ni_MIN_h_cutoff(i,:) = Inf*ones(1, opt_cutoff);
        g_Ni_MIN_cutoff(i) = Inf;
    else
        wait_flag(i) = true;
        mstatus(i) = 4; % wait
    end
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


    % lbest calculation
    l_g(i) = l_g_sum/l_n;

    % pbest calculation
    if g(i) <= MIN_p
        MIN = g(i);
        p_best = i;
    else
        MIN = MIN_p;
    end

    if MIN_s < MIN
        p_x(i,1) = source(p_best_s).x(Iter); p_x(i,2) = source(p_best_s).y(Iter);
        p_g(i) = MIN_s;
        g(i) = MIN_s;
    else
        p_x(i,:) =  x(p_best,:);
        p_g(i) = MIN;
        g(i) = MIN;
    end

  end % for
else % calculation of lbest, pbest for the next iterations
  for i = 1:1:N
    l_n = 1;
    l_g_sum = g(i);
    loc_g = zeros(1,N);
    MIN_p = Inf;
    for j = 1:1:N
        if  (j ~= i) && (norm(x(i,:) - x(j,:)) <= R)
            l_n = l_n + 1;
```

```
        l_g_sum = l_g_sum + g(j);
        if g(j) < MIN_p
            MIN_p = g(j);
            p_best = j;
        end
    end
end
MIN_s = Inf;
p_best_s = 0;
for s = 1:1:M
    if  norm(x(i,:) - [source(s).x(Iter) source(s).y(Iter)]) <= R
        l_n = l_n + 1;
        l_g_sum = l_g_sum + g_s(s);
        if g_s(s) < MIN_s
            MIN_s = g_s(s);
            p_best_s = s;
        end
    end
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% particle movement status calculation
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
g_Ni_MIN(i) = min(MIN_p, MIN_s);

if mstatus(i) == 3 % monitor
    if (wait_counter(i) < opt_cutoff)
        wait_flag(i) = true;
        mstatus(i) = 3; % monitor
        g_Ni_MIN_h_cutoff(i, wait_counter(i)+1) = g_Ni_MIN(i)/(2*theta);
        wait_counter(i) = wait_counter(i) + 1;
    elseif  (wait_counter(i) >= opt_cutoff && wait_counter(i) < op_st_interval) % testing
        g_Ni_MIN_cutoff(i) = min(g_Ni_MIN_h_cutoff(i,:));
        if (g_Ni_MIN(i)/(2*theta) < g_Ni_MIN_h_cutoff(i))
            wait_flag(i) = false;
            mstatus(i) = 1; % set swarm movement status
            wait_counter(i) = 0; % reset wait counter
            g_Ni_MIN_h_cutoff(i,:) = Inf*ones(1, opt_cutoff);
            g_Ni_MIN_cutoff(i) = Inf;
        else % continue testing
            wait_flag(i) = true;
            mstatus(i) = 3; % continue testing
            wait_counter(i) = wait_counter(i) + 1;
        end
    elseif (wait_counter(i) == op_st_interval)
            wait_flag(i) = false;
            mstatus(i) = 1; % set swarm movement status
            wait_counter(i) = 0; % reset wait counter
            g_Ni_MIN_h_cutoff(i,:) = Inf*ones(1, opt_cutoff);
            g_Ni_MIN_cutoff(i) = Inf;
    elseif (wait_counter(i) > op_st_interval)
        fprintf('Error in status 1\n');
    else
        fprintf('Error in status 2\n');
    end
end

if (g_Ni_MIN(i) < g(i) && mstatus(i) ~= 3)
    if g_Ni_MIN(i) == 0 % swarm movement
        wait_flag(i) = false;
```

```
            mstatus(i) = 1; % set swarm movement status
            wait_counter(i) = 0; % reset wait counter
            g_Ni_MIN_h_cutoff(i,:) = Inf*ones(1, opt_cutoff);
            g_Ni_MIN_cutoff(i) = Inf;
        elseif (0 < g_Ni_MIN(i) && g_Ni_MIN(i) < theta && mstatus_h(Iter-1,i) ~= 3) % monitor
            wait_flag(i) = true;
            mstatus(i) = 3; % monitor
            g_Ni_MIN_h_cutoff(i, wait_counter(i)+1) = g_Ni_MIN(i)/(2*theta);
            wait_counter(i) = wait_counter(i) + 1;
        elseif (g_Ni_MIN(i) >= theta) % wait
            wait_flag(i) = true;
            mstatus(i) = 4; % wait
        else
            % case: (0 < g_Ni_MIN(i)< theta && mstatus_h(Iter-1,i) == 3)
        end
    else
        % no change in movement
    end
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


