Τ.Ε.Ι ΠΕΙΡΑΙΑ
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ
ΤΜΗΜΑ ΑΥΤΟΜΑΤΙΣΜΟΥ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

PARTICLE SWARM OPTIMIZATION ΚΑΙ ΤΕΧΝΟΛΟΓΙΚΕΣ ΕΦΑΡΜΟΓΕΣ

ΣΠΟΥΔΑΣΤΗΣ: ΚΑΡΥΟΦΥΛΛΗΣ ΔΗΜΗΤΡΗΣ

ΕΞΑΜΗΝΟ: 18^{Ο}

Α.Μ: 32531

ΕΠΙΒΛΕΠΩΝ: Δρ. ΑΝΑΣΤΑΣΙΟΣ. Ι. ΝΤΟΥΝΗΣ

Αθήνα, Αύγουστος 2013

Ευχαριστίες

Οφείλω να πω ένα μεγάλο ευχαριστώ στο Δάσκαλο μου και Επιβλέποντα Καθηγητή της παρούσας πτυχιακής Δρ. Αναστάσιο Ντούνη, τόσο για το ότι με μύησε στον μαγικό κόσμο των Ευφυών Συστημάτων, όσο και για την υπομονή, την υποστήριξη και την πίστη που μου έδειξε καθ' όλη την διάρκεια της εργασίας μου.

Επίσης, ευχαριστώ την ομάδα του κυρίου Ντούνη, Σταύρο Σταυρινίδη και Παναγιώτη Κοφινά, για τις συμβουλές τους, τις ωραίες συζητήσεις και την ευκαιρία που μου έδωσαν να κατανοήσω βαθύτερα το αντικείμενο της εργασίας.

Τέλος, ευχαριστώ τους γονείς μου, που ανέχτηκαν την τόση προσήλωσή μου στην εργασία αυτή εις βάρος, κυρίως, της συμμετοχής μου στις οικογενειακές υποχρεώσεις.

Πίνακας περιεχομένων

Ευχαριστίες ..... **iii**

Περίληψη ..... **ix**

Μέρος Πρώτο **1**

Βελτιστοποίηση, Εξελικτικοί Αλγόριθμοι **1**

και PSO **1**

1.1 Βελτιστοποίηση **2**

1.1.1 Μαθηματικά προβλήματα *3*

1.1.2 Καθημερινά προβλήματα *4*

1.2 Εξελικτικοί Αλγόριθμοι **5**

1.3 Βελτιστοποίηση Σμήνους Σωματιδίων **8**

(Particle Swarm Optimization - PSO) **8**

Μέρος Δεύτερο **14**

Υλοποίηση Αλγορίθμου, Παραλλαγές **14**

και Συγκρίσεις **14**

2.1 Εισαγωγή **15**

2.2 -SPSO- (Standard Particle Swarm Optimization) **17**

2.2.1 Εύρεση και αντικατάσταση προσωπικού βέλτιστου *24*

2.2.2 Αντικατάσταση νέων θέσεων προσωπικού βέλτιστου *27*

2.2.3 Εύρεση του ολικού βέλτιστου και αντικατάσταση των θέσεων αυτού *27*

2.2.4 Πίνακες ταχυτήτων και θέσεων *28*

2.3 Ανάλυση της Μεθόδου **31**

2.3.1 Κίνηση Σωματιδίου *31*

2.3.2 Μέγεθος σμήνους *33*

2.3.3 Μέγιστες επιτρεπόμενες ταχύτητες *33*

2.3.4 Αριθμός Επαναλήψεων *33*

2.3.5 Συντελεστές Επιτάχυνσης *34*

2.3.6 Δοκιμές και Μετρήσεις *34*

2.4 -APSO- (Adaptive Particle Swarm Optimization)....37
2.4.1 Συντελεστής Αδρανείας (Inertia Weight)....37
2.4.2 Μετρήσεις και Συγκρίσεις....40
2.4.3 Διερεύνηση μη γραμμικών Συντελεστών Αδρανείας....43
2.5 -Νέος Προτεινόμενος PSO-....46
2.5.1 Ο PSO ως σύστημα ελέγχου....46
2.5.2 Έλεγχος σε διακριτά συστήματα....49
2.5.3 Διακριτός PSO – Discrete PSO....50
2.5.4 Διερεύνηση των συντελεστών....52
2.5.5 Βέλτιστη ρύθμιση των συντελεστών....55
2.5.6 Μετρήσεις και Συγκρίσεις....60
2.5.7 Συμπέρασμα....68

Μέρος Τρίτο....69
Εφαρμογές του PSO σε....69
προβλήματα βελτιστοποίησης....69
3.1 Εισαγωγή....70
3.2 Πρώτη Εφαρμογή του PSO....71
3.2.1 Μέθοδος....71
3.2.2 Αποτελέσματα....74
3.3 Δεύτερη Εφαρμογή το PSO....78
3.3.1 Μέθοδος....78
3.3.2 Αποτελέσματα....80

Μέρος Τέταρτο....83
Συμπεράσματα....83
4.1 Εισαγωγή....84
4.2 PSO και νέος DPSO....84
4.3 Οι παράμετροι του PSO....86
4.3.1 Σωματίδια και Επαναλήψεις....86
4.3.2 Σταθερές επιτάχυνσης....87
4.3.3 Σταθερά αδράνειας....88
4.3.4 Περιορισμός Ταχυτήτων....89

4.4 Εφαρμογή του PSO στον αυτόματο έλεγχο 89
4.5 Υλοποίηση βελτιστοποίησης PID και FLC σε Matlab – Simulink 90
4.5.1 Υλοποίηση PID 90
4.5.2 Υλοποίηση FLC 93
4.6 Προτάσεις για περαιτέρω έρευνα 95
4.6.1 Το πρόβλημα 95
4.6.2 Προτεινόμενες Λύσεις 96
Παράρτημα 98
Α. Ο αλγόριθμος του SPSO 98
Β. Ο αλγόριθμος των αντικειμενικών συναρτήσεων 101
Αναφορές 104

Λίστα Σχημάτων και Εικόνων

Εικόνα 1: Το μονοπάτι του Hamilton 3
Εικόνα 2: Το πρόβλημα του Σακιού 4
Εικόνα 3: Σχηματική αναπαράσταση του τρόπου υπολογισμού των ταχυτήτων 31
Εικόνα 4: Σύγκριση απόδοσης για διαφορετικό αριθμό σωματιδίων 35
Εικόνα 5: Σύγκριση απόδοσης για διαφορετικό αριθμό επαναλήψεων 36
Εικόνα 6: Κατανομή της σύγκλησης 1500 δοκιμών με 50 επαναλήψεις 41
Εικόνα 7: Κατανομή της σύγκλησης 1500 δοκιμών με 150 επαναλήψεις 42
Εικόνα 8: Γραμμικός και μη γραμμικός συντελεστής w 44
Εικόνα 9: Απόδοση των L-APSO, Function 1 και Function 2 45
Εικόνα 10: Ο SPSO σαν σύστημα ελέγχου 48
Εικόνα 11: Ο APSO σαν σύστημα ελέγχου 49
Εικόνα 12: Σύστημα ελέγχου με ανατροφοδότηση 50
Εικόνα 13: Μπλοκ διάγραμμα του Διακριτού PSO 52
Εικόνα 14: Σύγκριση ταχυτήτων σωματιδίων των SPSO, APSO και DPSO 54
Εικόνα 15: Σύγκλιση του APSO και DPSO στην συνάρτηση Schwefel 63
Εικόνα 16: Σύγκλιση του APSO και DPSO στην συνάρτηση Griewangk 64
Εικόνα 17: Σύγκλιση του APSO και DPSO στην συνάρτηση Rosenbrock 65
Εικόνα 18: Σύγκλιση του APSO και DPSO στην συνάρτηση Michalewicz 66
Εικόνα 19: Σύγκλιση του APSO και DPSO στην συνάρτηση Rastring 67

Εικόνα 20: Υλοποίηση του PID-Type Fuzzy Controller 70
Εικόνα 21: Το σύστημα ελέγχου του DC μοτέρ 72
Εικόνα 22: Συναρτήσεις συμμετοχής εισόδων (πάνω) και εξόδου (κάτω) 72
Εικόνα 23: Η επιφάνεια εξόδου του Ασαφούς Ελεγκτή 73
Εικόνα 24: Η σύγκλιση των APSO και DPSO στην πορεία των 40 επαναλήψεων 75
Εικόνα 25: Βηματική απόκριση βέλτιστων αποτελεσμάτων 76
Εικόνα 26: Το σύστημα ελέγχου πρώτου βαθμού με καθυστέρηση 79
Εικόνα 27: Η σύγκλιση των APSO και DPSO στην πορεία των 40 επαναλήψεων 81
Εικόνα 28: Βηματική απόκριση βέλτιστων αποτελεσμάτων 82
Εικόνα 29: Παρέμβαση-ρύθμιση στο μενού του PID 91
Εικόνα 30: Παράδειγμα συστήματος με κριτήριο IAE στην έξοδο 91
Εικόνα 31: Ονοματοδοσία του FLC 93

Λίστα Πινάκων

Πίνακας 1: Υπόδειγμα πίνακα ταχυτήτων και θέσεων 28
Πίνακας 2: Τιμές παραγόντων του APSO που χρησιμοποιούνται στην εργασία 40
Πίνακας 3: Μέση σύγκλιση SPSO και APSO σε 50 και 150 επαναλήψεις 41
Πίνακας 4: Τιμές σύγκλισης της συνάρτησης 1 και του γραμμικού APSO 44
Πίνακας 5: Τιμές σύγκλισης της συνάρτησης 2 και του γραμμικού APSO 45
Πίνακας 6: Σύγκριση συγκλίσεων των εξεταζόμενων αλγορίθμων 55
Πίνακας 7: Επιλεγμένες τιμές των συντελεστών ύστερα από βελτιστοποίηση 59
Πίνακας 8: Το προτεινόμενο σετ τιμών – βάση 60
Πίνακας 9: Σετ τιμών που χρησιμοποιήθηκε στις μετρήσεις και συγκρίσεις 60
Πίνακας 10: Benchmark συναρτήσεις που χρησιμοποιήθηκαν 61
Πίνακας 11: Χαρακτηριστικά της εκάστοτε συνάρτησης 62
Πίνακας 12: Σύγκριση συγκλίσεων των APSO και DPSO 63
Πίνακας 13: Σύγκριση συγκλίσεων των APSO και DPSO 64
Πίνακας 14: Σύγκριση συγκλίσεων των APSO και DPSO 65
Πίνακας 15: Σύγκριση συγκλίσεων των APSO και DPSO 66
Πίνακας 16: Σύγκριση συγκλίσεων των APSO και DPSO 67
Πίνακας 17: Οι ασαφείς κανόνες της εξόδου του ελεγκτή 73
Πίνακας 18: Σετ τιμών του DPSO που χρησιμοποιήθηκε στην πρώτη εφαρμογή 73
Πίνακας 19: Αποτελέσματα των δύο αλγορίθμων στην πρώτη εφαρμογή 74

Πίνακας 20: Βέλτιστες ρυθμίσεις των συντελεστών του ελεγκτή κατά τον APSO και DPSO 76
Πίνακας 21: Σετ τιμών του DPSO που χρησιμοποιήθηκε στην δεύτερη εφαρμογή ... 80
Πίνακας 22: Αποτελέσματα των δύο αλγορίθμων στην δεύτερη εφαρμογή 80
Πίνακας 23: Βέλτιστες ρυθμίσεις των συντελεστών του ελεγκτή κατά τον APSO και DPSO 82
Πίνακας 24: Κλασικές συναρτήσεις που χρησιμοποιήθηκαν για την σύγκριση του DPSO με τον APSO 85
Πίνακας 25: Μέσες τιμές συγκλίσεων των δύο αλγορίθμων και ποσοστική βελτίωση 85
Πίνακας 26: Αποτελέσματα των δύο αλγορίθμων στα δύο συστήματα που εξετάστηκαν 86

Πίνακας Ορολογιών / συμβολισμών

Μεθευρετικοί Αλγόριθμοι	Κατηγορία αλγορίθμων οι οποίοι μην έχοντας γνώση του προβλήματος προς επίλυση επιστρατεύουν μεθόδους τυχαίας αναζήτησης για την επίλυση
SPSO	Ο κλασικός, απλός (Simple) PSO
APSO	Προσαρμοστικός (Adaptive) PSO
DPSO	Διακριτός (Discrete) PSO
Benchmark Functions	Κλασικές μαθηματικές συναρτήσεις για την αξιολόγηση αλγορίθμων βελτιστοποίησης
Fitness	Αριθμητική τιμή ένδειξης της ποιότητας της λύσης
Agents	Σωματίδια. Αποτελούν τις προτεινόμενες λύσεις.
v_{id} / v(t)	Ταχύτητα σωματιδίου i στην διάσταση d / στον διακριτό χρόνο (επανάληψη) t
p_{id} / x(t)	Θέση σωματιδίου i στην διάσταση d / στον διακριτό χρόνο (επανάληψη) t
pbest	Προσωπική βέλτιστη λύση
gbest	Ολική βέλτιστη λύση
r1, r2 / rand	Τυχαίες τιμές κανονικής κατανομής στο [0 1]
c1, c2	Σταθερές επιτάχυνσης συνήθως ίσες με 2
w	Συντελεστής αδράνειας, συνήθως γραμμικός
firstval	Τιμή από την οποία το w ξεκινά στην αρχή των επαναλήψεων
finalval	Τιμή στην οποία το w καταλήγει στο πέρας των επαναλήψεων

Περίληψη

Στα πλαίσια της τεχνητής νοημοσύνης και τεχνητής ζωής έχουν αναπτυχθεί διάφοροι αλγόριθμοι που μιμούμενοι διεργασίες της φύσης προσδοκούν να δώσουν λύση σε διάφορα προβλήματα. Τέτοιοι αλγόριθμοι ονομάζονται μεθευρετικοί αλγόριθμοι βελτιστοποίησης. Αντικείμενο της ακόλουθης εργασίας είναι η παρουσίαση του αλγορίθμου Βελτιστοποίησης Σμήνους Σωματιδίων (Particle Swarm Optimization - PSO) και της χρήσης του σε εφαρμογές στα συστήματα αυτομάτου ελέγχου.

Στο πρώτο μέρος γίνεται μια αναφορά στις μεθόδους και τις τεχνικές βελτιστοποίησης της Υπολογιστικής Νοημοσύνης καθώς και μια ιστορική αναδρομή και επεξήγηση της φιλοσοφίας του αλγορίθμου του PSO.

Στο δεύτερο μέρος γίνεται μια εκτενής παρουσίαση του αλγορίθμου σε περιβάλλον προγραμματισμού Matlab. Ακόμα, παρουσιάζεται μια από της πιο γνωστές παραλλαγές και βελτιωμένες εκδόσεις του αλγορίθμου, APSO, καθώς και μια νέα έκδοση, DPSO, που αναπτύχθηκε και για πρώτη φορά παρουσιάζεται στα πλαίσια της παρούσας πτυχιακής. Οι δυο αυτές εκδόσεις συγκρίνονται μεταξύ τους στην επίλυση προβλημάτων μαθηματικής βελτιστοποίησης (test functions) και παρουσιάζονται τα αποτελέσματα.

Στο τρίτο μέρος αναλύεται το πρόβλημα ελέγχου δύο πραγματικών συστημάτων ελέγχου μέσω μοντελοποίησης στο Simulink. Για τον έλεγχο των συγκεκριμένων φυσικών συστημάτων υλοποιείται ο PID-Τύπου ασαφής ελεγκτής και ρυθμίζεται μέσω των δύο προαναφερθέντων εκδοχών του PSO.

Τέλος, τα ερευνητικά ευρήματα και συμπεράσματα που προκύπτουν, παρουσιάζονται στο τέταρτο και τελευταίο μέρος της εργασίας. Ακόμα, εκεί μπορεί κανείς να βρει προτεινόμενες λύσεις για την χρήση και ανάπτυξη του PSO σε περιβάλλον Matlab, καθώς και προτάσεις για περαιτέρω έρευνα και ανάπτυξη του προτεινόμενου DPSO.

Summary

In the context of Artificial Intelligence and Artificial Life several algorithms which imitate the processes of nature have developed, hoping to give a solution to various problems. Such algorithms are called metaheurisic optimization algorithms. Purpose of the following paper is to present the algorithm of Particle Swarm Optimization (Particle Swarm Optimization - PSO) and apply it in automatic control systems.

The first part is a reference to methods and optimization techniques of Computational Intelligence and a historical overview and explanation of the philosophy of the algorithm of PSO.

The second part is a detailed presentation of the algorithm in a programming environment called Matlab. Also, it presents one of the most famous variations and improved versions of the algorithm, APSO, and a new version, DPSO, developed and first presented in the framework of this thesis. These two versions are compared with each other in solving mathematical optimization (test functions) and the results presented.

The third part analyzes the problem of verification of two real control systems through modeling in Simulink. For the control of certain physical systems a PID-Type fuzzy controller is implemented the and regulated via the above two versions of PSO.

Finally, research findings and conclusions drawn are presented in the fourth and final part of the job. Additionally, one can find proposed solutions, use and development of PSO in Matlab environment, and suggestions for further research and development of the proposed DPSO.

Μέρος Πρώτο

Βελτιστοποίηση, Εξελικτικοί Αλγόριθμοι και PSO

1.1 Βελτιστοποίηση

Στην σημερινή εποχή όπου ο ανταγωνισμός είναι μεγάλος, η επιτυχία κρίνεται από τις λεπτομέρειες και το κόστος και η μείωσή του αποτελεί σημαντικότατο παράγοντα, οι απαιτήσεις για «καλά και γρήγορα» αποτελέσματα ολοένα και αυξάνονται. Λύση στην κατάσταση αυτή έρχονται να δώσουν οι αλγόριθμοι βελτιστοποίησης, που όπως δηλώνει και το όνομά τους, αναλαμβάνουν να βρουν την βέλτιστη ή τις βέλτιστες λύσεις σε δύσκολα και δυσεπίλυτα προβλήματα.

Η βελτιστοποίηση στον χώρο των εφαρμοσμένων μαθηματικών βρίσκει εφαρμογή σε διάφορους τομείς συνήθως με τη μορφή προβλημάτων ελαχιστοποίησης κόστους ή μεγιστοποίησης κέρδους. Με απλά λόγια, το μαθηματικό πρόβλημα που ζητά λύση είναι η εύρεση του ολικού ελάχιστου ή μέγιστου μιας συνάρτησης. Η συνάρτηση αυτή εμπεριέχει το πρόβλημα του οποίου η λύση αναζητείται και συνήθως είναι πολλών διαστάσεων (πρόβλημα με πολλούς παράγοντες). Πολλά πρακτικά προβλήματα από τον χώρο της επιστήμης, της μηχανικής, της φυσικής της βιολογίας και της χημείας, της οικονομίας, των υπολογιστών, της βιομηχανίας ή ακόμα και της διοίκησης μπορούν να μετατραπούν σε προβλήματα βελτιστοποίησης χρησιμοποιώντας μια αντικειμενική συνάρτηση που τα εκφράζει.

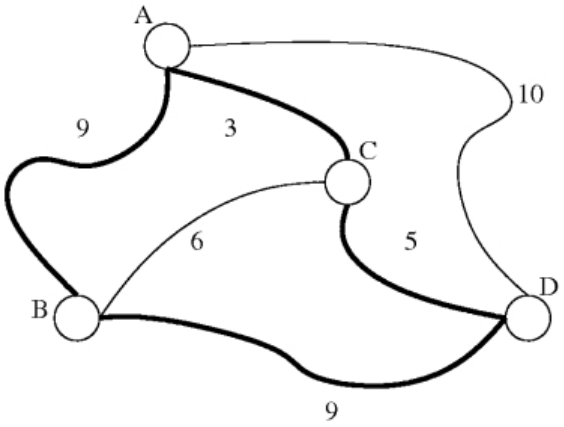
Η βελτιστοποίηση πολλαπλών κριτηρίων, όπως στις παραπάνω περιπτώσεις, έχει κάποια συγκεκριμένα χαρακτηριστικά που αποτελούν και το λόγο που τέτοιου είδους προβλήματα δεν επιδέχονται εύκολες λύσεις. Συγκεκριμένα, πρώτο χαρακτηριστικό είναι η ύπαρξη πολλών δυνατών λύσεων που δίνουν αποτελέσματα διαφορετικής «ποιότητας». Δεύτερο χαρακτηριστικό είναι η αλληλεπίδραση πολλών παραγόντων που αντικρούονται μεταξύ τους και η μεταβολή του ενός θα προκαλέσει την αντίθετη μεταβολή κάποιων άλλων με αποτέλεσμα να μειωθεί η ποιότητα της λύσης. Έτσι, η σύγκριση της ποιότητας της κάθε λύσης με βάση τα κριτήρια που τίθενται και η επιλογή της κατάλληλης είναι τελικός ο στόχος της βελτιστοποίησης.

Για να γίνουν πιο αντιληπτές οι έννοιες αυτές παρακάτω αναφέρονται μερικά γνωστά προβλήματα αναζήτησης βέλτιστης λύσης καθώς και μερικά παραδείγματα της καθημερινής ζωής.

1.1.1 Μαθηματικά προβλήματα

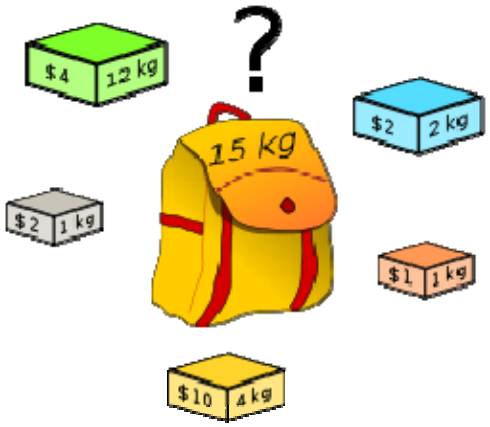
Ένα από τα πιο διάσημα προβλήματα είναι αυτό του Πλανόδιου Πωλητή (Travelling Salesman Problem-TSP). Σύμφωνα με αυτό, θεωρούμε πως ένας πλανόδιος πωλητής θέλει να επισκεφτεί κάποιες πόλεις με δεδομένες αποστάσεις μεταξύ τους. Το πρόβλημα της βελτιστοποίησης είναι να βρεθεί η συνολικά μικρότερη απόσταση που θα διανύσει ο πωλητής για να περάσει μόνο μία φορά από κάθε πόλη και να γυρίσει σε αυτή από όπου ξεκίνησε. Πρόβλημα παρόμοιο με αυτό, είναι το Μονοπάτι του Hamilton (Hamilton Path).

Στο επόμενο σχήμα φαίνονται οι πόλεις, οι πιθανές διαδρομές με τις αποστάσεις τους καθώς τονισμένη είναι η βέλτιστη λύση (A->B->D->C).



Εικόνα 1: Το μονοπάτι του Hamilton

Το 0-1 Σακί (0-1 Knapsack) είναι ένα άλλο παράδειγμα γνωστό στη μαθηματική κοινότητα. Σκοπός τώρα είναι να τοποθετηθούν σε ένα σακί με δεδομένο όριο βάρους αντικείμενα που ποικίλουν σε αξία και βάρος. Ζητείται τα αντικείμενα που τελικά θα μπουν στο σακί να έχουν τη μεγαλύτερη δυνατή αξία χωρίς να παραβιαστεί ο περιορισμός του βάρους. Έτσι, δεδομένων των πολλών πιθανών πρέπει να γίνει προσεκτική διαλογή των αντικειμένων ώστε να επιλεγεί ο βέλτιστος συνδυασμός.



Εικόνα 2: Το πρόβλημα του Σακιού

Στην προηγούμενη εικόνα φαίνεται σχηματικά ένα παράδειγμα του προβλήματος του Σακιού. Η αξία και το βάρος του κάθε αντικειμένου αναγράφονται σε αυτά και το μέγιστο βάρος που μπορεί να μπει στο σακίδιο είναι 15 κιλά.

1.1.2 Καθημερινά προβλήματα

Κάποια πολύ απλά και καθημερινά ζητήματα που ασυναίσθητα βελτιστοποιούν όλοι οι άνθρωποι αφορούν όπως έχει αναφερθεί στις επιλογές που γίνονται.

Έτσι, αν σκεφτούμε κάποιον που μπαίνει στο λεωφορείο όταν αυτό έχει αρκετές άδειες θέσεις και αποφασίζει σε μερικά δευτερόλεπτα το που θα καθίσει, λύνει άμεσα ένα πρόβλημα βελτιστοποίησης. Σαν πιθανές λύσεις εμφανίζονται όλες οι άδεις θέσεις που υπάρχουν και σαν κριτήρια αξιολόγησης της κάθε μιας μπορεί να είναι διάφοροι παράγοντες όπως η άνεση, ο ήλιος, ο αερισμός ή ακόμα το ποιός κάθεται στη διπλανή θέση.

Ένα άλλο παράδειγμα αφορά στο αυτοκίνητο το οποίο οδηγούμε. Πιθανές λύσεις είναι όλα τα αυτοκίνητα της αγοράς και κριτήρια αποτελούν οι τιμή αγοράς και άλλα πιθανά έξοδα, το είδος του αυτοκινήτου σε συνδυασμό με τη χρήση του, ο βαθμός ανταπόκρισής του στις απαιτήσεις του αγοραστή, η χώρα προέλευσης και πολλά άλλα.

Καταλαβαίνουμε λοιπόν πως το ζήτημα της επιλογής μεταξύ κάποιων λύσεων είναι τόσο συνηθισμένο στην καθημερινή μας ζωή, όσο και απαραίτητο σε πρακτικές,

τεχνολογικές και επιστημονικές εφαρμογές. Για αυτούς τους λόγους, οι μέθοδοι βελτιστοποίησης χρησιμοποιούνται και εξελίσσονται τα τελευταία χρόνια με γοργούς ρυθμούς εκμεταλλευόμενοι την ανάπτυξη στην επιστήμη των υπολογιστών. Η Υπολογιστική Νοημοσύνη είναι κινητήρια δύναμη ανάπτυξης τέτοιων μεθόδων και τεχνικών.

1.2 Εξελικτικοί Αλγόριθμοι

Στον χώρο της Τεχνητής Νοημοσύνης και με τη βοήθεια της συνεχώς αυξανόμενης υπολογιστικής ισχύος γίνονται τις τελευταίες δεκαετίες προσπάθειες να αναπαρασταθούν σε συνθήκες προσομοίωσης διάφορες διεργασίες της φύσης. Οι ερευνητές μελετώντας συστήματα σχετικά με τη ζωή προσπαθούν να προσομοιώσουν διαδικασίες όπως αυτή της φυσικής εξέλιξης των ειδών ή και να αναπαράγουν τις δραστηριότητες διαφόρων πληθυσμών ζώων με βάση τα κοινωνικά χαρακτηριστικά τους.

Αυτός ο αμφιλεγόμενος τομέας έρευνας, που ονομάζεται Τεχνητή Ζωή (Artificial Life), χωρίζεται σε τρεις κατηγορίες ως αναφορά τις εφαρμογές του: εμφανίζεται με τη μορφή λογισμικού (software), με τη μορφή ρομπότ (hardware) αλλά εφαρμόζεται και στα πεδία της βιοχημείας (synthetic biology).

Σκοπός της καθεμίας από τις τρεις αυτές κατηγορίες είναι η δημιουργία νέων ατόμων με συμπεριφορά που να προσεγγίζει αυτήν των πραγματικών ζωντανών οργανισμών. Παραδείγματος χάριν, η Τεχνητή Ζωή σε εφαρμογές υλικού μελετά τη μορφή οργανισμών και κατασκευάζονται ρομπότ που προσομοιάζουν είτε τη μορφή του ανθρώπου, είτε κάποιου ζώου όπως αράχνες ή εξάποδα και κινούνται όπως αυτά. Από την άλλη, σε επίπεδο λογισμικού γίνονται εικονικές προσομοιώσεις φυσικών διαδικασιών σε μοντέλα υλοποιημένα σε ηλεκτρονικούς υπολογιστές. Τέλος, η Συνθετική Βιολογία ή Συνθετική Ζωή θέτει ως στόχο της την κατασκευή συνθετικού DNA και εν τέλει τη δημιουργία ζωής εκ του μηδενός.

Στη συνέχεια θα ασχοληθούμε με την Τεχνητή Ζωή μόνο στο επίπεδο του λογισμικού και με τις εφαρμογές του στην βελτιστοποίηση.

Οι Εξελικτικοί Αλγόριθμοι (Evolutionary Algorithms - EA) δεν είναι παρά μια πρακτική εφαρμογή των παραπάνω τεχνικών που βασίζονται ως επί το πλείστον στην μελέτη οργανισμών ή διαδικασιών που παρατηρούνται στη φύση. Η μελέτη και η προσομοίωσή τους οδηγεί στην δημιουργία αλγορίθμων ικανών να εξελίσσονται και να προσαρμόζονται με τρόπο ανάλογο με αυτόν του οργανισμού τον οποίο μιμούνται.

Έχουν κατά καιρούς εμφανιστεί και προταθεί διάφορων ειδών ΕΑ. Ένας από τους πρώτους αλγορίθμους που έγινε ιδιαίτερα γνωστός στα μέσα της δεκαετίας του 1970 (*John Holland, Adaptation in Natural and Artificial Systems, 1975*) που έθεσε και τις βάσεις για τη συνέχεια σε αυτού του είδους τους αλγορίθμους ήταν οι Γενετικοί Αλγόριθμοι (Genetic Algorithms). Αυτοί, εμπνευσμένοι από την πρόταση για την εξέλιξη των ειδών σύμφωνα με τη Δαρβινική θεωρία, προσομοιώνουν τη διαδικασία του ανασυνδυασμού μεταξύ γονέων (χρωμοσωμάτων) που έχουν επιλεγεί ως πιο ικανοί για επιβίωση (ή αλλιώς, προσέφεραν καλύτερες λύσεις στο πρόβλημα που επιλύεται) και με τη μέθοδο της μετάλλαξης και με το πέρασμα των γενεών προσδοκούν στην εύρεση μιας βέλτιστης λύσης.

Ακόμα, εμφανίστηκαν αλγόριθμοι που μιμούνται πλήθη ζώων ή και εντόμων με αντικείμενο την προσομοίωση της κοινωνικής τους δράσης και με χαρακτηριστικό τους τις αξιοθαύμαστες μεθόδους που χρησιμοποιούν αυτοί οι οργανισμοί για να ανταλλάσσουν πληροφορίες και να σχηματίζουν οργανωμένους οικισμούς. Αυτοί οι αλγόριθμοι ανήκουν στην κατηγορία της Νοημοσύνης των Σμηνών (Swarm Intelligence) και πιο γνωστοί εκπρόσωποί τους είναι η Αποικία των Μυρμηγκιών (Ant Colony), η Βελτιστοποίηση Σμήνους Σωματιδίων (Particle Swarm Optimization), η Αποικία των Μελισσών (Bee Colony Algorithm) αλλά και η Αναζήτηση της Μαϊμούς (Monkey Search).

Όλοι αυτοί οι αλγόριθμοι εκμεταλλεύονται τους τρόπους επικοινωνίας των οργανισμών από τους οποίους πηγάζει η εμπνευστή τους και καταφέρνουν με εύκολο υπολογιστικά και γρήγορο τρόπο να δώσουν ικανοποιητικά αποτελέσματα σε προβλήματα βελτιστοποίησης.

Παρά τις όποιες διαφορές στη λειτουργία της κάθε μεθόδου, σε γενικές γραμμές όλες βασίζονται στην λογική που παρουσιάζεται στον παρακάτω ψευδοκώδικα:

Αρχή

Αρχικοποίηση Πληθυσμού

Αξιολόγηση

Επανέλαβε

Ανασυνδυασμός

Μετάλλαξη

Αξιολόγηση

Επιλογή

Μέχρι το κριτήριο τερματισμού να ικανοποιηθεί

Στην έξοδο η καλύτερη λύση

Τέλος

Έτσι, μια γενική εικόνα ενός ΕΑ αποτελείται από:

i. τη διαδικασία της δημιουργίας ενός τυχαίου αρχικού πληθυσμού, που θα αποτελεί ένα πλήθος πιθανών λύσεων του προβλήματος
ii. τη διαδικασία της αξιολόγησης που με βάση την αντικειμενική συνάρτηση (fitness function) που έχει οριστεί για το κάθε πρόβλημα, κρίνει την καταλληλότητα της κάθε λύσης
iii. τη διαδικασία του ανασυνδυασμού όπου ανάλογα με τον τρόπο λειτουργίας του κάθε αλγορίθμου δημιουργούνται οι απόγονοι, με βάση την προηγούμενη γενιά
iv. τη διαδικασία της μετάλλαξης όπου προστίθεται η «τυχαιότητα» και παραποιούνται οι απόγονοι
v. και τέλος τη διαδικασία της επιλογής όπου καθορίζονται ποιοι «γονείς» θα χρησιμοποιηθούν για τη δημιουργία της επόμενης γενιάς

Τα παραπάνω επαναλαμβάνονται όπως φαίνεται στον ψευδοκώδικα μέχρι να ικανοποιηθεί το κριτήριο τερματισμού που μπορεί να είναι ένας μέγιστος αριθμός επαναλήψεων ή η επίτευξη μιας τιμής στόχου της αντικειμενικής συνάρτησης.

Το χαρακτηριστικό των ΕΑ που τους κάνει να προτιμούνται από άλλες κλασικές μεθόδους βελτιστοποίησης όπως οι μέθοδοι quasi-Monte-Carlo, είναι το ότι είναι στοχαστικές ευρεστικές μέθοδοι ή αλλιώς μεθευρετικοί αλγόριθμοι. Αυτό σημαίνει πως ο αλγόριθμος έχει ελάχιστη ή καθόλου γνώση του προβλήματος ή της συνάρτησης που καλείται να λύσει και δεν χρειάζεται παραγώγους ή άλλες πολύπλοκα υπολογιζόμενες παραμέτρους. Εξετάζει ένα πληθυσμό λύσεων ταυτόχρονα αντί για ένα μόνο σημείο λύσης και γιαυτό μπορεί να ψάξει μεταξύ ενός τεραστίου πλήθους λύσεων για τις βέλτιστες. Από την άλλη όμως, λόγω της στοχαστικότητας που τους διακρίνει δεν μπορούν να εγγυηθούν πως θα βρουν το ολικό βέλτιστο.

1.3 Βελτιστοποίηση Σμήνους Σωματιδίων (Particle Swarm Optimization - PSO)

Όπως αναφέρθηκε παραπάνω, ο αλγόριθμος PSO χαρακτηρίζεται ως ένας εξελικτικός μεθευρετικός αλγόριθμος βελτιστοποίησης, που βασίζει τη λειτουργία του στη μίμηση και την προσομοίωση των κοινωνικών χαρακτηριστικών μιας ομάδας ατόμων. Συγκεκριμένα, μπορούμε να πούμε πως εμπνεύστηκε από τον τρόπο που κινούνται τα σμήνη των πουλιών στον αέρα, ή τα κοπάδια των ψαριών στο νερό.

Με μία παρουσίασή τους με τίτλο *Particle Swarm Optimization* οι Αμερικάνοι *James Kennedy, Κοινωνικός Ψυχολόγος* και *Russell Eberhart, Ηλεκτρολόγος Μηχανικός* σε συνέδριο του IEEE το 1995 [1] περιγράφουν τα κίνητρα με τα οποία ξεκίνησαν και τη διαδικασία που ακολούθησαν για να καταλήξουν εν αγνοία τους στην δημιουργία μιας γρήγορης και αποτελεσματικής μεθόδου βελτιστοποίησης. Αυτή η διαδικασία ακολουθεί παρακάτω.

Η αρχική πρόθεση, όπως περιγράφουν οι δύο ερευνητές, ήταν η εξομοίωση και η γραφική αναπαράσταση ενός σμήνους πουλιών που σε δύο διαστάσεις θα πραγματοποιούσε την χαρακτηριστική χορογραφία που παρατηρείται όταν ένα

σμήνος πετά. Σκοπός ήταν να προσομοιώσουν όχι μόνο την ταυτόχρονη κίνησή τους, αλλά και τις απότομες και φαινομενικά τυχαίες αλλαγές διεύθυνσης που κάνουν τον «χορό» τους τόσο ενδιαφέρον.

Για την δημιουργία της γραφικής αναπαράστασης, δημιουργήθηκε αρχικά ένα πλήθος πουλιών, ή καλύτερα *agents* όπως αρχικά ονομάστηκαν, και ανατέθηκε στο καθένα μια τυχαία ταχύτητα για τον άξονα **x** και άλλη μια για τον άξονα **y**. Με βάση αυτές τις ταχύτητες και σε κάθε επανάληψη του αλγορίθμου ο κάθε agent κινούνταν στον χώρο των δύο διαστάσεων. Για την αποφυγή των συγκρούσεων χρησιμοποιήθηκε η τεχνική της αλλαγής της ταχύτητας βάσει του πλησιέστερου γείτονα. Σύμφωνα με αυτή, σε κάθε επανάληψη υπολογιζόταν για κάθε agent ποιος άλλος agent βρισκόταν πιο κοντά του και οι ταχύτητες του πιο κοντινού δίνονταν στον υπό εξέταση agent. Έτσι μπόρεσαν να εμφανίσουν ικανοποιητικά την συγχρονισμένη κίνηση όλου του σμήνους χωρίς να υπάρχουν περιστατικά συγκρούσεων, αφού προτού δύο agents πλησιάσουν αρκετά μεταξύ τους ο ένας έστριβε προς την κατεύθυνση του άλλου.

Γρήγορα όμως όλο το σμήνος κατέληγε να κατευθύνεται προς την ίδια αμετάβλητη κατεύθυνση. Σε αυτό το σημείο έπρεπε να προστεθεί ένας παράγοντας τυχαιότητας (craziness). Με την προσθήκη κάποιας αλλαγής σε διάφορες τυχαίες ταχύτητες στους δύο άξονες επιτεύχθηκε η φαινομενικά τυχαία χορευτική κίνηση του σμήνους που το έκανε να δείχνει ζωντανό.

