ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΜΗΧΑΝΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΤΟΜΕΑΣ ΒΙΟΜΗΧΑΝΙΚΗΣ ΔΙΟΙΚΗΣΗΣ & ΕΠΙΧΕΙΡΗΣΙΑΚΗΣ ΕΡΕΥΝΑΣ

Διπλωματική Εργασία

Πειραματική μελέτη εξελικτικών αλγορίθμων σε προβλήματα βελτιστοποίησης υπό περιορισμούς

Μαρίνος Κ. Αντώνιος

Φοιτητής 10^{ου} εξαμήνου

Επιβλέπων: Τόλης Αθανάσιος, Επίκουρος Καθηγητής Ε.Μ.Π

Αθήνα, Ιούλιος 2016

Ευχαριστίες

Με την παρούσα διπλωματική εργασία ολοκληρώνεται ο πενταετής κύκλος σπουδών μου στη σχολή Μηχανολόγων Μηχανικών ΕΜΠ και θα ήθελα να ευχαριστήσω όλους όσους με βοήθησαν και συνέβαλαν στην προσπάθεια μου αυτή.

Πρώτα από όλα, θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή μου, τον Επίκουρο Καθηγητή Α. Τόλη, για τη βοήθεια και τη στήριξη του, για τις γνώσεις και τα ερεθίσματα που μου μετέδωσε, κατά τη διάρκεια των σπουδών μου και ειδικότερα κατά την εκπόνηση της διπλωματικής μου εργασίας. Η πόρτα του ήταν πάντα ανοιχτή για μένα για τυχόν απορίες και διευκρινήσεις.

Επίσης θα ήθελα να ευχαριστήσω όλους τους καθηγητές μου, οι οποίοι μου μετέδωσαν τις γνώσεις τους και με επηρέασαν, ο κάθε ένας με τον τρόπο του, αλλά και τους συμφοιτητές και φίλους μου, οι οποίοι με βοήθησαν όλα αυτά τα χρόνια.

Τέλος, θα ήθελα να ευχαριστήσω θερμά την οικογένειά μου για τη συνεχή υποστήριξη, οικονομική και ψυχολογική, που μου προσφέρουν όλα αυτά τα χρόνια των σπουδών μου και στην οποία χρωστάω μεγάλο μέρος της σημερινής μου επιτυχούς σταδιοδρομίας.

Περιεχόμενα

Έποψη Διπλωματικής Εργασίας

Οι Εξελικτικοί Αλγόριθμοι αποτελούν μια κατηγορία στοχαστικών μεθόδων βελτιστοποίησης και ανήκουν στην οικογένεια της Εξελικτικής Υπολογιστικής (Evolutionary Computation-EC) και βρίσκουν εφαρμογή σε διάφορους τομείς όπως η επιχειρησιακή έρευνα, ο μηχανικός σχεδιασμός, η ρομποτική, τα χρηματοοικονομικά κ.α. Οι δυνατότητα των αλγορίθμων να χρησιμοποιούνται σε πολύ μεγάλα προβλήματα γραμμικά και μη γραμμικά, και η εύκολη προσαρμογή τους σε νέα προβλήματα τους καθιστά ιδιαίτερα ελκυστικούς και δίκαια αποτελούν αυτή τη στιγμή αντικείμενο μελέτης από πολλούς ερευνητές.

Στα πλαίσια αυτής της διπλωματικής εργασίας μοντελοποιήθηκε ένας εξελικτικός αλγόριθμος, ο Self-Learning Particle Swarm Optimizer (SLPSO), τον οποίον εισήγαγαν το 2012 οι Li, Yang και Nguyen (Li et al., 2012), και δοκιμάστηκε πάνω σε προβλήματα και δοκιμαστικές συναρτήσεις με σκοπό να συγκρίνουμε αυτά τα αποτελέσματα με αυτά άλλων αλγορίθμων και να βγάλουμε συμπεράσματα για τον SLPSO. Πιο συγκεκριμένα δοκιμάστηκε ο αλγόριθμος πάνω σε έντεκα δοκιμαστικές συναρτήσεις και σε ένα πρακτικό πρόβλημα βέλτιστης ανάθεσης φορτίου. Όλα τα προβλήματα στα οποία έγιναν οι δοκιμές είναι προβλήματα βελτιστοποίησης υπό περιορισμούς. Επειδή ο SLPSO είναι κατασκευασμένος για προβλήματα βελτιστοποίησης χωρίς περιορισμούς, προστέθηκε σε αυτόν μια συνάρτηση ποινής.

Στο 1º κεφάλαιο γίνεται μια εισαγωγή στους εξελικτικούς αλγόριθμους. Στο 2º κεφάλαιο αναλύουμε πιο αναλυτικά τα χαρακτηριστικά των εξελικτικών αλγόριθμων, κάνουμε μια εισαγωγή στη βελτιστοποίηση και επίσης αναλύουμε τις μεθόδους χειρισμού των περιορισμών. Στο 3º κεφάλαιο παρουσιάζεται αναλυτικά το μαθηματικό μοντέλο των αλγορίθμων που μελετήθηκαν, δηλαδή του αλγόριθμου που μοντελοποιήθηκε και αυτών με τους οποίους τον συγκρίναμε. Στο 4º κεφάλαιο παρατίθενται όλα τα προβλήματα τα οποία μελετήσαμε. Στο 5º κεφάλαιο παρουσιάζονται τα αποτελέσματα από την πειραματική διαδικασία και η ανάλυση αυτών και στο 6º κεφάλαιο βρίσκονται τα συμπεράσματα τα οποία προέκυψαν.

Στόχος της διπλωματικής εργασίας είναι να μελετήσουμε τα αποτελέσματα αυτών των δοκιμών, να βγάλουμε κάποια συμπεράσματα για τις δυνάμεις και τις αδυναμίες του αλγόριθμου SLPSO, ώστε να εντοπίσουμε τα σημεία του αλγορίθμου στα οποία μπορεί να υπάρξει κάποια αλλαγή για να ξεπεραστούν αυτές οι αδυναμίες και να προκύψει ένας ακόμη πιο ισχυρός αλγόριθμος.

Συντομογραφίες

εDEag	ε constrained Differential Evolution with an Archive and Gradient-Based Mutation
CHT	Constraint Handling Technique
DE	Differential Evolution
EA	Evolutionary Algorithm
EC	Evolutionary Computation
EP	Evolutionary Programming
ES	Evolutionary Strategies
GA	Genetic Algorithms
GP	Genetic Programming
MA	Memetic Algorithm
MS	Management Science
OR	Operational Research
PSO	Particle Swarm Optimization
SLPSO	Self-Learning Particle Swarm Optimization
SRES	Stochastic Ranking for Constrained Evolutionary Optimization
TS	Tabu Search

1 Εισαγωγή στους Εξελικτικούς Αλγορίθμους

1.1 Ορισμός Εξελικτικών Αλγορίθμων

Οι Εξελικτικοί Αλγόριθμοι (ΕΑ) ανήκουν στην κατηγορία των στοχαστικών μεθόδων βελτιστοποίησης και βρίσκουν έμπνευση στη διαδικασία εξέλιξης στη φύση. Θεμελιωτής της θεωρίας της εξέλιξης θεωρείται ο Δαρβίνος, ο οποίος πρότεινε πως όλα τα είδη έχουν προκύψει από έναν πρόγονο, και σε κοινή δημοσίευση με τον R. Wallace εισήγαγαν την επιστημονική θεωρία ότι η εξέλιξη η οποία παρατηρείτε στη φύση προέκυψε από μια διαδικασία την οποία ονόμασαν φυσική επιλογή, κατά την οποία ο αγώνας των ατόμων για επιβίωση οδηγεί σε άτομα τα οποία είναι καλύτερα προσαρμοσμένα στο περιβάλλον (Darwin and Wallace, 1858), (Darwin, 1859).