    % lbest calculation
    loc_g(i) = l_g_sum/l_n;

    if loc_g(i) < l_g(i)
        l_g(i) = loc_g(i);
        l_x(i,:) = x(i,:);
    end

    g(i) = min([g(i), MIN_p, MIN_s]);

    % pbest calculation
    if p_g(i) > min(MIN_p, MIN_s)
        if (MIN_s < MIN_p)
            p_x(i,1) = source(p_best_s).x(Iter); p_x(i,2) = source(p_best_s).y(Iter);
            p_g(i) = MIN_s;
        else
            p_x(i,:) =  x(p_best,:);
            p_g(i) = MIN_p;
        end
    end

  end % for

end % if


end % function
```

randnum.m

```
function r = randnum (a,b)
% Random real number between a and b

   r= a + rand*(b - a);

end
```

inertia.m

```
% inertia.m
% function for the calculation of w for the various w update methods
%
% version 1.0
% March 2012

function w = inertia(w_method, Iter, w_last, w_max, w_min, I_max, f_w)

if  w_method == 1 % linear decreasing inertia weight
   w = w_max - (w_max - w_min)*Iter/I_max;
elseif w_method == 2 % non-linear decreasing inertia weight
   w = (w_max - w_min)*(I_max - Iter)/(I_max + w_min);
elseif w_method == 3 % non-linear decreasing inertia weight 2
   w = (w_last - w_min)*(I_max - Iter)/(I_max + w_min);
elseif w_method == 4 % random inertia weight
   w = 0.5 + randnum(w_min, w_max)/2;
elseif w_method == 5 % chaotic inertia weight
   z = randnum(0,1);
   z = 4*z*(1-z);
   w = (w_max - w_min)*(I_max - Iter)/I_max + w_min*z;
elseif w_method == 6 % chaotic random inertia weight
   z = randnum(0,1);
   z = 4*z*(1-z);
   w = 0.5*randnum(w_min, w_max) + 0.5*z;
elseif (w_method == 7) || (w_method == 8) % sigmoid mapping inertia weight
   w = 1/(1 + 1.5*exp(-2.6*f_w));
else
   fprintf('Error !\n');
end

end % function
```

r_calculation_v2.m

```
% r_calculation_v2.m
% function for the calculation of the distance covered by a swarm particle per iteration
%
% version 2.0
% March 2012

function r_h = r_calculation_v2(x, x_h, i, Iter, rMax, g, g_h)

r_h1 = [x(:,1) - x_h(Iter-1,:,1)' x(:,2) - x_h(Iter-1,:,2)'].^2;
r_h1 = sqrt(r_h1(:,1) + r_h1(:,2));
if (r_h1 > 8*rMax)
   if debug_mode == true
      fprintf('\nWarning!!!, particle=%d, Iteration=%d\n', i, Iter);
      fprintf('\n x=[%3.3f %3.3f]   x_h=[%3.3f %3.3f]\n', x(:,1), x(:,2), x_h(Iter-1,:,1), x_h(Iter-1,:,2));
      fprintf('\n g=%3.3f    g_h=%3.3f\n', g, g_h);
      fprintf('r_h1=%5.2f\n\n', r_h1);
   end
   r_h1 = 8*rMax;
end

r_h = r_h1;

end
```

p_clamping.m

```
% p_clamping.m
% function for the calculation of 2-D position clamping
% used in PSO, OSPSO algorithms
%
% version 1.0
% March 2012

function [x,v] = p_clamping(x, v, xMin, xMax, yMin, yMax)


if x(:,1) > xMax
   x(:,1) = xMax;
   v(:,1) = -v(:,1);
end
if x(:,1) < xMin
   x(:,1) = xMin;
   v(:,1) = -v(:,1);
end
if x(:,2) > yMax
   x(:,2) = yMax;
   v(:,2) = -v(:,2);
end
 if x(:,2) < yMin
    x(:,2) = yMin;
    v(:,2) = -v(:,2);
 end


end
```

v_clamping.m

```
% v_clamping.m
% function for the calculation of 2-D velocity clamping
% used in PSO, OSPSO algorithms
%
% version 1.0
% March 2012

function v = v_clamping(v, D, vMin, vMax)


for j = 1:D
   if abs(v(:,j)) > vMax
      v(:,j) = sign(v(:,j))*vMax;
   end
   if abs(v(:,j)) < vMin
      v(:,j) = sign(v(:,j))*vMin;
   end
end


end
```