Το ενδιαφέρον ενός ζωολόγου, του Heppner, για το πώς τα πουλιά βρίσκουν την τροφή τους και η προσπάθειά του να εξομοιώσει στην οθόνη την πτήση τους μέχρι την «ταΐστρα» έθεσε ένα άλλο ερώτημα. Σε αντίθεση με τα πουλιά του Heppner που ήξεραν που βρίσκεται η τροφή τους, τα πουλιά στη φύση δεν έχουν τέτοιου είδους πληροφόρηση. Παρ ‘όλα αυτά, εάν μία ποσότητα τροφής αφεθεί σε σημείο που μπορούν να προσεγγίσουν τα πουλιά, σε λίγη ώρα θα έχουν μαζευτεί αρκετά και σύντομα θα έρθουν κι άλλα. Προκύπτει λοιπόν λογικά το ερώτημα: πώς βρίσκουν τα πουλιά την τροφή τους;

Την απάντηση σε αυτή την ερώτηση προσπάθησαν να δώσουν με τον εξής τρόπο:

Στην οθόνη όπου κινούνταν οι agents προστέθηκε ένα σημείο (x,y) όπου βρισκόταν η υποτιθέμενη τροφή. Ύστερα προγραμμάτισαν κάθε agent να αξιολογεί την θέση που βρίσκεται στην κάθε επανάληψη με βάση την συνάρτηση:

$$Eval=\sqrt{(presentx-100)^2}+\sqrt{(presenty-100)^2}$$

Έτσι, όταν κάποιος agent βρισκόταν στην θέση (100,100) η τιμή της συνάρτησης γινόταν Eval=0.

Στο σημείο αυτό σε κάθε agent δόθηκαν χαρακτηριστικά μνήμης. Συγκεκριμένα, ο καθένας θυμόταν την καλύτερη τιμή (Eval) που είχε «ανακαλύψει» μέχρι εκείνο τη στιγμή, καθώς επίσης και σε ποια συντεταγμένη του χώρου (x,y) είχε σημειωθεί αυτή η τιμή. Ως «μνήμη» χρησιμοποιήθηκαν οι μεταβλητές *pbest[], pbestx[]* και *pbesty[]* για την καλύτερη τιμή (Eval), την θέση της στον οριζόντιο και στον κάθετο άξονα, αντίστοιχα. Σημειώνεται πως οι αγκύλες συμβολίζουν το ότι οι μεταβλητές αυτές είναι διανύσματα όπου κάθε στοιχείο τους αναφέρεται σε έναν άλλο agent. Αυτός ο συμβολισμός θα χρησιμοποιείται και στη συνέχεια της εργασίας.

Για την κίνηση λοιπόν του κάθε agent στον χώρο των λύσεων εφαρμόστηκε η εξής λογική: Εάν στον άξονα των τετμημένων η παρούσα θέση του agent βρισκόταν πιο αριστερά από την *pbestx* που αντιστοιχεί σε αυτόν τον agent τότε στην ταχύτητα του οριζόντιου άξονα (*vx[])* θα προστεθεί μια τυχαία ποσότητα (*rand()*p_increment,* όπου ο όρος *p_increment* είναι μια σταθερή παράμετρος) με την ταχύτητα να γίνεται:

$$vx[]=vx[]+rand()*p\_increment$$

Αντίστοιχα αν η θέση του *pbestx* είναι αριστερά από την παρούσα θέση τότε η τυχαία αυτή ποσότητα αφαιρείται από την ταχύτητα:

$$vx[]=vx[]-rand()*p\_increment$$

Ομοίως στον κάθετο άξονα αν το σημείο βρίσκεται χαμηλότερα από την παρούσα θέση του agent, η προηγούμενη ποσότητα αφαιρείται από την ταχύτητα του agent στον άξονα y:

$$vy[]=vy[]-rand()*p\_increment$$

Ή αν είναι ψηλότερα, προστίθεται σε αυτήν:

$$vy[]=vy[]+rand()*p\_increment$$

Έτσι, ο agent ρυθμίζει την ταχύτητά του ώστε να επιστρέφει στο σημείο όπου συνάντησε την καλύτερη τιμή *pbest* αλλά με τρόπο τυχαίο έτσι ώστε να ευνοείται το «ψάξιμο» και στις γύρω περιοχές.

Εκτός αυτής της μεταβλητής προστέθηκε μία ακόμα, η τοποθεσία του συνολικά καλύτερου σημείου που έχει βρεθεί από όλο το σμήνος. Οι αντίστοιχες ονομασίες ήταν *gbest[], gbestx[]* και *gbesty[]* που εκφράζουν την καλύτερη ολική τιμή και τις θέσεις της στον άξονα *x* και *y* αντίστοιχα. Στη γνώση αυτή είχε πρόσβαση όλο το σμήνος και με αυτόν τον τρόπο συμβολιζόταν η μετάδοση της γνώσης της τοποθεσίας του φαγητού που εμφανίζεται στη φύση. Με βάση τις παραπάνω μεταβλητές και με μία άλλη σταθερή παράμετρο, την *g_increment*, η ταχύτητα του κάθε agent ρυθμιζόταν με τρόπο ανάλογο με αυτόν που παρουσιάστηκε παραπάνω.

Εάν ο agent βρίσκεται πάνω από τη θέση στην οποία έχει σημειωθεί η καλύτερη ολική τιμή (*gbest*):

$$vy[]=vy[]-rand()*g\_increment$$

Εάν βρίσκεται από κάτω:

$$vy[]=vy[]+rand()*g\_increment$$

Εάν βρίσκεται αριστερά:

$$vx[]=vx[]+rand()*g\_increment$$

Εάν βρίσκεται δεξιά:

$$vy[]=vy[]-rand()*g\_increment$$

Με τη ρύθμιση της ταχύτητας του κάθε agent σύμφωνα με τις παραπάνω εξισώσεις και σύμφωνα με τους κανόνες του πλησιέστερου γείτονα και της τυχαιότητας που περιγράφηκαν πιο πάνω, κατέληξαν σε μία αρκετά αληθοφανή προσομοίωση της κίνησης του σμήνους προς την θέση στην οποία ήταν τοποθετημένο το «φαγητό». Τονίζεται ακόμα το πως επηρεάζει την κίνηση του σμήνους η αλλαγή των τιμών στις παραμέτρους *g_increment* και *p_increment.* Σύμφωνα με τους ερευνητές, με μεγάλες τιμές τα εικονικά πουλιά φαινόταν σαν να «ρουφιούνται» απότομα από την ίδια τους την ταΐστρα, ενώ αντίθετα, με μικρές τιμές,

κινούνταν ήρεμα ερευνώντας την περιοχή μέχρι που τελικά «προσγειώνονταν» σε αυτή.

Προσπαθώντας να εξελίξουν και να βελτιώσουν τον αλγόριθμο για την επίλυση απλών δισδιάστατων συναρτήσεων, κατέληξαν πως η επιπρόσθετη τυχαιότητα (craziness) καθώς και ο κανόνας του πλησιέστερου γείτονα δεν ήταν απαραίτητοι για τη λειτουργία του αλγορίθμου και μάλιστα έκαναν τη λειτουργία του πιο αργή. Αντίθετα, οι μεταβλητές γνώσης (*gbest[],pbest[]*) ήταν απαραίτητες. Ο παράγοντας *pbest[]* παρομοιάστηκε με την νοσταλγία, αφού είναι ο παράγοντας που κάνει τους agents να γυρνούν στην τοποθεσία που βρήκαν τις καλύτερες συνθήκες στο παρελθόν. Μελετήθηκε ακόμα η επίδραση που μπορεί να έχει στο τελικό αποτέλεσμα η σχέση μεγέθους της τιμής *p_increment* με την *g_increment,* για να διαπιστωθεί ότι για αρκετά μεγαλύτερο *p_increment* οι agents «πετούν» σε όλο τον χώρο λύσεων σε διάφορα σημεία ο καθένας, ενώ με αρκετά μεγαλύτερο *g_increment* συγκλίνουν όλοι γρήγορα σε ένα τοπικό ελάχιστο. Για αυτό τον λόγο οι Kennedy και Eberhart καταλήγουν να χρησιμοποιούν ίσες τιμές για τους δύο παράγοντες και συγκεκριμένα την τιμή 2.

Μετά από επιτυχημένες δοκιμές του αλγορίθμου στην εκπαίδευση ενός Νευρωνικού Δικτύου που αποτελούσε πρόβλημα 13 διαστάσεων, αποφάσισαν να αλλάξουν τον τρόπο με τον οποίο υπολογιζόταν η ταχύτητα του κάθε agent σε κάθε διάσταση. Η μέθοδος της σύγκρισης για την αλλαγή της ταχύτητας που περιγράφηκε παραπάνω θεωρήθηκε άκομψη και αργή, γιαυτό και οι ερευνητές κατέληξαν στη χρήση της παρακάτω συνάρτησης:

vx[][] = vx[][]
*+ 2 * rand * (pbestx[][] – presentx[][])*
*+ 2 * rand * (pbestx[][gbest] - presentx[][])*

Οι διπλές αγκύλες συμβολίζουν πίνακα δύο διαστάσεων (DxN) όπου στην μία είναι τοποθετημένο το πλήθος των agents (N) και στην άλλη ο αριθμός των διαστάσεων (D) του προς επίλυση προβλήματος.

Έτσι, βάσει αυτής της φόρμουλας η ταχύτητα του κάθε agent στην κάθε διάσταση εξαρτάται από τον συνδυασμό της απόστασής του από την προσωπική βέλτιστη θέση που έχει σημειωθεί στο παρελθόν και της απόστασής του από την συνολική βέλτιστη θέση που έχει βρεθεί από το σμήνος.

Στη συνέχεια θα γίνει αναφορά στις διαφορετικές παραλλαγές του αλγορίθμου που έχουν κατά καιρούς εμφανιστεί. Αυτές ως επί το πλείστον αλλάζουν τον τρόπο με τον οποίο γίνεται ο υπολογισμός των ταχυτήτων, έτσι ώστε να επιτυγχάνεται καλύτερη κάλυψη όλων των πιθανών λύσεων, για να μην εγκλωβίζεται ο αλγόριθμος σε τοπικά ελάχιστα (exploration), αλλά και να έχει μεγαλύτερη ακρίβεια στην τελική λύση (exploitation).

Μέρος Δεύτερο

Υλοποίηση Αλγορίθμου, Παραλλαγές και Συγκρίσεις

2.1 Εισαγωγή

Οι αλγόριθμοι Νοημοσύνης Σμηνών εμπνέονται από τη δράση και τα χαρακτηριστικά των κοινωνιών εντόμων, πουλιών, ψαριών και άλλων ζώων που χρησιμοποιούν την αυτό-οργάνωση και την κατανομή της εργασίας για την επίλυση προβλημάτων. Έτσι λοιπόν και ο PSO ως ένας αλγόριθμος βελτιστοποίησης Νοημοσύνης Σμηνών βασίζει την λειτουργία του στην από κοινού προσπάθεια όλου του σμήνους για εύρεση καλύτερων λύσεων, αλλά και στην ανταλλαγή της γνώσης αυτής μεταξύ των σωματιδίων που εργάζονται.

Ο PSO είναι ένας αλγόριθμος μίμησης της κοινωνικής συμπεριφοράς των πουλιών σε ένα σμήνος ή των ψαριών ενός κοπαδιού τόσο στον τρόπο που κινούνται, αλλά κυρίως στον τρόπο με τον οποίο βρίσκουν την τροφή τους και κατευθύνονται προς αυτή. Έτσι, χρησιμοποιείται με μεγάλη επιτυχία σαν αλγόριθμος βελτιστοποίησης, καθώς το εικονικό σμήνος που δημιουργεί έχει την ικανότητα να «ανακαλύπτει» τις βέλτιστες λύσεις με τρόπο ανάλογο αυτού των πουλιών.

Η προσομοίωση της συμπεριφοράς αυτής σε ηλεκτρονικό υπολογιστή μπορεί να πραγματοποιηθεί σε οποιαδήποτε γλώσσα προγραμματισμού, αρκεί να περιέχονται σε αυτή εντολές που διευκολύνουν την γραφική απεικόνιση του σμήνους σε δύο διαστάσεις. Για την πτυχιακή αυτή αρχικά ο κώδικας γράφηκε στη γλώσσα C++ και στη συνέχεια για να διευκολυνθούν οι μετρήσεις και οι απεικονίσεις, ο αλγόριθμος ξαναγράφτηκε στο περιβάλλον του Matlab.

Σημείωση: η χρήση του Matlab στον γραφή του κώδικα βοήθησε αρκετά, καθώς είναι μια γλώσσα προγραμματισμού που βασίζεται στη χρήση πινάκων και έκανε πολύ πιο εύκολο τον τρόπο δήλωσης και προσπέλασης των μεταβλητών που χρησιμοποιήθηκαν καθώς βελτίωσε εντυπωσιακά και την ταχύτητα εκτέλεσης του αλγορίθμου, αφού δεν χρειάστηκαν εμφωλευμένες επαναλήψεις.

Παρατηρώντας την ανάλυση του αλγορίθμου του PSO θα δούμε κάποιες ομοιότητες και κάποιες διαφορές σε σχέση με άλλους Εξελικτικούς Αλγορίθμους όπως οι Γενετικοί Αλγόριθμοι. Οι ΓΑ χρησιμοποιούν έναν πληθυσμό (population) λύσεων, αυτόν που στον PSO αποκαλείται σμήνος (swarm) καθώς και τα άτομα (individuals) στα οποία αντιστοιχούν τα σωματίδια (particles). Αντίθετα, στον PSO δεν χρησιμοποιούνται τεχνικές όπως η διασταύρωση (crossover) και η μετάλλαξη (mutation) για την εύρεση νέων λύσεων αλλά επιτρέπει στα σωματίδια να πετούν από λύση σε λύση. Ένας ψευδοκώδικας του αλγορίθμου του PSO ακολουθεί παρακάτω.

Αρχή

Αρχικοποίηση Θέσεων και Ταχυτήτων Σμήνους

Επανέλαβε

Αξιολόγηση

Επιλογή Καλύτερης Ολικής και Προσωπικής Θέσης

Υπολογισμός Νέων Ταχυτήτων

Υπολογισμός Νέων Θέσεων

Μέχρι το κριτήριο τερματισμού να ικανοποιηθεί

Αξιολόγηση

Επιλογή Καλύτερης Ολικής και Προσωπικής Θέσης

Στην έξοδο η καλύτερη λύση

Τέλος

Για να γίνει κατανοητός ο τρόπος που δομείται ο αλγόριθμος, παρακάτω γίνεται αναλυτική αναφορά στο πώς γράφηκε ο κώδικας στο γνωστό περιβάλλον του Matlab.

2.2 -SPSO- (Standard Particle Swarm Optimization)

Ο κλασικός αλγόριθμος Βελτιστοποίησης Σμήνους Σωματιδίων (SPSO) είναι αυτός που είχε αρχικά προταθεί το 1995 από τους Kennedy και Eberhart και ο απλούστερος αλλά και λιγότερο αποτελεσματικός από όλες τις παραλλαγές που τα επόμενα χρόνια προτάθηκαν. Αυτός ήταν και ο πρώτος που γράφηκε για την εργασία αυτή και παρουσιάζεται παρακάτω.

Δημιουργούμε ένα καινούριο m-file από το μενού του Matlab, το σώζουμε με το όνομα SPSO.m και ακολουθεί η συγγραφή του κώδικα.

- Αρχικά δημιουργούνται και ρυθμίζονται κάποιες μεταβλητές που είναι απαραίτητες για τη συνέχεια του κώδικα. Αυτές είναι: ο αριθμός των σωματιδίων που θα χρησιμοποιηθεί (part_num), ο αριθμός των διαστάσεων του προβλήματος που πρόκειται να λυθεί (dimensions) και ο αριθμός των επαναλήψεων που θα πραγματοποιηθούν για να λήξει η διαδικασία (iterations). Ακόμα αρχικοποιείται το διάνυσμα που περιέχει τις τιμές των προσωπικών βέλτιστων λύσεων (pbest), ο πίνακας με τις θέσεις που επιτεύχθηκαν οι λύσεις αυτές (pbestpos) καθώς και η τιμή του συνολικού βέλτιστου του σμήνους (gbest). Αυτές οι τιμές τοποθετούνται πολύ ψηλά ώστε πάντα στην πρώτη σύγκριση τους στον αλγόριθμο να αντικαθιστούνται από τις νέες μικρότερες τιμές. Η αρχικοποίησή τους είναι απαραίτητη καθώς η πρώτη εμφάνισή τους στον κώδικα γίνεται σε σύγκριση και όχι σε καταχώρηση.

```
part_num=10; % Particle number

dimensions=3; % Dimensions of Problem

iterations=300; % Maximum Iterations Number

% pbest, pbestpos and gbest close to +oo
pbest(1:part_num,1)=1e+200;

pbestpos(1:part_num,1:dimentions)=1e+200;

gbest=1e+200;
```

- Στη συνέχεια γίνεται μια επαναληπτική διαδικασία που επιτρέπει την εισαγωγή από το χρήστη των ορίων του χώρου αναζήτησης για κάθε διαφορετική διάσταση (limits). Αυτό επιτρέπει στον αλγόριθμο να αναζητά τις βέλτιστες λύσεις μεταξύ κάποιων ορίων, διαφορετικών για κάθε διάσταση, αποκλείοντας έτσι την έρευνα σε τόπους που ξέρουμε πως δεν παρέχουν καλές λύσεις ή που αποκλείονται λόγω περιορισμών του προβλήματος. Σε αυτό το στάδιο δημιουργούνται επίσης οι πίνακες που περιέχουν τις θέσεις των σωματιδίων (position) και τις ταχύτητές τους (speed). Επίσης φτιάχνονται διανύσματα τα οποία περιέχουν τις μέγιστες (speedmax) και ελάχιστες (speedmin) επιτρεπόμενες ταχύτητες που μπορεί να έχει ένα σωματίδιο σε κάθε διάσταση. Έχει παρατηρηθεί πως όταν οι ταχύτητες των σωματιδίων βρίσκονται μεταξύ κάποιων ορίων ο αλγόριθμος αποδίδει καλύτερα. Αυτά τα όρια εδώ έχουν τοποθετηθεί στο ±10% του εύρους των ορίων για κάθε διάσταση,

  Τονίζεται ότι οι πίνακες των ταχυτήτων και θέσεων των σωματιδίων, καθώς και όλοι οι άλλοι που εμφανίζονται στη συνέχεια είναι δομημένοι με τον εξής τρόπο: οι στήλες τους αναφέρονται στις διαστάσεις του προβλήματος, ενώ οι γραμμές τους στα σωματίδια που ψάχνουν τη λύση. Έτσι, για την επίλυση ενός προβλήματος 3 διαστάσεων με 10 σωματίδια ο πίνακας θέσεων θα είχε διαστάσεις 10x3.

```
for diastash=1:dimensions

      %Εδώ, σε κάθε επανάληψη, δηλαδή σε κάθε μία διάσταση, εμφανίζεται στην
      %οθόνη ο αριθμός της διάστασης και ο χρήστης δίνει για την διάσταση αυτή
      %τα όρια μέσα στα οποία θα γίνει η έρευνα
      display(diastash)
      limits(diastash,:)= input('Δώσε όρια διάστασης');


      %Σε αυτό το σημείο υπολογίζονται τυχαία οι αρχικές θέσεις των σωματιδίων
      %μέσα στα όρια που τοποθετούνται στο αμέσως προηγούμενο βήμα
      position(:,diastash)=
           rand(part_num,1)*(limits(diastash,2) -
           - limits(diastash,1)) + limits(diastash,1);


  %Στο επόμενο βήμα υπολογίζονται τυχαία οι αρχικές ταχύτητες των
      %σωματιδίων μέσα στο 20% του εύρους των ορίων που τέθηκαν
      speed(:,diastash)=
           rand(part_num,1)*0.2*(limits(diastash,2)-
           -limits(diastash,1))-.1*(limits(diastash,2)-
           -limits(diastash,1));
```

```
      %Εδώ με ανάλογο τρόπο υπολογίζονται τα όρια των ταχυτήτων
      speedmax(diastash)=0.1*(limits(diastash,2)-
           -limits(diastash,1));

      speedmin(diastash)=0.1*(limits(diastash,1)-
           -limits(diastash,2));

end
```

Εδώ, ας δούμε με ποιον τρόπο κατασκευάζονται οι πίνακες των τυχαίων θέσεων και ταχυτήτων και πως περιορίζονται μέσα σε κάποια όρια. Η εντολή rand(a,b) που περιλαμβάνεται στο σετ εντολών του Matlab είναι μια γεννήτρια ψευδοτυχαίων ομοιόμορφα κατανεμημένων τιμών που επιστρέφει έναν πίνακα **a** x **b** στοιχείων στο ανοικτό διάστημα (0,1). Έτσι, σε κάθε επανάληψη φτιάχνεται ένα διάνυσμα τυχαίων αριθμών από το 0 μέχρι το 1 με πλήθος ίσο με τον αριθμό των σωματιδίων στο σμήνος.

Πολλαπλασιάζοντας το τυχαίο διάνυσμα με το εύρος των ορίων που έχουν τεθεί για το συγκεκριμένο διάστημα, μέγιστο-ελάχιστο [limits(diastasi,2)-limits(diastasi,1)] μετατρέπουμε τις τυχαίες τιμές σε τιμές μεταξύ του μηδενός και της τιμής του εύρους. Στη συνέχεια προσθέτοντας το χαμηλό όριο, [limits(diastasi,1)], στις νέες τιμές του διανύσματος καταλήγουμε σε τυχαίες θέσεις positions() μεταξύ των ορίων limits() για όλα τα σωματίδια και σε όλες τις διαστάσεις.

Για την κατασκευή του πίνακα των ταχυτήτων, δεδομένου του περιορισμού που έπρεπε να υπάρχει (η ταχύτητες να βρίσκονται στο ±10% του μέγιστου εύρους των ορίων για κάθε διάσταση) φτιάχνουμε όπως και πριν ένα τυχαίο διάνυσμα με πλήθος τον αριθμό των σωματιδίων. Αυτή τη φορά πολλαπλασιάζεται με το 0.2 της τιμής του εύρους και μας δίνει τιμές από το μηδέν μέχρι το 0.2*«εύρος». Στο τέλος, αφαιρείται μια ποσότητα ίση με το 10% του εύρους, έτσι ώστε το η μέγιστη ταχύτητα να είναι η: 0.1*«εύρος», ενώ η ελάχιστη η: -0.1*«εύρος».

Για να γίνουν αυτά κατανοητά ας δούμε ένα παράδειγμα:

Έστω ότι θέλουμε να φτιάξουμε διάνυσμα 5 στοιχείων στο διάστημα (-3,5):

Η εντολή rand(5,1) δίνει:	Πολλαπλασιάζοντας με το εύρος των τιμών-στόχος: (εύρος=5-(-3)=8)	Τέλος, προσθέτοντας το χαμηλό όριο (-3):
>> pos=rand(5,1)	*>> pos=pos*8*	*>> pos=pos-3*
pos =	*pos =*	*pos =*
0.0759	*0.6072*	*-2.3928*
0.0540	*0.4320*	*-2.5680*
0.5308	*4.2464*	*1.2464*
0.7792	*6.2336*	*3.2336*
0.9340	*7.4720*	*4.4720*

Βλέπουμε λοιπόν πως έχουμε δημιουργήσει ένα διάνυσμα με τυχαίους αριθμούς στο διάστημα που επιθυμούσαμε (-3,5).

Για τη δημιουργία του διανύσματος των ταχυτήτων σε αυτό το πρόβλημα ο περιορισμός είναι το ±10% του εύρους, δηλαδή το διάστημα (-0.8,0.8):

Η εντολή rand(5,1) δίνει:	Πολλαπλασιάζουμε με το 20% του εύρους:	Αφαιρούμε το 10% του εύρους:
>> speed=rand(5,1)	*>>speed=speed*0.2*8*	*>> speed=speed -0.1*8*
speed =	*speed =*	*speed =*
0.2077	*0.3324*	*-0.4676*
0.3012	*0.4820*	*-0.3180*
0.4709	*0.7535*	*-0.0465*
0.2305	*0.3688*	*-0.4312*
0.8443	*1.3509*	*0.5509*

- Στο επόμενο κομμάτι κώδικα πραγματοποιείται το κυρίως έργο του PSO, δηλαδή η επαναληπτική διαδικασία που εξασφαλίζει τη βέλτιστη λύση μετά το τέλος των επαναλήψεων αυτών.

```
%Εδώ  γίνεται η εκκίνηση των επαναλήψεων έχοντας ως μετρητή τον cnt
for cnt=1:iterations

      %Σε αυτό το σημείο γίνεται ο έλεγχος της αποτελεσματικότητας της
      %κάθε λύσης. Καλείται η συνάρτηση test_function στην οποία παρέχονται
      %ο αριθμός των διαστάσεων (dimensions), οι θέσεις όλων των σωματιδίων
      %σε όλες τις διαστάσεις (position) που είναι ένας πίνακας μεγέθους
      % [part_num x dimensions]. Αυτή η συνάρτηση επιστρέφει έναν διάνυσμα
      %μεγέθους [part_num,1]  το οποίο δίνει τον «βαθμό ικανοποίησης» της
      %συνάρτησης για κάθε σωματίδιο (fitness value).
      eval=test_function(dimensions,position);


      %Εύρεση των προσωπικών καλύτερων τιμών και αντικατάστασή τους
      pbest=~(eval<pbest).*pbest+eval.*(eval<pbest);


      %Εύρεση και αντικατάσταση των παλιών καλύτερων προσωπικών θέσεων
      %με τις νέες και καλύτερες
      for i=1:dimentions

            pbestpos(:,i)=~(eval>pbest).*pbestpos(:,i)+
                  +position(:,i).*(eval>pbest);
      end

      %Εύρεση και αντικατάσταση της ολικής βέλτιστης τιμής και των αντίστοιχων
      %θέσεων σε όλες τις διαστάσεις
      [minim,thesi]=min(pbest);
      if minim<gbest
            gbest=minim;
            gbestpos=position(thesi,:);
      end
```

```
%Τώρα ξεκινά η επανάληψη που για κάθε διάσταση ξεχωριστά θα
%υπολογιστούν οι νέες ταχύτητες και θέσεις των σωματιδίων
for diastash=1:dimentions

      %Αυτή η γραμμή ανανεώνει την ταχύτητα του κάθε σωματιδίου στην
      %αντίστοιχη διάσταση
      speed(:,diastash) = speed(:,diastash) +
      + 2*rand(part_num,1).*
      .* (pbestpos(:,diastash)-position(:,diastash))+
      + 2 * rand(part_num,1).*
      .* (ones(part_num,1) * gbestpos(1,diastash) -
      - position(:,diastash) );


%Εδώ ρυθμίζεται η νέα ταχύτητα του κάθε σωματιδίου σε
      % περίπτωση που είναι μεγαλύτερη από την μέγιστη επιτρεπόμενη
      speed(:,diastash)=speed(:,diastash).*
      .* ~(speed(:,diastash)>speedmax(diastash))+
      + speedmax(diastash).*
      .* (speed(:,diastash)>speedmax(diastash));

      %Εδώ ρυθμίζεται η νέα ταχύτητα του κάθε σωματιδίου σε
      % περίπτωση που είναι μικρότερη από την ελάχιστη επιτρεπόμενη
      speed(:,diastash)=speed(:,diastash).*
      .*~(speed(:,diastash)<speedmin(diastash))+
      +speedmin(diastash).*
      .*(speed(:,diastash)<speedmin(diastash));


%Γίνεται η εύρεση των νέων θέσεων βάσει των νέων ταχυτήτων
      position(:,diastash)=
           position(:,diastash)+speed(:,diastash);


%Εδώ γίνεται η συγκράτηση των σωματιδίων εντός των μέγιστων
      %ορίων για κάθε διάσταση
      position(:,diastash)=position(:,diastash).*
      .*~(position(:,diastash)>limits(diastash,2))+
      +limits(diastash,2).*
      .*(position(:,diastash)>limits(diastash,2));

%Εδώ γίνεται η συγκράτηση των σωματιδίων εντός των μέγιστων
      %ορίων για κάθε διάσταση
      position(:,diastash)=position(:,diastash).*
      .*~(position(:,diastash)<limits(diastash,1))+
      +limits(diastash,1).*
      .*(position(:,diastash)<limits(diastash,1));


end
```

```
        %Στο διάνυσμα best_history αποθηκεύεται σε κάθε επανάληψη η
        %καλύτερη ολική βέλτιστη τιμή έτσι ώστε μετά το τέλος των επαναλήψεων
        %να έχουμε μία εικόνα της προόδου του αλγορίθμου
         best_history(cnt)=gbest;

end
```

Τώρα, ας συζητήσουμε κάποια από αυτά που είδαμε στον κώδικα παραπάνω. Αρχικά, τονίζεται ότι ως γλώσσα προγραμματισμού το Matlab έχει την ιδιαιτερότητα να λειτουργεί κάνοντας πράξεις μεταξύ πινάκων. Έτσι, οποιαδήποτε επαναληπτική διαδικασία που θα μπορούσε να αντικατασταθεί και να εμφανιστεί με τη μορφή πινάκων είναι προτιμότερο, για λόγους ταχύτητας εκτέλεσης, να εμφανίζεται με τη μορφή αυτή. Για αυτό το λόγο, ενώ η πρώτη έκδοση του αλγορίθμου αυτού στα πλαίσια της εργασίας δομήθηκε με εμφωλευμένες επαναλήψεις, η τελική του μορφή κάνει εκτεταμένη χρήση πράξεων μεταξύ πινάκων ώστε να επιταχυνθεί η ταχύτητα εκτέλεσης.

Συγκεκριμένα, αντί οι θέσεις και ταχύτητες να εξετάζονται ξεχωριστά για κάθε σωματίδιο και σε κάθε διάσταση, εξετάζονται και γίνονται πράξεις για όλα τα σωματίδια ταυτόχρονα για κάθε μια διάσταση.

Παράδειγμα του κώδικα με επαναλήψεις:

```
for i=1:max_iterations
     for j=1:diastaseis

          speed(i,j)=...
          position(i,j)=...

     end
end
```

Σε αυτή την περίπτωση για να κατασκευαστούν οι δύο πίνακες της ταχύτητας και των θέσεων με μέγεθος (**max_iterations** x **diastaseis**) θα θέλαμε **max_iterations*diastaseis** επαναλήψεις καθώς φτιάχνονται τα στοιχεία τους ένα προς ένα.

Στην περίπτωση των πράξεων μεταξύ πινάκων υπάρχει ένα διάνυσμα για όλα τα σωματίδια (το ένα κάτω απ' το άλλο) και χρησιμοποιείται η επανάληψη των διαστάσεων ως δείκτης της εκάστοτε στήλης του κάθε πίνακα:

```
for i=1:diastaseis

    speed(:,i)=...
    position(:,i)=...

end
```

Τώρα, ενώ οι πίνακες έχουν το ίδιο μέγεθος με πριν, οι επαναλήψεις που γίνονται είναι μόλις όσες είναι και οι διαστάσεις του προβλήματος. Υπενθυμίζεται πως το σύμβολο της άνω-κάτω τελείας αντί για ένα συγκεκριμένο αριθμό που αντιστοιχεί σε κάποια γραμμή σημαίνει πως οι πράξεις που ακολουθούν αφορούν όλες τις γραμμές του πίνακα.

Μεγάλη δυσκολία υπήρξε στην μετατροπή κάποιων διαδικασιών σε συμπιεσμένο κώδικα όσο το δυνατόν λιγότερων γραμμών. Αυτές οι διαδικασίες αφορούσαν την εύρεση του προσωπικού και ολικού βέλτιστου, τον σχηματισμού των πινάκων με τις αντίστοιχες θέσεις των βέλτιστων και τη διαδικασία της συγκράτησης των θέσεων και ταχυτήτων εντός ορίων. Παρακάτω παρουσιάζονται οι τρόποι που τελικώς ακολουθήθηκαν για να γίνουν οι απαιτούμενες συγκρίσεις και μεταθέσεις.

2.2.1 Εύρεση και αντικατάσταση προσωπικού βέλτιστου

Όπως είχε παρουσιαστεί στο προηγούμενο κεφάλαιο το κάθε πουλί από το σμήνος των Kennedy και Eberhart θυμόταν ένα στοιχείο για το προσωπικό του παρελθόν. Αυτό ήταν το καλύτερο αποτέλεσμα που είχε καταφέρει μέχρι εκείνη τη στιγμή καθώς και τη θέση στην οποία το είχε βρει. Η δημιουργία αυτής της μνήμης γίνεται με την προσθήκη ενός διανύσματος που κάθε στοιχείο του είναι το καλύτερο αποτέλεσμα που έχει φέρει το κάθε σωματίδιο. Για τη συνεχή ενημέρωση του διανύσματος αυτού είναι απαραίτητη η σύγκρισή του με το διάνυσμα που επιστρέφει η συνάρτηση test_function. Η συνεχής σύγκριση δηλαδή των προηγούμενων βέλτιστων τιμών με τις νέες τιμές που προκύπτουν ύστερα από κάθε επανάληψη και η

αντικατάστασή τους με τις νέες σε περίπτωση που αυτές είναι μικρότερες. (Μην ξεχνάμε πως σε προβλήματα βελτιστοποίησης το ζητούμενο συνήθως είναι η μικρότερη δυνατή λύση.)

Για να γίνει αυτή η σύγκριση και αντικατάσταση ακολουθήθηκε η εξής λογική:

— Καταρχήν, πρέπει να γίνει μια σύγκριση ώστε να διαπιστωθεί ποια σωματίδια έχουν μετακινηθεί προς καλύτερες θέσεις από αυτές που βρίσκονταν πριν. Αυτή έγινε με απλή σύγκριση του διανύσματος με τις προηγούμενες καλύτερες τιμές, με το διάνυσμα που φέρει τα αποτελέσματα των τωρινών θέσεων. Αυτή η σύγκριση δίνει ένα νέο διάνυσμα με λογικές τιμές (1 ή 0) που είναι το αποτέλεσμα της σύγκρισης των στοιχείων των δύο διανυσμάτων ένα προς ένα:

Για παράδειγμα φτιάχνουμε δύο διανύσματα **a** και **b** και κάνουμε τη μεταξύ τους σύγκριση. Και παίρνουμε το αποτέλεσμα:

```
>> a=[1;2;3;4;5];
>> b=[5;4;3;2;1];
>> pin=a>b
pin =
```

```
0
0
0
1
1
```

— Από τη σύγκριση (eval<pbest) παίρνουμε ένα διάνυσμα με μηδενικά και άσσους, ανάλογα με το αν η νέες τιμές (eval) είναι μικρότερες ή όχι από τις παλιές βέλτιστες (pbest). Με την χρήση του συμβόλου (~) αντιστρέφονται τα αποτελέσματα. Τελικώς, αυτό που προκύπτει ένα διάνυσμα που μας δείχνει ποια από τα στοιχεία του διανύσματος pbest θα χρειαστούν αλλαγή! Με 0 συμβολίζονται αυτά που δεν χρειάζονται αλλαγή και με 1 αυτά όπου τα αντίστοιχα σωματίδια έχουν σημειώσει πρόοδο. Πολλαπλασιάζοντας στοιχείο προς στοιχείο το αποτέλεσμα αυτό με το διάνυσμα των νέων τιμών (eval) οι τιμές που δεν χρειάζονταν αλλαγή παραμένουν μηδενικές ενώ οι άσσοι γίνονται ίσοι με την αντίστοιχη τιμή που έχει σημειώσει το κάθε σωματίδιο στην επανάληψη αυτή. Για να γίνει αυτός ο πολλαπλασιασμός χρησιμοποιείται ο συνδυασμός τελεία-επί (.*) . Με βάση το προηγούμενο παράδειγμα:

```
                    1
>> pin1=~pin.*a     2
pin1 =              3
                    0
                    0
```

— Ανάλογα, πολλαπλασιάζοντας τις προηγούμενες τιμές των προσωπικών βέλτιστων (pbest) με το μη ανεστραμμένο διάνυσμα που προκύπτει από τη σύγκριση παίρνουμε ένα νέο διάνυσμα με: τις προηγούμενες τιμές στις θέσεις των σωματιδίων που δεν χρειάζονται αλλαγή και μηδενικά στις θέσεις αυτών που χρειάζονται ανανέωση. Με βάση το προηγούμενο παράδειγμα:

```
                    0
>> pin2=pin.*b      0
pin2 =              0
                    2
                    1
```

— Τέλος, έχοντας αυτά που πρέπει να αντικατασταθούν και αυτά που δεν πρέπει να αντικατασταθούν προσθέτοντάς τα προκύπτει το τελικό αποτέλεσμα που είναι τα μικρότερα στοιχεία και από τα δύο διανύσματα, όλα μαζί σε ένα, το νέο pbest με τις καλύτερες τιμές που έχει πετύχει το κάθε σωματίδιο μέχρι στιγμής. Το τελικό αποτέλεσμα του παραδείγματος γίνεται:

```
                    1
>> apot=pin1+pin2   2
apot =              3
                    2
                    1
```

Με την τεχνική που ακολουθήθηκε παραπάνω μπορούμε να συγκρίνουμε δύο διανύσματα μεταξύ τους και σταδιακά να πάρουμε κάποια από τα στοιχεία του ενός, και σε συνδυασμό με τα αντίστοιχα στοιχεία του άλλου να φτιάξουμε ένα νέο διάνυσμα με το σύνολο των μεγαλύτερων ή μικρότερων τιμών των δύο αρχικών διανυσμάτων. Αυτή η διαδικασία μπορεί να συμπτυχθεί σε μία μόνο γραμμή κώδικα που θα πραγματοποιεί όλα τα βήματα του παραπάνω παραδείγματος:

```
>> apot=~(a>b).*a+(a>b).*b

apot =
     1
     2
     3
     2
     1
```

Τονίζεται ότι επιλέγεται η χρήση του αντιστρόφου (~) μιας ανισότητας αντί για το αντίθετο σύμβολο ανισότητας καθώς κάτι τέτοιο θα αλλοίωνε τα αποτελέσματα βάζοντας μηδενικά στις περιπτώσεις όπου δύο τιμές είναι ίσες.