Όπως είπαμε η βασική ιδέα από την οποία εμπνεύστηκαν οι (ΕΑ) είναι η διαδικασία της εξέλιξης και πιο συγκεκριμένα αυτή της φυσικής επιλογής και της επικράτησης του ισχυρότερου. Σύμφωνα με τη θεωρία αυτή σε έναν πληθυσμό ατόμων, τα άτομα τα οποία προσαρμόζονται καλύτερα στο περιβάλλον έχουν περισσότερες πιθανότητες επιβίωσης και αναπαραγωγής. Με τη σειρά τους, τα άτομα αυτά τα οποία είναι καλύτερα προσαρμοσμένα στο περιβάλλον, αναπαράγονται και περνάνε τα χαρακτηριστικά τους στους απογόνους τους μέσω του μηχανισμού της αναπαραγωγής. Έτσι, σε κάθε γενιά τα πιο ικανά άτομα επιβιώνουν και αναπαράγονται περνώντας τα χαρακτηριστικά τους στην επόμενη γενιά. Οι απόγονοι αυτών συνδυάζουν τα χαρακτηριστικά των γονέων και παρουσιάζουν καλύτερη προσαρμογή στο περιβάλλον.

Οι (ΕΑ) χρησιμοποιούν πληθυσμούς υποψήφιων λύσεων, οι οποίες μπορούν να αντιστοιχηθούν με τα άτομα του πληθυσμού. Η πίεση του περιβάλλοντος προκαλείται από την αντικειμενική συνάρτηση με την οποία ελέγχεται και η καταλληλότητα των ατόμων. Αν έχουμε πρόβλημα ελαχιστοποίησης τα άτομα τα οποία πετυχαίνουν τη χαμηλότερη τιμή είναι πιο πετυχημένα, ενώ αν έχουμε πρόβλημα μεγιστοποίησης το αντίθετο. Έτσι, τα άτομα τα οποία έχουν καλύτερη τιμή αντικειμενικής συνάρτησης, θεωρούνται πιο ικανά, και έχουν μεγαλύτερες πιθανότητες να αφήσουν απογόνους και να μεταβιβάσουν τα χαρακτηριστικά τους στις επόμενες γενιές. Συνδυάζοντας χαρακτηριστικά από ικανά άτομα του πληθυσμού, προκύπτει η επόμενη γενιά. Η εξέλιξη στους (ΕΑ) πραγματοποιείται μέσω συντελεστών μετάλλαξης, γονιδιακή διασταύρωση και επιλογή γονέων. Η μετάλλαξη και η διασταύρωση δημιουργούν την απαραίτητη ποικιλομορφία του πληθυσμού και διευκολύνουν την καινοτομία, ενώ η επιλογή γονέων αυξάνει τη μέση ποιότητα των λύσεων. Αυτή είναι μια επαναληπτική διαδικασία η οποία συνεχίζεται έως ότου βρεθεί κάποιο άτομο το οποίο να έχει κάποια επιθυμητή τιμή αντικειμενικής συνάρτησης ή να φτάσουμε κάποιο υπολογιστικό όριο το οποίο έχουμε θέσει (Eiben and Smith, 2003).

Οι (EA) έχουν αποδεχτεί πολύ χρήσιμα εργαλεία επίλυσης ακόμη και σε πολύ δύσκολα προβλήματα βελτιστοποίησης. Το μεγάλο τους πλεονέκτημα έγκειται στο γεγονός ότι προσφέρουν ευελιξία και προσαρμοστικότητα σε συνδυασμό με σταθερή απόδοση, η οποία εξαρτάται όμως και από το είδος του προβλήματος. Γενικά, οι (EA) αποτελούν μια κατηγορία αλγορίθμων, οι οποίοι μπορούν να προσαρμοστούν και να χρησιμοποιηθούν σε ένα ευρύ φάσμα επίλυσης προβλημάτων και ιδιαίτερα σε προβλήματα βελτιστοποίησης (Bäck et al., 1997). Για τους παραπάνω λόγους η χρήση τους βρίσκει μεγάλο φάσμα εφαρμογών όπως η οικονομική μοντελοποίηση και προσομοίωση, η μελέτη διαφόρων βιολογικών διεργασιών, η επιχειρησιακή έρευνα, η εκμάθηση μηχανών κ.α.

1.2 Ιστορική αναδρομή Εξελικτικών Αλγορίθμων

Η ιδέα της χρήσης μηχανών ως μεθόδου αυτόματης επίλυσης προβλημάτων ξεκίνησε από τη δεκαετία του 1940, πολύ πριν από την ανακάλυψη των υπολογιστών (Fogel, 1998). Σε γραπτά του A.Turing τα οποία χρονολογούνται το 1948, περιγράφονται τέτοιες ιδέες. Η ιδέα της προσομοίωσης της διαδικασίας εξέλιξης και της φυσικής επιλογής με σκοπό την επίλυση υπολογιστικών προβλημάτων προσεγγίστηκε για πρώτη φορά στα τέλη της δεκαετίας του 1950 από τους Bremermann (Bremermann,1962), Friedberg (Friedberg, 1958) και Box (Box, 1957). Το 1962 ο Bremermann δοκίμασε ένα υπολογιστικό πείραμα πάνω στο θέμα “Optimization Through Evolution and Recombination” (Bremermann, 1962).

Τη δεκαετία του 1960 αναπτύχθηκαν σε τρία διαφορετικά μέρη του κόσμου τρείς διαφορετικές προσεγγίσεις της βασικής ιδέας της προσομοίωσης της φυσικής διαδικασίας εξέλιξης. Παρόλα αυτά το πεδίο έμεινε για σχεδόν τρεις δεκαετίες άγνωστο στην ευρύτερη επιστημονική κοινότητα λόγω της έλλειψης υπολογιστικής ισχύος, αλλά και επίσης λόγω κάποιων μεθοδολογικών αδυναμιών των προσεγγίσεων εκείνων (T.Bäck et al., 1997).

Πρώτη γενιά: Οι κύριες κατηγορίες στις οποίες μπορούμε να χωρίσουμε τους (EA) είναι οι Εξελικτικές Στρατηγικές (Evolutionary Strategies-ES), ο Εξελικτικός Προγραμματισμός (Evolutionary Programming-EP) και οι Γενετικοί Αλγόριθμοι (Genetic Algorithms-GA).

- Οι Fogel, Owens και Walsh εισήγαγαν την έννοια του EP ως μια προσπάθεια να δημιουργήσουν τεχνητή νοημοσύνη, εξελίσσοντας μηχανές οι οποίες θα μπορούν να προβλέψουν γεγονότα με βάση τις ιστορικές παρατηρήσεις. (Fogel et al., 1966)
- Η λογική των GA όπως τη γνωρίζουμε, προτάθηκε για πρώτη φορά από τον Holland (Holland, 1975) και τον μαθητή του (Jong, 1975) το 1975 παρόλο που κάποιες ιδέες στο πλαίσιο της προσομοίωσης γενετικών συστημάτων είχαν ήδη εμφανιστεί το 1957 (Fraser, 1957).
- Η μέθοδος των ES αναπτύχθηκε από τους Rechenberg και Schwefel (Rechenberg, 1973) (Schwefel, 1975). Οι πρώτες προσπάθειες όμως σχετικά με τις Εξελικτικές Στρατηγικές ξεκίνησαν το 1964 στο Τεχνικό Πανεπιστήμιο του Βερολίνου σε πρακτικά προβλήματα στον τομέα της υδροδυναμικής όπως ο βέλτιστος σχεδιασμός σωλήνων και ακροφύσιων. Διάφορες εκδοχές αυτής της στρατηγικής δοκιμάστηκαν στον πρώτο διαθέσιμο υπολογιστή στο Τεχνικό Πανεπιστήμιο του Βερολίνου (Schwefel, 1965)

Αυτές είναι οι τρείς πρώτες προσεγγίσεις της εφαρμογής της θεωρίας της εξέλιξης σε υπολογιστικά συστήματα, στις οποίες βασίστηκαν και οι μετέπειτα μελέτες. Σε φιλοσοφικό επίπεδο αυτές διαφέρουν στον βαθμό με τον οποίο χρησιμοποιούν την εξέλιξη. Σε αλγοριθμικό επίπεδο διαφέρουν κυρίως στην παρουσίαση υποψηφίων λύσεων και στους συντελεστές που χρησιμοποιούν. (Yao, 2003)

Δεύτερη γενιά: Τη δεκαετία του 1980 η πρόοδος στην απόδοση των υπολογιστών έδωσε τη δυνατότητα να εφαρμοστούν οι (EA) σε μια σειρά από δύσκολα πρακτικά προβλήματα βελτιστοποίησης, και η εφαρμογή τους απέκτησε ευρύτερο κοινό. Έτσι, προέκυψαν μια σειρά από προσπάθειες οι οποίες βασίστηκαν στην πρώτη γενιά που αναφέραμε:

- Το 1986 ο Glover εισήγαγε τη μέθοδο Tabu Search (TS), η οποία μπορεί να θεωρηθεί ως μια μετα-ευρετική μέθοδος η οποία εφαρμόζεται πάνω σε μια άλλη ευρετική. Η προσέγγιση της μεθόδου είναι να υπερβαίνει τις τοπικά βέλτιστες λύσεις, με μια στρατηγική η οποία απαγορεύει-τιμωρεί (για αυτό και “Tabu”) συγκεκριμένες κινήσεις. Ο σκοπός της παραπάνω στρατηγικής είναι να αποφύγουμε την κυκλική κίνηση των λύσεων, αποκλείοντας μέσω του “Tabu” συγκεκριμένες κινήσεις (πρόκειται για κλάδους πιθανών λύσεων). Μια κίνηση μπορεί γρήγορα να χάσει την ιδιότητα “Tabu”, σε σχετικά μικρό χρονικό διάστημα. Η μέθοδος TS προήλθε ως μια προσπάθεια να εφαρμοστεί η στρατηγική "Oscillating Assignment" σε προβλήματα ακέραιου προγραμματισμού στη δημοσίευση "Heuristics for integer programming using surrogate constraints" (Glover, 1977) (Glover, 1986).

- Το 1990 αναπτύχθηκε ο Γενετικός Προγραμματισμός (Genetic Programming-GP) από τον Koza, μέθοδος η οποία παρέχει έναν τρόπο αναζήτησης του πιο ικανού υπολογιστικού προγράμματος για την επίλυση πολύπλοκων προβλημάτων. Στη μέθοδο GP υπάρχει πληθυσμός υποψηφίων υπολογιστικών προγραμμάτων ο οποίος εξελίσσεται σύμφωνα με τα Δαρβινικά πρότυπα (Koza, 1990)
- Το 1991 οι Kelly και Davis ανέπτυξαν έναν υβριδικό γενετικό αλγόριθμο ο οποίος έχει προκύψει από συνδυασμό τεχνητής νοημοσύνης και στατιστικής ταξινόμησης. Χρησιμοποιείται μια τεχνική τεχνητής νοημοσύνης, ένας GA, με σκοπό να ενισχύσει την απόδοση ενός κλασσικού αλγόριθμου, του K Nearest Neighbour Classification. (Kelly and Davis, 1991).
- Το 1992 ο Michalewicz πρότεινε τη χρήση εξελικτικών προγραμμάτων. Το εξελικτικό πρόγραμμα αυτό είναι ένας γενετικός αλγόριθμος ενισχυμένος με γνώση ανάλογα με το πρόβλημα, η οποία είναι ενσωματωμένη σε κατάλληλες δομές δεδομένων (Data Structures) και χρησιμοποιεί συντελεστές που εξαρτώνται από το πρόβλημα. Τέτοια προγράμματα μπορούν να διαφέρουν στα παρακάτω:
    - Στις διαφορετικές δομές δεδομένων.
    - Στους διαφορετικούς γενετικούς συντελεστές για να μεταλλάσσουν τα άτομα.
    - Στις μεθόδους για να δημιουργηθεί ο αρχικός πληθυσμός.
    - Στον τρόπο με τον οποίο χειρίζονται τους περιορισμούς του προβλήματος.
    - Στις παραμέτρους όπως ο πληθυσμός, οι πιθανότητες να εφαρμοστούν οι διάφοροι συντελεστές κλπ.

  Παρόλα αυτά, έχουν μια κοινή αρχή, η οποία είναι ότι χρησιμοποιούν έναν πληθυσμό ατόμων ο οποίος υφίσταται διάφορους μετασχηματισμούς, και στη διάρκεια αυτής της εξελικτικής διαδικασίας τα άτομα αγωνίζονται για επιβίωση (Michalewicz, 1992).

Τρίτη γενιά: Οι EA αποδείχτηκαν ισχυρό εργαλείο στην επίλυση προβλημάτων και όλο και περισσότεροι επιστήμονες ασχολήθηκαν με αυτούς, το οποίο οδήγησε σε νέες προσεγγίσεις και καινούργιες στρατηγικές. Κάποιες από τις σημαντικότερες από αυτές είναι οι παρακάτω:

- Μία από αυτές τις τεχνικές είναι ο Μιμητικός Αλγόριθμος (Memetic Algorithm -MA), έννοια την οποία εισήγαγε για πρώτη φορά ο Moscato (Moscato, 1989). Τη στιγμή που οι GA είναι εμπνευσμένοι από τη βιολογική εξέλιξη, την οποία και προσπαθούν να μιμηθούν, οι MA προσπαθούν να μιμηθούν την πολιτιστική εξέλιξη. Ο MA είναι ο συνδυασμός μεταξύ της πληθυσμιακής καθολικής αναζήτησης και της ευρετικής τοπικής αναζήτησης την οποία κάνει κάθε άτομο.
- Το 1991 οι Colorni, Dorigo και Maniezzo προτείνουν ένα νέο αλγόριθμο βελτιστοποίησης ο οποίος μιμείται μια αποικία μυρμηγκιών (Ant Colony Optimization Algorithm) (Colorni et al., 1991).
- Το 1994 ο Reynolds εισήγαγε τους Πολιτισμικούς Αλγόριθμους (Cultural Algorithms). Οι αλγόριθμοι αυτοί χρησιμοποιούν ένα πλήθος ατόμων το οποίο περιγράφεται από ένα σύνολο γνωρίσματα και συμπεριφορές και επίσης από μια γενικότερη περιγραφή των εμπειριών τους η οποία προκύπτει από επικοινωνία και ανταλλαγή πληροφοριών μεταξύ του πληθυσμού, το οποίο αποτελεί την πολιτισμική συνιστώσα του αλγόριθμου (Reynolds, 1994).
- Το 1994 ο Adleman προσπάθησε να λύσει συνδυαστικά προβλήματα χρησιμοποιώντας εργαλείο της μοριακής βιολογίας. Ο ίδιος περιγράφει πως κωδικοποίησε έναν μικρό γράφο σε μοριακό επίπεδο DNA, και πως οι “λειτουργίες” των υπολογισμών διεξήχθησαν με κανονικά πρωτόκολλα και ένζυμα (Adleman, 1994).
- Το 1995 οι Kennedy και Eberhart αναπτύσσουν έναν καινούργιο αλγόριθμο βελτιστοποίησης, ο οποίος ονομάζεται Particle Swarm Optimization (PSO) (Kennedy and Eberhart, 1995) (Kennedy, 1997). Όπως μας προϊδεάζει το όνομα, ο αλγόριθμος αυτός χρησιμοποιεί σμήνος σωματιδίων. Ο PSO αναπαριστά την κίνηση που κάνει ένα σμήνος πτηνών ή ένα κοπάδι ψαριών. Πέραν από την προφανή σχέση του αλγόριθμου με την τεχνητή ζωή (Artificial Life), αυτός σχετίζεται και με τους GA και τον EP.
- Το 1997 οι Storn και Price προτείνουν μια νέα ευρετική προσέγγιση βελτιστοποίησης η οποία ονομάζεται Differential Evolution (DE). Όπως γράφουν η νέα μέθοδος αυτή μπορεί να αντιμετωπίσει μη-γραμμικές και μη- διαφορίσιμες συναρτήσεις. Ο DE χρησιμοποιεί πληθυσμό ατόμων και επίσης ένα είδος μετάλλαξης και διασταύρωσης (Storn and Price, 1997).
- Το 1999 οι Dasgupta και Forrest ανέπτυξαν μια μέθοδο που ονομάζεται Artificial Immune Systems (Dasgupta and Forrest, 1999). Η μέθοδος αυτή προσπαθεί να μιμηθεί

το ανθρώπινο ανοσοποιητικό σύστημα ώστε να επιλύσει προβλήματα και έχει εμπνευστεί από μια πρωτύτερη δημοσίευση της Forrest (Forrest et al., 1994).

- Το 2002 οι Larranaga και Lozano ανέπτυξαν μια νέα οικογένεια αλγορίθμων, τους Estimation of distribution algorithms. Οι αλγόριθμοι αυτοί ανήκουν στους GA και EA, καθώς βασίζονται στη φυσική επιλογή, αλλά αντί τη γενετική (συντελεστές, μετάλλαξη κλπ.) χρησιμοποιούν πιθανολογικά μοντέλα των παρατηρούμενων καλύτερων σημείων. Για αυτό μερικές φορές ονομάζονται και probabilistic model-building GAs (Larranaga and Lozano, 2002).