2.2.2 Αντικατάσταση νέων θέσεων προσωπικού βέλτιστου

Για να «θυμούνται» τα σωματίδια τις θέσεις σε όλες τις διαστάσεις που έχουν συναντήσει το προσωπικό τους βέλτιστο, γίνεται χρήση ενός πίνακα καλύτερων θέσεων του pbestpos. Αυτός είναι διαστάσεων (**part_num** x **dimentions**) που σημαίνει ότι στις γραμμές του αντιστοιχούν τα σωματίδια ενώ στις στήλες του οι διαστάσεις του προβλήματος. Για να φτιαχτεί αυτός ο πίνακας επιστρατεύεται η προηγούμενη τεχνική της σύγκρισης και αντικατάστασης. Η βασική διαφορά αυτή τη φορά είναι ότι για κάθε στήλη (διάσταση) επαναλαμβάνεται η διαδικασία της σύγκρισης, αφού αυτή πραγματοποιείται για τα στοιχεία του διανύσματος μίας διάστασης κάθε φορά.

Αναλυτικότερα αν ξαναδούμε αυτό το κομμάτι του κώδικα θα δούμε πως χρησιμοποιείται ο μετρητής των επαναλήψεων ως δείκτης της εκάστοτε διάστασης και ότι σε κάθε επανάληψη ανανεώνεται η κάθε μία στήλη του πίνακα των καλύτερων προσωπικών θέσεων. Η σύγκριση και αντικατάσταση επιτυγχάνεται με τον ίδιο ακριβώς τρόπο που παρουσιάστηκε παραπάνω.

2.2.3 Εύρεση του ολικού βέλτιστου και αντικατάσταση των θέσεων αυτού

Για να βρεθεί το ολικό βέλτιστο σε κάθε επανάληψη γίνεται χρήση κάποιων βοηθητικών μεταβλητών. Αυτές δημιουργούνται μέσω της εντολής [minim,thesi] = min(pbest). Η συνάρτηση min() επιστρέφει τη μικρότερη τιμή που περιέχεται στο

διάνυσμα pbest και την αποθηκεύει στη μεταβλητή minim, αλλά βάζει και στη μεταβλητή thesi την θέση στην οποία βρήκε την τιμή αυτή. Έτσι, μετά τον έλεγχο του pbest με όλες τις καλύτερες τιμές όλων των σωματιδίων, επιλέγεται η βέλτιστη καθώς απομνημονεύεται και ο αριθμός του σωματιδίου που την πέτυχε.

Αυτή η καλύτερη τιμή για την κάθε επανάληψη συγκρίνεται με την ολική βέλτιστη τιμή που έχει πετύχει το σμήνος μέχρι στιγμής. Αν είναι καλύτερη, που θα σημαίνει πως το σμήνος βρήκε καλύτερη λύση για το πρόβλημα, θα πάρει τη θέση της προηγούμενης βέλτιστης τιμής και στο διάνυσμα όπου σώζονται οι αντίστοιχες καλύτερες θέσεις θα μπουν οι νέες. Αυτές θα βρεθούν βάσει της θέσης του καλύτερου σωματιδίου με αριθμό: thesi.

2.2.4 Πίνακες ταχυτήτων και θέσεων

Στη συνέχεια πρέπει να ανανεωθούν οι πίνακες των ταχυτήτων και των θέσεων του κάθε σωματιδίου και για κάθε διάσταση. Οι πίνακες αυτοί έχουν διαστάσεις (**part_num** x **dimensions**). Έχουν δηλαδή τη μορφή:

Σωματίδια\Διάσταση	$1^η$ διάσταση	$2^η$ διάσταση	$3^η$ διάσταση
$1^ο$ σωματίδιο	Θέση (1,1)	Θέση (1,2)	Θέση (1,3)
$2^ο$ σωματίδιο	Θέση (2,1)	Θέση (2,2)	Θέση (2,3)
$3^ο$ σωματίδιο	Θέση (3,1)	Θέση (3,2)	Θέση (3,3)

Πίνακας 1: Υπόδειγμα πίνακα ταχυτήτων και θέσεων

Ξεκινώντας η επανάληψη από την πρώτη διάσταση, διαμορφώνονται αρχικά τις ταχύτητες με βάση την πρώτη εξίσωση των Kennedy και Eberhart. Αυτή λέει απλά ότι η νέα ταχύτητα δημιουργείται προσθέτοντας την παλιά ταχύτητα με την απόσταση του σωματιδίου από την βέλτιστη προσωπική θέση και την απόστασή του από την ολική βέλτιστη θέση. Βέβαια, για την προσθήκη της στοχαστικότητας στον αλγόριθμο οι αποστάσεις αυτές είναι πολλαπλασιασμένες με ένα τυχαίο παράγοντα μεταξύ 0 και 1.

Η εξίσωση των ταχυτήτων βάσει των Kennedy και Eberhart όπως παρουσιάστηκε παραπάνω είναι η:

vx[][] = vx[][]
*+ 2 * rand * (pbestx[][] – presentx[][])*
*+ 2 * rand * (pbestx[][gbest] - presentx[][])*

Ενώ στον αλγόριθμο που γράφηκε για να γίνονται οι πράξεις μεταξύ διανυσμάτων σε κάθε μία διάσταση ξεχωριστά η αντίστοιχη εξίσωση είναι η:

```
speed(:,diastash) = speed(:,diastash)
          + 2*rand(part_num,1)
          .* (pbestpos(:,diastash)-position(:,diastash))
          + 2*rand(part_num,1)
          .* (ones(part_num,1) * gbestpos(1,diastash)
          - position(:,diastash) );
```

Αναλύοντας τον κώδικα αυτόν, βλέπουμε να κατασκευάζεται το διάνυσμα που περιέχει τις ταχύτητες όλων των σωματιδίων (χρήση άνω-κάτω τελείας) για μία μόνο διάσταση, αυτήν με τον αριθμό: diastasi.

Για να βρεθεί η απόσταση της παρούσας θέσης των σωματιδίων από την προσωπική βέλτιστη επιλέγονται από τους πίνακες των ταχυτήτων τα διανύσματα αυτά που αντιστοιχούν στην εκάστοτε διάσταση και αφαιρούνται μεταξύ τους ένα προς ένα, όπως ορίζει η πράξη της αφαίρεσης μεταξύ πινάκων. Κατασκευάζεται επίσης με την εντολή rand() ένα νέο διάνυσμα με τόσα στοιχεία όσα είναι και τα σωματίδια, γεμάτο με τυχαίους αριθμούς. Αυτοί, διπλασιασμένοι, πολλαπλασιάζουν ένα προς ένα τα στοιχεία που προκύπτουν από την προηγούμενη αφαίρεση. Στη συνέχεια πρέπει να γίνει η αντίστοιχη πράξη μεταξύ των θέσεων των σωματιδίων και του ολικού βέλτιστου. Καθώς όμως η θέση του ολικού βέλτιστου είναι ένα μόνο σημείο για κάθε διάσταση, δεν είναι δυνατή η άμεση αφαίρεσή του από ένα διάνυσμα. Για να αντιμετωπιστεί αυτή η δυσκολία χρησιμοποιείται η συνάρτηση ones(). Αυτή δημιουργεί ένα διάνυσμα στο μέγεθος που επιθυμούμε (όσα είναι και τα σωματίδια) γεμάτο από άσσους. Πολλαπλασιάζοντας έτσι αυτό το διάνυσμα με τη θέση του ολικού βέλτιστου δημιουργείται ένα νέο διάνυσμα από το οποίο πλέον μπορεί να αφαιρεθεί το διάνυσμα των θέσεων. Το αποτέλεσμα της διαίρεσης πολλαπλασιάζεται όπως και πριν με τυχαία φτιαγμένους αριθμούς. Το άθροισμα όλων

αυτών των διανυσμάτων (παλαιές ταχύτητες, αποστάσεις από προσωπικά βέλτιστα και αποστάσεις από ολικά βέλτιστα) δίνει το τελικό αποτέλεσμα που δεν είναι άλλο από τις νέες ταχύτητες.

Αυτό που ακολουθεί τους παραπάνω υπολογισμούς είναι η σύγκριση των αποτελεσμάτων με τις οριακές ταχύτητες που έχουν υπολογιστεί νωρίτερα. Η μέθοδος της σύγκρισης και αντικατάστασης που ακολουθείται είναι η ίδια με αυτή που αναλύθηκε παραπάνω. Επαναλαμβάνεται η ίδια διαδικασία σε κάθε διάσταση τόσο για τον περιορισμό των ελάχιστων όσο και για τον περιορισμό των μέγιστων ταχυτήτων (speedmin - speedmax).

Απλά προσθέτοντας στις προηγούμενες θέσεις τις ταχύτητες που μόλις υπολογίστηκαν, όπως ορίζει η βασική έκδοση του αλγορίθμου, προκύπτουν οι επόμενες θέσεις. Αυτές θα συγκριθούν με τα κατώτερα και τα ανώτερα όρια (limits(:,1) και limits(:,2) αντίστοιχα) για να περιοριστούν τα σωματίδια εντός του χώρου αναζήτησης.

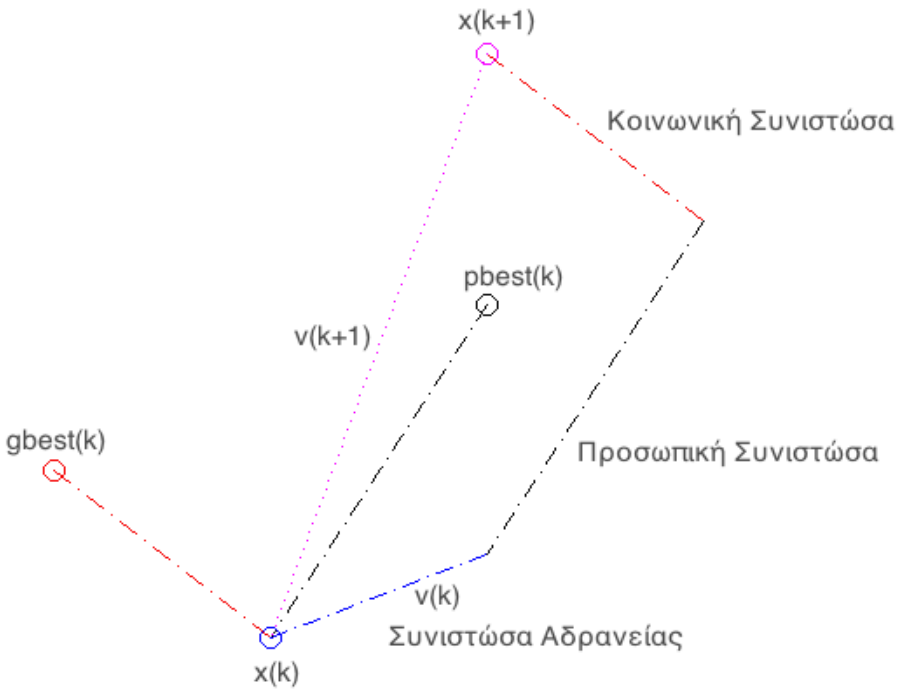
Τέλος, με τη λήξη κάθε επανάληψης καταγράφεται σε μία μεταβλητή με όνομα best_history η τιμή της βέλτιστης μέχρι στιγμής λύσης. Αυτό εξυπηρετεί στο να γίνει μετά τη λήξη του αλγορίθμου μία συνολική αξιολόγηση της απόδοσής του. Θα χρησιμοποιηθεί κυρίως στη συνέχεια για τη σύγκριση διαφορετικών παραλλαγών του αλγορίθμου.

2.3 Ανάλυση της Μεθόδου

Με τη γνώση του τρόπου με τον οποίο έχει δομηθεί ο αλγόριθμος μπορούν πλέον να εξαχθούν κάποια συμπεράσματα για τον τρόπο με τον οποίο κινούνται τα σωματίδια κατά τη διάρκεια της επίλυσης του προβλήματος αλλά και να εντοπιστούν οι παράγοντες που επηρεάζουν την αποτελεσματικότητα του αλγορίθμου. Για να γίνει πλήρως κατανοητός ο τρόπος κίνησης των σωματιδίων, ακολουθεί ένα παράδειγμα σε περιβάλλον (πρόβλημα) δύο διαστάσεων με ένα μόνο σωματίδιο.

2.3.1 Κίνηση Σωματιδίου

Στο παρακάτω σχήμα παρουσιάζεται το σύνολο των παραγόντων που επηρεάζουν την κίνηση του σωματιδίου από την παλιά του θέση x(k) στην καινούρια x(k+1). Αυτοί οι παράγοντες όπως ορίζει η εξίσωση της ταχύτητας είναι: η απόσταση από το ολικό βέλτιστο (Κοινωνική Συνιστώσα), η απόσταση από το προσωπικό βέλτιστο (Προσωπική Συνιστώσα) και η προηγούμενη ταχύτητα του σωματιδίου. Προσθέτοντας τα διανύσματα αυτά (στην πράξη στον αλγόριθμο πολλαπλασιάζονται με τυχαίους συντελεστές) προκύπτει το διάνυσμα της νέας ταχύτητας και κατά συνέπεια η νέα θέση του σωματιδίου.



Εικόνα 3: Σχηματική αναπαράσταση του τρόπου υπολογισμού των ταχυτήτων

Για να γίνει πλήρως κατανοητός ο τρόπος με τον οποίο τα σωματίδια κινούνται στον χώρο, γίνονται κάποιες αναλογίες. Συγκεκριμένα, η ταχύτητα των σωματιδίων συσχετίζεται με την ταχύτητα ενός σώματος που ορίζεται ως η διαφορά της παρούσας θέσης από την προηγούμενη δια τον χρόνο:

$v = \frac{dx}{dt}$ και στο διακριτό επίπεδο του PSO:
$v(k) = x(k) - x(k-1)$ και επίσης: $v(k+1) = x(k+1) - x(k)$
άρα η επόμενη θέση θα είναι : $x(k+1) = x(k) + v(k+1)$

Επίσης, η επιτάχυνση που ευθύνεται για αλλαγές στην ταχύτητα των σωμάτων μπορεί να αντιστοιχιστεί με μία επιτάχυνση που οφείλεται για την αλλαγή της ταχύτητας των σωματιδίων.

$a = \frac{dv}{dt}$ και στο διακριτό επίπεδο του PSO:
$a(k) = v(k) - v(k-1)$ και επίσης: $a(k+1) = v(k+1) - v(k)$
άρα, η επόμενη ταχύτητα θα είναι: $v(k+1) = v(k) + a(k+1)$

Από τον αλγόριθμο, και συγκεκριμένα από την εξίσωση που μας δίνει την νέα ταχύτητα των σωματιδίων, γίνεται αντιληπτό ότι η επιτάχυνση των σωματιδίων ταυτίζεται με το άθροισμα της κοινωνικής και προσωπικής συνιστώσας. Έτσι, μπορούμε να πούμε ότι η «κινητήρια δύναμη» για την κίνηση των σωματιδίων είναι η απόσταση της παρούσας θέσης τους από την θέση του προσωπικού και ολικού βέλτιστου. Γίνεται κατανοητό ότι η ταχύτητα και η επιτάχυνση ενός σωματιδίου θα γίνουν μηδενικές όταν και μόνο όταν οι θέσεις του ολικού και προσωπικού βέλτιστου ταυτιστούν και είναι ίδιες με την παρούσα θέση.

Σαν παράδειγμα, ας θεωρήσουμε πως οι θέσεις των δύο βέλτιστων ενός σωματιδίου ταυτίζονται. Καθώς το σωματίδιο κινείται με μία ταχύτητα προς το σημείο αυτό, η απόσταση μειώνεται, και άρα η επιτάχυνση προς την κατεύθυνση αυτή μειώνεται. Όταν το σωματίδιο περάσει «πάνω» από το σημείο και αρχίσει να απομακρύνεται προς την άλλη πλευρά λόγω κεκτημένης ταχύτητας, η απόσταση θα αρχίσει να μεγαλώνει, η επιτάχυνση θα γίνει αρνητική (επιβράδυνση) και το

σωματίδιο θα σταματήσει. Στη συνέχεια θα επιταχυνθεί ξανά προς το σημείο και ου το καθεξής. Αυτή η συμπεριφορά θυμίζει έντονα ταλάντωση.

Μετά από την ανάλυση της κίνησης των σωματιδίων στον χώρο αναζήτησης θα εξεταστούν οι παράμετροι που επηρεάζουν την αποτελεσματικότητα του αλγορίθμου. Η αποτελεσματικότητα αυτή αναφέρεται τόσο στην ταχύτητα επίλυσης των προβλημάτων, όσο και στην ακρίβεια εύρεσης της βέλτιστης λύσης. Οι παράμετροι που παρουσιάζονται παρακάτω είναι το μέγεθος του σμήνους, οι μέγιστες επιτρεπόμενες ταχύτητες, ο αριθμός των επαναλήψεων και οι συντελεστές επιτάχυνσης.

2.3.2 Μέγεθος σμήνους

Το μέγεθος του σμήνους αφορά στον συνολικό αριθμός των σωματιδίων που χρησιμοποιούνται για την εύρεση της λύσης. Ένας μεγάλος αριθμός ευνοεί την μεγάλη διασπορά τους στο χώρο των λύσεων κάτι που βοηθάει στην παράλληλη εξέταση πολλών πιθανών λύσεων. Από την άλλη, ένας συνεχώς αυξανόμενος αριθμός δεν εγγυάται την εύρεση καλύτερων λύσεων, ενώ επιφέρει σημαντικές καθυστερήσεις στην ταχύτητα επίλυσης του αλγορίθμου λόγω αυξημένης πολυπλοκότητας.

2.3.3 Μέγιστες επιτρεπόμενες ταχύτητες

Οι ταχύτητες αυτές υπολογίζονται με βάση το εύρος του χώρου αναζήτησης και είναι για κάθε μία διάσταση ξεχωριστά, αφού κάθε διάσταση μπορεί να έχει διαφορετικά όρια από τις άλλες. Είναι ένας απαραίτητος περιορισμός που κάνει τα σωματίδια να πετούν μεταξύ κοντινών λύσεων αντί να «διακτινίζονται» σε μακρινές θέσεις. Με αυτόν τον τρόπο αναγκάζονται να εξετάζουν και περιοχές πιθανώς καλών λύσεων, τις οποίες αλλιώς θα παρέκαμπταν.

2.3.4 Αριθμός Επαναλήψεων

Ο μέγιστος αριθμός των επαναλήψεων που θα εκτελέσει ο αλγόριθμος αποτελεί ένα από τα δύο κριτήρια τερματισμού του αλγορίθμου. Το άλλο κριτήριο τερματισμού είναι η επίτευξη ενός αρκετά ικανοποιητικού αποτελέσματος. Ο μέγιστος αριθμός επαναλήψεων καθορίζεται από τον χρήστη και έχει άμεση σχέση με

την πολυπλοκότητα του προβλήματος, με τις μέγιστες ταχύτητες αλλά και με το μέγεθος του σμήνους. Σε πολυδιάστατα προβλήματα που πρέπει να εξεταστεί μεγάλος αριθμός πιθανών λύσεων πρέπει να προσφερθεί στον αλγόριθμο αρκετός χρόνος για να μπορέσει να κατευθυνθεί προς τη βέλτιστη λύση. Σε αντιστοιχία, όταν το μέγεθος του σμήνους είναι μικρό ή οι μέγιστες ταχύτητες χαμηλές, πάλι ο αλγόριθμος πρέπει να κάνει περισσότερες προσπάθειες. Από την άλλη, ένας μεγάλος αριθμός επαναλήψεων δεν βοηθάει πάντα στην επίτευξη ακόμα καλύτερης λύσης καθώς ο PSO είναι πολλές φορές ευαίσθητος σε τοπικά ελάχιστα. Αυτό σημαίνει πως είναι πιθανό το σμήνος να «κολλήσει» σε κάποιο μη βέλτιστο σημείο του χώρου λύσεων. Σε αυτή την περίπτωση, περισσότερες επαναλήψεις δεν θα βοηθήσουν.

2.3.5 Συντελεστές Επιτάχυνσης

Οι συντελεστές επιτάχυνσης είναι οι αριθμοί *p_increment* και *g_increment* που παρουσιάστηκαν στο 1^{ο} Μέρος. Αυτοί όπως είδαμε είναι συνήθως ίσοι και έχουν την τιμή 2. Αυξάνοντας όμως τον ένα συντελεστή εις βάρος του άλλου βλέπουμε διαφορετικές συμπεριφορές κατά την εκτέλεση του αλγορίθμου. Έτσι λοιπόν, μεγάλο *p_increment* θα οδηγήσει την έρευνα σε όλο τον χώρο αναζήτησης καθώς κάθε σωματίδιο θα δρα σχεδόν αυτόνομο επηρεαζόμενο ελάχιστα από τα άλλα, δηλαδή από την θέση του ολικού βέλτιστου. Στην αντίθετη περίπτωση τα σωματίδια θα κινούνται προς την θέση του ολικού βέλτιστου ξεχνώντας το προσωπικό. Αυτές οι διαφορές στην κίνηση του σμήνους οδήγησαν στην δημιουργία νέων αλγορίθμων που προσαρμόζουν αυτές τις τιμές στην πορεία της εξέλιξης του αλγορίθμου επιτυγχάνοντας καλύτερα αποτελέσματα.

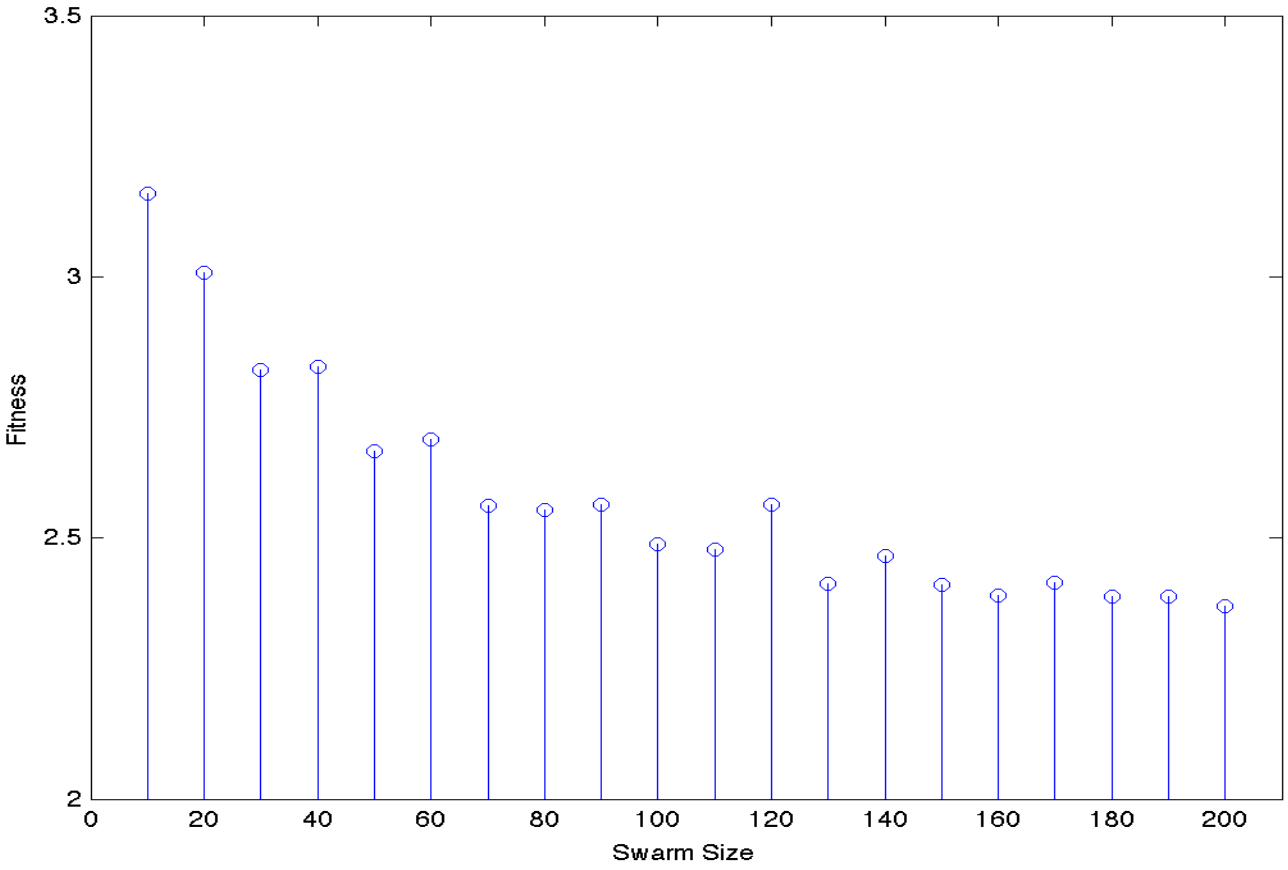
2.3.6 Δοκιμές και Μετρήσεις

Για να γίνει κατανοητή η επίδραση του μεγέθους του σμήνους και ο αριθμός των επαναλήψεων στο τελικό αποτέλεσμα του αλγορίθμου, παρακάτω παρουσιάζονται τα αποτελέσματα κάποιων συγκρίσεων.

Οι μετρήσεις έγιναν για τον αλγόριθμο SPSO που παρουσιάστηκε παραπάνω. Σκοπός ήταν η λύση της συνάρτησης του Griewangk σε 18 διαστάσεις. Μετά τη λήξη των καθορισμένων επαναλήψεων, τα εκάστοτε αποτελέσματα (ποιότητα λύσης - fitness) συγκρίνονται στις παρακάτω εικόνες.

Στην πρώτη περίπτωση χρησιμοποιήθηκαν σμήνη από 10 μέχρι 200 σωματιδίων και ο αριθμός των επαναλήψεων ήταν 1500. Στη δεύτερη, έγινε προσπάθεια επίλυσης του ίδιου προβλήματος αλλά αυτή τη φορά ενώ το μέγεθος του σμήνους ήταν σταθερό στα 80 σωματίδια, οι επαναλήψεις αυξάνονταν από 100 και φτάνουν τις 2000. Και στις δύο περιπτώσεις, παρουσιάζεται ο μέσος όρος των αποτελεσμάτων 30 προσπαθειών.

Παρακάτω φαίνονται τα αποτελέσματα των μετρήσεων για διαφορετικό μέγεθος σμήνους.

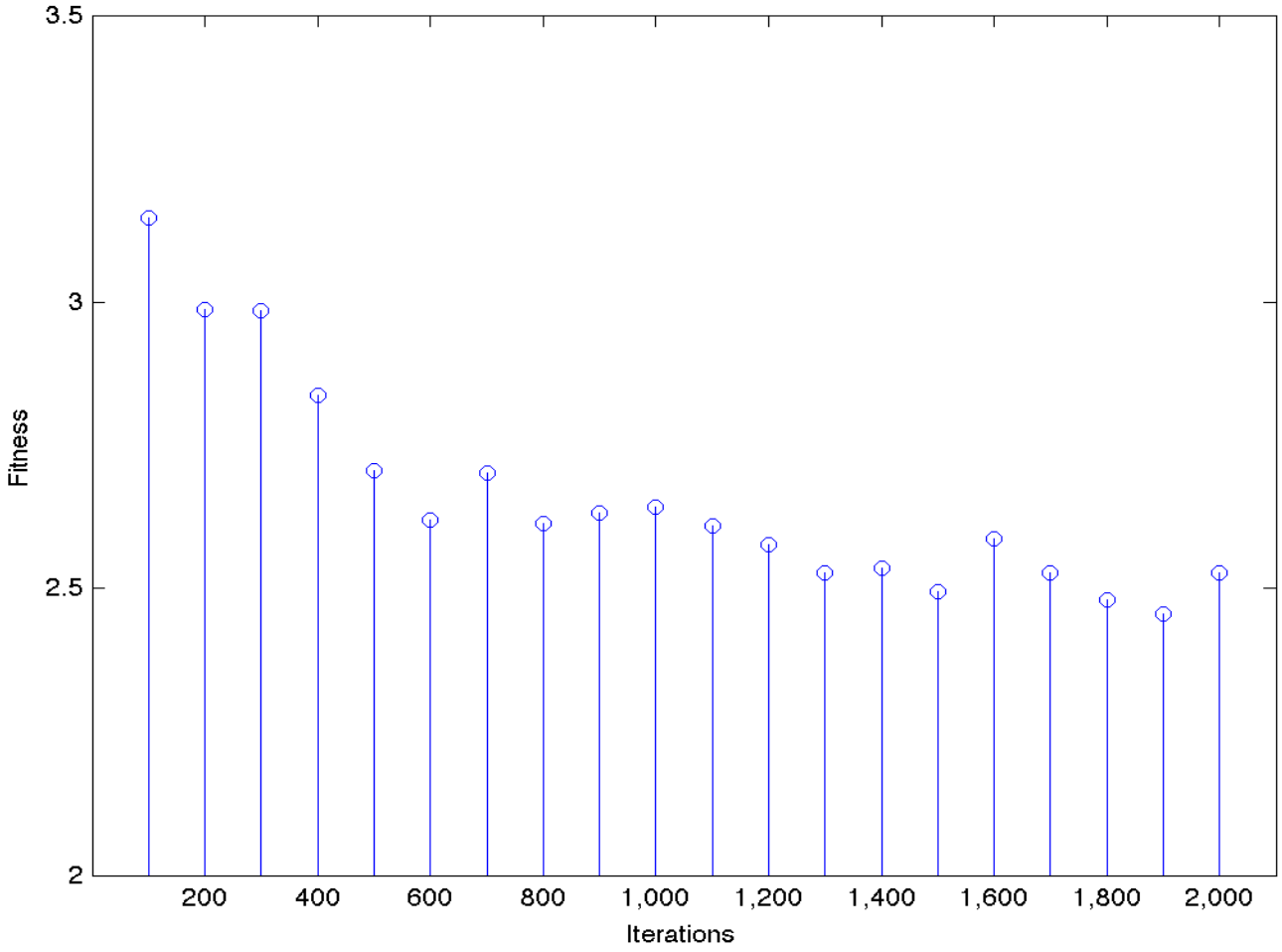


Εικόνα 4: Σύγκριση απόδοσης για διαφορετικό αριθμό σωματιδίων

Φαίνεται πως υπάρχει μια σημαντική βελτίωση της απόδοσης του αλγορίθμου για περισσότερα από 10 σωματίδια, αλλά η βελτίωση αυτή δεν είναι αναλογική του αριθμού των σωματιδίων. Έτσι, για σμήνη από 50 σωματίδια και πάνω, η βελτίωση

δεν είναι αξιοσημείωτη και ενδεχομένως στις περιπτώσεις όπου το ζητούμενο είναι η ταχύτητα εκτέλεσης, η χρήση περισσότερων σωματιδίων είναι ανώφελη.

Παρακάτω παρουσιάζονται τα αποτελέσματα του αλγορίθμου για διαφορετικό αριθμό επαναλήψεων.



Εικόνα 5: Σύγκριση απόδοσης για διαφορετικό αριθμό επαναλήψεων

Εδώ, όπως αναμενόταν, περισσότερες επαναλήψεις βοηθούν μεν την ακρίβεια του αλγορίθμου, αλλά όπως φαίνεται από τις 600 επαναλήψεις και μετά η βελτίωση είναι μικρή σε σχέση με την αύξηση του χρόνου εκτέλεσης.

Αφού μελετήθηκαν οι παράγοντες που επηρεάζουν τη λειτουργία του PSO και αναλύθηκε ο αλγόριθμος εκτέλεσης του SPSO, παρακάτω θα γίνει μια αναφορά σε παραλλαγές του αλγορίθμου.

2.4 -APSO- (Adaptive Particle Swarm Optimization)

2.4.1 Συντελεστής Αδρανείας (Inertia Weight)

Κάποιες παραλλαγές του Κλασικού αλγορίθμου του PSO ξεκίνησαν να παρουσιάζονται κιόλας από το 1998 με τους Shi και Eberhart να προτείνουν την πρώτη προσαρμοστική εκδοχή (Adaptive).

Σύμφωνα με τους ερευνητές ο όρος της προηγούμενης ταχύτητας κατά τον υπολογισμό της νέας ταχύτητας στην κλασική έκδοση του αλγορίθμου αποτελεί ένα είδος αδράνειας του σωματιδίου. Πραγματικά, η ύπαρξη του όρου αυτού στην εξίσωση κάνει τα σωματίδια λιγότερο «υπάκουα» σε απότομες αλλαγές κατευθύνσεων. Όπως παρατήρησαν, αυτή η συμπεριφορά είναι ιδιαιτέρως χρήσιμη στην εκκίνηση του αλγορίθμου καθώς τα σωματίδια παρασύρονται από το ίδιο τους το «βάρος» και εξερευνούν περιοχές που αλλιώς δεν θα εξερευνούσαν. Έτσι εξετάζεται αποτελεσματικά μεγάλο μέρος του χώρου λύσεων. Στη συνέχεια όμως και όσο τα σωματίδια συγκεντρώνονται γύρω από μία λύση η αδράνεια αυτή λειτουργεί ανασταλτικά στην εύρεση λεπτομερούς λύσης. Αντίθετα, μικρότερη αδράνεια προς το τέλος του αλγορίθμου βοηθά στην λεπτομερή αναζήτηση της βέλτιστης λύσης.

Αυτό λοιπόν που προτάθηκε ήταν η χρήση ενός προσαρμοζόμενου συντελεστή βαρύτητας που θα ξεκινούσε με μια μεγάλη αρχικά τιμή η οποία με τη πάροδο των επαναλήψεων θα μειωνόταν σε μια τιμή που προσέγγιζε το μηδέν. Έτσι, οι ερευνητές κατέληξαν στην χρήση ενός συντελεστή w, του οποίου η τιμή μειώνεται γραμμικά κατά το πέρας των επαναλήψεων.

Η εκδοχή αυτή του αλγορίθμου έχει γίνει ιδιαιτέρως δημοφιλής αφού είναι πολύ απλή στην εκτέλεση και εντυπωσιακά αποτελεσματική. Εμφανίζει σημαντική βελτίωση σε σχέση με τον κλασικό αλγόριθμο και συναντάται με την μορφή:

$$
\begin{aligned}
vx[][] = {} & w * vx[][] \\
& + 2 * rand * (pbestx[][] - presentx[][]) \\
& + 2 * rand * (pbestx[][gbest] - presentx[][])
\end{aligned}
$$

Για να επιτευχθεί η γραμμική μείωση του συντελεστή αδράνειας w, γίνεται ο υπολογισμός αυτού σε κάθε επανάληψη. Ο προσαρμοζόμενος συντελεστής w υπολογίζεται συνεχώς και σε κάθε επανάληψη βάσει της εξίσωσης ευθείας:

$$w = \boldsymbol{firstval} - \frac{\boldsymbol{firstval} - \boldsymbol{finalval}}{\boldsymbol{maxiterations}} * \boldsymbol{iteration}$$

Όπου *firstval* είναι η τιμή του w στην αρχή του αλγορίθμου, *finalval* είναι η τιμή του στο τέλος, *maxiterations* είναι το σύνολο των επαναλήψεων και *iteration* είναι ο αριθμός της τρέχουσας επανάληψης.

Με αυτόν τον τρόπο, ο παραπάνω συντελεστής w ξεκινά από μια αρχική τιμή firstval και με γραμμικό τρόπο ως προς τον αριθμό των επαναλήψεων καταλήγει στην τελική τιμή finalval. Ο υπολογισμός της τιμής του συντελεστή w μπορεί να υλοποιηθεί μέσα στον αλγόριθμο με δύο τρόπους: ταυτόχρονα με τον υπολογισμό της ταχύτητας των σωματιδίων για κάθε διάσταση σε κάθε επανάληψη ή καλύτερα, σε κάθε επανάληψη για όλες τις διαστάσεις και σωματίδια. Αυτές οι δύο υλοποιήσεις φαίνονται στα δύο παρακάτω αποσπάσματα του αλγορίθμου:

Υπολογισμός της αδράνειας σε κάθε επανάληψη και για κάθε διάσταση

Αρχή

Αρχικοποίηση Θέσεων και Ταχυτήτων Σμήνους

Επανέλαβε

Επανέλαβε για διαστάσεις

Αξιολόγηση

Επιλογή Καλύτερης Ολικής και Προσωπικής Θέσης

Υπολογισμός Αδράνειας και Νέων Ταχυτήτων

Υπολογισμός Νέων Θέσεων

Τέλος Διαστάσεων

Μέχρι το κριτήριο τερματισμού να ικανοποιηθεί

Αξιολόγηση

Επιλογή Καλύτερης Ολικής και Προσωπικής Θέσης

Στην έξοδο η καλύτερη λύση

Τέλος

Υπολογισμός της αδράνειας σε κάθε επανάληψη μόνον

Αρχή

Αρχικοποίηση Θέσεων και Ταχυτήτων Σμήνους

Επανέλαβε

Υπολογισμός Αδράνειας

Επανέλαβε για διαστάσεις

Αξιολόγηση

Επιλογή Καλύτερης Ολικής και Προσωπικής Θέσης

Υπολογισμός Νέων Ταχυτήτων

Υπολογισμός Νέων Θέσεων

Τέλος Διαστάσεων

Μέχρι το κριτήριο τερματισμού να ικανοποιηθεί

Αξιολόγηση

Επιλογή Καλύτερης Ολικής και Προσωπικής Θέσης

Στην έξοδο η καλύτερη λύση

Τέλος

Οι παραπάνω αλγόριθμοι μεταφέρονται σε κώδικα Matlab στις παρακάτω γραμμές. Πρώτα, παρουσιάζεται η μέθοδος όπου η αδράνεια υπολογίζεται σε κάθε επανάληψη του αλγορίθμου στην γραμμή του κώδικα όπου υπολογίζονται και οι νέες ταχύτητες:

```
% Υπολογισμός της αδράνειας σε κάθε επανάληψη και για κάθε διάσταση
speed(:,diastash)=
          (firstval-((firstval-finalval)/iterations)*cnt)
          .*speed(:,diastash) + 2*rand(part_num,1)……
```

Εναλλακτικά, για την αποφυγή του άσκοπου υπολογισμού της αδράνειας «επαναλήψεις*διαστάσεις» φορές αλλά μόνο τόσες φορές, όσες είναι και οι επαναλήψεις :

```
% Υπολογισμός της ¨αδράνειας¨ μία μόνο φορά για κάθε επανάληψη
inertia=(firstval-((firstval-finalval)/iterations)*cnt);

for diastash=1:dimentions
     % Χρήση της αδράνειας σε κάθε διάσταση
     speed(:,diastash) = inertia .* speed(:,diastash)
          +2*rand(part_num,1)……
```

Με την πρώτη παρουσίαση του αλγορίθμου APSO από τους Shi και Eberhart έγινε σαφές ότι οι τιμές firstval και finalval επηρεάζουν σημαντικά την αποτελεσματικότητα του αλγορίθμου. Έτσι, μετά από διερεύνηση οδηγήθηκαν στην πρόταση δύο τιμών, των firstval = 0.9 και finalval = 0.4. Παρατηρώντας την υπάρχουσα βιβλιογραφία, αυτές οι δύο τιμές φαίνονται να κυριαρχούν. Έτσι, η εξίσωση υπολογισμού της αδράνειας διαμορφώνεται ως εξής:

$$w = \mathbf{0.9} - \frac{\mathbf{0.5}}{\boldsymbol{maxiterations}} * \boldsymbol{iteration}$$

Οι παράγοντες που επηρεάζουν την αποτελεσματικότητα του APSO όπως θα χρησιμοποιούνται στο εξής στην παρούσα εργασία συνοψίζονται εδώ:

Παράγοντας	**Τιμή / Εύρος**
r1, r2	Τυχαία τιμή [0 1] Κανονική Κατανομή
c1 = c2	2 Σταθερή τιμή
w	[0.9 0.4] Γραμμικά φθίνουσα

Πίνακας 2: Τιμές παραγόντων του APSO που χρησιμοποιούνται στην εργασία

Όπως έχει αναφερθεί και παραπάνω, η παραλλαγή αυτή του βασικού αλγορίθμου παρουσιάζει εντυπωσιακή βελτίωση των επιδόσεων. Έτσι, ο Adaptive PSO αυξάνοντας ελάχιστα μόνο την πολυπλοκότητα και τον χρόνο εκτέλεσης έχει πλέον γίνει σημείο αναφοράς και σύγκρισης του PSO.