1.3 Πρακτικές εφαρμογές Εξελικτικών Αλγορίθμων

Όπως είπαμε οι EA έχουν αποδειχτεί πολύ χρήσιμοι λόγω της αποτελεσματικότητας τους σε προβλήματα βελτιστοποίησης αλλά και λόγω της προσαρμοστικότητας και της ευελιξίας που προσφέρουν. Για τους παραπάνω λόγους οι EA χρησιμοποιούνται σε πρακτικές εφαρμογές σε επιστήμες όπως η μηχανική, τα οικονομικά, η βιολογία κ.α. Οι Bäck, Hammel και Schwefel αναφέρουν μερικές από αυτές τις πρακτικές εφαρμογές (Bäck et al.,1997):

- Εφαρμογές μηχανικού σχεδιασμού που χρησιμοποιούν συνεχής παραμέτρους. Τέτοια παραδείγματα είναι η κατασκευή αεροσκαφών, προβλήματα δομικής μηχανικής βασισμένα σε σχηματική αναπαράσταση δύο διαστάσεων, ηλεκτρομαγνητικά συστήματα κ.α.
- Εφαρμογές μηχανικού σχεδιασμού που χρησιμοποιούν διακριτές παραμέτρους. Μερικά παραδείγματα εφαρμογών είναι η βελτιστοποίηση ψηφιακού φίλτρου, ο σχεδιασμός γραμμικού επιταχυντή και βελτιστοποίηση της ρύθμισης καυσίμου σε πυρηνικό αντιδραστήρα.
- Εφαρμογές μηχανικού σχεδιασμού mixed-integer, που χρησιμοποιούν δηλαδή και ακέραιες και συνεχής παραμέτρους. Κάποιες τέτοιες εφαρμογές είναι ο σχεδιασμός βιώσιμων δικτύων και ο σχεδιασμός οπτικών συστημάτων.
- Εφαρμογές σε προβλήματα βελτιστοποίησης συνδυαστικής τα οποία χρησιμοποιούν δυαδική αναπαράσταση των λύσεων, όπως η εφαρμογή στον προγραμματισμό του πληρώματος σε αεροπορικές εταιρείες, το πρόβλημα του σάκου (Knapsack problem) και άλλα προβλήματα συνδυαστικής.

- Εφαρμογές σχετικά με προβλήματα συνδυαστικής στις οποίες χρησιμοποιείται μετάλλαξη στην αναπαράσταση των λύσεων βρίσκονται στον τομέα του χρονικού προγραμματισμού. Τέτοιες είναι ο προγραμματισμός παραγωγής, ο προγραμματισμός ανάθεσης βέλτιστων διαδρομών, η δρομολόγηση κλήσεων σε δίκτυα τηλεπικοινωνίας και ο προγραμματισμός φόρτωσης παλετών.
- Επίσης διάφορες εφαρμογές στα χρηματοοικονομικά, στις επιχειρήσεις και τα οικονομικά.

2 Βελτιστοποίηση με τη χρήση Εξελικτικών Αλγορίθμων

2.1 Γενικευμένη περιγραφή ενός Εξελικτικού Αλγόριθμου

Σε αυτό το σημείο θα εξηγήσουμε πιο αναλυτικά τη λειτουργία των ΕΑ. Όπως είπαμε, οι ΕΑ μιμούνται τη βιολογική διαδικασία εξέλιξης και συγκεκριμένα βασίζονται στην φυσική επιλογή και την επιβίωση του ισχυρότερου. Στους ΕΑ η αντικειμενική συνάρτηση έχει τον ρόλο του περιβάλλοντος, οι υποψήφιες λύσεις αντιπροσωπεύουν τα άτομα του πληθυσμού και οι καταλληλότητα των ατόμων ελέγχεται με την τιμή της αντικειμενικής συνάρτησης για κάθε υποψήφια λύση.

Αρχικά σε κάθε ΕΑ θέτουμε κάποιο κριτήριο τερματισμού, το οποίο μπορεί να είναι είτε το υπολογιστικό κόστος το οποίο είμαστε διατεθειμένοι να δαπανήσουμε για την επίλυση του προβλήματος, είτε κάποιο επιθυμητό αποτέλεσμα της αντικειμενικής συνάρτησης είτε κάποιο χρονικό όριο είτε κάτι πιο πολύπλοκο όπως για παράδειγμα η μη βελτίωση της βέλτιστης λύσης για κάποιον αριθμό επαναλήψεων. Το υπολογιστικό κόστος μπορεί να εκφραστεί είτε με τις συνολικές επαναλήψεις ή αλλιώς με τις συνολικές γενιές τις οποίες θέλουμε να δημιουργήσουμε, είτε με τον συνολικό αριθμό αξιολογήσεων της αντικειμενικής συνάρτησης.

Εκτός από το κριτήριο τερματισμού, θα πρέπει να ορίσουμε και τιμές για τις μεταβλητές που χρησιμοποιεί ο ΕΑ. Τέτοιες είναι για παράδειγμα το μέγεθος του πληθυσμού, το πλήθος των γονέων για κάθε επόμενη γενιά, οι τιμές των συντελεστών μετάλλαξης και διασταύρωσης. Γενικά ένας ΕΑ είναι προτιμότερο να μην έχει πολλές μεταβλητές ρυθμίσεις και να είναι αποτελεσματικός σε διάφορα προβλήματα με όσο το δυνατόν λιγότερες ρυθμίσεις γίνεται.

Με την εκκίνηση του ΕΑ δημιουργείται ένας πληθυσμός από υποψήφιες λύσεις, συνήθως με τυχαίο τρόπο, λαμβάνοντας όμως υπόψιν το πεδίο ορισμού του προβλήματος. Ο πληθυσμός αυτός αποτελεί την πρώτη γενιά. Κατόπιν, για κάθε άτομο του πληθυσμού, ελέγχεται η τιμή της αντικειμενικής συνάρτησης. Έτσι ελέγχεται η καταλληλότητα του κάθε ατόμου και με βάση αυτήν επιλέγεται ποια άτομα θα συμμετέχουν στην δημιουργία της επόμενης γενιάς.

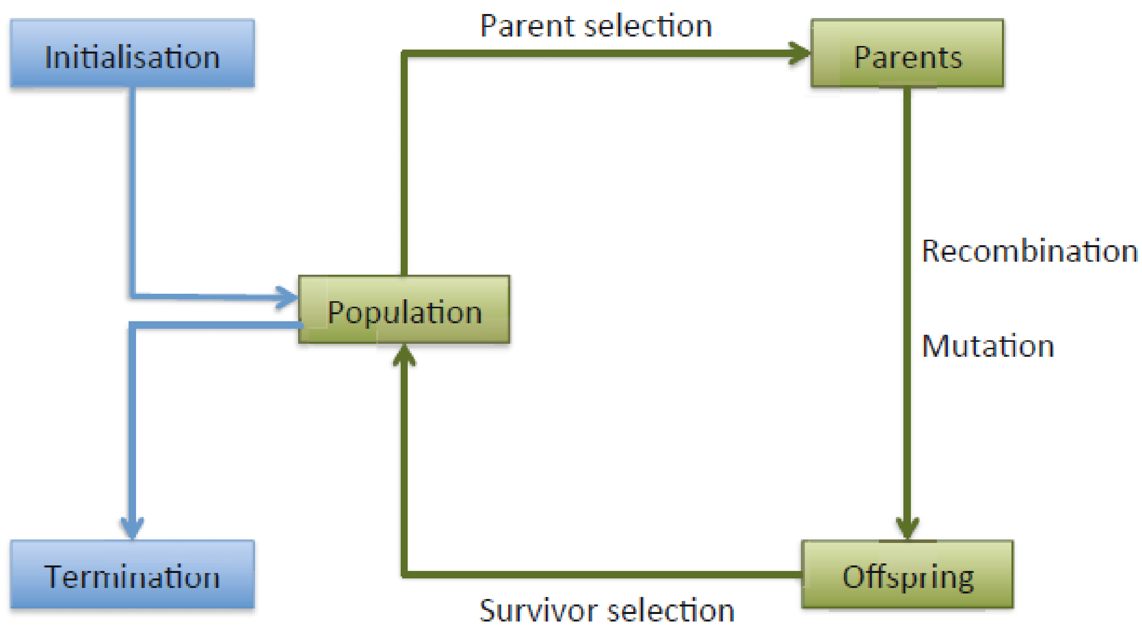
Σε αυτό το σημείο επιλέγονται οι γονείς οι οποίοι θα δημιουργήσουν την επόμενη γενιά και αρχίζει η διαδικασία αναπαραγωγής, η οποία διαφέρει από αλγόριθμο σε αλγόριθμο. Ο αριθμός των γονέων είναι κάτι το οποίο ορίζεται στην αρχή του αλγορίθμου. Η επιλογή των γονέων μπορεί βέβαια να μην βασίζεται πάντα μόνο στην καταλληλότητα η οποία εκφράζεται μέσω της αντικειμενικής συνάρτησης, αλλά να προστίθεται και κάποιος παράγοντας τύχης.

Τώρα έχει δημιουργηθεί η νέα γενιά και πρέπει να αποφασιστεί ποια άτομα θα επιζήσουν. Αυτή είναι μια επαναληπτική διαδικασία η οποία θα τερματιστεί όταν ικανοποιηθεί το κριτήριο τερματισμού το οποίο έχουμε ορίσει.

Παρακάτω στο Σχήμα 2.1 παρουσιάζεται ο ψευδό-κώδικας ενός γενικευμένου ΕΑ όπως περιγράφηκε παραπάνω και στο Σχήμα 2.2 παρουσιάζεται η επαναληπτική διαδικασία της διαδοχής των γενεών εκφρασμένη σε διάγραμμα ροής:

```
BEGIN
  INITIALISE population with random candidate solutions;
  EVALUATE each candidate;
  REPEAT UNTIL ( TERMINATION CONDITION is satisfied ) DO
    1 SELECT parents;
    2 RECOMBINE pairs of parents;
    3 MUTATE the resulting offspring;
    4 EVALUATE new candidates;
    5 SELECT individuals for the next generation;
  OD
END
```

Σχήμα 2.1: Γενικό σχεδιάγραμμα ενός ΕΑ σε ψευδό-κώδικα (Eiben and Smith, 2003)



Σχήμα 2.2: Γενικό σχεδιάγραμμα ενός ΕΑ εκφρασμένο σε διάγραμμα ροής (Eiben and Smith, 2003)

2.2 Τα μέρη ενός Εξελικτικού Αλγόριθμου

Πιο πάνω περιγράψαμε το γενικό σχεδιάγραμμα των ΕΑ. Τώρα θα δούμε τα συστατικά, τις διαδικασίες και τους συντελεστές οι οποίοι θα πρέπει να καθοριστούν ώστε να οριστεί ένας ΕΑ. Σύμφωνα με τους Eiben και Smith τα μέρη αυτά είναι (Eiben and Smith, 2003):

- Αναπαράσταση (καθορισμός των ατόμων)
- Αντικειμενική συνάρτηση
- Πληθυσμός
- Μηχανισμός επιλογής γονέων
- Συντελεστές παραλλαγής
- Μηχανισμός επιλογής επιζώντων

2.2.1 **Αναπαράσταση – Καθορισμός των ατόμων**

Το πρώτο βήμα στον ορισμό ενός ΕΑ είναι να συνδέσουμε τον «πραγματικό κόσμο» με τον «κόσμο του ΕΑ». Αυτό περιλαμβάνει πολλές φορές την απλοποίηση του προβλήματος στον πραγματικό κόσμο, έτσι ώστε να προκύψει ένα σαφώς καθορισμένο πρόβλημα «μοντέλο» μέσω του οποίου μπορούν να υπάρξουν και να υπολογιστούν πιθανές λύσεις. Το πρώτο βήμα για να υλοποιηθεί αυτό είναι να βρούμε έναν τρόπο με τον οποίο θα καθορίζονται και θα αποθηκεύονται οι λύσεις ώστε να μπορούν να χειριστούν από υπολογιστή.

Τα αντικείμενα τα οποία σχηματίζουν υποψήφιες λύσεις στο πρόβλημα στον πραγματικό κόσμο αναφέρονται ως φαινότυποι (phenotypes), ενώ η κωδικοποίησή τους, η οποία είναι ουσιαστικά τα άτομα μέσα στον ΕΑ αναφέρονται ως γενότυποι (genotypes). Το πρώτο αυτό σκέλος της δημιουργίας του «κόσμου του ΕΑ», το οποίο είναι ο καθορισμός κάποιας σχέσης μεταξύ φαινοτύπων και γενοτύπων, αναφέρεται ως αναπαράσταση (representation) στη διεθνή βιβλιογραφία. Συνοψίζοντας, μπορούμε να πούμε πως σε αυτό το βήμα πρόκειται για κωδικοποίηση και αποκωδικοποίηση των λύσεων μεταξύ του «πραγματικού κόσμου» και του «ΕΑ κόσμου».

2.2.2 **Αντικειμενική συνάρτηση**

Ο ρόλος της αντικειμενικής συνάρτησης είναι να αντιπροσωπεύει τις απαιτήσεις στις οποίες θα πρέπει να προσαρμοστεί ο πληθυσμός. Αποτελεί τη βάση για τη φυσική επιλογή και καθορίζει τι σημαίνει βελτίωση. Από την οπτική της επίλυσης του προβλήματος, αυτή αντιπροσωπεύει το πρόβλημα το οποίο πρέπει να επιλυθεί. Από τεχνικής απόψεως, η αντικειμενική συνάρτηση είναι η αυτή η οποία εκχωρεί ποιοτική κλίμακα στους γενοτύπους.

2.2.3 **Πληθυσμός**

Ο ρόλος του πληθυσμού είναι να αναπαριστά υποψήφιες λύσεις. Ο πληθυσμός είναι ένα σύνολο γενοτύπων και ο πληθυσμός είναι αυτός ο οποίος εξελίσσεται. Δεδομένου ότι έχουμε ορίσει την αναπαράσταση, ο καθορισμός ενός πληθυσμού απαιτεί μόνο τον καθορισμό του πλήθους των ατόμων σε αυτόν. Σχεδόν σε όλους τους ΕΑ το μέγεθος του πληθυσμού είναι σταθερό καθ’ όλη την διάρκεια της εξελικτικής διαδικασίας.

2.2.4 **Μηχανισμός επιλογής γονέων**

Ο ρόλος του μηχανισμού επιλογής γονέων είναι να καθορίσει τους γονείς ανάμεσα στα άτομα, βάση της καταλληλότητας τους, ώστε να επιτρέψει στα καταλληλότερα άτομα να γίνουν οι γονείς της επόμενης γενιάς. Μαζί με τον μηχανισμό επιλογής επιζώντων, ο μηχανισμός επιλογής γονέων ευθύνονται για την δημιουργία πίεσης για βελτίωση της ποιότητας του πληθυσμού.

2.2.5 Συντελεστές παραλλαγής

Ο ρόλος αυτών των συντελεστών είναι να δημιουργήσουν άτομα από παλιά άτομα, συγκεκριμένα από αυτά των γονέων. Οι συντελεστές παραλλαγής χωρίζονται σε δύο βασικές κατηγορίες, βασιζόμενες στο πλήθος ατόμων: Αυτόν που εφαρμόζεται σε ένα άτομο (μετάλλαξη-mutation) και αυτόν που εφαρμόζεται σε n-πλήθος ατόμων (διασταύρωση-recombination).

Ο συντελεστής μετάλλαξης είναι ένας συντελεστής ο οποίος εφαρμόζεται σε έναν γενότυπο και προσφέρει έναν τροποποιημένο απόγονο. Ένας συντελεστής μετάλλαξης είναι πάντα στοχαστικός: το αποτέλεσμά του, ο απόγονος, εξαρτάται από την έκβαση μιας σειράς τυχαίων επιλογών. Θα πρέπει να διευκρινιστεί πως όλοι οι συντελεστές οι οποίοι εφαρμόζονται σε ένα άτομο, δεν μπορούν να οριστούν αναγκαστικά και ως συντελεστές μετάλλαξης. Για παράδειγμα ένας συντελεστής ο οποίος δρα συστηματικά σε κάποιο άτομο προσπαθώντας να επιδιορθώσει μια αδυναμία, δεν μπορεί να θεωρηθεί συντελεστής μετάλλαξης. Ο συντελεστής μετάλλαξης υποτίθεται πως προκαλεί τυχαία και αμερόληπτη αλλαγή.