2.4.2 Μετρήσεις και Συγκρίσεις

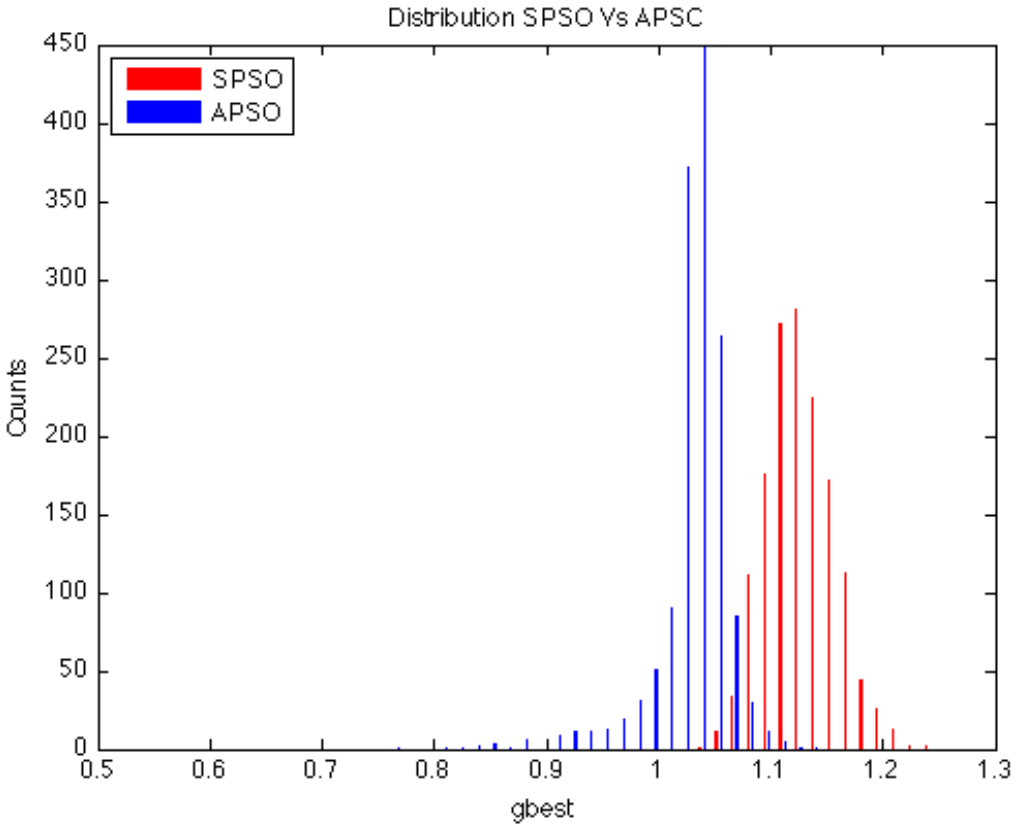
Για να γίνουν κατανοητά τα πλεονεκτήματα της συγκεκριμένης παραλλαγής, παρακάτω γίνεται άμεση σύγκριση των δύο αλγορίθμων με χρίση της συνάρτησης

Griewangk. Συγκεκριμένα, μετριέται η σύγκλιση (gbest) των δύο μεθόδων επιλύοντας την συνάρτηση Griewangk 20 διαστάσεων με 20 σωματίδια και σε δυο διαφορετικές περιπτώσεις: με 150 και 50 επαναλήψεις. Ο κάθε αλγόριθμος εξετάζεται 1500 φορές και υπολογίζεται ο μέσος όρος. Τα αποτελέσματα φαίνονται παρακάτω:

Επαναλήψεις	**SPSO**	**APSO**
50	1.1266	1.0285
150	1.0823	0.2153

Πίνακας 3: Μέση σύγκλιση SPSO και APSO σε 50 και 150 επαναλήψεις

Παρατηρώντας τα αποτελέσματα των μετρήσεων εντύπωση προκαλεί το εξής γεγονός: όταν οι επαναλήψεις είναι λίγες, οι δύο αλγόριθμοι διαφέρουν ελάχιστα μεταξύ τους. Όταν όμως οι επαναλήψεις γίνουν περισσότερες, ο APSO επιδεικνύει εντυπωσιακά καλύτερη συμπεριφορά, ενώ η απόδοση του SPSO βελτιώνεται ανεπαίσθητα. Αυτή η παρατήρηση φαίνεται γραφικά στις παρακάτω κατανομές των αποτελεσμάτων από τις 1500 μετρήσεις.

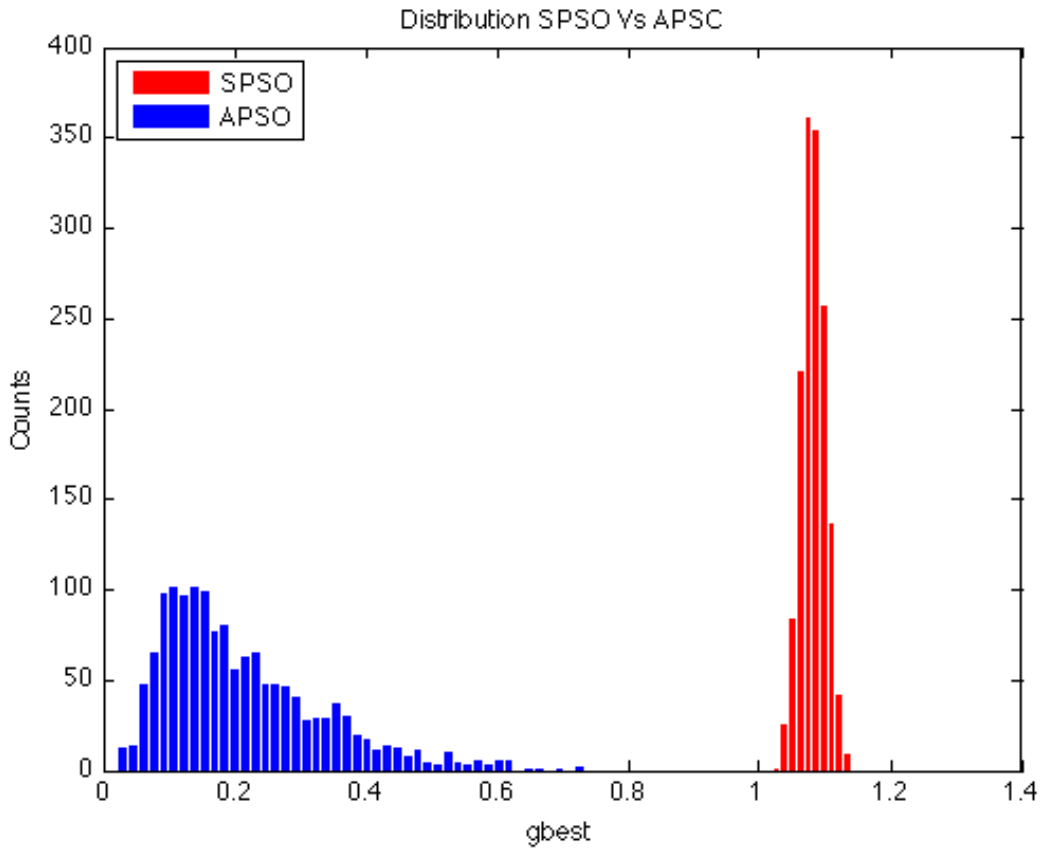


Εικόνα 6: Κατανομή της σύγκλησης 1500 δοκιμών με 50 επαναλήψεις

Στην προηγούμενη εικόνα φαίνεται ότι ο SPSO (κόκκινο) είναι ελαφρώς χαμηλότερης απόδοσης από τον APSO (μπλε). Επίσης φαίνεται ότι ο SPSO εμφανίζει μια κανονική κατανομή, ενώ ο ASPSO τείνει να επεκτείνεται προς καλύτερες λύσεις.

Στην επόμενη γραφική εμφανίζεται η εντυπωσιακή βελτίωση της ποιότητας των λύσεων που δίνει ο APSO σε σχέση με τον SPSO. Ενώ η κόκκινη κατανομή έχει πλησιάσει ελάχιστα προς την αρχή των αξόνων, η μπλε έχει πλησιάσει εντυπωσιακά. Ενδιαφέρον εμφανίζεται και στο γεγονός ότι λόγω της τυχαιότητας που χαρακτηρίζει τον PSO η κατανομή του APSO απέχει πολύ από την κανονική.

Από την παραπάνω σύγκριση αποδεικνύεται η ποιοτική ανωτερότητα του APSO. Η ποιότητα των λύσεων του APSO φαίνεται να βελτιώνεται συνεχώς με την αύξηση των επαναλήψεων του αλγορίθμου, σε αντίθεση με τον SPSO ο οποίος φαίνεται να «κολλά» εύκολα σε τοπικά ελάχιστα.



Εικόνα 7: Κατανομή της σύγκλησης 1500 δοκιμών με 150 επαναλήψεις

2.4.3 Διερεύνηση μη γραμμικών Συντελεστών Αδρανείας

Στα πλαίσια της παρούσας εργασίας διερευνήθηκε η ιδέα της ύπαρξης APSO με μη γραμμικό Συντελεστή Αδρανείας. Όπως έχει αναφερθεί παραπάνω, ο συντελεστής w ξεκινά από την τιμή 0.9 στην αρχή εκτέλεσης του αλγορίθμου και καταλήγει γραμμικά στην τιμή 0.4 με το τέλος των προκαθορισμένων επαναλήψεων. Στις επόμενες παραγράφους εξετάζεται η απόδοση του αλγορίθμου για μη γραμμική μείωση της τιμής του w.

Για να γίνει αυτή η διερεύνηση ο γραμμικός συντελεστής w αντικαταστάθηκε από καμπύλες. Συγκεκριμένα η εξίσωση υπολογισμού του w που αναφέρθηκε παραπάνω αντικαταστάθηκε με τις παρακάτω:

Εξίσωση 1:

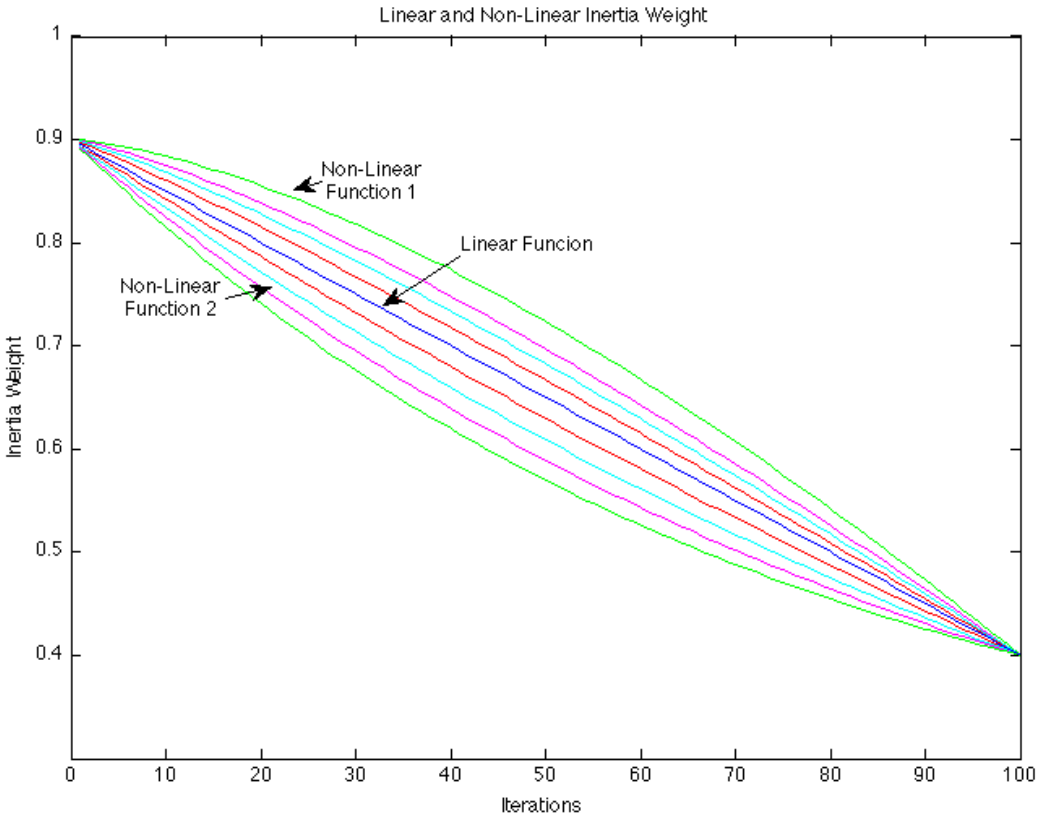
$$\boldsymbol{w = (firstval - finalval) * \left(1 - \frac{iteration^X}{maxiterations^X}\right) + finalval}$$

Εξίσωση 2:

$$\boldsymbol{w = \frac{firstval - finalval}{1 - e^{-\frac{X}{60} * maxiterations}} * e^{-\frac{X}{60} * iteration} + finalval - \frac{firstval - finalval}{1 - e^{-\frac{X}{60} * maxiterations}}}$$

όπου η μεταβλητή *X* είναι ένας παράγοντας που καθορίζει το μέγεθος της κυρτότητας που εμφανίζει ο w.

Για να γίνουν τα παραπάνω κατανοητά, παρατίθεται η παρακάτω γραφική παράσταση. Εκεί εμφανίζεται ο συντελεστής w σύμφωνα με τη γραμμική πρόταση των Shi και Eberhart (μπλε γραμμή) καθώς και οι καμπύλες σύμφωνα με τις παραπάνω εξισώσεις 1 και 2 (καμπύλες από πάνω και από κάτω της γραμμικής, αντίστοιχα).



Εικόνα 8: Γραμμικός και μη γραμμικός συντελεστής w

Στην παραπάνω γραφική παράσταση εμφανίζεται η εξέλιξη τόσο του γραμμικού όσο και των μη γραμμικών συντελεστών w που εξετάστηκαν. Οι καμπύλες που εμφανίζονται πάνω από την μπλε γραμμή (γραμμικός w) οφείλονται στην πρώτη εξίσωση, ενώ εκείνες που βρίσκονται από κάτω, στην δεύτερη. Αύξηση στην τιμή του παράγοντα X προκαλεί απομάκρυνση των καμπυλών από την γραμμική εκδοχή. Παρακάτω παρουσιάζονται τα αποτελέσματα των δοκιμών που εκτελέστηκαν για όλες τις καμπύλες που εμφανίστηκαν στην εικόνα. Οι δοκιμές αφορούν την συνάρτηση Griewangk 20 διαστάσεων με 20 σωματίδια και για 150 επαναλήψεις. Οι τιμές σύγκλισης είναι ο μέσος όρος 1500 δοκιμών, σε κάθε περίπτωση.

Συνάρτηση 1 σε σύγκριση με τον Linear APSO:

L-APSO	X = 1.1	X = 1.2	X = 1.3	X = 1.4	X = 1.5
0.2183	0.2268	0.2478	0.2633	0.2759	0.3007

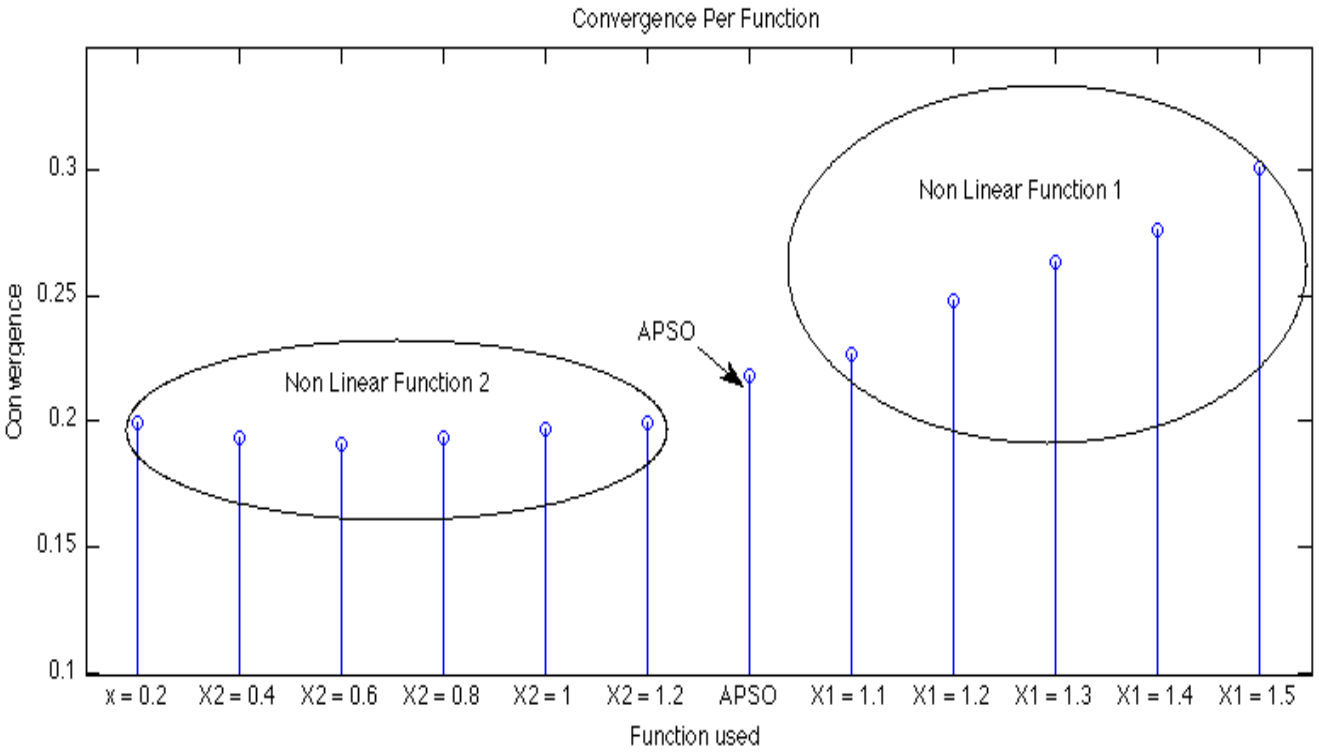
Πίνακας 4: Τιμές σύγκλισης της συνάρτησης 1 και του γραμμικού APSO

Συνάρτηση 2 σε σύγκριση με τον Linear APSO:

L-APSO	X = 0.2	X = 0.4	X = 0.6	X = 0.8	X = 1	X = 1.2
0.2183	0.1992	0.1937	0.1910	0.1933	0.1966	0.1997

Πίνακας 5: Τιμές σύγκλισης της συνάρτησης 2 και του γραμμικού APSO

Παρατηρώντας τους παραπάνω πίνακες εμφανίζεται μια διαφοροποίηση στις αποδόσεις του αλγορίθμου. Συγκεκριμένα, χρησιμοποιώντας την Συνάρτηση 1 ο αλγόριθμος γίνεται υποδεέστερος του γραμμικού APSO, με την απόδοσή του να χειροτερεύει όσο μεγαλώνει ο συντελεστής X. Αντιθέτως, η Συνάρτηση 2 εμφανίζεται να έχει αποδόσεις καλύτερες της γραμμικής για μικρές τιμές το X, ενώ όσο αυτή αυξάνεται, τόσο φαίνεται να χειροτερεύει πάλι ο αλγόριθμος. Προφανώς, η καλύτερη λύση για το παρών παράδειγμα είναι η Συνάρτηση 2 με X = 0.6. Αυτά τα στοιχεία φαίνονται γραφικά στην παρακάτω εικόνα.



Εικόνα 9: Απόδοση των L-APSO, Function 1 και Function 2

2.5 -Νέος Προτεινόμενος PSO-

Οι δυνατότητες εξέλιξης και βελτίωσης του PSO είναι πολλές, πράγμα που αποδεικνύεται από την ύπαρξη αρκετών προτάσεων εναλλακτικών εκδοχών του αλγορίθμου. Κάθε μία από αυτές, με πρώτη εκδοχή τον APSO, εμφανίζει βελτιωμένα χαρακτηριστικά ως προς την ποιότητα των λύσεων, δίνοντας αυξημένη ταχύτητα εύρεσης ικανοποιητικής λύσης ή ανώτερης ποιότητας λύση αποφεύγοντας τοπικά και βρίσκοντας ακριβέστερα ολικά ελάχιστα.

Στα πλαίσια της παρούσας πτυχιακής εργασίας, αναπτύχθηκε και παρουσιάζεται μια καινοτόμα μέθοδος που εμφανίζει αποτελέσματα ανώτερης ποιότητας από αυτά του SPSO αλλά και αυτά του APSO. Στην βιβλιογραφία που μελετήθηκε δεν βρέθηκε κάποια παρόμοια προσέγγιση με αυτήν που παρουσιάζεται παρακάτω. Έτσι, γίνεται για πρώτη φορά παρουσίαση και διερεύνηση των δυνατοτήτων της μεθόδου αυτής.

2.5.1 Ο PSO ως σύστημα ελέγχου

Για να καταστεί κατανοητή η λειτουργία της προτεινόμενης εκδοχής, παρακάτω γίνεται μια νέα προσέγγιση στην υλοποίηση του PSO: ως η δομή ενός ελεγκτή. Ακολουθώντας την λογική που παρουσιάζεται στην αναφορά [6], ο τρόπος με τον οποίο ο PSO βρίσκει μια βέλτιστη λύση ανανεώνοντας τις θέσεις των σωματιδίων, μπορεί να παρομοιαστεί με ένα διακριτό σύστημα με ανατροφοδότηση. Συγκεκριμένα, παρουσιάζεται ένα διακριτό σύστημα ελέγχου με ανατροφοδότηση, όπου τον ρόλο του ελεγκτή παίζει η μέθοδος υπολογισμού των νέων θέσεων του PSO. Έτσι γίνεται δυνατή η δράση ενός ελεγκτή στο σύστημα αυτό, με στόχο την γρήγορη (χρόνος αποκατάστασης) και ακριβή εύρεση (μόνιμο σφάλμα) ενός σημείου-στόχου που συνεχώς μεταβάλλεται (pbest και gbest). Ο PSO μπορεί να θεωρηθεί ως ο ελεγκτής σε ένα σύστημα ελέγχου με ανατροφοδότηση και αναλύεται όπως εξηγείται παρακάτω.

Το ελεγχόμενο σύστημα ουσιαστικά αποτελείται από το ίδιο το πρόβλημα προς επίλυση. Την είσοδο του ελεγχόμενου συστήματος αυτού αποτελούν οι θέσεις στις

οποίες βρίσκονται τα σωματίδια και η μετρήσιμη έξοδός του είναι η ποιότητα των λύσεων (pbest και gbest) για αυτές τις θέσεις. Στην συνέχεια, μέσω ενός βρόχου ανατροφοδότησης οι καλύτερες λύσεις (pbest και gbest) συγκρίνονται με τις παρούσες θέσεις και δίνεται το σφάλμα της θέσης των σωματιδίων. Τέλος, το σφάλμα αυτό εισέρχεται σε έναν ελεγκτή ο οποίος υπολογίζει τις ταχύτητες των σωματιδίων και ανανεώνει τις θέσεις τους. Οι εξισώσεις θέσης και ταχύτητας της βασικής εκδοχής του PSO (SPSO) σε διακριτό χρόνο φαίνονται παρακάτω. Επίσης, για την απλοποίηση του μπλοκ διαγράμματος του PSO, προστίθενται τρεις νέοι συμβολισμοί (e_{pid}, $e_{gi\delta}$ και e):

$$e_{pid}(t)=2*r_1*(pbest-x_{id}(t))$$

$$e_{gid}(t)=2*r_2*(gbest-x_{id}(t))$$

$$e(t)=e_{pid}(t)+e_{gid}(t)$$

$$v_{id}(t+1)=v_{id}(t)+e(t)$$

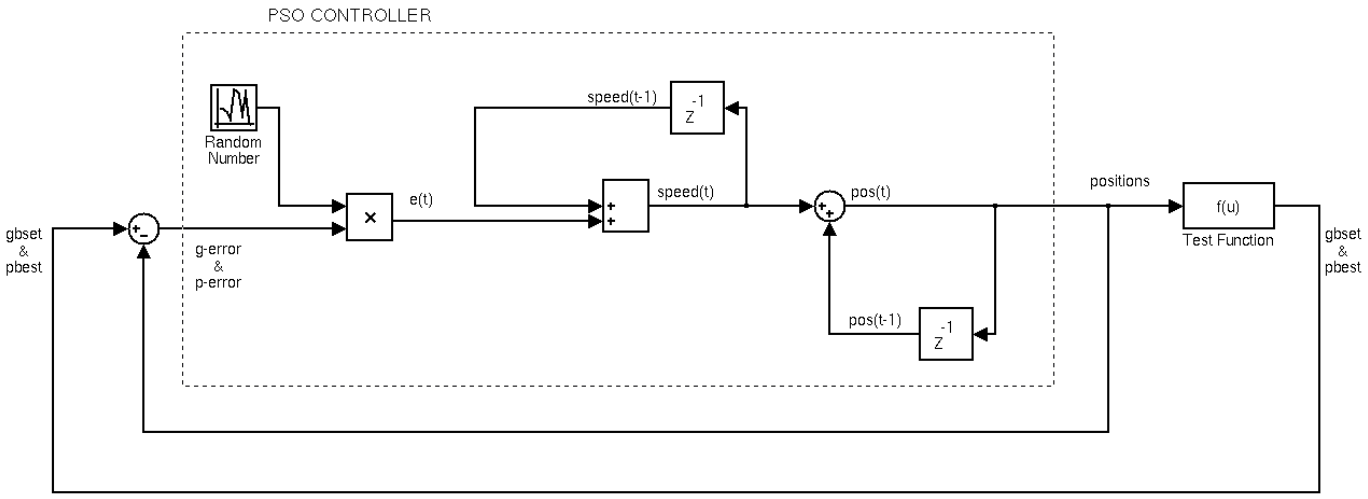
$$x_{id}(t+1)=x_{id}(t)+v_{id}(t+1)$$

Όπου v_{id} η ταχύτητα και x_{id} η θέση του σωματιδίου i ($i=1...part\_num$) στην διάσταση d ($d=1...dimentions$), e_{pid} το σφάλμα θέσης ως προς το προσωπικό βέλτιστο, e_{gid} το σφάλμα ως προς το ολικό, e το συνολικό σφάλμα και r_1, r_2 τυχαίοι αριθμοί με κανονική κατανομή μεταξύ 0 και 1. Έτσι, η εξίσωση υπολογισμού των θέσεων γίνεται:

$$x_{id}(t+1) = x_{id}(t) + v_{id}(t) + e(t)$$

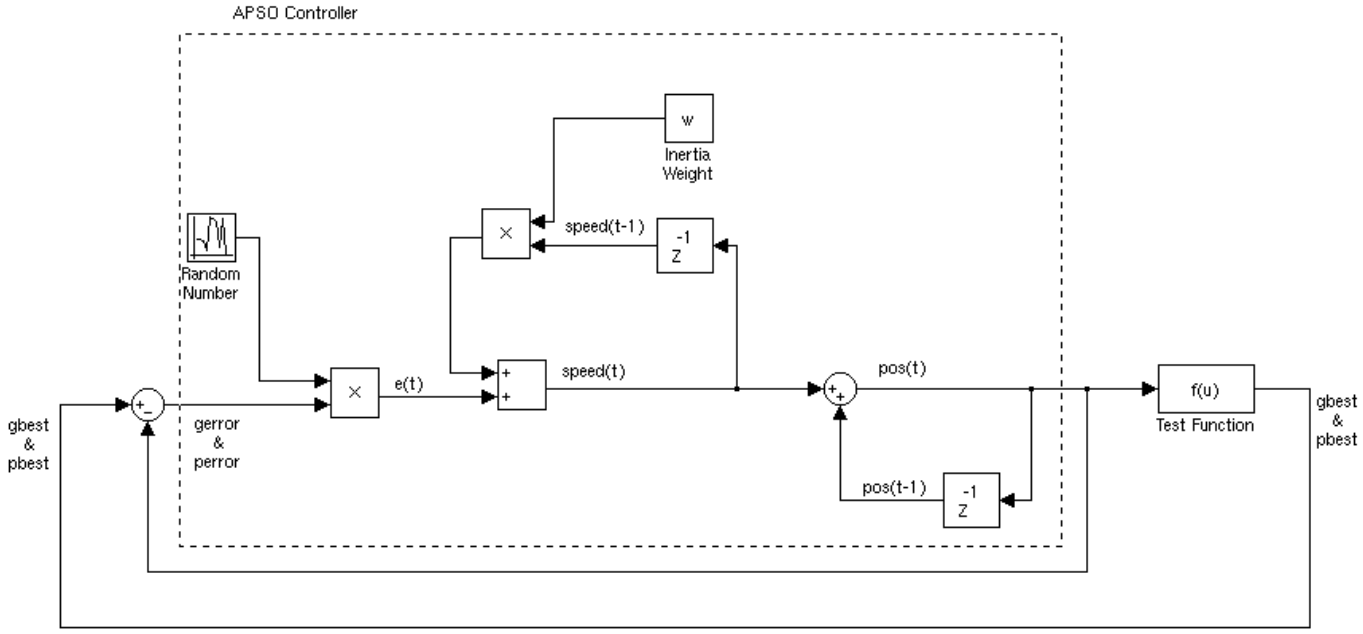
Εκφράζοντας την παραπάνω εξίσωση με λόγια, η επόμενη θέση ενός σωματιδίου i στην διάσταση d διαμορφώνεται από το άθροισμα της προηγούμενης θέσης του, της προηγούμενης ταχύτητάς του και της απόστασής του από το προσωπικό και το ολικό βέλτιστο.

Συνοπτικά, η παραπάνω περιγραφή απεικονίζεται στην επόμενη εικόνα. Το πλαίσιο περιλαμβάνει τα στοιχεία του SPSO, δηλαδή τον ελεγκτή του συστήματος όπου φαίνεται ο τρόπος υπολογισμού των νέων θέσεων των σωματιδίων. Στο σχήμα γίνεται μια απλοποίηση ως προς τον τρόπο υπολογισμού του σφάλματος, e(t), που δεν θα πρέπει να μπερδέψει τον αναγνώστη.



Εικόνα 10: Ο SPSO σαν σύστημα ελέγχου

Ακολουθώντας την παραπάνω λογική, εκτός του κλασικού SPSO, μπορεί να γίνει η αντίστοιχη αναπαράσταση με μπλοκ-διάγραμμα και για τον APSO. Έτσι, η εκδοχή του προσαρμοζόμενου PSO διαφοροποιείται μόνο στην προσθήκη του συντελεστή αδρανείας w. Το στοιχείο που πολλαπλασιάζεται με τον συντελεστή αυτόν δεν είναι άλλο από την προηγούμενη ταχύτητα *speed(t-1)* προτού αθροιστεί με το σφάλμα *e(t)* για να δώσουν την νέα ταχύτητα *speed(t)*. Το μπλοκ-διάγραμμα του περιγραφόμενου ελεγκτή φαίνεται παρακάτω.



Εικόνα 11: Ο APSO σαν σύστημα ελέγχου

Ο σκοπός για τον οποίο γίνεται η παραπάνω παρομοίωση του PSO με αυτήν ενός συστήματος ελέγχου γίνεται αντιληπτή κατανοώντας την λειτουργία ενός διακριτού ελεγκτή. Μια εισαγωγή και ανάλυση της λειτουργίας αυτής ακολουθεί στις επόμενες παραγράφους.

2.5.2 Έλεγχος σε διακριτά συστήματα

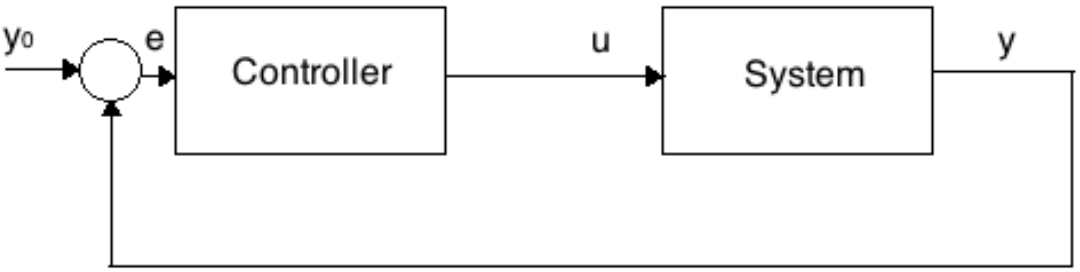
Στο πεδίο του ψηφιακού αυτομάτου ελέγχου, ένας ψηφιακός ελεγκτής αναλαμβάνει να ελέγξει ένα αναλογικό σύστημα. Σε ένα τέτοιο σύστημα, ανά τακτά χρονικά διαστήματα (χρόνος δειγματοληψίας) μετράται η ελεγχόμενη έξοδος του αναλογικού συστήματος. Ακολουθεί μετατροπή της τιμής αυτής από αναλογική σε ψηφιακή, σύγκρισή της με το σήμα εισόδου και υπολογίζεται ένα σφάλμα. Με κριτήριο αυτό το σφάλμα και μέσω απλών μαθηματικών πράξεων, ο μικροεπεξεργαστής υπολογίζει το νέο σήμα ελέγχου προς το σύστημα. Αυτή η νέα ψηφιακή έξοδος του ελεγκτή μετατρέπεται σε αναλογική και τροφοδοτεί το σύστημα.

Από μαθηματικής απόψεως, αυτός ο υπολογισμός δεν είναι κάτι άλλο από την υλοποίηση της εξίσωσης διαφορών που περιγράφει τον ελεγκτή στο διακριτό πεδίο.

Έτσι, ένα παράδειγμα υπολογισμού μιας τέτοιας εξίσωσης διαφορών - εξόδου διακριτού ελεγκτή μπορεί να είναι η παρακάτω εξίσωση (νόμος ελέγχου) [17]:

$u(t+1) = a*u(t) + b*u(t-1) + c*e(t) + d*e(t-1)$

Όπου *u* το σήμα ελέγχου προς το ελεγχόμενο σύστημα και *e* το σφάλμα, όπως φαίνονται στο επόμενο σχήμα.



Εικόνα 12: Σύστημα ελέγχου με ανατροφοδότηση

Αναλόγως του ελεγχόμενου συστήματος, ο ελεγκτής και άρα η εξίσωση υπολογισμού του σήματος εξόδου, μπορεί να ποικίλει ως προς τον αριθμό των προηγούμενων καταστάσεων και τους συντελεστές (a, b, ...).

Περιγράφοντας έναν τέτοιο νόμο ελέγχου με λόγια, το νέο σήμα ελέγχου θα είναι το άθροισμα του προηγούμενου σήματος ελέγχου, του προηγούμενου σφάλματος, και του προ-προηγούμενου σήματος ελέγχου και σφάλματος.

Χρησιμοποιώντας αυτή την περιγραφή του «αθροίσματος προηγούμενων καταστάσεων», φαίνεται μια έντονη ομοιότητα με αυτήν της εξίσωσης υπολογισμού των ταχυτήτων στον PSO. Η νέα εκδοχή PSO που παρουσιάζεται εδώ, βασίζεται ακριβώς σε αυτή την παρατήρηση.

2.5.3 Διακριτός PSO – Discrete PSO

Όπως περιγράφηκε παραπάνω, παρατηρείται μια έντονη ομοιότητα του τρόπου περιγραφής των εξισώσεων διαφορών ενός διακριτού ελεγκτή και του PSO. Για τον υπολογισμό της νέας θέσης ενός σωματιδίου ο PSO χρησιμοποιεί την προηγούμενη

θέση, την προηγούμενη ταχύτητα και το παρόν σφάλμα θέσης. Αντιθέτως, ένας ελεγκτής εκτός από το παρόν σφάλμα και το προηγούμενο σήμα ελέγχου, χρησιμοποιεί και παλαιότερες τιμές. Ο έλεγχος σε αυτήν την περίπτωση βελτιώνεται, καθώς ο ελεγκτής φαίνεται να «γνωρίζει» καλύτερα την κατάσταση στην οποία βρίσκεται το σύστημα.