Η διασταύρωση βασίζεται στην απλή ιδέα: ζευγαρώνοντας δυο άτομα με διαφορετικά αλλά επιθυμητά χαρακτηριστικά, μπορούμε να δημιουργήσουμε απόγονους οι οποίοι συνδυάζουν αυτά τα χαρακτηριστικά. Η αρχή αυτή έχει βάση καθώς για χιλιετίες εφαρμόζεται στα φυτά και στα ζώα από τους ανθρώπους με σκοπό να δημιουργήσουν είδη με καλύτερα χαρακτηριστικά. Ο συντελεστής διασταύρωσης είναι ένας δυαδικός συντελεστής ο οποίος συγχωνεύει πληροφορίες από δύο γονείς σε έναν απόγονο. Όπως και στην μετάλλαξη, ο συντελεστής διασταύρωσης είναι στοχαστικός: οι επιλογές για το ποια μέρη ποιου γονέα θα συνδυαστούν, και πως ακριβώς θα συνδυαστούν, βασίζεται σε μια σειρά τυχαίων γεγονότων ή αλλιώς ας τις πούμε «κληρώσεις». Ο ρόλος της διασταύρωσης διαφέρει στους διάφορους τομείς της EC. Για παράδειγμα στον GP είναι συχνά ο μόνος συντελεστής παραλλαγής ο οποίος χρησιμοποιείται, ενώ στον EP δεν χρησιμοποιείται ποτέ. Συντελεστές διασταύρωσης για παραπάνω από δύο άτομα εφαρμόζονται σπάνια. Παρόλα αυτά κάποιες μελέτες δείχνουν ότι μπορεί να έχουν θετική επίδραση στην εξέλιξη.

2.2.6 **Μηχανισμός επιλογής επιζώντων**

Ομοίως με τον μηχανισμό επιλογής γονέων, ο ρόλος του μηχανισμού επιλογής επιζώντων είναι να καθορίζει τα άτομα με τα καλύτερα χαρακτηριστικά. Ο μηχανισμός αυτός εφαρμόζεται μετά από τη δημιουργία των απογόνων από τους επιλεγμένους γονείς. Το πόσα άτομα θα επιβιώσουν από τον πληθυσμό καθορίζεται εξ αρχής. Το ποια άτομα θα είναι αυτά τα οποία θα περάσουν στον επόμενο γύρο βασίζεται στις τιμές της αντικειμενικής συνάρτησης για αυτά. Ο μηχανισμός αυτός είναι τις περισσότερες φορές ντετερμινιστικός.

2.3 Εισαγωγή στη Βελτιστοποίηση

Η Επιχειρησιακή Έρευνα (Operational Research-OR) και η Διοικητική Επιστήμη (Management Science-MS) είναι κλάδοι οι οποίοι προσπαθούν να συνεισφέρουν σε προβλήματα λήψης αποφάσεων, αναπτύσσοντας μαθηματικά μοντέλα τα οποία περιγράφουν προβλήματα του πραγματικού κόσμου, και μετά να τα επιλύσουν εφαρμόζοντας μαθηματικές διαδικασίες (Hillier and Hillier, 2003).

Κάθε πρόβλημα βελτιστοποίησης περιγράφετε από μία αντικειμενική συνάρτηση ή αλλιώς συνάρτηση στόχος (Objective Function), η οποία πρέπει να ελαχιστοποιηθεί ή να μεγιστοποιηθεί, το οποίο δεν έχει σημασία καθώς πολύ εύκολα μετατρέπετε ένα πρόβλημα μεγιστοποίησης σε πρόβλημα ελαχιστοποίησης αλλάζοντας το πρόσημο. Η αντικειμενική συνάρτηση έχει n μεταβλητές, οι οποίες είναι n ποσοτικά εκφρασμένες αποφάσεις οι οποίες θα πρέπει να παρθούν, βλέποντας το πρόβλημα από την σκοπιά των OR και MS. Οι μεταβλητές αυτές έχουν η κάθε μια και πεδίο ορισμού. Τα προβλήματα βελτιστοποίησης συχνά έχουν περιορισμούς ανισότητας ή/και ισότητας, οι οποίοι θα πρέπει να πρέπει να ικανοποιούνται ώστε να έχουμε εφικτή λύση (feasible solution). Το σύνολο όλων των εφικτών λύσεων ονομάζεται εφικτή περιοχή (feasible region). Μια λύση η οποία ανήκει στην εφικτή περιοχή και μεγιστοποιεί ή ελαχιστοποιεί την αντικειμενική συνάρτηση, ανάλογα το τι πρόβλημα έχουμε, ονομάζεται βέλτιστη λύση (optimal solution).

Η γενική μορφή ενός προβλήματος βελτιστοποίησης είναι:

$$Minimize\ f(\vec{x}) \tag{2.1}$$

Υπό τους περιορισμούς:

$$g_i(\vec{x}_i) \leq 0, \quad \mathrm{i} = 1, \dots, \mathrm{m} \tag{2.2}$$

$$h_i(\vec{x}_i) = 0, \quad \mathrm{i} = 1, \dots, \mathrm{p} \tag{2.3}$$

όπου $\vec{x}$ είναι ένα διάνυσμα λύσης $\vec{x}$=[x_1, x_2,..., x_n] και το πεδίο ορισμού του διανύσματος ορίζεται για κάθε διάσταση:

$$x_{min,j} \leq x_j \leq x_{max,j} \quad j=1,2,\dots,n \tag{2.4}$$

m είναι ο αριθμός των περιορισμών ανισότητας, και p ο αριθμός περιορισμών ισότητας. Αν ορίσουμε ως S το πεδίο εφικτών λύσεων και με R ολόκληρο το πεδίο τιμών, τότε ισχύει $S \subseteq R$. Οι περιορισμοί ισότητας συνήθως μετατρέπονται σε περιορισμούς ανισότητας, κάνοντάς τους λίγο πιο ευέλικτους ως εξής:

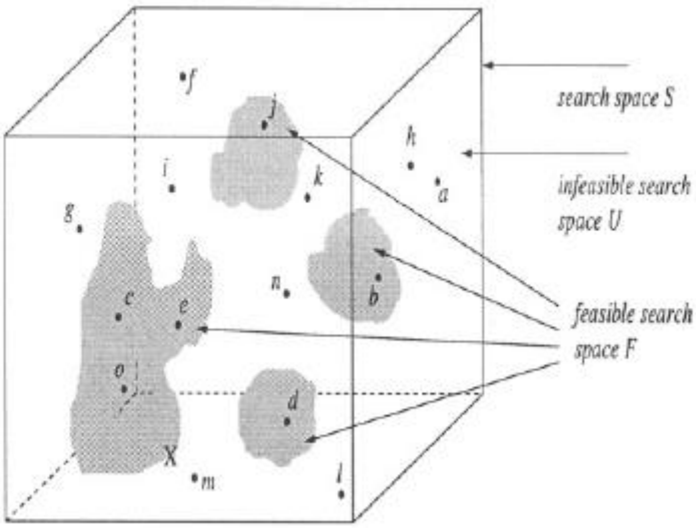
$$|h_i(x_i)| - \varepsilon \leq 0, \mathrm{i} = 1, \dots, \mathrm{p} \tag{2.5}$$

όπου ε ένας μικρός αριθμός, για παράδειγμα 0.0001.

2.4 Μέθοδοι αντιμετώπισης περιορισμών

Όπως είπαμε μέχρι τώρα οι ΕΑ αποτελούν μια οικογένεια από αλγόριθμους οι οποίοι έχουν αποδειχτεί πολύ ικανοί στο να αντιμετωπίζουν δύσκολα προβλήματα βελτιστοποίησης, και για αυτό έχουν τραβήξει το ενδιαφέρον μεγάλου μέρους της επιστημονικής κοινότητας. Παρόλα αυτά, οι ΕΑ μπορούν να αντιμετωπίσουν μόνο προβλήματα βελτιστοποίησης χωρίς περιορισμούς (unconstrained problems). Αν θέλουμε να επιλύσουμε μια συνάρτηση με περιορισμούς, θα χρειαστεί να ενσωματώσουμε σε αυτόν (τον ΕΑ) μια τεχνική χειρισμού περιορισμών (Constraint Handling Technique-CHT). Σε αυτό το κεφάλαιο θα παρουσιάσουμε μερικές από αυτές.