Με βάση αυτή την λογική, ο PSO τροποποιήθηκε και εξελίχθηκε έτσι ώστε να μπορεί και αυτός, όπως και ο ελεγκτής, να γνωρίζει την πρόοδό που σημειώνει. Έτσι, χρησιμοποιώντας τους συμβολισμούς του σφάλματος που παρουσιάστηκαν παραπάνω, ο τρόπος υπολογισμού των ταχυτήτων του Διακριτού PSO (D-PSO) ακολουθεί:

...

$e_2(t)=e_1(t)$ ή αλλιώς $e_2(t) = e(t-2)$

$e_1(t)=e(t)$ ή αλλιώς $e_1(t) = e(t-1)$

$e_{pid}(t)=2r_1(pbest-x_{id}(t))$

$e_{gid}(t)=2r_2(gbest-x_{id}(t))$

$e(t)=e_{pid}(t) + e_{gid}(t)$

$v_{id}(t+1) = \beta*v_{id}(t) + \gamma*e(t) + \delta*e_1(t) + \varepsilon* e_2(t) + ...$

$x_{id}(t+1)= \alpha*x_{id}(t)+v_{id}(t+1)$

Συνολικά, η εξίσωση υπολογισμού των θέσεων μπορεί να εκφραστεί και ως:

$x_{id}(t+1) = \alpha*x_{id}(t)+ \beta*v_{id}(t) + \gamma*e(t) + \delta*e_1(t) + \varepsilon* e_2(t) +...$

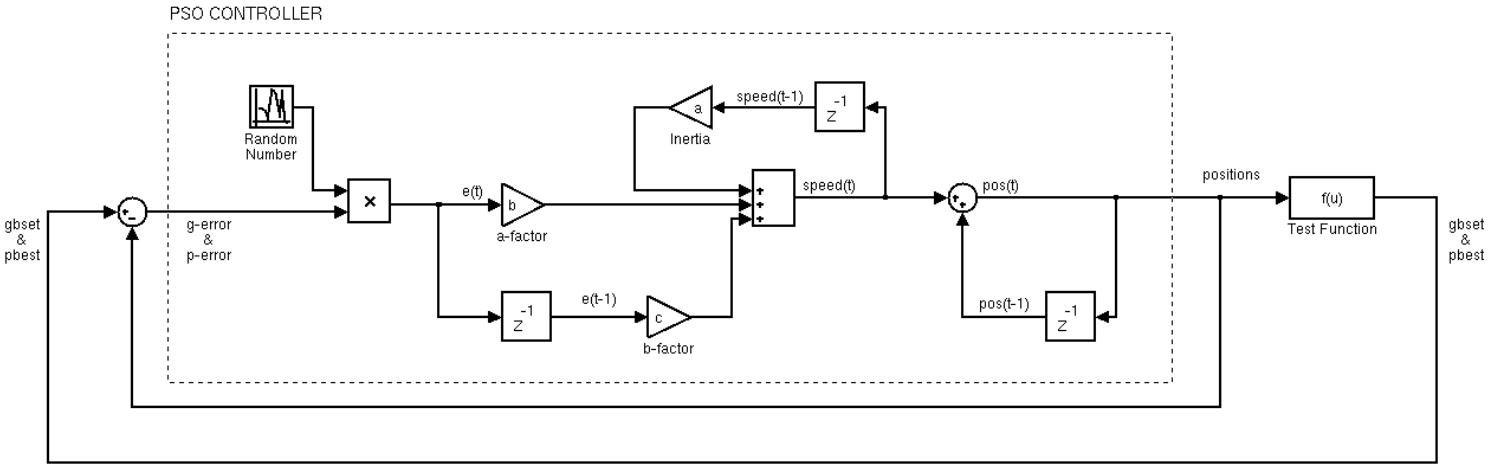
ή αλλιώς

$x_{id}(t+1) = \alpha*x_{id}(t)+ \beta*v_{id}(t) + \gamma*e(t) + \delta*e(t-1) + \varepsilon* e(t-2) + ...$

Όπως φαίνεται από τα παραπάνω, οι νέες θέσεις των σωματιδίων του προτεινόμενου DPSO δεν ρυθμίζονται με βάση μόνο τις προηγούμενες θέσεις, ταχύτητες και σφάλματα, αλλά και με βάση τα προηγούμενα και προ-προηγούμενα

σφάλματα. Σημαντικότατη αλλαγή αποτελεί και η χρήση του συντελεστή (α) στις προηγούμενες θέσεις. Οι τρεις τελείες δηλώνουν πως με επιλογή του χρήστη μπορούν να συμπεριληφθούν και ακόμα περισσότεροι όροι. Οι συντελεστές α, β, γ και δ ρυθμίζονται έτσι ώστε να επιτευχθούν οι καλύτερες δυνατές επιδόσεις.

Στο επόμενο μπλοκ διάγραμμα εμφανίζεται ο DPSO σαν ελεγκτής ενός συστήματος. Στο συγκεκριμένο σύστημα χρησιμοποιείται μόνο μία τιμή προηγούμενων καταστάσεων, το προηγούμενο σφάλμα e(t-1).



Εικόνα 13: Μπλοκ διάγραμμα του Διακριτού PSO

Κάτι που έγινε σαφές κατά την χρήση του DPSO για την επίλυση διαφόρων προβλημάτων ήταν το ότι η επιλογή διαφορετικού αριθμού όρων και διαφορετικών συντελεστών, οδηγούσε σε αποτελέσματα διαφορετικής ποιότητας για το κάθε πρόβλημα. Αυτό είναι αναμενόμενο κοιτώντας το παραπάνω μπλοκ διάγραμμα. Συγκεκριμένα, και αφού ο PSO έχει πια τον ρόλο του ελεγκτή, γίνεται κατανοητό πως κάθε διαφορετικό σύστημα θα χρειαστεί και έναν ειδικά ρυθμισμένο ελεγκτή. Παρακάτω, γίνεται μια προσπάθεια κατανόησης της επιρροής των συντελεστών του DPSO στην επίλυση κάποιων συγκεκριμένων προβλημάτων.

2.5.4 Διερεύνηση των συντελεστών

Όπως αναφέρθηκε παραπάνω, οι συντελεστές α, β, γ, δ... της εξίσωσης των νέων θέσεων του DPSO μπορούν να ρυθμιστούν για να επιτευχθούν οι καλύτερες δυνατές επιδόσεις. Θέτοντας όλους αυτούς τους όρους στην μονάδα και συγκρίνοντας την

εξίσωση του DPSO με αυτήν του SPSO είναι προφανές ότι η διαφορά μεταξύ τους είναι οι προηγούμενες καταστάσεις των σφαλμάτων.

SPSO:

$x_{id}(t+1) = x_{id}(t) + v_{id}(t) + e(t)$

DPSO:

$x_{id}(t+1) = x_{id}(t) + v_{id}(t) + e(t) + e(t-1) + \dots$

Παρακάτω, γίνεται μια προσπάθεια να κατανοηθούν οι επιδράσεις των συντελεστών στην λειτουργία και επιδόσεις του αλγορίθμου. Ξεκινώντας την διερεύνηση από την ειδοποιό διαφορά μεταξύ των δύο εξισώσεων, τον όρο $\delta * e(t-1)$, ο συντελεστής δ παίρνει διάφορες τιμές και γίνονται κάποιες συγκρίσεις. Παρατηρώντας την εξίσωση της ταχύτητας στην περίπτωση του DPSO βλέπουμε τουλάχιστον τέσσερις συντελεστές που αθροίζονται μεταξύ τους. Δύο από αυτούς τους όρους συμβολίζουν την βελτίωση ή μη της σύγκλισης του αλγορίθμου. Συγκεκριμένα, αυτοί οι δύο συντελεστές είναι το παρόν και το προηγούμενο σφάλμα. Η διαφορά μεταξύ τους δηλώνει το ρυθμό και την ακρίβεια με την οποίο ο αλγόριθμος συγκλίνει σε μία βέλτιστη λύση. Έτσι, γίνεται κατανοητό ότι για να μπορέσουν τα σωματίδια να διερευνήσουν με μεγάλη ακρίβεια μια μικρή περιοχή, η διαφορά των σφαλμάτων και άρα η ταχύτητά τους πρέπει να τείνει στο μηδέν.

Χρησιμοποιώντας την αναλογία της ταχύτητας και επιτάχυνσης όπου:

$v(k+1) = v(k) + a(k+1)$

είναι ανάλογο του:

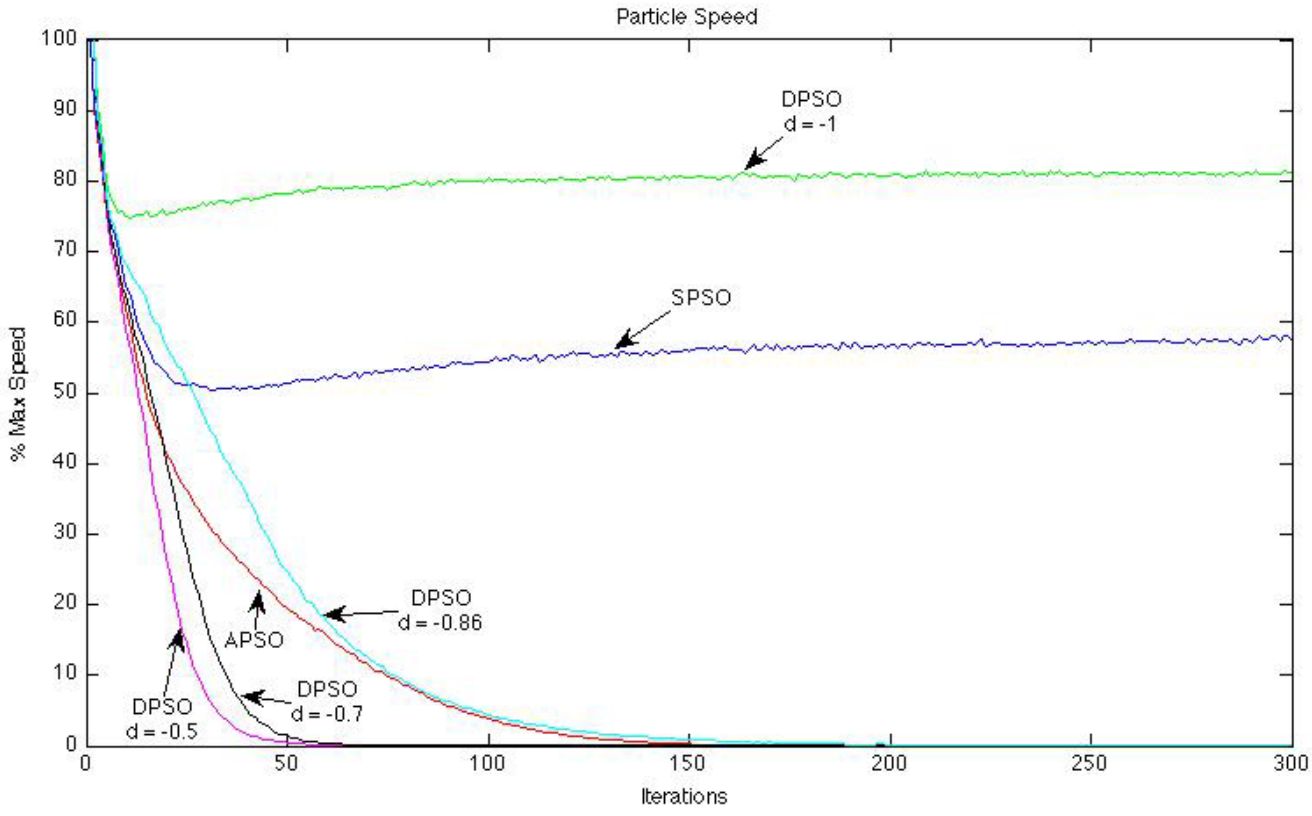
$v_{id}(t+1) = v_{id}(t) + e(t) + \delta * e(t-1) + \varepsilon * e(t-2) + \dots$

φαίνεται ότι η επιτάχυνση $a(k+1)$ είναι ανάλογη του αθροίσματος των σφαλμάτων: $e(t) + \delta * e(t-1) + \varepsilon * e(t-2) + \dots$

Γνωρίζοντας, όμως, ότι η επιτάχυνση πρέπει να τείνει στο μηδέν, προκύπτει ότι για σύνολο δύο συντελεστών ($e(t)$ και $e(t-1)$) η πράξη μεταξύ τους θα πρέπει να είναι η αφαίρεση ($\delta<0$). Αν δεν γίνεται αυτό, και αντιθέτως οι συντελεστές αθροίζονται, η επιτάχυνση των σωματιδίων παίρνει μεγάλες τιμές με αποτέλεσμα η

σύγκλιση του DPSO και τα αποτελέσματά του να μην είναι ικανοποιητικά. Επίσης, σημαντικό ρόλο παίζει και η τιμή του δ, όχι μόνο το πρόσημό του. Έτσι, όταν οι επιτάχυνση ($e(t) + \delta * e(t-1)$) παίρνει μεγάλες τιμές, ανεξαρτήτου προσήμου, τα σωματίδια τείνουν να ταλαντώνονται έντονα και «άσκοπα» στον χώρο.

Χαρακτηριστικό παράδειγμα της σημαντικότητας να τείνουν στο μηδέν οι επιταχύνσεις και οι ταχύτητες είναι η μεγάλη βελτίωση που σημειώνει ο APSO σε σύγκριση με τον SPSO. Παρότι ο APSO δεν επηρεάζει με κάποιο τρόπο τις επιταχύνσεις, δρα όμως και μειώνει την προηγούμενη ταχύτητα. Με αυτόν τον τρόπο και καθώς περνούν οι επαναλήψεις οι νέες ταχύτητες εξαναγκάζονται να γίνουν όλο και πιο μικρές. Παρακάτω, γίνεται μια σύγκριση των μέσων ταχυτήτων του συνόλου των σωματιδίων και σε όλες τις διαστάσεις ενός προβλήματος, έπειτα από 100 επάλληλες επιλύσεις του προβλήματος. Οι ταχύτητες είναι εκφρασμένες σε ποσοστό επί της μέγιστης επιτρεπόμενης ταχύτητας, και οι αλγόριθμοι που εξετάστηκαν είναι οι SPSO, APSO και DPSO. Συγκεκριμένα για τον DPSO οι συντελεστές
α, β και γ ήταν μονάδα ενώ ο δ είχε τιμές: -1, -0.86, -0.7 και -0.5.



Εικόνα 14: Σύγκριση ταχυτήτων σωματιδίων των SPSO, APSO και DPSO

Στην εικόνα φαίνεται καθαρά πως ο SPSO καθώς και ο DPSO με δ = -1 έχουν αρκετά μεγάλες ταχύτητες μέχρι εξάντλησης των επαναλήψεων. Από την άλλη, οι

ταχύτητες των APSO και ο DPSO με τιμές δ > -1 προσεγγίζουν και αγγίζουν το μηδέν, πράγμα που σημαίνει ότι ερευνούν με λεπτομέρεια ένα μικρό μέρος του χώρου των λύσεων. Στον παρακάτω πίνακα φαίνονται τα αποτελέσματα της παραπάνω δοκιμής. Γίνεται σαφές πως η ταχύτητες των σωματιδίων συνδέονται άμεσα με την αποτελεσματικότητα της σύγκλισης που μπορούν να επιτύχουν. Συγκεκριμένα, φαίνεται πως ταχύτητες που σταδιακά τείνουν στο μηδέν ευνοούν την εύρεση ακριβούς λύσης, αλλά όταν αυτές μηδενίζονται γρήγορα, τα σωματίδια τείνουν να παγιδεύονται σε τοπικά ελάχιστα. Έτσι, ο DPSO για *δ=-1* και μεγάλη ταχύτητα δίνει μη ικανοποιητικό αποτέλεσμα, ακριβώς όπως και ο SPSO. Στο άλλο άκρο, όταν ο συντελεστής *δ* παίρνει τιμές μεγαλύτερες του -0.86 οι ταχύτητες είναι μικρές και τα αποτελέσματα χαμηλής ποιότητας. Τέλος, ο APSO και ο DPSO με *δ=-0.86* δίνουν τα καλύτερα αποτελέσματα.

Αλγόριθμος	**SPSO**	**APSO**	**DPSO δ = -1**	**DPSO δ = -0.86**	**DPSO δ = -0.7**	**DPSO δ = -0.5**
Μ.Ο. Σύγκλισης	0.524	0.095	11.399	0.151	0.252	0.230
Μ.Ο. Ταχυτήτων (%)	56.12	9.03	80.05	11.15	5.66	4.59

Πίνακας 6: Σύγκριση συγκλίσεων των εξεταζόμενων αλγορίθμων

Στον παραπάνω πίνακα, ως Μέσος Όρος (Μ.Ο.) Σύγκλισης νοείται ο αριθμητικός μέσος όρος των συγκλίσεων των 100 επιλύσεων για τον κάθε αλγόριθμο, ενώ ως Μέσος Όρος Ταχυτήτων, νοείται ο αριθμητικός μέσος όρος των ποσοστιαίων ταχυτήτων του κάθε αλγορίθμου, όπως παρουσιάζονται στην Εικόνα 14.

2.5.5 Βέλτιστη ρύθμιση των συντελεστών

Όπως φαίνεται από τις εξισώσεις ταχύτητας του Διακριτού PSO, ο αριθμός των όρων των παρελθόντων σφαλμάτων και οι τιμές των συντελεστών, είναι ζήτημα της επιλογής του χρήστη. Στην συγκεκριμένη περίπτωση επιλέχθηκε η χρήση πέντε συνολικά όρων: αυτός της ταχύτητας, των δύο προηγούμενων σφαλμάτων και των δύο προηγούμενων θέσεων. Εξ ορισμού, η επιλογή των τιμών των πέντε αυτών

συντελεστών, για την επίτευξη της καλύτερης δυνατής λύσης μιας συνάρτησης, αποτελεί το ίδιο ένα πρόβλημα βελτιστοποίησης. Έτσι, οι συντελεστές αυτοί μπορούν να επιλεγούν είτε με χειροκίνητη μέθοδο (trial and error) είτε να αποτελέσουν πρόβλημα επίλυσης ενός αλγορίθμου βελτιστοποίησης σαν τους γενετικούς αλγορίθμους, το big bang.

Στην συγκεκριμένη περίπτωση, οι συντελεστές των πέντε αυτών όρων βελτιστοποιήθηκαν κάνοντας χρήση του ίδιου του PSO. Χρησιμοποιήθηκε ο αλγόριθμος APSO με δέκα σωματίδια και είκοσι επαναλήψεις για να λύσει το πρόβλημα των πέντε διαστάσεων. Οι πέντε αυτές διαστάσεις δεν είναι άλλες από τους συντελεστές α, β, γ, δ και ε που πολλαπλασιάζουν τις προηγούμενες θέσεις, ταχύτητες, σφάλματα και προ-προηγούμενα θέσεις και σφάλματα αντίστοιχα.

Για να γίνει αυτή η βελτιστοποίηση σχεδιάστηκε ένας αλγόριθμος-παραλλαγή του test_function. Μια συνάρτηση δηλαδή, που καλούνταν από τον APSO, δεχόταν τις θέσεις όλων των σωματιδίων σε κάθε διάσταση και επέστρεφε μια τιμή (fitness value) χαρακτηριστική της επίδοσης του κάθε σωματιδίου. Για την εύρεση της επίδοσης του κάθε σωματιδίου η test_function καλούσε 600 φορές μια άλλη συνάρτηση και υπολόγιζε έναν μέσο όρο. Αυτή η συνάρτηση με τη σειρά της ήταν ο DPSO που με 20 σωματίδια και 150 επαναλήψεις έλυνε ένα σύνολο συναρτήσεων διαφορετικών ορίων και διαστάσεων.

Συγκεκριμένα, αυτή η βελτιστοποίηση πραγματοποιήθηκε με κριτήριο ένα σύνολο τριών συναρτήσεων: Michalewicz, Schwefel και η Trid function. Για την συγκεκριμένη περίσταση, φτιάχτηκε μια εξίσωση που κανονικοποιούσε και άθροιζε τα επιμέρους αποτελέσματα:

$$eval_p = \sum_{n=1}^{3} \frac{(min_n - result_{np})}{min_n}$$

Όπου n = 1...3 οι τρεις συναρτήσεις που εξετάστηκαν, p το εκάστοτε σωματίδιο, min_n τα γνωστά ελάχιστα της εκάστοτε συνάρτησης, $result_{np}$ ο μέσος όρος των 600 επαναλήψεων του κάθε σωματιδίου για την κάθε συνάρτηση.

Για να γίνουν τα παραπάνω κατανοητά, ακολουθεί εκφρασμένος με λόγια ο αλγόριθμος που πραγματοποίησε την βελτιστοποίηση. Αυτός αποτελείται από πέντε

συναρτήσεις που καλούνται με την σειρά που εμφανίζονται: APSO, test_function, DPSO1, DPSO2 και DPSO3.

Αλγόριθμος του APSO που εξετάζει διάφορες τιμές για τους συντελεστές

Αρχή

Αρχικοποίηση Θέσεων και Ταχυτήτων Σμήνους

Επανέλαβε 20 φορές

Αξιολόγηση=κλήση της test_function [θέσεις σωματιδίων (30x4)]

Επιλογή Καλύτερης Ολικής και Προσωπικής Θέσης

Υπολογισμός Νέων Ταχυτήτων

Υπολογισμός Νέων Θέσεων

Τέλος

Στην έξοδο η καλύτερη λύση

Τέλος

Αλγόριθμος του test_function που αξιολογεί την επίδοση των σωματιδίων

Αρχή

Λήψη θέσεων σωματιδίων

Επανέλαβε για τα 30 σωματίδια

Επανέλαβε 600 φορές

Τιμή_1(φορά) = κλήση συνάρτησης DPSO_1[συντελεστές (1x5)]

Τιμή_2(φορά) = κλήση συνάρτησης DPSO_2(1x5)

Τιμή_3(φορά) = κλήση συνάρτησης DPSO_3(1x5)

Τέλος

$Result_1$(σωματίδιο) = M.O.(Τιμή_1)

$Result_2$(σωματίδιο) = M.O.(Τιμή_2)

$Result_3$(σωματίδιο) = M.O.(Τιμή_3)

Υπολογισμός *eval*(σωματιδιο)

Τέλος

Στην έξοδο το *eval*

Τέλος

Αλγόριθμος του DPSO_1 που βρίσκει βέλτιστη λύση στη συνάρτηση Michalewicz βάσει των συντελεστών

Αρχή

Λήψη τιμών για συντελεστές

Αρχικοποίηση Θέσεων και Ταχυτήτων Σμήνους

Επανέλαβε 150 φορές

Αξιολόγηση = λύση της Michalewicz [θέσεις σωματιδίων (20x5)]

Επιλογή Καλύτερης Ολικής και Προσωπικής Θέσης

Υπολογισμός Νέων Ταχυτήτων

Υπολογισμός Νέων Θέσεων

Τέλος

Στην έξοδο η καλύτερη λύση

Τέλος

Αλγόριθμος του DPSO_2 που βρίσκει βέλτιστη λύση στη συνάρτηση Schwefel βάσει των συντελεστών

Αρχή

Λήψη τιμών για συντελεστές

Αρχικοποίηση Θέσεων και Ταχυτήτων Σμήνους

Επανέλαβε 150 φορές

Αξιολόγηση = λύση της Schwefel [θέσεις σωματιδίων (20x4)]

Επιλογή Καλύτερης Ολικής και Προσωπικής Θέσης

Υπολογισμός Νέων Ταχυτήτων

Υπολογισμός Νέων Θέσεων

Τέλος

Στην έξοδο η καλύτερη λύση

Τέλος

Αλγόριθμος του DPSO_3 που βρίσκει βέλτιστη λύση στη συνάρτηση Trid βάσει των συντελεστών

Αρχή

Λήψη τιμών για συντελεστές

Αρχικοποίηση Θέσεων και Ταχυτήτων Σμήνους

Επανέλαβε 150 φορές

Αξιολόγηση = λύση της Trid [θέσεις σωματιδίων (20x5)]

Επιλογή Καλύτερης Ολικής και Προσωπικής Θέσης

Υπολογισμός Νέων Ταχυτήτων

Υπολογισμός Νέων Θέσεων

Τέλος

Στην έξοδο η καλύτερη λύση

Τέλος

Συνολικά, όπως περιγράφεται παραπάνω, χρησιμοποιήθηκε ο APSO για να ρυθμίσει τους συντελεστές α, β, γ, δ και ε του DPSO με κριτήριο ένα σύνολο τριών συναρτήσεων. Τα αποτελέσματα της ρύθμισης αυτής παρουσιάζονται παρακάτω και οδηγούν στην εξίσωση διαφορών των επόμενων θέσεων:

α	β	γ	δ	ε
1.2	*1.1*	*-0.3*	*0.8*	*0.2*

Πίνακας 7: Επιλεγμένες τιμές των συντελεστών ύστερα από βελτιστοποίηση

Και άρα:

$$x_{id}(t+1) = 0.8*x_{id}(t) + 0.2*x_{id}(t-1) + 1.2*v_{id}(t) + 1.1*e(t) - 0.3*e(t-1)$$

Παρότι το συγκεκριμένο σετ τιμών των συντελεστών παρέχει πολύ ικανοποιητικά αποτελέσματα για το δεδομένο πολύπλοκο πρόβλημα, παρουσιάζει προβληματική συμπεριφορά σε άλλα προβλήματα. Για τον λόγο αυτό, στην συνέχεια της εργασίας χρησιμοποιείται ένα άλλο σετ τιμών και προτείνεται ως «βάση» για την χρίση του DPSO. Με τον όρο «βάση» δηλώνεται η ανάγκη της χειροκίνητης βέλτιστης ρύθμισης των συντελεστών του DPSO. Λόγω του ότι το κάθε πρόβλημα

αποτελεί και ένα διαφορετικό σύστημα, ο ελεγκτής του -το σετ τιμών- χρειάζεται να είναι διαφορετικό. Το παρακάτω προτεινόμενο σετ-βάση αποτελεί μια καλή αρχή για ένα μεγάλο σύνολο προβλημάτων. Με πολύ μικρές παρεμβάσεις σε αυτό, τα αποτελέσματα του DPSO μπορούν να βελτιωθούν σημαντικά. Το σετ τιμών-βάση είναι το:

α	β	γ	δ	ε
0.8	*0.5*	*0.2*	*0.8*	*0.2*

Πίνακας 8: Το προτεινόμενο σετ τιμών – βάση

Και άρα, η προτεινόμενη εξίσωση ταχυτήτων γίνεται:

$$x_{id}(t+1) = 0.8*x_{id}(t) + 0.2*x_{id}(t-1) + 0.8*v_{id}(t) + 0.5*e(t) + 0.3*e(t-1)$$

Σε όλα τα προβλήματα που ακολουθούν στην παρούσα εργασία, γίνεται χρίση του παραπάνω σετ τιμών, με κάποιες μικρο-ρυθμίσεις στους συντελεστές α και γ και μόνο.

2.5.6 Μετρήσεις και Συγκρίσεις

Έχοντας ήδη αποδείξει την υπεροχή του APSO σε σύγκριση με τον SPSO στις προηγούμενες σελίδες, εδώ γίνεται μια απόπειρα να εξεταστούν οι επιδόσεις του DPSO. Για τον σκοπό αυτό, θα εκτελεστούν διάφορες δοκιμασίες και θα συγκριθούν τα αποτελέσματα του DPSO, με αυτά του APSO. Ο DPSO των πέντε συντελεστών που χρησιμοποιήθηκε, είχε τιμές:

α	β	γ	δ	ε
0.7 ή 0.75	0.5	0.2	0.8	0.2

Πίνακας 9: Σετ τιμών που χρησιμοποιήθηκε στις μετρήσεις και συγκρίσεις

Με τον συντελεστή α να ρυθμίζεται χειροκίνητα στο 0.7 ή 0.75 ανάλογα με τις επιδόσεις του αλγορίθμου στην εκάστοτε συνάρτηση. Για ευκολία επιλέχθηκαν αυτές οι δύο τιμές για το α και δεν έγινε περαιτέρω διερεύνηση.

Αρχικά, εξετάζεται η σύγκλιση και των δύο αλγορίθμων στην λύση της συνάρτησης Schwefel. Η αναζήτηση γίνεται στον χώρο [-500 500], για πρόβλημα 4 διαστάσεων, κάνοντας χρήση 20 σωματιδίων και για 500 επαναλήψεις, με *α=0.75*. Ακολουθεί η συνάρτηση Griewangk με όρια αναζήτησης τα όρια [-600 600], έξι διαστάσεις, 20 σωματίδια και 500 επαναλήψεις και *α=0.7*. Στη συνέχεια, οι επιδόσεις των δύο αλγορίθμων συγκρίνονται ως προς τη συνάρτηση Rosenbrock's Valley έξι διαστάσεων στα όρια [-2.048 2.048] με ίδιο αριθμό επαναλήψεων και σωματιδίων και με *α=0.7*. Ακολούθως, εξετάστηκε η συνάρτηση Michalewicz με όρια τα [0 π], πέντε διαστάσεις, ίδια σωματίδια και επαναλήψεις και *α=0.75*. Τέλος, εξετάζεται η συνάρτηση Rastring έξι διαστάσεων με όρια τα [-5.12 5.12] και σωματίδια, επαναλήψεις και συντελεστή α όπως παραπάνω.

Παρακάτω παρουσιάζονται με την προαναφερθείσα σειρά οι εξεταζόμενες συναρτήσεις, και στον πίνακα 10 συνοψίζονται πληροφορίες σχετικά με τις διαστάσεις του κάθε προβλήματος, τον χώρο όπου έγινε η έρευνα, το ολικό ελάχιστο της κάθε συνάρτησης καθώς και η τιμή του συντελεστή *α* όπως ρυθμίστηκε στην κάθε περίπτωση.

Schwefel's Function	$\boldsymbol{f_1(x) = \sum_{i=1}^{n} \left[-x_i sin(\sqrt{\lvert x_i \rvert})\right]}$
Griewangk's Function	$\boldsymbol{f_2(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} cos\left(\frac{x_i}{\sqrt{i}}\right) + 1}$
Rosenbrock's Valley	$\boldsymbol{f_3(x) = \sum_{i=1}^{n-1} \left[100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2\right]}$
Michalewicz's Function	$\boldsymbol{f_4(x) = -\sum_{i=1}^{n} sin(x_i)\left[sin\left(\frac{ix_i^2}{\pi}\right)\right]^{2m}, m = 10}$
Rastring's Function	$\boldsymbol{f_5(x) = 10n + \sum_{i=1}^{n}\left[x_i^2 - 10cos(2\pi x_i)\right]}$

Πίνακας 10: Benchmark συναρτήσεις που χρησιμοποιήθηκαν

Συνάρτηση	**Διαστάσεις προβλήματος**	**Χώρος έρευνας**	**Ολικό ελάχιστο**	**Συντελεστής α**
$f_1(x)$	4	$[-500 \;\; 500]$	$-418.9829 * 4$	0.75
$f_2(x)$	6	$[-600 \;\; 600]$	0	0.7
$f_3(x)$	6	$[-2.048 \;\; 2.048]$	0	0.7
$f_4(x)$	5	$[0 \;\; \pi]$	-4.687	0.75
$f_5(x)$	6	$[-5.12 \;\; 5.12]$	0	0.75

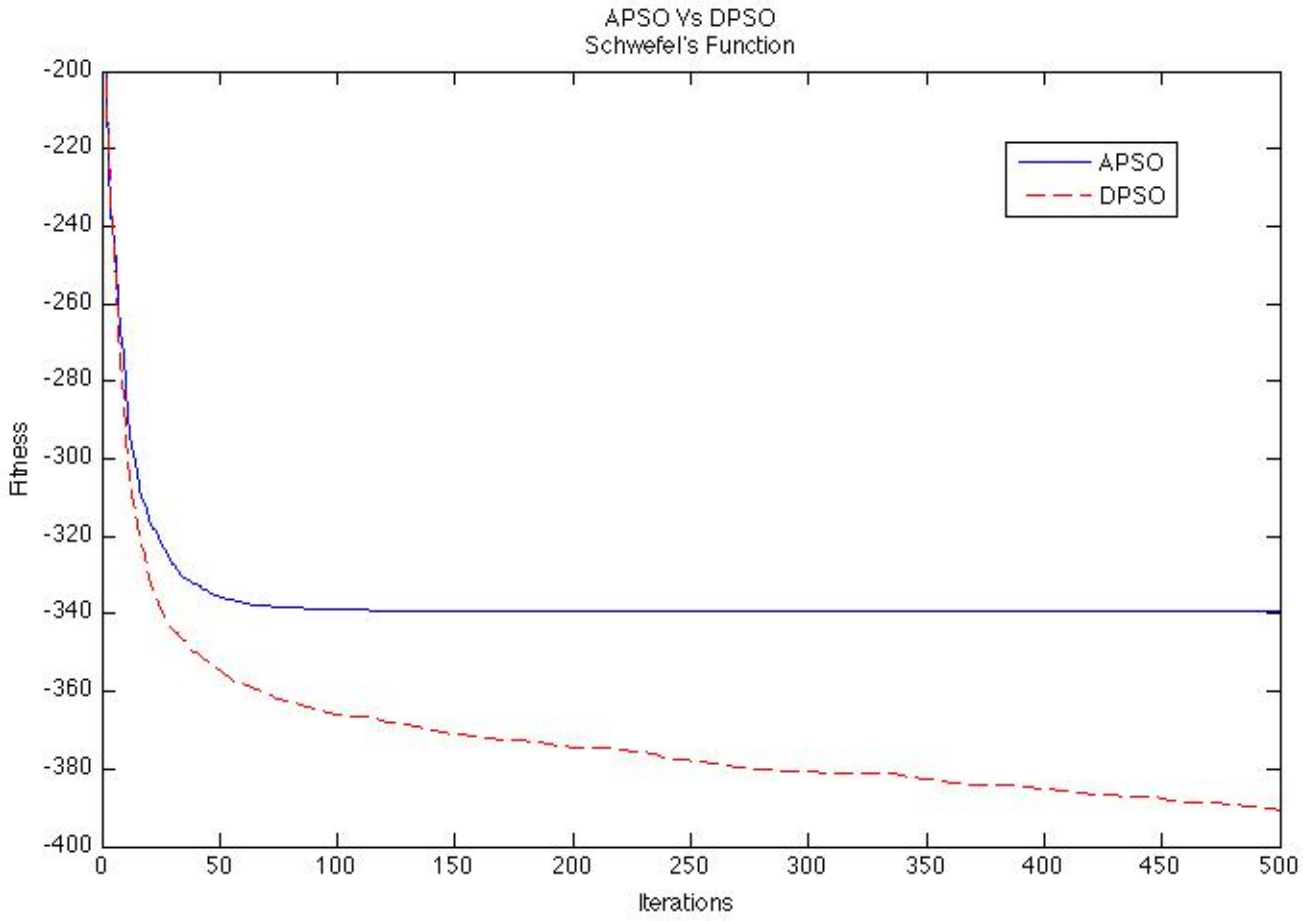
Πίνακας 11: Χαρακτηριστικά της εκάστοτε συνάρτησης

Κάθε αλγόριθμος εκτελέστηκε 200 φορές για αυξημένη ακρίβεια, και τα αποτελέσματά τους συγκρίνονται στους παρακάτω πίνακες. Ακόμα, στις εικόνες φαίνονται οι συγκλίσεις των δύο αλγορίθμων για τις πέντε εξεταζόμενες συναρτήσεις.