Οι τεχνικές αυτές δίνουν τη δυνατότητα στον ΕΑ να επεξεργάζεται προβλήματα constrained όπως θα επεξεργαζόταν κάποιο πρόβλημα unconstrained. Κάθε τεχνική έχει πλεονεκτήματα και μειονεκτήματα. Άλλες τεχνικές απορρίπτουν απευθείας τις τιμές που βρίσκονται σε περιοχές μη εφικτές ενώ άλλες τεχνικές τις επεξεργάζονται, και σε κάποιες μελέτες έχει διατυπωθεί ότι είναι πολύ σημαντικό να επεξεργάζεται ο αλγόριθμος και μη εφικτές λύσεις (Michalewicz, 1995). Τον λόγο μπορούμε να τον σκεφτούμε εύκολα αν αναλογιστούμε ένα πεδίο εφικτών λύσεων το οποίο αποτελείται από δύο μη συνεχή υποσύνολα. Το Σχήμα 2.3 απεικονίζει ένα πεδίο αναζήτησης το οποίο αποτελείται από πεδία αναζήτησης εφικτών λύσεων τα οποία είναι ασύνδετα. Αν κάποιος αλγόριθμος δεν επεξεργάζεται μη εφικτές λύσεις μπορεί να εγκλωβιστεί σε ένα από τα πεδία εφικτών λύσεων, ενώ το βέλτιστο βρίσκεται σε κάποιο από τα υπόλοιπα.



Σχήμα 2.3: Ένα πεδίο αναζήτησης λύσεων το οποίο αποτελείται από μη εφικτά και εφικτά υποσύνολα με έναν πληθυσμό 15 ατόμων (Michalewicz and Schmidt, 2003).

Γεγονός είναι, πως κάθε πρόβλημα στον πραγματικό κόσμο συνοδεύεται από περιορισμούς, τους οποίους δεν μπορείς να τους αποφύγεις (Michalewicz and Schmidt, 2003). Το 1990 οι Dhar και Ranganathan έγραψαν (Dhar and Ranganathan, 1990):

“Virtually all decision making situations involve constraints. What distinguishes various types of problems is the form of these constraints. Depending on how the problem is visualized, they can arise as rules, data dependencies, algebraic expressions, or other forms.”

Στο παραπάνω απόσπασμα οι Dhar και Ranganathan κάνουν σαφές πως η λήψη αποφάσεων συνοδεύεται πάντα από περιορισμούς και πως αυτό που διακρίνει τα διάφορα είδη προβλημάτων είναι ο τύπος αυτόν τον περιορισμών.

Οι CHT συνοψίζονται στις παρακάτω κατηγορίες (Coello, 2002):

- Συνάρτηση ποινής (Penalty function)
- Ειδική αναπαράσταση και συντελεστές (Special representations and operators)
- Αλγόριθμοι επιδιόρθωσης (Repair algorithms)
- Διαχωρισμός αντικειμενικής συνάρτηση και περιορισμών (Separation of objectives and constraints)
- Υβριδικές μέθοδοι (Hybrid Methods)

Παρακάτω θα δούμε διάφορες μεθόδους αυτών των κατηγοριών. Σε άλλες κατηγορίες θα εστιάσουμε περισσότερο και σε άλλες λιγότερο. Το αντικείμενο του χειρισμού περιορισμών σε προβλήματα βελτιστοποίησης έχει πάρα πολύ μεγάλο όγκο δημοσιεύσεων στη διεθνή βιβλιογραφία, και σε καμία περίπτωση δεν μπορεί να αναλυθεί πλήρως στα πλαίσια αυτής της διπλωματικής. Η πηγή για αυτές τις μεθόδους είναι κυρίως η εργασία του Coello: *"Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: A survey of the state of the Art "* (Coello,2002).

2.4.1 **Συνάρτηση ποινής (Penalty function)**

Η πιο συνήθης μέθοδος αντιμετώπισης περιορισμών στην κοινότητα των ΕΑ είναι η συνάρτηση ποινής. Οι συναρτήσεις ποινής αρχικά είχαν προταθεί από τον Courant την δεκαετία του 1940 (Courant, 1943) και αργότερα επεκτάθηκαν από τον Carroll (Carroll, 1961) και τους Fiacco και McCormick (Fiacco and McCormick, 1966). Η ιδέα πίσω από την μέθοδο αυτήν είναι να μετατρέπεται ένα πρόβλημα βελτιστοποίησης constrained σε unconstrained, προσθέτοντας μια συγκεκριμένη τιμή στην αντικειμενική συνάρτηση η οποία θα εξαρτάται από την ικανοποίηση ή όχι των περιορισμών από την συγκεκριμένη λύση.

Υπάρχουν δύο είδη συναρτήσεων ποινής: εξωτερικές (external) και εσωτερικές (internal). Στην περίπτωση των εξωτερικών μεθόδων ξεκινάμε από μια μη εφικτή λύση (infeasible) και από εκεί κατευθυνόμαστε προς την περιοχή εφικτών λύσεων (feasible region). Στην περίπτωση των εσωτερικών μεθόδων, η ποινή επιλέγεται έτσι ώστε σε περιοχές εφικτών λύσεων και μακριά από το όριο εφικτών και μη εφικτών λύσεων, αυτή να είναι πολύ μικρή, ενώ όταν κοντεύει αυτό το όριο τότε η ποινή τείνει στο άπειρο, με αποτέλεσμα όταν ξεκινάμε την αναζήτηση στο πεδίο εφικτών λύσεων, η αναζήτηση δεν μπορεί να βγει έξω από αυτό το πεδίο.

Η πιο κοινή μέθοδος η οποία χρησιμοποιείται στους ΕΑ είναι της εξωτερικής συνάρτησης ποινής, και αυτή θα αναλύσουμε. Πολύ σημαντικός παράγοντας για αυτό είναι διότι στην βελτιστοποίηση με ΕΑ δεν απαιτείται να ξεκινήσουμε με μια εφικτή λύση. Στην βελτιστοποίηση με ΕΑ είναι σημαντικό να μπορεί να δουλέψει ο αλγόριθμος στα όρια εφικτών και μη εφικτών λύσεων. Αυτό είναι και το κύριο μειονέκτημα των μεθόδων εσωτερικής συνάρτησης ποινής.

Η γενική διατύπωση της εξωτερικής συνάρτησης ποινής είναι η παρακάτω:

$$\varphi(\vec{x}) = f(\vec{x}) \pm \left[\sum_{i=1}^{n} r_i \times G_i + \sum_{j=1}^{p} c_j \times H_j \right]^{\beta} \tag{2.6}$$

Στην παραπάνω εξίσωση η f είναι η αντικειμενική συνάρτηση η αρχική, και η φ είναι η νέα αντικειμενική συνάρτηση η οποία συμπεριλαμβάνει τους περιορισμούς. Πλέον καλούμαστε να ελαχιστοποιήσουμε ή να μεγιστοποιήσουμε την φ. Οι G_i και H_j είναι συναρτήσεις οι οποίες εξαρτώνται από τους περιορισμούς ανισότητας και ισότητας αντίστοιχα. Όταν οι περιορισμοί ικανοποιούνται, τότε οι G_i και H_j έχουν τιμή ίση με μηδέν, ενώ όταν καταπατούνται είναι θετικές και ίσες με αυτούς. Οι συντελεστές r_i και c_j είναι θετικές σταθερές οι οποίες ονομάζονται συντελεστές ποινής (penalty factors). Το β θέτεται συνήθως ίσο με 1 ή 2. Επίσης το + χρησιμοποιείται όταν έχουμε πρόβλημα ελαχιστοποίησης, ενώ το – όταν έχουμε πρόβλημα μεγιστοποίησης, έτσι ώστε σε κάθε περίπτωση η καταπάτηση των περιορισμών να επιβαρύνει την αντικειμενική συνάρτηση όταν έχουμε μη εφικτή λύση. Για τις G_i και H_j ισχύει:

$$H_j = |h_i(\vec{x})| \tag{2.7}$$

Ιδανικά, η ποινή, ή αλλιώς η επιβάρυνση στην αντικειμενική συνάρτηση, θα έπρεπε να κρατείται τόσο χαμηλά ώστε να είναι ακριβώς πάνω από το επίπεδο στο οποίο μη εφικτές λύσεις θεωρούνται βέλτιστες. Αυτό ονομάζεται κανόνας ελάχιστης ποινής (minimum penalty rule). Στην γενική εξίσωση (2.6) φαίνεται πως η ποινή εξαρτάται από τους συντελεστές ποινής. Αν εμείς θέσουμε υψηλές τιμές σε αυτούς τότε η ποινή θα είναι και αυτή υψηλή. Το ποια τιμή θεωρείται υψηλή διαφέρει από πρόβλημα σε πρόβλημα.