Συνάρτηση 1: Schwefel's Function

Συνάρτηση	Αλγόριθμος	Μέσος Όρος	Μέγιστο	Τυπική Απόκλιση	Προσέγγιση Ολικού Ελαχίστου (< -418)
Schwefel	**APSO**	-339.05	-201.84	41.3	5%
	DPSO	-390.49	-269.08	30.7	41%

Πίνακας 12: Σύγκριση συγκλίσεων των APSO και DPSO

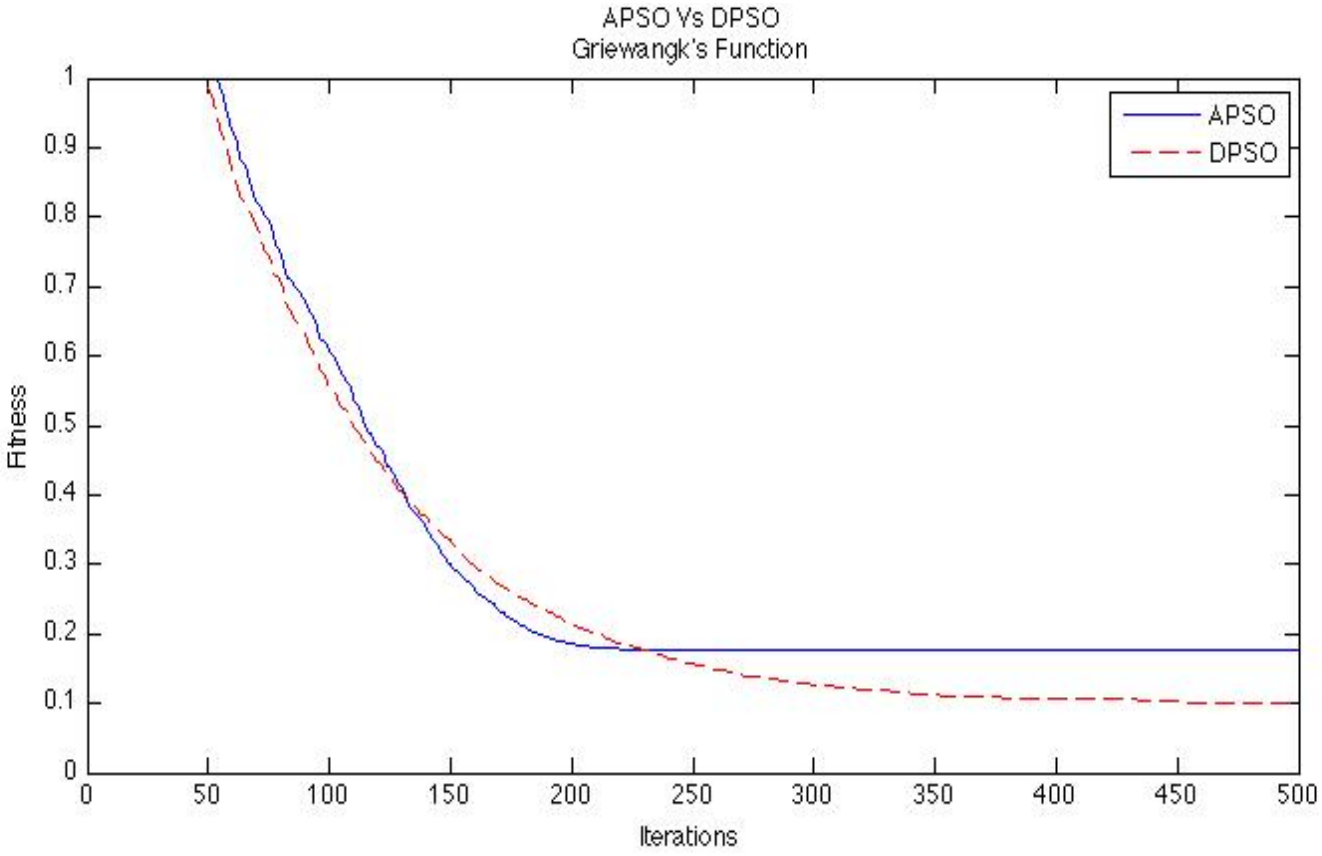


Εικόνα 15: Σύγκλιση του APSO και DPSO στην συνάρτηση Schwefel

Συνάρτηση 2: Griewangk's Function

Συνάρτηση	Αλγόριθμος	Μέσος Όρος	Μέγιστο	Τυπική Απόκλιση	Προσέγγιση Ολικού Ελαχίστου (< 0.05)
Griewangk	**APSO**	0.1767	0.5837	0.11	6 %
	DPSO	0.1045	0.3572	0.06	21.5 %

Πίνακας 13: Σύγκριση συγκλίσεων των APSO και DPSO

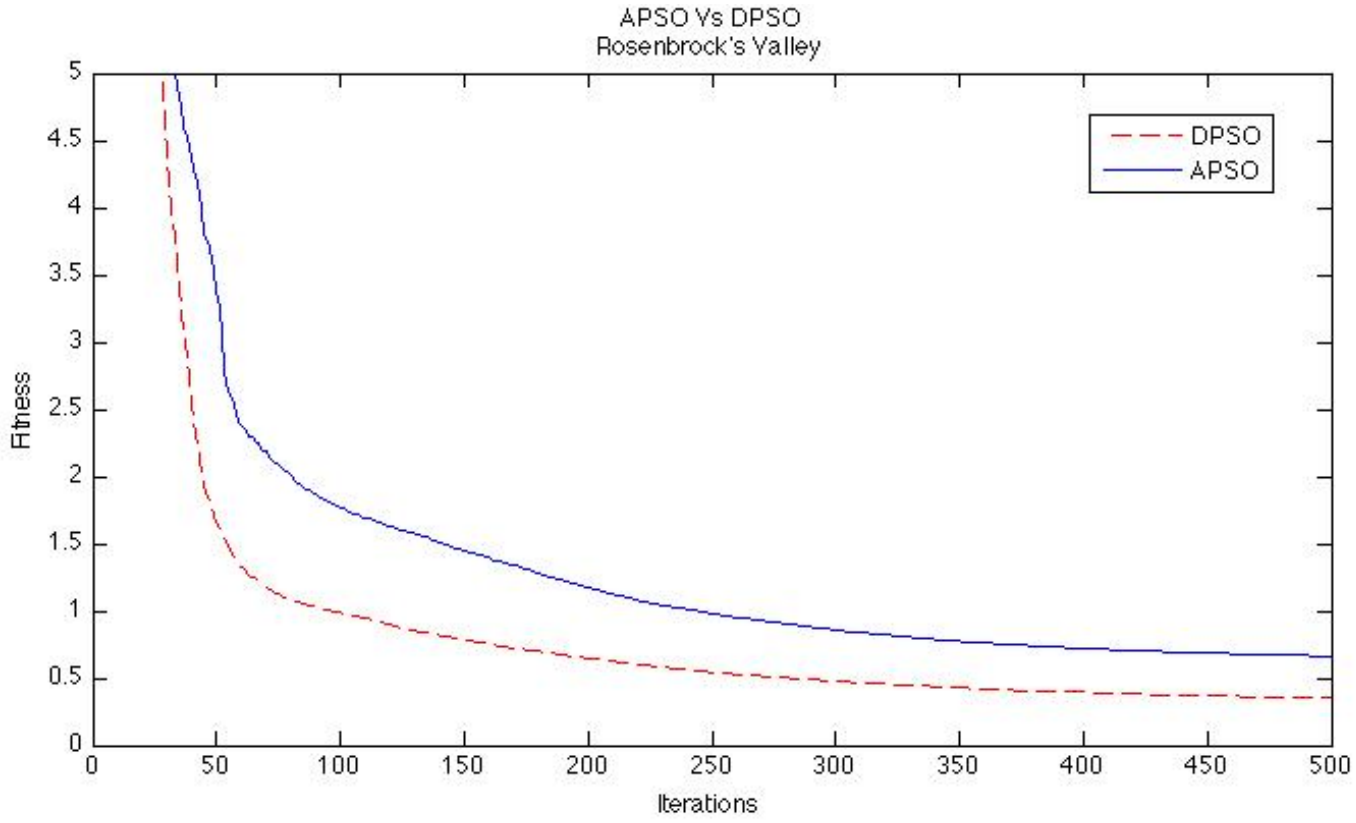


Εικόνα 16: Σύγκλιση του APSO και DPSO στην συνάρτηση Griewangk

Συνάρτηση 3: Rosenbrock's Function

Συνάρτηση	Αλγόριθμος	Μέσος Όρος	Μέγιστο	Τυπική Απόκλιση	Προσέγγιση Ολικού Ελαχίστου (< 0.005)
Rosenbrock	**APSO**	0.6634	4.3261	1.38	12%
	DPSO	0.3515	4.3209	1.06	19.5%

Πίνακας 14: Σύγκριση συγκλίσεων των APSO και DPSO

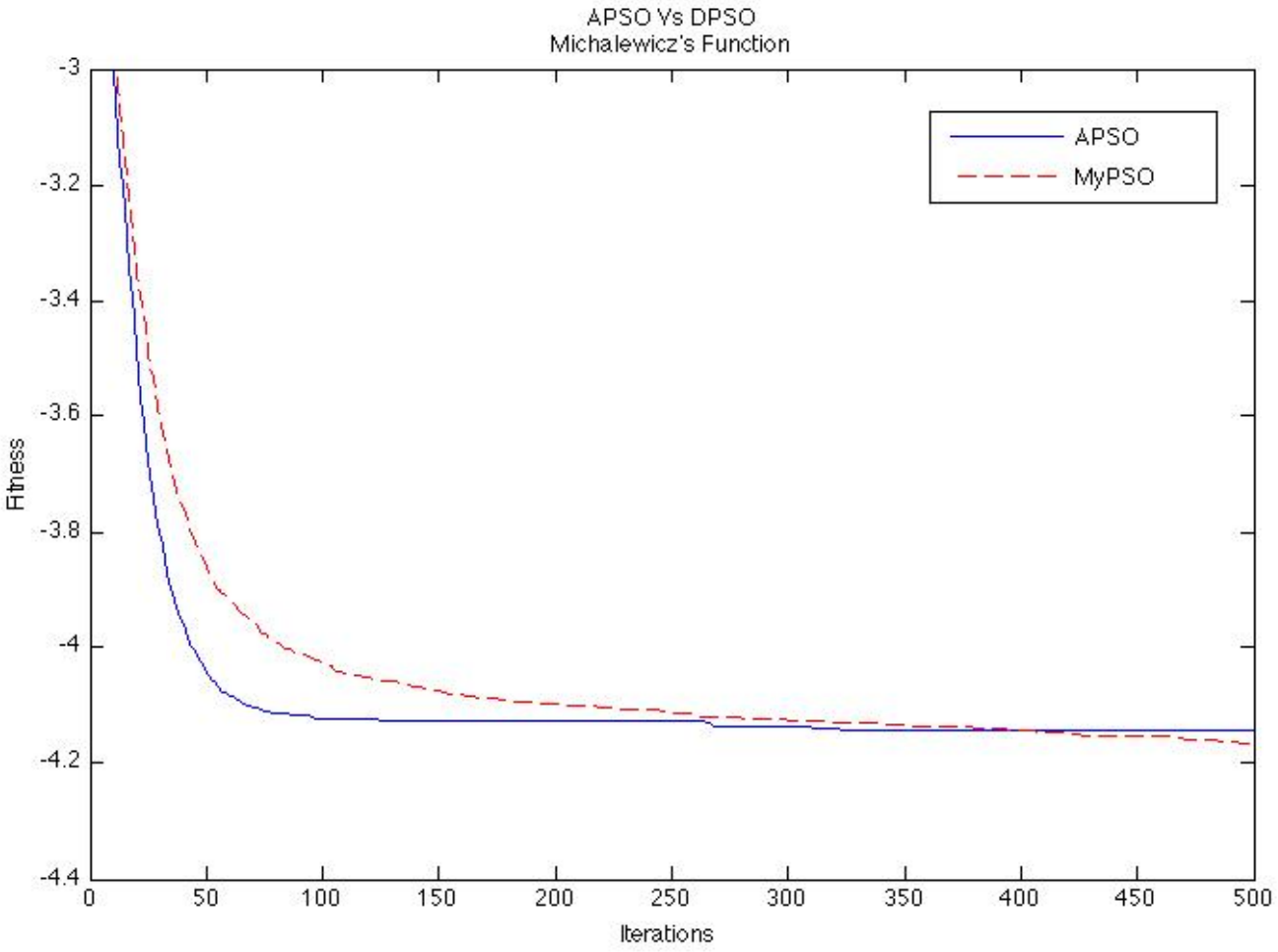


Εικόνα 17: Σύγκλιση του APSO και DPSO στην συνάρτηση Rosenbrock

Συνάρτηση 4: Michalewicz's Function

Συνάρτηση	Αλγόριθμος	Μέσος Όρος	Μέγιστο	Τυπική Απόκλιση	Προσέγγιση Ολικού Ελαχίστου (< -4.687)
Michalewicz	**APSO**	-4.1441	-2.2680	0.45	7.5%
	DPSO	-4.1641	-2.6571	0.43	4.5%

Πίνακας 15: Σύγκριση συγκλίσεων των APSO και DPSO

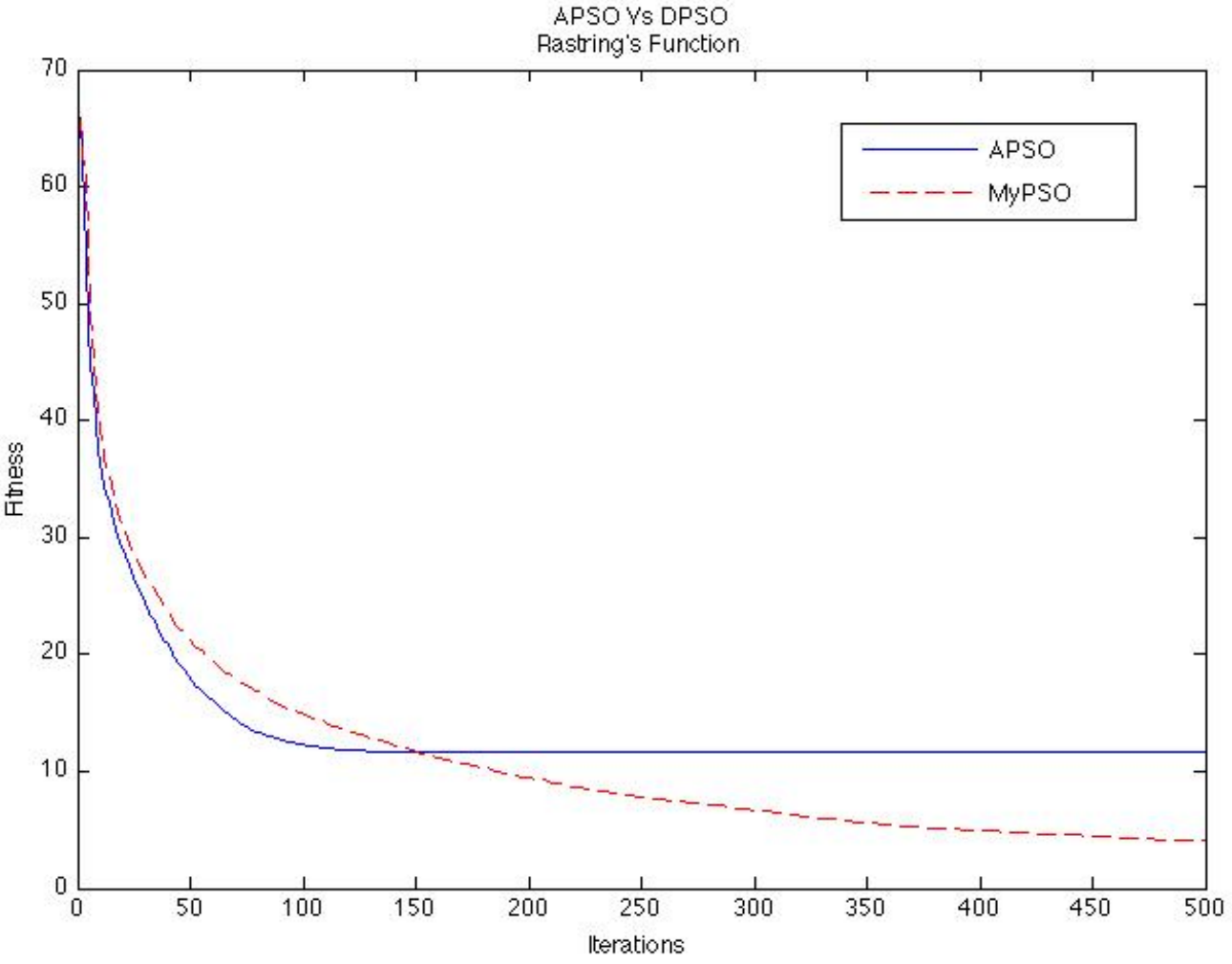


Εικόνα 18: Σύγκλιση του APSO και DPSO στην συνάρτηση Michalewicz

Συνάρτηση 5: Rastring's Function

Συνάρτηση	Αλγόριθμος	Μέσος Όρος	Μέγιστο	Τυπική Απόκλιση	Προσέγγιση Ολικού Ελαχίστου (< 1.5)
Rastring	**APSO**	11.6669	37.8082	7.06	1%
	DPSO	4.0263	34.9071	4.88	16%

Πίνακας 16: Σύγκριση συγκλίσεων των APSO και DPSO



Εικόνα 19: Σύγκλιση του APSO και DPSO στην συνάρτηση Rastring

Στους παραπάνω πίνακες και εικόνες φαίνονται τα αποτελέσματα των συγκρίσεων του DPSO ως προς αυτά του APSO για τις πέντε συναρτήσεις. Σε όλα τα προβλήματα φαίνεται καθαρά η υπεροχή του DPSO, όπου κάποιες φορές η διαφορά είναι μεγαλύτερη και κάποιες άλλες μικρότερη.

Σε κάποιες συναρτήσεις, όπως στην συνάρτηση Schwefel και την Rastring, ο DPSO εμφανίζει ιδιαιτέρως καλή συμπεριφορά ως προς τον APSO. Συνήθως έχει σταθερά καλύτερη σύγκλιση από τον APSO και μέχρι και το τέλος των πεντακοσίων επαναλήψεων βελτιώνει την επίδοσή του με σταθερό ρυθμό. Έτσι, φαίνεται πως με περισσότερες επαναλήψεις θα μπορούσε να βελτιώνει την επίδοσή του μέχρι την εύρεση του ολικού ελαχίστου. Αντίθετα, ο APSO φαίνεται να παγιδεύεται σε τοπικά ελάχιστα.

Ακόμα, όπως φαίνεται από τους πίνακες ο DPSO φτάνει κοντά στο ολικό ελάχιστο της κάθε συνάρτησης περισσότερες φορές από τον APSO, με εξαίρεση μόνο την συνάρτηση Michalewicz.

2.5.7 Συμπέρασμα

Στην παραπάνω ενότητα ο προτεινόμενος αλγόριθμος DPSO συγκρίθηκε με τον συχνότερα χρησιμοποιούμενο APSO. Οι συγκρίσεις δείχνουν πως ο προτεινόμενος αλγόριθμος υπερέχει του APSO τόσο ως προς την ακρίβεια (απόλυτο μέσο όρο) όσο και ως προς την επαναληψιμότητα.

Ακόμα, σημαντικό χαρακτηριστικό της λειτουργίας του DPSO είναι η ικανοποιητική αποφυγή των τοπικών ελαχίστων. Όπως φαίνεται στις γραφικές παραστάσεις των συγκλίσεων, σε κανένα από τα προβλήματα δεν εμφανίζεται κορεσμός στις βέλτιστες λύσεις. Έτσι, όσο αυξάνονται οι επαναλήψεις ο αλγόριθμος δείχνει να μπορεί να βρει συνεχώς καλύτερες λύσεις.

Μέρος Τρίτο

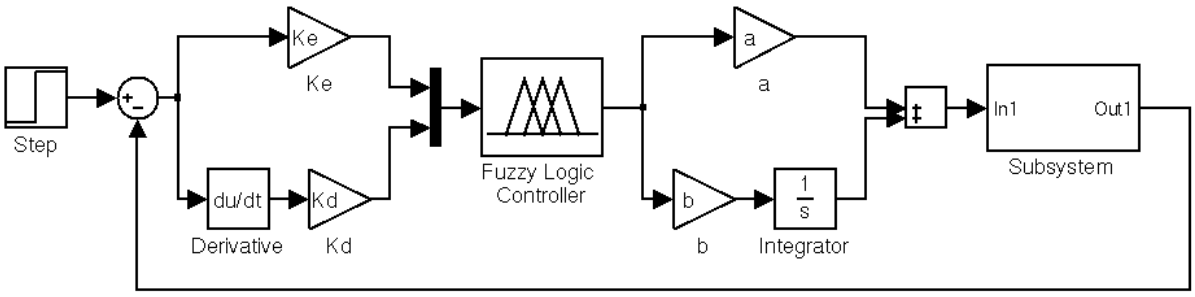
Εφαρμογές του PSO σε προβλήματα βελτιστοποίησης

3.1 Εισαγωγή

Ένας ελεγκτής μπορεί ρυθμιστεί ώστε να ελέγξει διαφορετικά συστήματα απλά και μόνο αλλάζοντας κάποιες τιμές που αφορούν τα εσωτερικά του χαρακτηριστικά. Έτσι, αλγόριθμοι βελτιστοποίησης όπως ο PSO έχουν κατά καιρούς προταθεί και χρησιμοποιηθεί σε προβλήματα αυτομάτου ελέγχου. Σε αυτά τα προβλήματα, αναζητάται ένας ελεγκτής ιδανικά ρυθμισμένος για να ελέγχει το σύστημα του εκάστοτε προβλήματος. Αυτό λοιπόν, αποτελεί ιδανική εφαρμογή για έναν αλγόριθμο βελτιστοποίησης.

Αναλόγως με τον ελεγκτή προς βελτιστοποίηση, οι διαστάσεις του προβλήματος διαφέρουν. Σε έναν PI, PD ή PID ελεγκτή, αυτές μπορεί να είναι το αναλογικό, ολοκληρωτικό και παραγωγικό του μέρος, δηλαδή ένα πρόβλημα μέχρι τριών διαστάσεων. Σε έναν πιο περίπλοκο ασαφή ελεγκτή, τα χαρακτηριστικά που επηρεάζουν την ποιότητα ελέγχου μπορεί να είναι δεκάδες. Μπορεί να είναι το σχήμα και οι θέσεις των συναρτήσεων συμμετοχής, οι κανόνες ή και οι συντελεστές πολλαπλασιασμού των σημάτων εξόδου και εισόδου.

Στα προβλήματα που ακολουθούν χρησιμοποιείται ένας ασαφής ελεγκτής σε μία διάταξη που ονομάζεται Ασαφής Ελεγκτής Τύπου PID (PID-Type Fuzzy Controller) που πρωτοπαρουσιάζεται από τους Qiao και Mizumoto [11]. Ένας τέτοιος ελεγκτής έχει μόνο τέσσερις μεταβλητές για βελτιστοποίηση και είναι οι τέσσερις συντελεστές στα σήματα εισόδου και εξόδου a, b, Ke και Kd, όπως φαίνονται παρακάτω.



Εικόνα 20: Υλοποίηση του PID-Type Fuzzy Controller

Στις επόμενες σελίδες ο PSO θα εφαρμοστεί για την βελτιστοποίηση του Ασαφή Ελεγκτή Τύπου PID σε δύο προβλήματα:

- ένα DC Drive Motor
- και ένα σύστημα πρώτης τάξης με καθυστέρηση

3.2 Πρώτη Εφαρμογή του PSO

3.2.1 Μέθοδος

Ο αλγόριθμος PSO μπαίνει στην πράξη για να βελτιστοποιήσει έναν Ασαφή Ελεγκτή Τύπου PID που ελέγχει ένα DC Motor. Το ηλεκτρικό αυτό μοτέρ είναι αυτό που χρησιμοποιείται και στο [14] και περιγράφεται από την παρακάτω συνάρτηση μεταφοράς:

$$G(s) = \frac{k_m}{(1+\tau_m s)(1+\tau_e s)}$$

Οι τιμές των παραμέτρων του είναι:

Παράμετρος	Τιμή
k_m	0.05
τ_m	300 ms
τ_e	14 ms

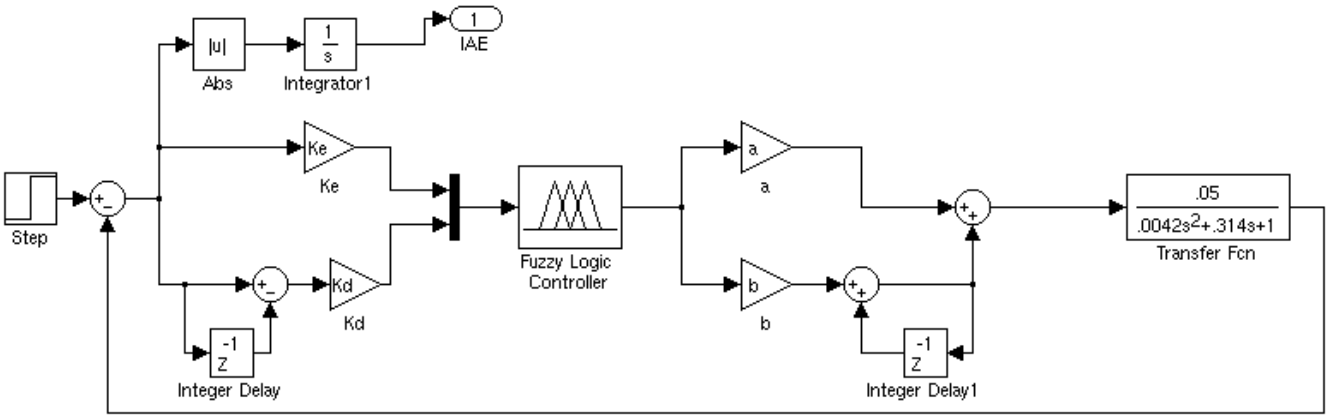
Έτσι, η συνάρτηση μεταφοράς γίνεται:

$$G(s) = \frac{0.05}{0.0042s^2 + 0.314s + 1}$$

Για τη βελτιστοποίηση του συστήματος χρησιμοποιήθηκε το κριτήριο IAE (Integral Absolute Error) σε συνδυασμό με κάποιους άλλους όρους. Το κριτήριο IAE, όπως δηλώνει και το όνομά του, κρίνει την ποιότητα του ελέγχου με βάση το ολοκλήρωμα του απόλυτου σφάλματος ελέγχου:

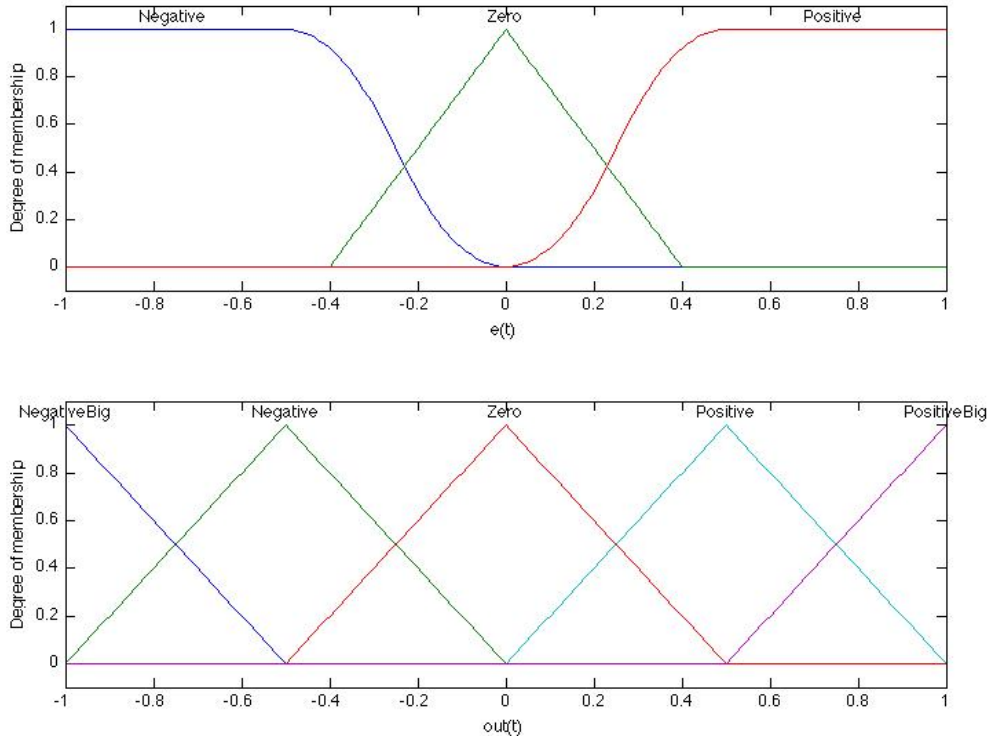
$$IAE = \int |u(t) - y(t)|\, dt$$

Έτσι, το σύστημα ελέγχου φτιαγμένο στο Simulink, συμπεριλαμβανομένου του κριτηρίου IAE ακολουθεί στην παρακάτω εικόνα.



Εικόνα 21: Το σύστημα ελέγχου του DC μοτέρ

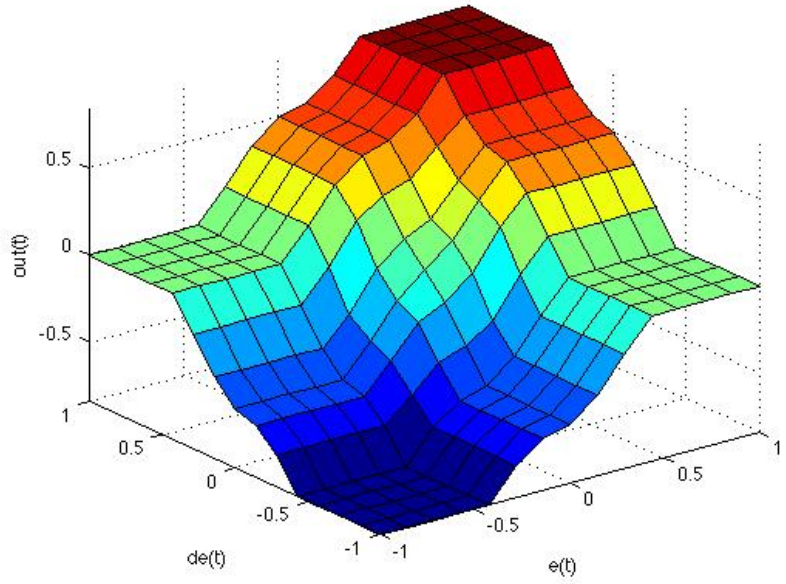
Ο ασαφής ελεγκτής που χρησιμοποιήθηκε είχε δυο πανομοιότυπες εισόδους και μια έξοδο με τρεις και με πέντε συναρτήσεις συμμετοχής αντίστοιχα. Οι συναρτήσεις συμμετοχής αυτών φαίνονται στην εικόνα 21. Οι ασαφείς κανόνες ακολουθούν στον πίνακα στην εικόνα 22 και η επιφάνεια ελέγχου που προκύπτει στην εικόνα 23. Τέλος, χρησιμοποιήθηκε product-sum inference method και η ασαφοποίηση έγινε με την μέθοδο του κέντρου βάρους.



Εικόνα 22: Συναρτήσεις συμμετοχής εισόδων (πάνω) και εξόδου (κάτω)

Είσοδος 1/2	**Αρνητική**	**Μηδενική**	**Θετική**
Αρνητική	*Μεγάλη Αρνητική*	*Αρνητική*	*Μηδενική*
Μηδική	*Αρνητική*	*Μηδενική*	*Θετική*
Θετική	*Μηδενική*	*Θετική*	*Μεγάλη Θετική*

Πίνακας 17: Οι ασαφείς κανόνες της εξόδου του ελεγκτή



Εικόνα 23: Η επιφάνεια εξόδου του Ασαφούς Ελεγκτή

Για την βελτιστοποίηση του παραπάνω ελεγκτή στο σύστημα της εικόνας 20, χρησιμοποιήθηκε ο APSO αλγόριθμος καθώς και ο προτεινόμενος αλγόριθμος DPSO με ελαφρώς τροποποιημένο σετ τιμών:

α	***β***	***γ***	***δ***	***ε***
0.8	0.5	0.27	0.8	0.2

Πίνακας 18: Σετ τιμών του DPSO που χρησιμοποιήθηκε στην πρώτη εφαρμογή

Για επιβεβαίωση των αποτελεσμάτων, οι δύο αλγόριθμοι κλήθηκαν να λύσουν το πρόβλημα 20 φορές και τα αποτελέσματα τους φαίνονται και συγκρίνονται παρακάτω. Όλες οι προσομοιώσεις έγιναν με τους ίδιους όρους για να είναι άμεσα συγκρίσιμες. Συγκεκριμένα, τα σμήνη που χρησιμοποιήθηκαν αποτελούνταν από 30

σωματίδια και η κάθε μία από τις 20 προσπάθειες ήταν ρυθμισμένη στις 40 επαναλήψεις. Ακόμα, ο χρόνος για τον οποίο έτρεχε η προσομοίωση στο Simulink ήταν 0.5 δευτερόλεπτα και ο χρόνος δειγματοληψίας 0.0005. Τέλος, το πεδίο στο οποίο γινόταν η έρευνα για τις βέλτιστες τιμές ήταν το [0 500] και το κριτήριο για το οποίο έγινε η βελτιστοποίηση ήταν ένας συνδυασμός του κριτηρίου IAE και της μέγιστης και ελάχιστης τιμής ταλάντωσης στην βηματική απόκριση. Συγκεκριμένα, οι επιδόσεις του ελεγκτή στο σύστημα κρίνονταν από την εξίσωση:

$$fitness = 2 * IAE + 0.2 * (SettlingMax - SettlingMin)$$

Όπου *IAE* το ολοκλήρωμα του απολύτου σφάλματος, *SettlingMax* η μέγιστη υπερύψωση και *SettlingMin* η μέγιστη βύθιση στην μόνιμη κατάσταση. Η αφαίρεση των δύο αυτών όρων μεταξύ τους, δείχνει πόσο κοντά στο Set Point βρίσκεται η απόκριση του συστήματος.

Η επιλογή της παραπάνω συνδυαστικής αντικειμενικής συνάρτησης προέκυψε μετά από πειραματισμό και έγινε για να πετύχει δύο στόχους. Πρώτον, την ελαχιστοποίηση των ταλαντώσεων στην μόνιμη κατάσταση για επίτευξη μηδενικού μόνιμου σφάλματος και δεύτερον το να επιτευχθεί μικρός χρόνος αποκατάστασης.

3.2.2 Αποτελέσματα

Κάθε αλγόριθμος εκτελέστηκε 20 φορές για την βελτιστοποίηση του προβλήματος του DC μοτέρ. Τα αποτελέσματα (fitness) που προέκυψαν από τις 20 εκτελέσεις και των δύο μεθόδων συγκεντρώθηκαν και κάποια στατιστικά στοιχεία τους εμφανίζονται στον παρακάτω πίνακα. Συγκεκριμένα φαίνονται οι καταλληλότερες καθώς και χειρότερες λύσεις που βρήκε ο κάθε αλγόριθμος, ο αριθμητικός μέσος των 20 αποτελεσμάτων, και η τυπική απόκλισή τους.

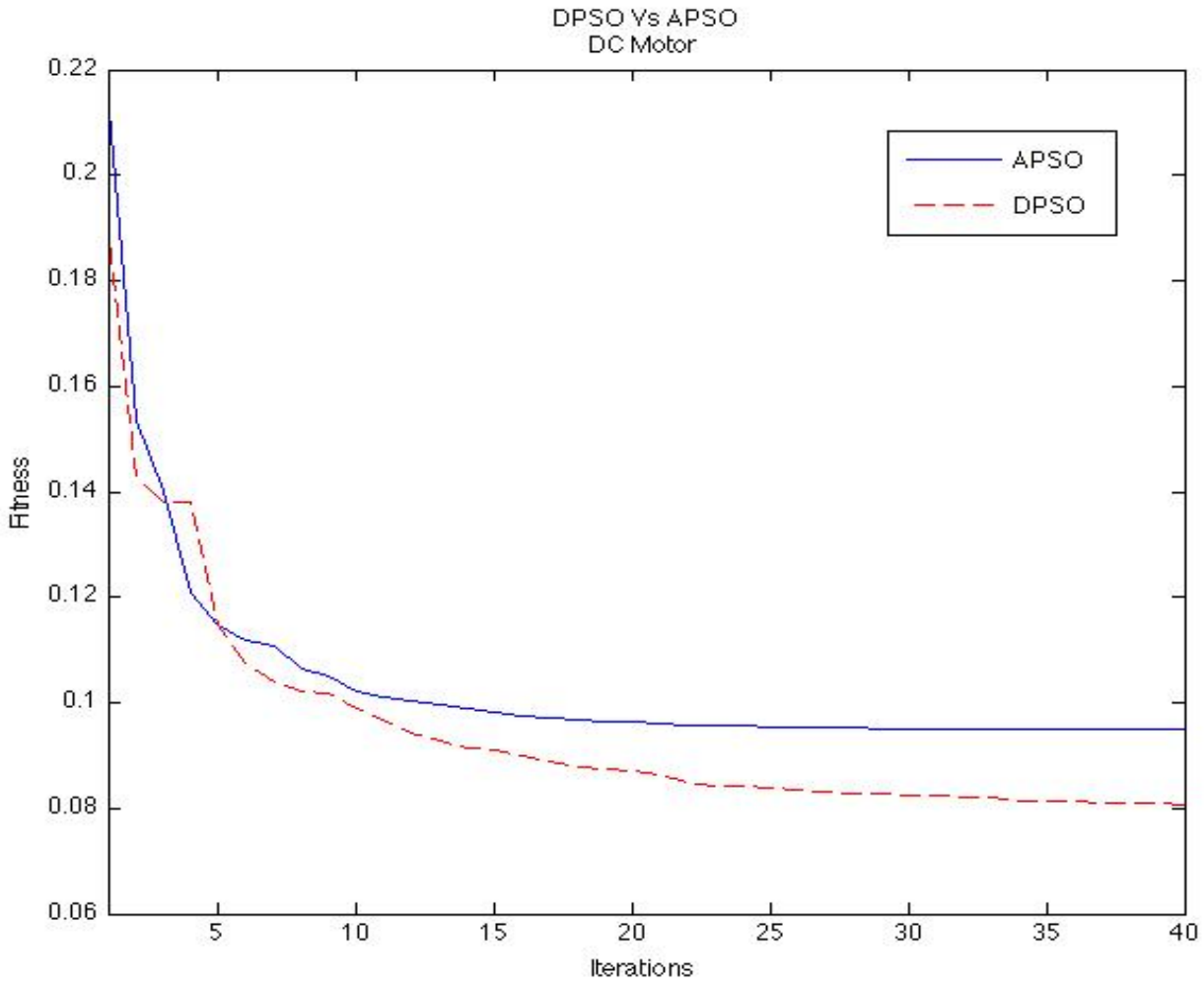
Αλγόριθμος	Καλύτερο αποτέλεσμα	Μέσος Όρος	Χειρότερο αποτέλεσμα	Std.Deviation
APSO	0.0627	0.0948	0.1306	0.0153
DPSO	0.0555	0.0808	0.1062	0.0193

Πίνακας 19: Αποτελέσματα των δύο αλγορίθμων στην πρώτη εφαρμογή

Στον παραπάνω πίνακα γίνεται εμφανής η υπεροχή του DPSO σε σύγκριση με τον APSO. Ο μέσος όρος των 20 προσπαθειών στην περίπτωση του APSO είναι 0.0948, ενώ ο DPSO εμφανίζει μέση τιμή 0.0808, δηλαδή βελτίωση της τάξης του14.8%.

Στην επόμενη εικόνα εμφανίζεται η βελτίωση της ρύθμισης των τεσσάρων συντελεστών στην πορεία των επαναλήψεων, και για τους δύο αλγόριθμους. Κατά την πορεία των επαναλήψεων, και συγκεκριμένα μετά το δεύτερο μισό, η σύγκλιση του APSO φαίνεται να σταθεροποιείται και να μην επιδέχεται περαιτέρω βελτίωση.

Αντιθέτως, ο DPSO συνεχίζει να βελτιώνει την επίδοσή του με σταθερό ρυθμό. Ακόμα, από την πέμπτη κιόλας επανάληψη η απόδοση του DPSO γίνεται καλύτερη από αυτήν του APSO και από εκεί και μετά βελτιώνεται συνεχώς.

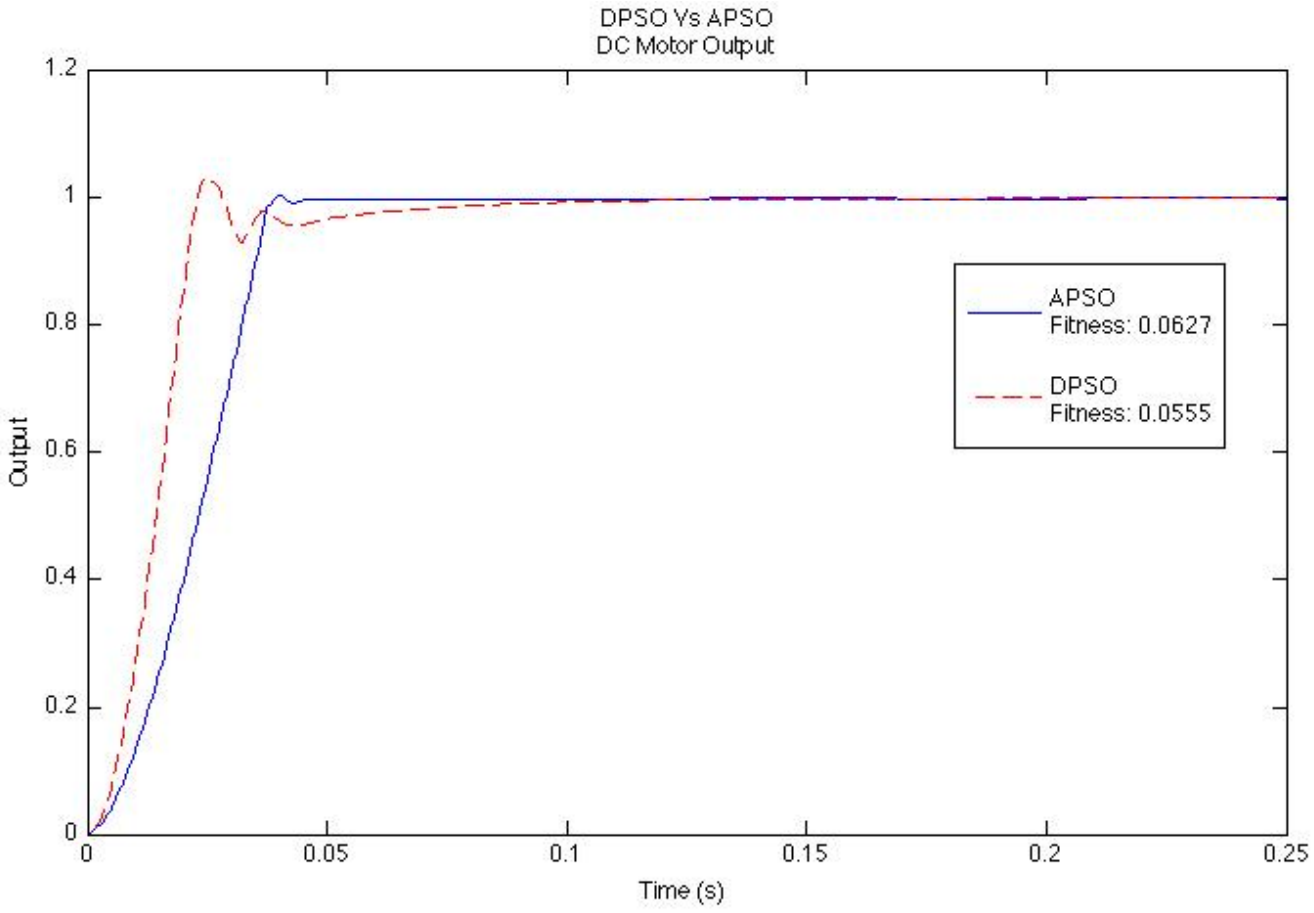


Εικόνα 24: Η σύγκλιση των APSO και DPSO στην πορεία των 40 επαναλήψεων

Για την απεικόνιση της βηματικής απόκρισης του συστήματος του DC κινητήρα χρησιμοποιήθηκαν οι βέλτιστες τιμές που πέτυχαν και οι δύο αλγόριθμοι. Στο πίνακα φαίνονται οι βέλτιστες τιμές των Ke, Kd, a και b και στην εικόνα οι βηματικές αποκρίσεις. Είναι εμφανές ότι με το κριτήριο που χρησιμοποιήθηκε γίνεται προτίμηση ενός γρήγορου χρόνου αποκατάστασης σε βάρος της υπερύψωσης.

Αλγόριθμος	**Ke**	**Kd**	**α**	**β**	**Fitness**
APSO	18.84	421.02	500	292.94	0.0627
DPSO	1.13	70.62	500	323.65	0.0555

Πίνακας 20: Βέλτιστες ρυθμίσεις των συντελεστών του ελεγκτή κατά τον APSO και DPSO



Εικόνα 25: Βηματική απόκριση βέλτιστων αποτελεσμάτων

Στην εικόνα 25 φαίνονται οι βέλτιστες βηματικές αποκρίσεις του συστήματος για τους δύο αλγορίθμους. Υπάρχουν δύο βασικές διαφορές μεταξύ των δύο αποκρίσεων.

Πρώτον, ο DPSO φαίνεται να έχει αρκετά μικρότερο χρόνο αποκατάστασης σε σχέση με τον APSO. Έτσι, απέσπασε καλύτερο σκορ στο κριτήριο IAE.

Δεύτερον, ο APSO εμφανίζει μια πιο ομαλή μετάβαση στη μόνιμη κατάσταση χωρίς τις μεγάλες διακυμάνσεις του DPSO, πλεονεκτώντας του DPSO. Προφανώς όμως, με το παρόν κριτήριο, το πλεονέκτημα αυτό του APSO δεν φαίνεται να είναι αρκετά σημαντικό σε σχέση με τον χρόνο αποκατάστασης. Έτσι, η βηματική απόκριση του APSO «τιμωρείται» για την αργή κλίση ανόδου της με καταλληλότητα *(fitness)* χαμηλότερη αυτής του DPSO.

3.3 Δεύτερη Εφαρμογή το PSO

3.3.1 Μέθοδος

Σαν δεύτερη εφαρμογή για τον PSO επιλέχτηκε ένα σύστημα πρώτου βαθμού με καθυστέρηση. Το ίδιο σύστημα εμφανίζεται και στο [14] και περιγράφει ένα «απλό πιστολάκι». Συγκεκριμένα, αποτελείται από έναν ανεμιστήρα που, μέσα από έναν σωλήνα, φυσά αέρα με σταθερή ροή. Ο αέρας αυτός ζεσταίνεται από μια αντίσταση που βρίσκεται μέσα στο σωλήνα. Το ζητούμενο του προβλήματος εδώ, είναι να ελέγχεται η τάση στην αντίσταση έτσι ώστε η θερμοκρασία του αέρα στην έξοδο να είναι η επιθυμητή. Το πρωτοβάθμιο αυτό μοντέλο περιγράφεται από την συνάρτηση μεταφοράς:

$$G(s) = \frac{k_m}{1+\tau s} e^{-\tau_d s}$$

Όπου,

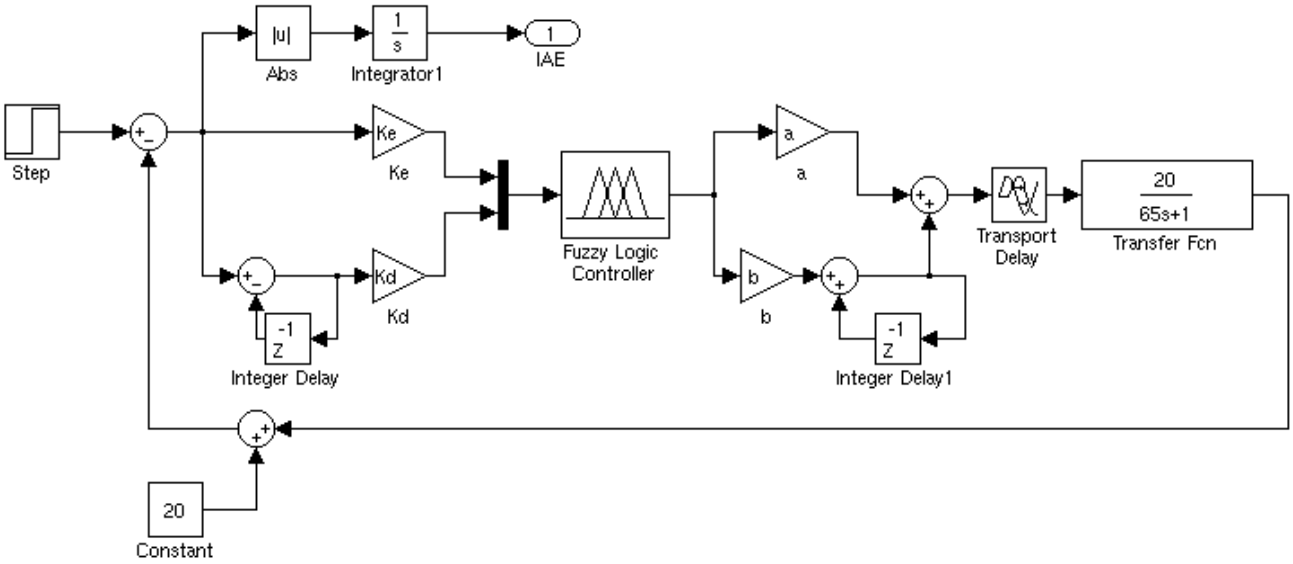
Παράμετρος	Τιμή
k_m	20
τ	65 s
τ_d	1 s

Έτσι, το μοντέλο του συστήματος είναι το:

$$G(s) = \frac{20}{1+65s} e^{-s}$$

Για την δεύτερη εφαρμογή, παρότι το σύστημα είναι διαφορετικό από αυτό της πρώτης, χρησιμοποιήθηκε ο ίδιος ελεγκτής. Ίδια μέθοδος ασαφοποίησης και αποασαφοποίησης, ίδιες συναρτήσεις συμμετοχής και ίδιοι κανόνες.

Το σύστημα που προσομοιώθηκε φαίνεται στην επόμενη εικόνα.



Εικόνα 26: Το σύστημα ελέγχου πρώτου βαθμού με καθυστέρηση

Το παραπάνω σύστημα αποτελείται από τον PID-Type Fuzzy ελεγκτή με τους τέσσερις συντελεστές του, το κριτήριο IAE, το σύστημα με την καθυστέρηση του, μια βηματική είσοδο και μία σταθερά. Αυτή η σταθερά που προστίθεται στην έξοδο του συστήματος συμβολίζει την θερμοκρασία του χώρου, την θερμοκρασία του αέρα δηλαδή που εισέρχεται στο «πιστολάκι» για να θερμανθεί. Αυτή είναι ρυθμισμένη στους 20^{o}C, ενώ η θερμοκρασία-στόχος είναι οι 50^{o}C. Ο χρόνος της προσομοίωσης είναι 60 δευτερόλεπτα, ο χρόνος δειγματοληψίας τα 0,2 δευτερόλεπτα και ο χώρος των λύσεων είναι ο [0 100]. Οι συντελεστές προς βελτιστοποίηση παραμένουν οι συντελεστές εισόδου Ke, Kd και εξόδου α, β.

Η αντικειμενική συνάρτηση – κριτήριο που χρησιμοποιήθηκε για την βελτιστοποίηση των τεσσάρων συντελεστών και σε αυτό το πρόβλημα είναι συνδυαστικό. Συγκεκριμένα, είναι ένας συνδυασμός του κριτηρίου IAE και των και της μέγιστης και ελάχιστης τιμής ταλάντωσης στην βηματική απόκριση και δίνεται από την εξίσωση:

$$fitness = IAE + 3 * (SettlingMax - SettlingMin)$$

Όπου όπως και στο προηγούμενο πρόβλημα, *IAE* είναι το ολοκλήρωμα του απολύτου σφάλματος, *SettlingMax* η μέγιστη υπερύψωση και *SettlingMin* η μέγιστη βύθιση

στην μόνιμη κατάσταση. Ο συντελεστής 3 σε αυτή την περίπτωση δηλώνει την σημαντικότητα της μη ύπαρξης υπερύψωσης αλλά ούτε και ταλαντώσεων στην μόνιμη κατάσταση.

Πρέπει να σημειωθεί ότι την συγκεκριμένη περίπτωση αρχικά η έξοδος του συστήματος σημείωνε έντονες, απαράδεκτες ταλαντώσεις. Αυτό συνέβαινε λόγω της απότομης μεταβολής των εισόδων του ελεγκτή από τιμές μεγαλύτερες του 1 σε τιμές μικρότερες του -1. Αυτό σήμαινε πως ο ελεγκτής λειτουργούσε συνεχώς στις δύο ακραίες εξόδους του, -1 και 1, χωρίς να γίνεται χρήση των υπολοίπων συναρτήσεων συμμετοχής. Για να διορθωθεί αυτό το πρόβλημα, έγινε κατανοητό πως οι τιμές των Ke και Kd έπρεπε να είναι αρκετά μικρές.

Έτσι, αντί τα σήματα εισόδου του PID-Type Fuzzy ελεγκτή να πολλαπλασιάζονται με Ke και Kd, πολλαπλασιάζονταν με 1/Ke και 1/Kd.

Τέλος, για την βελτιστοποίηση του ελεγκτή αυτού του συστήματος με τον DPSO χρησιμοποιήθηκε αυτούσιο το σετ τιμών-βάση που προτάθηκε παραπάνω:

α	β	γ	δ	ε
0.8	0.5	0.27	0.8	0.2

Πίνακας 21: Σετ τιμών του DPSO που χρησιμοποιήθηκε στην δεύτερη εφαρμογή

3.3.2 Αποτελέσματα

Τα αποτελέσματα των δύο αλγορίθμων μετά από την εκτέλεση των 20 επαναλήψεις εμφανίζονται στον παρακάτω πίνακα:

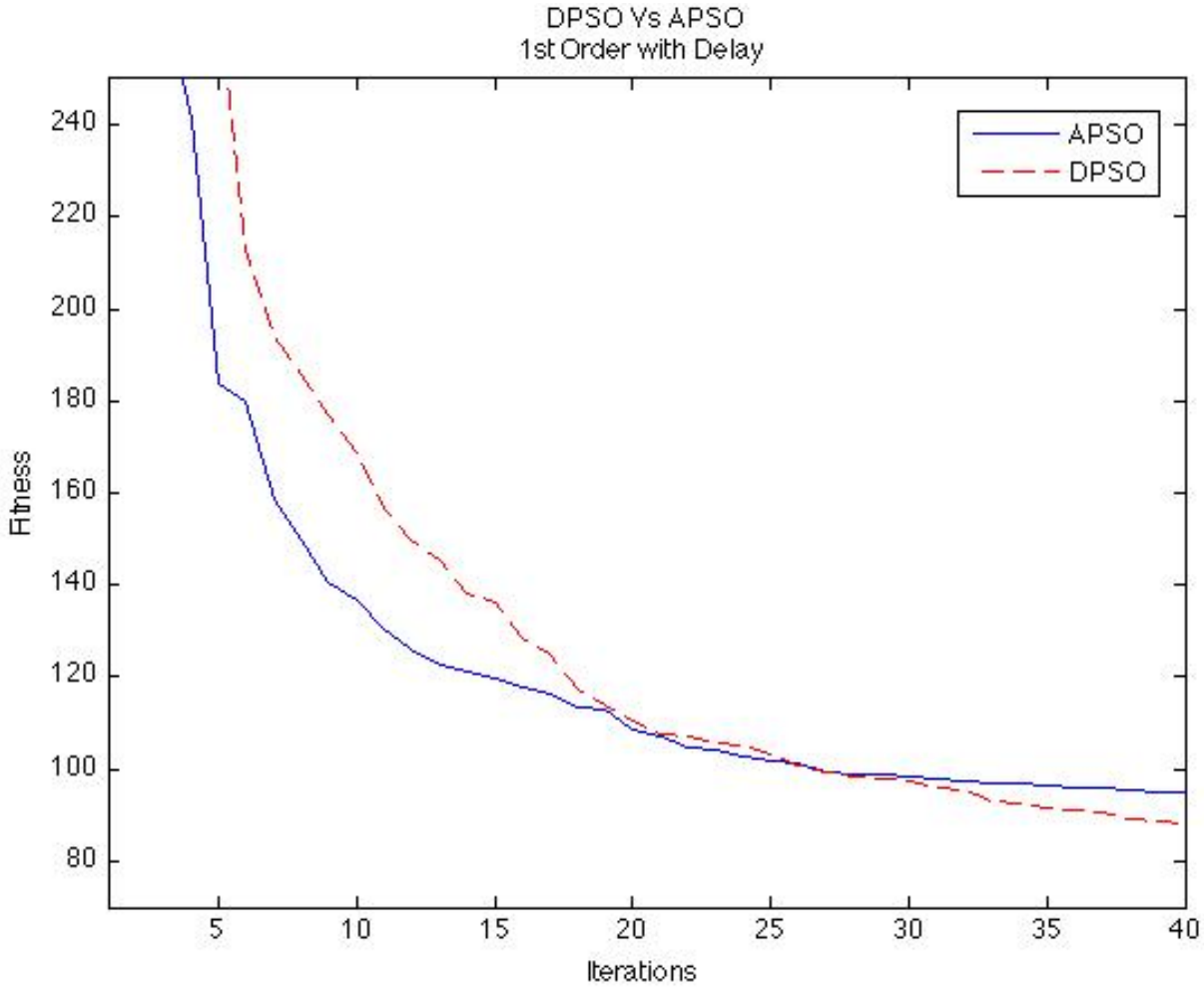
Αλγόριθμος	Ελάχιστο	Μέσος Όρος	Μέγιστο	Std.Deviation
APSO	72.1142	94.9413	190.0944	25.0812
DPSO	71.4466	88.2188	104.4150	8.0566

Πίνακας 22: Αποτελέσματα των δύο αλγορίθμων στην δεύτερη εφαρμογή

Σε αυτό το σύστημα και χωρίς να γίνει καμία τροποποίηση στο προτεινόμενο σετ τιμών, τα αποτελέσματα του DPSO υπερέχουν αυτών του APSO. Εντυπωσιακή είναι

η διαφορά στην τυπική απόκλιση των δύο αλγορίθμων και η συνολικά καλύτερη εικόνα του DPSO.

Στην παρακάτω εικόνα φαίνεται η σύγκλιση των δύο αλγορίθμων. Αρχικά και μέχρι περίπου τις μισές επαναλήψεις, ο DPSO παρουσιάζει χειρότερη απόδοση από τον APSO, αλλά στην πορεία ο DPSO καταλήγει να είναι καλύτερος. Ακόμα, ενώ προς το τέλος των 40 επαναλήψεων ο APSO τείνει να μην βελτιώνεται περαιτέρω, ο DPSO βελτιώνεται δείχνοντας πως με λίγες περισσότερες επαναλήψεις θα έφερνε ακόμα καλύτερα αποτελέσματα.



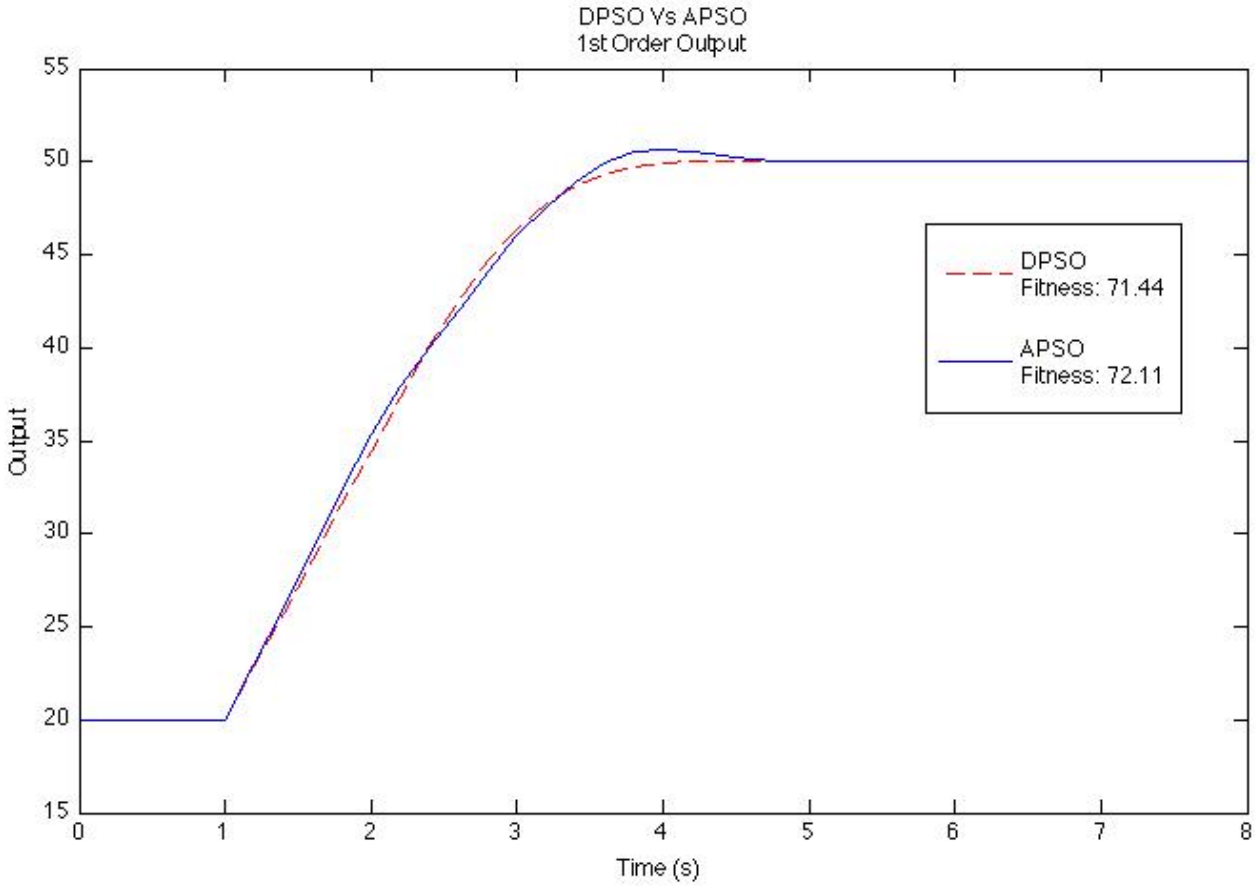
Εικόνα 27: Η σύγκλιση των APSO και DPSO στην πορεία των 40 επαναλήψεων

Οι βηματικές αποκρίσεις που φαίνονται στην παρακάτω εικόνα αντιστοιχούν στις βέλτιστες λύσεις που πέτυχαν οι δύο αλγόριθμοι. Συγκεκριμένα, οι τιμές που

αντιστοιχούν στους τέσσερις συντελεστές προς βελτιστοποίηση και τα αντίστοιχα σκορ με βάση το κριτήριο φαίνονται στον παρακάτω πίνακα.

Αλγόριθμος	Ke	Kd	α	β	Fitness
APSO	0.0247	0.055	100	0.2745	72.1142
DPSO	0.0162	0.0168	94.26	0.2595	71.4466

Πίνακας 23: Βέλτιστες ρυθμίσεις των συντελεστών του ελεγκτή κατά τον APSO και DPSO



Εικόνα 28: Βηματική απόκριση βέλτιστων αποτελεσμάτων

Στον παραπάνω πίνακα και την εικόνα φαίνονται οι τιμές και αποκρίσεις που αντιστοιχούν στις βέλτιστες αποδόσεις του DPSO και του APSO. Φαίνεται μια μικρή μόνο διαφορά μεταξύ των σκορ των δύο αλγορίθμων, που οφείλεται στην ελαφρά υπερύψωση που εμφανίζει η βηματική απόκριση του APSO. Καθώς όμως το κριτήριο με το οποίο έγινε η αξιολόγηση τιμωρεί έντονα τις υπερυψώσεις, η καταλληλότητα *(fitness)* του DPSO είναι καλύτερη.

Μέρος Τέταρτο

Συμπεράσματα

4.1 Εισαγωγή

Στις παραπάνω σελίδες παρουσιάστηκε, υλοποιήθηκε, μελετήθηκε και εξελίχθηκε μια μέθοδος βελτιστοποίησης. Ο αλγόριθμος PSO αποτελεί μία ακόμα μέθοδο βελτιστοποίησης βασισμένη στην παρατήρηση της φύσης και αντιγραφή της δράσης διαφόρων πληθυσμών και διεργασιών της.

Συγκεκριμένα, ο PSO παρομοιάζει τον τρόπο με τον οποίο ένα σμήνος πουλιών ή ένα κοπάδι ψαριών κινούνται αρμονικά στον χώρο. Έτσι, όπως και τα πουλιά ή τα ψάρια «μαζεύονται» γύρω από την τροφή τους, έτσι και τα «σωματίδια» του PSO συγκεντρώνονται γύρω από τις λεγόμενες βέλτιστες λύσεις. Η ιδέα της μίμησης αυτής οδήγησε στην δημιουργία ενός γρήγορου και αξιόπιστου εξελικτικού αλγορίθμου βελτιστοποίησης. Από τον πρώτο καιρό της εμφάνισής του, πολλοί ερευνητές έχουν ασχοληθεί με την μελέτη των δυνατοτήτων και πιθανών εφαρμογών του αλγορίθμου. Κάτι ανάλογο επιχειρήθηκε και στα πλαίσια της παρούσας πτυχιακής εργασίας.

4.2 PSO και νέος DPSO

Στην ενότητα 2.5 έγινε παρουσίαση και ανάπτυξη μιας νέας εκδοχής του αλγορίθμου που αναπτύχθηκε στα πλαίσια της παρούσας πτυχιακής εργασίας. Εμπνευσμένη από τον αυτόματο έλεγχο σε διακριτά συστήματα, παρομοιάζει τον PSO ως έναν ελεγκτή σε σύστημα κλειστού βρόχου. Αυτός ο ελεγκτής, όπως γίνεται και στους διακριτούς ελεγκτές [17], χρησιμοποιεί πρόσθετα παρελθόντα στοιχεία για τον υπολογισμό των ταχυτήτων των σωματιδίων, κάτι που ο απλός SPSO δεν κάνει. Η μόνη διαφορά του έγκειται σε αυτή την λεπτομέρεια, αλλά καταφέρνει να αποδώσει καλύτερα αποτελέσματα σε σχέση με την πιο αξιόπιστη και πλέον συνηθισμένη έκδοση του αλγορίθμου, τον APSO. Αυτό συμπεραίνεται από τις δοκιμές που εκτελέστηκαν στα πλαίσια της εργασίας, και παρουσιάζονται ξανά παρακάτω.

Αρχικά, η επιδόσεις του DPSO συγκρίθηκαν με αυτές του APSO στη λύση πέντε κλασικών συναρτήσεων. Αυτές ονομαστικά είναι οι:

1	Schwefel’s Function
2	Griewangk’s Function
3	Rosenbrock's Valley
4	Michalewicz’s Function
5	Rastring’s Function

Πίνακας 24: Κλασικές συναρτήσεις που χρησιμοποιήθηκαν για την σύγκριση του DPSO με τον APSO

Τα αποτελέσματα των δύο αλγορίθμων, σύμφωνα με την διαδικασία που περιγράφεται στην ενότητα 2.5.6 φαίνονται συνοπτικά παρακάτω.

Συνάρτηση	**APSO**	**DPSO**	**Βελτίωση**
Schwefel	-339.05	-390.49	*15.2 %*
Griewangk	0.1767	0.1045	*40.9 %*
Rosenbrock	0.6634	0.3515	*47.0 %*
Michalewicz	-4.1441	-4.1641	*0.5 %*
Rastring	11.6669	4.0263	*65.5 %*

Πίνακας 25: Μέσες τιμές συγκλίσεων των δύο αλγορίθμων και ποσοστική βελτίωση

Όπως φαίνεται στον παραπάνω πίνακα, σε όλες τις περιπτώσεις ο DPSOυπερτερεί του APSO, κάποιες φορές εντυπωσιακά, κάποιες άλλες όχι. Σημαντικό όμως χαρακτηριστικό είναι το ότι τα αποτελέσματα προκύπτουν σε ευρύ φάσμα προβλημάτων, κάτι που φαίνεται και στην εφαρμογή του αλγορίθμου DPSO και σε προβλήματα βελτιστοποίησης Ασαφών Ελεγκτών. Στον παρακάτω πίνακα συνοψίζονται τα αποτελέσματα των εν λόγω δοκιμών που αναλυτικά παρουσιάζονται στο Τρίτο Μέρος της εργασίας.

Σύστημα - Πρόβλημα	APSO	DPSO	Βελτίωση
1ης Τάξης με Καθ/ηση	94.9413	88.2188	7.1 %
2ης Τάξης	0.0948	0.0808	14.8 %

Πίνακας 26: Αποτελέσματα των δύο αλγορίθμων στα δύο συστήματα που εξετάστηκαν

Και εδώ βάσει των δοκιμών που εκτελέστηκαν, φαίνεται μια ικανοποιητική βελτίωση της απόδοσης του αλγορίθμου.

Βάσει των παραπάνω δοκιμών τόσο σε δοκιμαστικές συναρτήσεις, όσο και σε προσομοιώσεις συστημάτων, ο νέος προτεινόμενος αλγόριθμος DPSO εμφανίζεται να υπερτερεί έναντι του κλασικού APSO.

4.3 Οι παράμετροι του PSO

Παρακάτω, αναλύονται οι παράμετροι που επηρεάζουν την λειτουργία και αποτελεσματικότητα του PSO. Οι προτεινόμενες επιλογές και ρυθμίσεις είναι αποτέλεσμα δοκιμών και προκύπτουν από ανασκόπηση της βιβλιογραφίας.

4.3.1 Σωματίδια και Επαναλήψεις

Η λειτουργία του PSO, όπως αναφέρθηκε και παραπάνω, βασίζεται σε έναν αριθμό σωματιδίων που «πετάνε», προσπαθώντας να βρουν την βέλτιστη λύση. Η πτήση τους αυτή διαρκεί για συγκεκριμένο χρονικό διάστημα, που ορίζεται από τις επαναλήψεις του αλγορίθμου. Έτσι, προκύπτουν δύο μεταβλητές, εξαρτώμενες από τον χρήστη, που μπορούν να επηρεάσουν την απόδοση του αλγορίθμου: ο αριθμός των σωματιδίων και ο αριθμός των επαναλήψεων. Όπως φαίνεται στην ενότητα 2.3.6, η αλλαγή των δύο παραπάνω παραμέτρων επιφέρει αλλαγές και στις επιδόσεις του PSO. Αυξάνοντας τον αριθμό των σωματιδίων καθώς και αυτόν των επαναλήψεων, όπως είναι και λογικό, ο αλγόριθμος επιτυγχάνει καλύτερες λύσεις. Αυτή η βελτίωση όμως, δεν έρχεται χωρίς κόστος. Περισσότερες επαναλήψεις και σωματίδια σημαίνει και μεγαλύτερο χρόνο εκτέλεσης. Έτσι, είναι στην διακριτική ευχέρεια του χρήστη η

ρύθμιση των παραμέτρων αυτών ώστε να επιτύχει τα καλύτερα δυνατά αποτελέσματα στα πλαίσια του διαθέσιμου χρόνου.

4.3.2 Σταθερές επιτάχυνσης

Όπως έχει αναφερθεί και παραπάνω, τα σωματίδια «πετάνε» στον χώρο ψάχνοντας για τις καλύτερες λύσεις. Για να αλλάξουν θέση λοιπόν, ρυθμίζουν την ταχύτητά τους με βάση την προηγούμενη ταχύτητα, μια προσωπική και μια κοινωνική συνιστώσα. Όλοι αυτοί οι παράγοντες εμπεριέχονται στην εξίσωση των ταχυτήτων που έχει ως εξής:

$$v_{id}\,(t+1) = v_{id}\,(t) + c_1\,r_1\,(p_{id}\,(t) - x_{id}\,(t)) + c_2\,r_2\,(p_{gd}\,(t) - x_{id}\,(t))$$

Όπως φαίνεται παραπάνω, οι δύο συνιστώσες (προσωπική και κοινωνική) πολλαπλασιάζονται με τους συντελεστές $c_1\ r_1$ και $c_2\ r_2$ αντίστοιχα. Από αυτούς, οι r_1 και r_2 είναι εξ ‘ορισμού τυχαίοι αριθμοί από το 0 μέχρι το 1. Οι c_1 και c_2 λέγονται σταθερές επιτάχυνσης και είναι παράγοντες που επηρεάζουν το βάρος της κοινωνικής και προσωπικής επιρροής στον υπολογισμό της ταχύτητας. Όπως έχει μελετηθεί [1] οι δύο αυτοί παράγοντες επιδρούν ως εξής:

- Όταν ο c_1 αυξηθεί σημαντικά, τα σωματίδια τείνουν να παραμένουν στις θέσεις που έχουν επιτεύξεις τις καλύτερες προσωπικές τους λύσεις, διεσπαρμένα στον χώρο. Έτσι, χάνεται το πλεονέκτημα της «κοινής γνώσης» και ο χαρακτήρας της ομαδικότητας.
- Αντιθέτως, όταν αυξηθεί σημαντικά ο c_2, όλα τα σωματίδια μαζεύονται άμεσα σε μια λύση χωρίς να ερευνήσουν τον γύρω χώρο.
- Όταν και οι δύο συντελεστές είναι αρκετά μεγάλοι, οι ταχύτητες των σωματιδίων παραμένουν υψηλές, χωρίς αυτά να μπορούν να βρουν με επαρκή ακρίβεια την λύση (exploitation).
- Τέλος, όταν και οι δύο συντελεστές είναι αρκετά μικροί, η ταχύτητα γίνεται τόσο μικρή ώστε τα σωματίδια να μην μπορούν να ερευνήσουν αποτελεσματικά τον χώρο.

Σύμφωνα με τα παραπάνω και για να αποφευχθεί η σύγχυση, έχει καθιερωθεί από τα πρώτα στάδια της εξέλιξης του PSO η χρήση ίσων συντελεστών, και συγκεκριμένα:

$$c_1 = c_2 = 2$$

4.3.3 Σταθερά αδράνειας

Μια από τις πρώτες βελτιωτικές κινήσεις για την αύξηση των επιδόσεων του αλγορίθμου ήταν η προσθήκη ενός ακόμα περιοριστικού παράγοντα στην εξίσωση των ταχυτήτων, αυτού της αδράνειας. Συγκεκριμένα, ο όρος της προηγούμενης ταχύτητας θεωρείται ως ένα είδος αδράνειας που παρασέρνει τα σωματίδια σε περιοχές που αλλιώς δεν θα εξερευνούσαν. Όπως όμως έγινε κατανοητό όταν αυτή η αδράνεια είναι μεγάλη, δυσχεραίνει το έργο του αλγορίθμου. Για αυτόν τον λόγο έχουν κατά καιρούς προταθεί διάφοροι περιοριστικοί παράγοντες με σημαντικότερους τους constriction factor (φ) [3] και inertia weight (w) [2] που ορίζονται ως εξής:

- Constriction Factor

Σύμφωνα με αυτή τη μέθοδο ο υπολογισμός των ταχυτήτων εξαρτάται και από έναν παράγοντα K. Αυτός με τη σειρά του, είναι συνάρτηση των c_1 και c_2 σύμφωνα με τα παρακάτω:

$$K = \frac{2}{\left|2-\varphi-\sqrt{\varphi^2-4\varphi}\right|}, \text{ όπου } \varphi = c1 + c2, \ \varphi > 4$$

$$v_{id}(t+1) = K\left[v_{id}(t) + c_1 r_1 (p_{id}(t) - x_{id}(t)) + c_2 r_2 (p_{gd}(t) - x_{id}(t)) \right]$$

- Inertia Weight

Αυτή η μέθοδος χρησιμοποιεί έναν μεταβαλλόμενο συντελεστή w για να περιορίσει όχι όλη την εξίσωση των ταχυτήτων όπως παραπάνω, αλλά μόνο την αδράνεια που οφείλεται στην προηγούμενη ταχύτητα $v_{id}(t+1)$. Όπως περιγράφηκε στην ενότητα 2.4, ο συντελεστής w μειώνεται γραμμικά με την πάροδο των επαναλήψεων ξεκινώντας με τιμή ίση με 0.9, φτάνοντας να έχει τιμή ίση με 0.4 στο τέλος των επαναλήψεων. Αυτή η εκδοχή είναι η πιο διαδεδομένη και αποδεκτή και αποτελεί το πλέον συνηθισμένο σημείο αναφοράς.

4.3.4 Περιορισμός Ταχυτήτων

Ένας ακόμα πολύ σημαντικός παράγοντας που επηρεάζει καθοριστικά την αποτελεσματικότητα του PSO είναι αυτός των μέγιστων επιτρεπόμενων ταχυτήτων. Όταν τα σωματίδια πετούν για να βρουν την καλύτερη δυνατή λύση μπορεί να πάρουν διάφορες ταχύτητες, εξαρτώμενες πάντα από την εξίσωση των ταχυτήτων. Σε έναν χώρο έρευνας με δεδομένες διαστάσεις όμως, οι πολύ μεγάλες ταχύτητες δεν έχει νόημα. Εάν αυτές επιτρέποντα, τα σωματίδια απλά «διακτινίζονται» από άκρη σε άκρη του χώρου, χωρίς στην πράξη να ερευνούν μέσα σε αυτόν. Είναι απαραίτητο λοιπόν να υπάρχει έλεγχος στις ταχύτητες με τις οποίες κινούνται. Αυτός ο έλεγχος δεν είναι τίποτα παραπάνω από την θέσπιση ενός κατωφλίου το οποίο δεν μπορεί να υπερβεί. Μέσα από την βιβλιογραφία αλλά μετά από πειραματισμούς στα πλαίσια της παρούσας εργασίας το κατώφλι για την κάθε διάσταση ορίζεται στο 10% του εύρους της εκάστοτε διάστασης. Αυτό σημαίνει πως εάν μία διάσταση έχει όρια τα [-5 5], δηλαδή εύρος 10, τότε η μέγιστη ταχύτητα πρέπει να οριστεί στο 1.

4.4 Εφαρμογή του PSO στον αυτόματο έλεγχο

Όπως άλλοι εξελικτικοί αλγόριθμοι βελτιστοποίησης, έτσι και ο PSO χρησιμοποιείται επιτυχώς για την επίλυση πολύπλοκων πολυδιάστατων μη-γραμμικών προβλημάτων. Τέτοια προβλήματα συναντώνται και σε εφαρμογές αυτομάτου ελέγχου. Η βαθμονόμιση ενός PID ελεγκτή, η ρύθμιση ενός Ασαφούς Ελεγκτή ή και ενός πολύπλοκου Νευρωνικού δικτύου αποτελούν συχνές εφαρμογές για τον PSO [4, 5, 8, 9, 10,12...17]. Τα παραδείγματα των ελεγκτών αυτών αποτελούν πολυδιάστατα και μη-γραμμικά προβλήματα βελτιστοποίησης.

Συγκεκριμένα, οι διαστάσεις του προβλήματος δεν είναι άλλες από τις μεταβλητές του εκάστοτε ελεγκτή. Για παράδειγμα, ένα σύστημα ελεγχόμενο από έναν PID ελεγκτή αποτελεί πρόβλημα τριών διαστάσεων. Αυτών του P, του I και του D. Αντίστοιχα, ένας Ασαφής Ελεγκτής μπορεί να αποτελεί ένα πρόβλημα πολλών διαστάσεων. Όπως φαίνεται στο Τρίτο Μέρος της παρούσας εργασίας, λύνεται ένα

πρόβλημα τεσσάρων διαστάσεων. Μπορεί όμως η πολυπλοκότητα του προβλήματος να ανέβει όταν αντικείμενο βελτιστοποίησης αποτελούν:

- Το σχήμα των Συναρτήσεων Συμμετοχής
- Τα κάτω ή/και πάνω όρια των Συναρτήσεων Συμμετοχής
- Ο αριθμός των Συναρτήσεων Συμμετοχής σε κάθε είσοδο και έξοδο
- Οι κανόνες που συνδέουν εισόδους και έξοδο

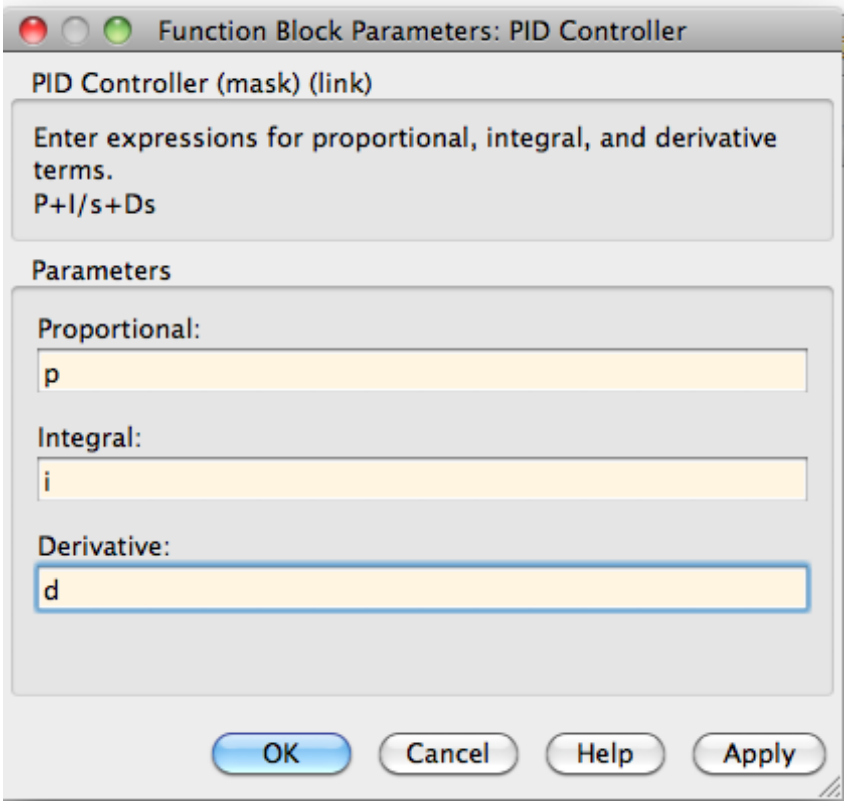
Εύκολα καταλαβαίνει κανείς ότι αναθέτοντας την ρύθμιση τόσων πολλών μεταβλητών στον PSO το πρόβλημα προς επίλυση γίνεται εντυπωσιακά περίπλοκο. Αξίζει να αναφερθεί πως στα πλαίσια της εργασίας αυτής ρυθμίστηκε ελεγκτής που αποτελούσε πρόβλημα 44άρων διαστάσεων: 4 διαστάσεις από τους συντελεστές στα σήματα εισόδου-εξόδου του PID-Type FLC, 15 από τις κορυφές των πέντε τριγωνικών συναρτήσεων συμμετοχής των δύο εισόδων και μιας εξόδου και άλλες 25 που αποτελούσαν τον πίνακα 5x5 με τους κανόνες.

4.5 Υλοποίηση βελτιστοποίησης PID και FLC σε Matlab – Simulink

4.5.1 Υλοποίηση PID

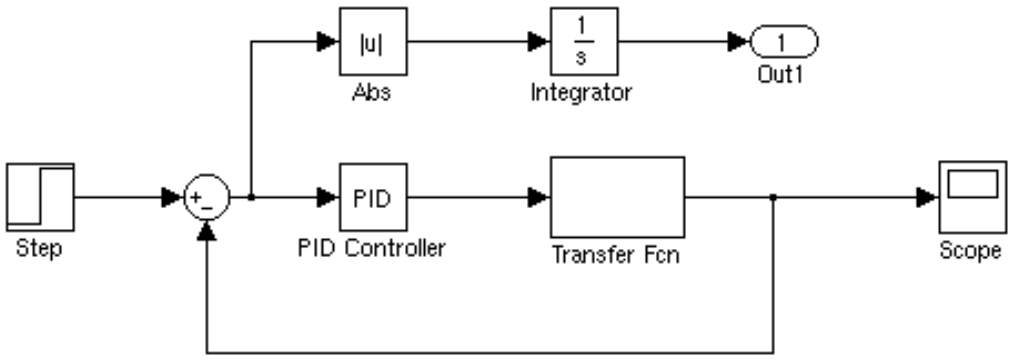
Ο εκάστοτε προς βελτιστοποίηση ελεγκτής ρυθμίζεται με βάση κάποιο κριτήριο. Το κριτήριο αυτό όπως φαίνεται στο Τρίτο Μέρος μπορεί να είναι ένα από τα συνήθη IAE, ITAE, ISE ή ITSE, ή και διάφοροι άλλοι συνδυασμοί. Ο αλγόριθμος, σε συνεργασία με το σύστημα που πρέπει να είναι μοντελοποιημένο στο Simulink, κρίνει τις αποκρίσεις του συστήματος για τις προτεινόμενες από τον PSO ρυθμίσεις του ελεγκτή, βάσει του κριτηρίου που έχει επιλεγεί.

Για την ρύθμιση ενός ελεγκτή PID με PSO σε περιβάλλον Matlab-Simulink πρέπει το σύστημα κλειστού βρόχου να είναι μοντελοποιημένο, και ο ελεγκτής να είναι ρυθμισμένος όχι με κάποιες τιμές, αλλά με τα ονόματα των μεταβλητών προς βελτιστοποίηση, δηλαδή p, i και d. Αυτές οι μεταβλητές παίρνουν τιμές απευθείας από το Workspace, και εκεί δίνει τις προτεινόμενες λύσεις ο αλγόριθμος. Η ρύθμιση-παρέμβαση φαίνεται στην παρακάτω εικόνα.



Εικόνα 29: Παρέμβαση-ρύθμιση στο μενού του PID

Ακόμα, το μοντέλο πρέπει να δίνει το αποτέλεσμα της προσομοίωσης μέσω του κριτηρίου στο Workspace, μέσω μιας απλής εξόδου "Out". Ένα παράδειγμα του πως πρέπει είναι ένα μοντέλο με κριτήριο IAE φαίνεται παρακάτω.



Εικόνα 30: Παράδειγμα συστήματος με κριτήριο IAE στην έξοδο

Πέρα από το μοντέλο του συστήματος στο Simulink, πρέπει να γίνει μια και μοναδική παρέμβαση στον κώδικα του PSO. Αυτή η αλλαγή αφορά την αντικειμενική

συνάρτηση της οποίας την τιμή θα προσπαθήσει να ελαχιστοποιήσει. Πρέπει λοιπόν για κάθε ένα από τα σωματίδια να γίνουν τα εξής:

- Οι 3 διαστάσεις του προβλήματος (προτεινόμενες λύσεις από το εκάστοτε σωματίδιο) να αντιστοιχιστούν στις μεταβλητές p, i, και d ώστε να μπορεί το Simulink να τις «βλέπει» και να τις χρησιμοποιεί
- Να εκτελεστεί η προσομοίωση στο Simulink
- Να καταχωρηθεί η καταλληλότητα της λύσης (fitness) του κάθε σωματιδίου

Αυτές οι αλλαγές μπορούν να συμπεριληφθούν είτε στον κώδικα ”Αρχικός PSO” όπως φαίνεται στο παράρτημα, είτε ως μια παραλλαγή της συνάρτησης test_function χωρίς να γίνει καμία αλλαγή στον κώδικα ”Αρχικός PSO”. Η δεύτερη επιλογή φαίνεται παρακάτω:

```
% Συνάρτηση που δέχεται αριθμό διαστάσεων προβλήματος και πίνακα με τις θέσεις
% όλων των σωματιδίων σε όλες τις διαστάσεις και επιστρέφει πίνακα value με τις
% επιδόσεις όλων των σωματιδίων
function value=test_function(diastaseis,theseis)

      % Εκτέλεση της προσομοίωσης και καταχώρηση των αποτελεσμάτων
      % για κάθε σωματίδιο ξεχωριστά
      for part=1:length(theseis)

            % Αντιστοίχηση των τριών διαστάσεων κάθε σωματιδίου στις
            % μεταβλητές p, i, και d
            p=theseis(part,1);
            i=theseis(part,2);
            d=theseis(part,3);

            % Προσομοίωση του μοντέλου «Systhma_me_PID» για 10 sec
            sim('Systhma_me_PID',10);

            % Καταχώρηση του αποτελέσματος του εκάστοτε σωματιδίου
            value(part,1)=yout(length(yout),1);

      end

end
```

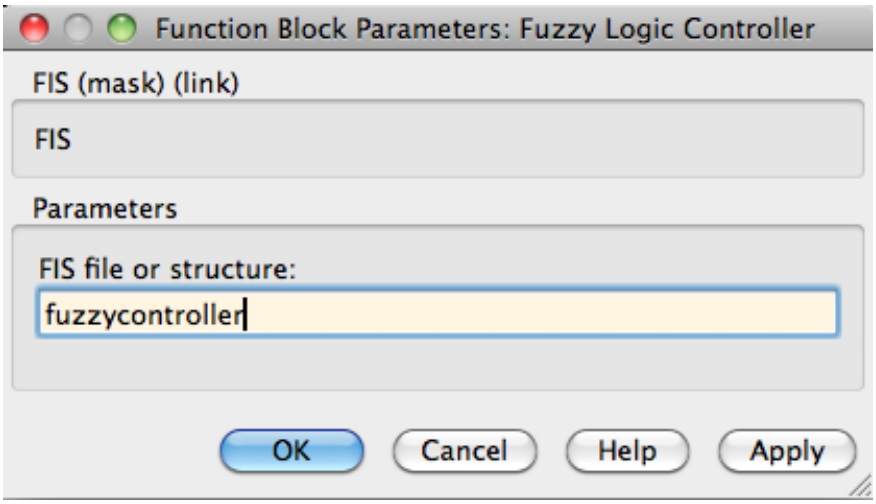
4.5.2 Υλοποίηση FLC

Η υλοποίηση της βελτιστοποίησης ενός Ασαφούς Ελεγκτή σε περιβάλλον Matlab - Simulink παρουσιάζει αρκετά περισσότερες δυσκολίες. Λόγος αυτού είναι η δυσκολία με την οποία ρυθμίζεται ένας FLC στο Matlab. Πρέπει αρχικά να δημιουργηθεί χειροκίνητα ένας ελεγκτής μέσω του Fuzzy Tool του Simulink και να αποθηκευτεί. Το συγκεκριμένο εργαλείο εμφανίζεται γράφοντας «fuzzy» στο παράθυρο εντολών (Command Window) του Matlab. Από εκεί μπορεί κανείς να κατασκευάσει έναν Ασαφή Ελεγκτή και να τον σώσει.

Για να μπορέσει ο PSO να επέμβει στον παραπάνω ελεγκτή, πρέπει πρώτον να φορτωθεί ο ελεγκτής στο Workspace. Αυτό γίνεται στην αρχή της εκτέλεσης του PSO και με την εντολή *readfis*.

```
% Άνοιγμα του Ελεγκτή Controller και χρήση του με το όνομα fuzzycontroller
fuzzycontroller2=readfis('Controller3');
```

Στην περίπτωση αυτή, ο ελεγκτής φορτώνεται σαν *structure* με ένα νέο όνομα. Η χρήση του ίδιου ονόματος είναι αναγκαία, καθώς το block του FLC στο Simulink αντλεί τις πληροφορίες του από το Workspace. Το ίδιο όνομα λοιπόν, πρέπει να χρησιμοποιηθεί και στο μοντέλο του Simulink, από το παράθυρο των ιδιοκτητών του FLC.



Εικόνα 31: Ονοματοδοσία του FLC

Αφού ο ελεγκτής φορτωθεί και ονοματιστεί σωστά, μπορούν πλέον οι μεταβλητές του να είναι προσβάσιμες από τον PSO. Όλες οι αλλαγές γίνονται είτε από τον ίδιο τον PSO είτε μέσω της συνάρτησης test_function, όπως και στην περίπτωση του PID. Για λόγους καθαρότητας στον κώδικα, προτιμάται αυτή η λύση.

Αναλόγως με το ποιο χαρακτηριστικό του FLC θέλουμε να βελτιστοποιήσουμε, το Matlab δίνει τη δυνατότητα χρήσης διαφόρων εντολών. Εδώ θα αναπτυχθούν δύο τεχνικές που αποδείχτηκαν εύχρηστες σε συνδυασμό με τον PSO.

Για την αλλαγή της θέσης των κορυφών και βάσεων των Συναρτήσεων Συμμετοχής μπορεί να χρησιμοποιηθεί η εντολή *setfis* για το κάθε σωματίδιο ξεχωριστά, όπως έγινε και παραπάνω.

```
% Η setfis αλλάζει τις παραμέτρους της mf No 1 στην είσοδο No 1
fuzzycontroller=setfis
(fuzzycontroller,'input',1,'mf',1,'params', [a  b c]);
```

Έτσι, ακολουθείται το «μονοπάτι»:
Ελεγκτής: Fuzzycontroller →Είσοδος: 1 →Συνάρτηση Συμμετοχής: 1 → Παράμετροι

Οι μεταβλητές a, b και c προκύπτουν όπως και πριν από τις διαστάσεις του προβλήματος και τις προτεινόμενες λύσεις του κάθε σωματιδίου. Με αυτόν τον ομολογουμένως άκομψο τρόπο και επαναλαμβάνοντας αυτή τη γραμμή για όλες τις Συναρτήσεις Συμμετοχής, κάποιος μπορεί να βελτιστοποιήσει όλες τις παραμέτρους όλων των εισόδων και εξόδων.

Για την αλλαγή των κανόνων χρησιμοποιήθηκε μια άλλη προσέγγιση. Σύμφωνα με αυτή, αντί να επεμβαίνουμε στον ελεγκτή μέσω εντολών, αλλάζουμε κατευθείαν τον επίμαχο κανόνα, δίνοντάς του νέα τιμή. Στον παραπάνω Ασαφή ελεγκτή *fuzzycontroller*, όπως και σε κάθε ασαφή ελεγκτή, οι κανόνες είναι αριθμημένοι και μπορούν να προσπελαστούν με το όνομα του ελεγκτή όπως παρακάτω:

```
% Απευθείας αλλαγή του κανόνα No 1
fuzzycontroller.rule(1).consequent = new_value;
```

Επαναλαμβάνοντας αυτή την γραμμή κώδικα και για όλους τους υπόλοιπους κανόνες μπορεί κανείς να βελτιστοποιήσει και να διαμορφώσει ολόκληρο το σύνολο κανόνων ενός ελεγκτή, όσοι και αν είναι αυτοί. Ένας μικρός περιορισμός που τίθεται εδώ, είναι το ότι ο PSO δίνει ως πιθανές λύσεις δεκαδικούς αριθμούς. Από την άλλη, όπως φαίνεται από την παραπάνω ανάθεση του νέου κανόνα, η τιμή *new_value* πρέπει να είναι ακέραιος αριθμός. Αυτός ο αριθμός παίρνει τιμές από το 1 και πάνω, και παραπέμπει στις Συναρτήσεις Συμμετοχής της εξόδου. Έτσι, για μία έξοδο με 5 Συναρτήσεις Συμμετοχής, η μεταβλητή *new_value* παίρνει τιμές 1, 2, 3, 4 ή 5. Για αυτόν τον λόγο, οι δεκαδικές τιμές που δίνονται από τον PSO θα πρέπει να στρογγυλοποιούνται. Όλη αυτή η διαδικασία συνοψίζεται στην επόμενη επανάληψη:

```
% Ανάθεση των προτεινόμενων τιμών του PSO (θέση) σαν τους 25 κανόνες του FIS
for i=1:25

     fuzzycontroller.rule(i).consequent=round(thesh(i));

end
```

Μετά από την ρύθμιση όλων των συντελεστών του FLC, πρέπει να γίνει η προσομοίωση του συστήματος στο Simulik και να καταχωρηθούν τα αποτελέσματα του προτεινόμενου από το κάθε σωματίδιο ελεγκτή. Αυτό γίνεται όπως και στο παράδειγμα του PID, κάνοντας χρήση ενός κριτηρίου και της εντολής *sim()*.

4.6 Προτάσεις για περαιτέρω έρευνα

Στην παραπάνω εργασία παρουσιάστηκε μια νέα προσέγγιση του PSO, αυτή του συστήματος ελέγχου. Στις μετρήσεις που πραγματοποιήθηκαν ο DPSO έδωσε αρκετά καλά αποτελέσματα, καθιστώντας την τεχνική αρκετά αξιόλογη. Στις παρακάτω παραγράφους αναπτύσσονται κάποιες ιδέες για συνέχιση της εξέλιξης, βελτίωσης και διερεύνηση της μεθόδους αυτής και των δυνατοτήτων της.

4.6.1 Το πρόβλημα

Το σημαντικότερο πρόβλημα που αντιμετωπίστηκε στις δοκιμές και την χρήση του DPSO ήταν χωρίς αμφιβολία η ρύθμιση των παραμέτρων του. Ο DPSO μπορεί να χρησιμοποιεί αρκετούς συντελεστές που πολλαπλασιάζουν τις προηγούμενες ταχύτητες, θέσεις και σφάλματα, όπως φαίνεται από την εξίσωση υπολογισμού των νέων θέσεων:

$$x_{id}(t+1) = \alpha * x_{id}(t) + \beta * v_{id}(t) + \gamma * e(t) + \delta * e(t-1) + \varepsilon * e(t-2) + \ldots$$

Όπως είχε αναφερθεί και στην ενότητα 2.5, δεν βρέθηκε ένα συγκεκριμένο σετ τιμών για τους συντελεστές α, β, γ, δ και ε που να δίνει λύσεις καλύτερες του APSO σε όλα τα προβλήματα.

Έχοντας βασικές γνώσεις πάνω στον αυτόματο έλεγχο και έχοντας την αναλογία ελεγκτή-συστήματος κατά νου, είναι λογικό το κάθε διαφορετικό σύστημα (προς επίλυση πρόβλημα) να χρειάζεται και διαφορετικά ρυθμισμένο ελεγκτή (PSO). Η ανάγκη διερεύνησης διαφορετικών τιμών για τους συντελεστές του DPSO όμως, τον κάνουν δύσκολο και άβολο στην χρήση. Για αυτόν τον λόγο, πρέπει να δοθούν λύσεις που να επιτρέπουν την αξιόπιστη χρήση του DPSO σε όλων των ειδών τα προβλήματα, χωρίς την επέμβαση του χρήστη.

4.6.2 Προτεινόμενες Λύσεις

Μέχρι το τέλος της παρούσας εργασίας, πραγματοποιήθηκε αρκετή έρευνα σε τρόπους που θα μπορούσαν να λύσουν το παραπάνω πρόβλημα. Έγιναν προσπάθειες να εξηγηθεί το φαινόμενο αυτό ώστε να μπορέσει να βρεθεί μια τεχνική για τον έλεγχο και τη διόρθωσή του.

Η σκέψη πως δεν υπάρχει ένα γνωστό σετ τιμών για τους συντελεστές αλλά αντιθέτως, πρέπει να γίνεται πειραματισμός για να βρεθεί, οδήγησε στην εξέταση σετ τιμών που προκύπτουν τυχαία κατά την εκτέλεση του αλγορίθμου. Συγκεκριμένα, οι σταθεροί συντελεστές α, β, και γ αντικαταστάθηκαν από τυχαίες τιμές μεταξύ 0 και 1. Επίσης, εξετάστηκε και η χρήση συντελεστών που, πάλι τυχαία, αυξομοιώνουν ελαφρώς τις τιμές τους. Οι τεχνικές αυτές έδωσαν εντυπωσιακά αποτελέσματα σε

κάποια προβλήματα, αλλά υποδεέστερα σε άλλα. Συνολικά, φαίνεται να είναι ικανές τεχνικές, αλλά σίγουρα, χρειάζεται περισσότερος πειραματισμός και έρευνα.

Αυτό που πάνω από όλα έγινε κατανοητό, ήταν η επίδραση που έχουν οι συντελεστές στις ταχύτητες με τις οποίες τα σωματίδια κινούνται στον χώρο. Συγκεκριμένα, όπως φαίνεται στην εικόνα 14 της ενότητας 2.5.4, οι συντελεστές αυξάνουν και μειώνουν δραστικά τις ταχύτητες. Έτσι, έγινε μια προσπάθεια να αναπτυχθεί μια μέθοδος η οποία να ελέγχει και να περιορίζει τις ταχύτητες αυτές, αλλάζοντας τις τιμές των συντελεστών α ή β ή γ ή και συνδυασμού αυτών.

Αυτό έγινε με χρήση ενός ασαφούς συστήματος που δρα ως ελεγκτής. Συγκεκριμένα, το σύστημα αυτό είχε ως ασαφείς εισόδους την μέση ταχύτητα όλων των σωματιδίων σε όλες τις διαστάσεις και τον αριθμό της τρέχουσας επανάληψης. Η έξοδός του μπορούσε να είναι είτε ένας ρυθμός μεταβολής των συντελεστών, ή η τιμή των συντελεστών των ίδιων. Η βασική ιδέα εδώ, ήταν να μην χρειάζεται χειροκίνητη ρύθμιση των συντελεστών με βάση την ταχύτητα, αλλά ο έλεγχός τους μέσω του Ασαφούς συστήματος.

Έτσι, οι ταχύτητες μπορούν να ακολουθούν το ίδιο «προφίλ» ασχέτως με το πρόβλημα που ο DPSO λύνει. Έτσι, ένα πρόβλημα που θα χρειαζόταν μικρούς συντελεστές μπορεί να λυθεί με τον ίδιο ακριβώς τρόπο με ένα που θα χρειαζόταν μεγάλους. Και αυτή η ιδέα έδωσε καλά αποτελέσματα, αλλά χρειάζεται αρκετή έρευνα για να μπορέσουν να βγουν συμπεράσματα.

Καταλήγοντας, αυτή η πτυχιακή εργασία εξέτασε τον αλγόριθμο βελτιστοποίησης PSO και την πιο γνωστή παραλλαγή του, τον APSO. Ακόμα, αναπτύχθηκε μια καινούρια τεχνική που ονομάστηκε DPSO και διερευνήθηκαν οι δυνατότητές της. Αρχικά, αυτή φαίνεται να υπερτερεί του APSO αλλά χρειάζεται περισσότερη έρευνα και εξέλιξή της ώστε να μπορεί με έναν απλό και γρήγορο τρόπο να δίνει λύσεις καλύτερες του διαδεδομένου APSO.

Παράρτημα

Υπόδειγμα αλγορίθμου του απλού SPSO γραμμένο σε περιβάλλον Matlab, όπως περιγράφεται στο Δεύτερο Μέρος της εργασίας. Φαίνεται να είναι άκομψα γραμμένο, αλλά αυτό είναι λόγω ασυμβατότητας του Matlab με τον κειμενογράφο. Για να λειτουργήσει ο παρακάτω αλγόριθμος χρειάζεται μια επιπλέον συνάρτηση που εμπεριέχει τις αντικειμενικές συναρτήσεις. Στο Δεύτερο Μέρος αυτή έχει περιγραφεί ως test_function και ακολουθεί μετά τον αλγόριθμο του SPSO

Α. Ο αλγόριθμος του SPSO

```
%Dimitrios Karyofyllis

% SPSO

% P R O S O X H ! ! !

% Prepei na einai swmeno sto idio directory me thn synarthsh test_function!

part_num=10; % Particle number

dimensions=3; % Dimensions of Problem

iterations=300; % Maximum Iterations Number

% pbest, pbestpos and gbest close to +oo
pbest(1:part_num,1)=1e+200;

pbestpos(1:part_num,1:dimensions)=1e+200;

gbest=1e+200;

for diastash=1:dimensions

      %Εδώ, σε κάθε επανάληψη, δηλαδή σε κάθε μία διάσταση, εμφανίζεται στην
      %οθόνη ο αριθμός της διάστασης και ο χρήστης δίνει για την διάσταση αυτή
      %τα όρια μέσα στα οποία θα γίνει η έρευνα
      display(diastash)
      limits(diastash,:)= input('Δώσε όρια διάστασης');

      %Σε αυτό το σημείο υπολογίζονται τυχαία οι αρχικές θέσεις των σωματιδίων
      %μέσα στα όρια που τοποθετούνται στο αμέσως προηγούμενο βήμα
```

```
     position(:,diastash)=rand(part_num,1)*
(limits(diastash,2) - limits(diastash,1)) +
limits(diastash,1);

     %Στο επόμενο βήμα υπολογίζονται τυχαία οι αρχικές ταχύτητες των
     %σωματιδίων μέσα στο 20% του εύρους των ορίων που τέθηκαν
     speed(:,diastash)=rand(part_num,1)*0.2*
(limits(diastash,2)-limits(diastash,1))-
.1*(limits(diastash,2)   -limits(diastash,1));

%Εδώ με ανάλογο τρόπο υπολογίζονται τα όρια των ταχυτήτων
     speedmax(diastash)=0.1*(limits(diastash,2)-
limits(diastash,1));

     speedmin(diastash)=0.1*(limits(diastash,1)-
limits(diastash,2));

end


%Εδώ  γίνεται η εκκίνηση των επαναλήψεων έχοντας ως μετρητή τον cnt
for cnt=1:iterations

     %Σε αυτό το σημείο γίνεται ο έλεγχος της αποτελεσματικότητας της
     %κάθε λύσης. Καλείται η συνάρτηση test_function στην οποία παρέχονται
     %ο αριθμός της αντικειμενικής συνάρτησης προς επίλυση (1...7),
     %ο αριθμός των διαστάσεων (dimensions), οι θέσεις όλων των σωματιδίων
     %σε όλες τις διαστάσεις (position) που είναι ένας πίνακας μεγέθους
     % [part_num x dimensions]. Αυτή η συνάρτηση επιστρέφει έναν διάνυσμα
     %μεγέθους [part_num,1]  το οποίο δίνει τον «βαθμό ικανοποίησης» της
     %συνάρτησης για κάθε σωματίδιο (fitness value).
     eval=test_function(1,dimensions,position);


     %Εύρεση των προσωπικών καλύτερων τιμών και αντικατάστασή τους
     pbest=~(eval<pbest).*pbest+eval.*(eval<pbest);


     %Εύρεση και αντικατάσταση των παλιών καλύτερων προσωπικών θέσεων
     %με τις νέες και καλύτερες
     for i=1:dimensions

          pbestpos(:,i)=~(eval>pbest).*pbestpos(:,i)
+position(:,i).*(eval>pbest);
     end

     %Εύρεση και αντικατάσταση της ολικής βέλτιστης τιμής και των αντίστοιχων
     %θέσεων σε όλες τις διαστάσεις
     [minim,thesi]=min(pbest);
     if minim<gbest
```

```
            gbest=minim;
            gbestpos=position(thesi,:);
      end


      %Τώρα ξεκινά η επανάληψη που για κάθε διάσταση ξεχωριστά θα
      %υπολογιστούν οι νέες ταχύτητες και θέσεις των σωματιδίων
      for diastash=1:dimensions

            %Αυτή η γραμμή ανανεώνει την ταχύτητα του κάθε σωματιδίου στην
            %αντίστοιχη διάσταση
            speed(:,diastash) = speed(:,diastash) +
2*rand(part_num,1).* (pbestpos(:,diastash)-
position(:,diastash)) + 2 * rand(part_num,1).*
(ones(part_num,1) * gbestpos(1,diastash)-
position(:,diastash) );


      %Εδώ ρυθμίζεται η νέα ταχύτητα του κάθε σωματιδίου σε
            % περίπτωση που είναι μεγαλύτερη από την μέγιστη επιτρεπόμενη
            speed(:,diastash)=speed(:,diastash).*
~(speed(:,diastash)>speedmax(diastash)) +
speedmax(diastash).*
(speed(:,diastash)>speedmax(diastash));

            %Εδώ ρυθμίζεται η νέα ταχύτητα του κάθε σωματιδίου σε
            % περίπτωση που είναι μικρότερη από την ελάχιστη επιτρεπόμενη
            speed(:,diastash)=speed(:,diastash)
.*~(speed(:,diastash)<speedmin(diastash))
+speedmin(diastash)
.*(speed(:,diastash)<speedmin(diastash));


      %Γίνεται η εύρεση των νέων θέσεων βάσει των νέων ταχυτήτων
            position(:,diastash)=
position(:,diastash)+speed(:,diastash);


      %Εδώ γίνεται η συγκράτηση των σωματιδίων εντός των μέγιστων
            %ορίων για κάθε διάσταση
            position(:,diastash)=position(:,diastash)
.*~(position(:,diastash)>limits(diastash,2))
+limits(diastash,2)
.*(position(:,diastash)>limits(diastash,2));

      %Εδώ γίνεται η συγκράτηση των σωματιδίων εντός των μέγιστων
            %ορίων για κάθε διάσταση
            position(:,diastash)=position(:,diastash)
.*~(position(:,diastash)<limits(diastash,1))
+limits(diastash,1)
.*(position(:,diastash)<limits(diastash,1));
```

```
    end

      %Στο διάνυσμα best_history αποθηκεύεται σε κάθε επανάληψη η
      %καλύτερη ολική βέλτιστη τιμή έτσι ώστε μετά το τέλος των επαναλήψεων
      %να έχουμε μία εικόνα της προόδου του αλγορίθμου
       best_history(cnt)=gbest;

end
```

Β. Ο αλγόριθμος των αντικειμενικών συναρτήσεων

```
%Dimitrios Karyofyllis

% Function gia thn axiologhsh twn thesewn twn particles tou PSO

% P R O S O X H ! ! ! ! !   Na einai sto idio directory me to Arxikos_PSO.m

% Dexetai to noumero ths synarthshs pou theloume na exetasoume ( fun ), tis
% diastaseis tou provlhmatos ( diastaseis ) kai ton pinaka me tis theseis
% olwn twn particles se oles tis diastaseis ( thesesis ).

% Epistrefei to dianysma "value" poy periexei thn axiologhsh toy kathe swmatidiou


function value=test_function(fun,diastaseis,theseis)

switch fun

    case 1
        %De Jong's Function Limits [-5.12 5.12] Min=0 @ (0,0,0,..)
        value=0;
        for i=1:diastaseis
            value=value+ theseis(:,i).^2;
        end

    case 2
        %Rastring's Function Limits [-5.12 5.12] Min=0 @ (0,0,...)
        value=10*diastaseis;
        for i=1:diastaseis
            value=value + theseis(:,i).^2-
10*cos(2*pi.*theseis(:,i));
        end

    case 3
        %Rosenbrock's Valley Function Limits [-2.048 2.048] Min=0 @ (1,1,1..)
        value=0;
```

```
        for i=1:(diastaseis-1)
            value= value + 100*(theseis(:,i+1)-
theseis(:,i).^2).^2 + (1-theseis(:,i)).^2;
        end

    case 4
        %Griewangk's Function Limits [-600 600] Min=0 (0,0,0..)
        sum=0;
        prod=1;
        for i=1:diastaseis
            sum=sum+theseis(:,i).^2;
            prod=prod.*cos(theseis(:,i)/sqrt(i));
        end
        value=sum/4000 - prod + 1;

    case 5
        %Michalewicz's Function Limits [0 3.14] Min: -4.687 @ 5D, -9.66 @
10D
        n = diastaseis;
        m = 10;
        s = 0;
        for i = 1:n;
            s = s+sin(theseis(:,i)).*
(sin(i.*(theseis(:,i).^2)/pi)).^(2*m);
        end
        value = -s;

    case 6
        %Schwefel's Function Limits [-500 500] Min=-418.983*dimensions @
(420.9687, 420.9687,..)
        value=0;
        for i=1:diastaseis
            value=value -
theseis(:,i).*sin(sqrt(abs(theseis(:,i))));
        end

    case 7
        %Trid Function Limits [-Dimentions^2 Dimentions^2] Min=-50 @ 6D, -
200 @10D

        s1 = 0;
        s2 = 0;
        for j = 1:diastaseis
            s1 = s1+(theseis(:,j)-1).^2;
        end
        for j = 2:diastaseis;
            s2 = s2+theseis(:,j).*theseis(:,(j-1));
        end
        value = s1-s2;
```

```
end
```

Αναφορές

[1] Kennedy, J. and Eberhart, R. (1995) "Particle swarm optimization". *Proceedings of 1995 IEEE International Conference on Neural Networks*, Perth, Australia, pp.1942-1948.

[2] Eberhart, R.C., Shi, Y.: "Comparing inertia weights and constriction factors in particle swarm optimization". *Proceedings of IEEE Congress Evolutionary Computation, San Diego, CA, pp. 84–88 (2000*)

[3] Clerc, M. and Kennedy, J. (2002) "The particle swarm - explosion, stability, and convergence in a multidimensional complex space". *IEEE Transactions on Evolutionary Computation*, Vol 6, No. 1, pp. 58-73.

[4] Z.-W. Woo et al., "A PID type fuzzy controller with self-tuning scaling factors", *Fuzzy Sets and Systems 115 (2000) 321–326.*

[5] Gu Fang, Ngai Ming Kwok and Quang Ha, "Automatic Fuzzy Membership Function Tuning Using the Particle Swarm Optimisation", *2008 IEEE Pacific-Asia Workshop on Computational Intelligence and Industrial Application*

[6] Lu Baiquan, Gao Gaiqin and Lu Zeyu, "The block diagram method for designing the particle swarm optimization algorithm". *2011 © Springer Science+Business Media, LLC. 2011*

[7] Shi, Y., Eberhart, RC.: "A modified particle swarm optimizer". *Proceedings of the IEEE International Conference on Evolutionary Computation. IEEE Press, Piscataway, NJ, pp. 69–73 (1998)*

[8] Magdalene Marinaki, Yannis Marinakis, Georgios E. Stavroulakis, "Fuzzy control optimized by a Multi-Objective Particle Swarm Optimization algorithm for vibration suppression of smart structures", *©Springer-Verlag 2010*

[9] A. S. Elwer* and S. A. Wahsh†, "Improved Performance of Permanent Magnet Synchronous Motor by using Particle Swarm Optimization Techniques", *Journal of Power Electronics, Vol. 9, No. 2, 2009*

[10] Hsuan-Ming Feng et al. "Particle Swarm Optimization Learning Fuzzy Systems Design",*Proceedings of the Third International Conference on Information Technology and Applications (ICITA'05)*

[11] Wu Zhi Qiao*, Masaharu Mizumoto, "PID type fuzzy controller and parameters adaptive method", *Fuzzy Sets and Systems78 (1996)23-35*

[12] Boumediene Allaoua et al., “The Efficiency of Particle Swarm Optimization Applied on Fuzzy Logic DC Motor Speed Control*”,SERBIAN JOURNAL OF ELECTRICAL ENGINEERING Vol. 5, No. 2, 2008, 247-262*

[13] Boumediene Allaoua et al., “Setting Up PID DC Motor Speed Control Alteration Parameters Using Particle Swarm Optimization Strategy”,*Leonardo Electronic Journal of Practices and Issue 14, 2009 Technologies p. 19-32 ISSN 1583-1078*

[14] Bouallègue et al.,”A New Method for Tuning PID-Type Fuzzy Controllers Using Particle Swarm Optimization*”, licensee InTech©2012*

[15] Del Valle *Et Al.*: “Particle Swarm Optimization: Basic Concepts, Variants And Applications In Power Systems”. *IEEE Transactions On Evolutionary Computation, Vol. 12, No. 2, 2008*

[16] M. Gu.zelkaya, I’. Eksin*, E. Yes,il.“Self-tuning of PID-type fuzzy logic controller coefficients via relative rate observer”. *Engineering Applications of Artificial Intelligence 16 (2003) 227–236*

[17] Γεώργιος Σύρκος, ¨Ψηφιακός Έλεγχος, Κλασσικός, Σύγχρονος, Εξελικτικός Με Matlab¨, 2/2004, Ιδ. Έκδοση

[18] Jing Jie, Jianchao Zeng, Chongzhao Han, Adaptive Particle Swarm Optimization with PD controller

[19] http://techteach.no/fag/sesm3401/h08/pid_diskret/tidsdiskret_pid_reg.pdf

[20] http://www.atmel.com/images/doc2558.pdf

[21] http://www-optima.amp.i.kyoto-u.ac.jp/member/student/hedar/Hedar_files/TestGO_files/Page364.htm

[22] http://www.cds.caltech.edu/~murray/courses/cds101/fa04/caltech/am04_ch8-3nov04.pdf

Τα αναφερόμενα websites προσπελάστηκαν κατά την περίοδο Ιανουαρίου 2013 - Αυγούστου 2013.