Όταν η ποινή παίρνει πολύ υψηλές ή πολύ χαμηλές τιμές τότε η βελτιστοποίηση από τον EA καθίσταται πολύ δύσκολη. Όταν η ποινή είναι πολύ μεγάλη και η βέλτιστη λύση βρίσκεται κοντά στα όρια εφικτών και μη εφικτών λύσεων, τότε ο EA ωθείται πολύ γρήγορα μακριά από το όριο αυτό και μέσα στην περιοχή εφικτών λύσεων. Η κατάσταση αυτή λέγεται over-penalization, δηλαδή ότι η αντικειμενική συνάρτηση «τιμωρείται» παραπάνω από όσο θα έπρεπε ή με άλλα λόγια οι περιορισμοί έχουν μεγαλύτερη επιρροή στην αντικειμενική συνάρτηση από όσο θα έπρεπε. Όταν σε αντίθετη περίπτωση η ποινή είναι πολύ χαμηλή τότε λέμε πως έχουμε under-penalization. Σε αυτήν την περίπτωση ο EA ωθείται εκτός της περιοχής εφικτών λύσεων, καθώς η επιρροή των περιορισμών στην αντικειμενική συνάρτηση είναι αμελητέα.

Τώρα θα παρουσιάσουμε μερικές από τις διαφορετικές προσεγγίσεις των συναρτήσεων ποινής:

2.4.1.1 *Σταθερές ποινές (Static Penalties)*

Σε αυτή την κατηγορία ανήκουν οι προσεγγίσεις στις οποίες οι συντελεστές ποινής δεν εξαρτώνται από την τρέχουσα επανάληψη/γενιά με κανέναν τρόπο, και έτσι παραμένουν σταθερές καθ’ όλη τη διάρκεια της εξελικτικής διαδικασίας. Μερικές από αυτές είναι:

- Οι Homifar, Lai και Qi πρότειναν μια προσέγγιση στην οποία ο χρήστης-ερευνητής θέτει διάφορα επίπεδα παραβίασης, και ένας συντελεστής ποινής ορίζεται για κάθε ένα από αυτά τα επίπεδα, με τέτοια λογική, ώστε οι συντελεστές να αυξάνονται όσο αυξάνεται και το επίπεδο τις παραβίασης. Αυτή η προσέγγιση ξεκινάει με έναν πληθυσμό του οποίου τα άτομα ορίζονται τυχαία (feasible or infeasible) (Homaifar et al., 1994). Και η εξίσωση (2.6) μετατρέπεται:

$$fitness(\vec{x}) = f(\vec{x}) \; + \left[\sum_{j=1}^{m} R_{k,i} \times \max\big(0, g_i(\vec{x})\big)^{\beta} \right] \tag{2.8}$$

Όπου $f(\vec{x})$ είναι η αρχική αντικειμενική συνάρτηση, $fitness(\vec{x})$ είναι η νέα αντικειμενική συνάρτηση η οποία περιλαμβάνει και τους περιορισμούς. $R_{k,i}$ είναι οι συντελεστές ποινές, όπου $i=1,\ldots m$ είναι ο αριθμός των περιορισμών και g_i οι περιορισμοί, και *k=1,…p* είναι ο αριθμός των επιπέδων παραβίασης τα οποία έχει ορίσει ο χρήστης.

- Μία διαφορετική προσέγγιση είναι αυτή που προτάθηκε από τους Morales και Quezada (Morales and Quezada, 1998), με την οποία η αντικειμενική υπολογίζεται με τον παρακάτω τρόπο:

$$fitness(\vec{x}) = \begin{cases} f(\vec{x}), & if\ solution\ feasible \\ K - \sum_{i=1}^{s} \frac{K}{m}, & otherwise \end{cases} \tag{2.9}$$

Όπου *s* είναι ο αριθμός των περιορισμών οι οποίοι ικανοποιούνται, *m* είναι ο συνολικός αριθμός περιορισμών και το *K* είναι μια μεγάλη σταθερά (συνήθως την έθεταν ίση με 1×10^9 στο πείραμα). Η παραπάνω μέθοδος βλέπουμε ότι δεν λαμβάνει υπόψιν κατά πόσο καταπατείται ένας περιορισμός.

- Οι Hoffmeister και Sprave πρότειναν την παρακάτω συνάρτηση ποινής (Hoffmeister and Sprave, 1996):

$$fitness(\vec{x}) = f(\vec{x}) + \sqrt{\sum_{i=0}^{m} H(-g_i(\vec{x}))\, g_i(\vec{x})^2} \tag{2.10}$$

Όπου H :R {0,1} είναι η συνάρτηση Heavyside:

$$H(y) = \begin{cases} 1 : y > 0 \\ 0 : y \leq 0 \end{cases} \tag{2.11}$$

- Οι Runarsson και Yao προτείνουν μια απλή προσέγγιση στην οποία ο συντελεστής ποινής είναι ένας, για το άθροισμα των παραβιάσεων όλων των περιορισμών (Runarsson and Yao, 2003).

$$\varphi(\vec{x}) = f(\vec{x}) + \lambda \times \sum_{i=1}^{m} \max\left(0, g_i(\vec{x})\right) \tag{2.12}$$

Ο συντελεστής ποινής, λ, ορίζεται αυθαίρετα από τον χρήστη, και για να βρεθεί το κατάλληλο λ θα χρειαστεί ο χρήστης να δουλέψει με την μέθοδο δοκιμής και πλάνης (trial and error) ώσπου να βρεθεί το λ για το οποίο ο ΕΑ βρίσκει το βέλτιστο αποτέλεσμα. Η παραπάνω διαδικασία γίνετε με τον λογική πως αν βρίσκουμε λύσεις οι οποίες παραβιάζουν τους περιορισμούς, τότε αυξάνουμε το λ, αν αντίθετα οι περιορισμοί ικανοποιούνται αλλά οι λύσεις, όσον αφορά την αντικειμενική συνάρτηση, δεν είναι ικανοποιητικές, τότε μειώνουμε το λ. Αν το λ είναι μικρό ή μεγάλο εξαρτάται πάντα από την συνάρτηση και τους περιορισμούς με τους οποίους δουλεύουμε. Για παράδειγμα μπορεί το λ ίσο με 1000 να είναι μεγάλο για μια συνάρτηση, ενώ για μια άλλη να είναι μικρό. Για τους παραπάνω λόγους η διαδικασία αυτή μπορεί να είναι επίπονη μερικές φορές, και να απαιτεί αρκετές δοκιμές, που ισοδυναμούν με υπολογιστικό κόστος, ώσπου να βρεθεί ένα λ για το οποίο να βρίσκουμε ικανοποιητικές λύσεις.

Θα πρέπει να αναφέρουμε πως η παραπάνω μεθοδολογία, στην οποία δοκιμάζουμε συντελεστή ποινής λ, και τον αυξάνουμε αν έχουμε παραβίαση περιορισμών, ενώ τον μειώνουμε αν δεν έχουμε παραβίαση αλλά έχουμε κακές λύσεις αντικειμενικής (της αρχικής αντικειμενικής), μπορεί να μην δουλεύει πάντα. Η μεθοδολογία αυτή μπορεί να μοιάζει πολύ λογική, καθώς ουσιαστικά αυξάνουμε και μειώνουμε επιρροή των περιορισμών μέσω του λ, αλλά μερικές φορές δεν έχει αποτέλεσμα και αυτό σχέση με την τοπολογία της αντικειμενικής συνάρτησης και των περιορισμών.

2.4.1.2 *Δυναμικές ποινές (Dynamic Penalties)*

Σε αυτή την κατηγορία συμπεριλαμβάνονται όλες οι συναρτήσεις ποινής στις οποίες η τρέχουσα γενιά ή αλλιώς επανάληψη της εξελικτικής διαδικασίας επηρεάζει τον υπολογισμό των συντελεστών ποινής, και συνήθως η συνάρτηση ποινής ορίζεται έτσι ώστε να αυξάνει η τιμή της με την πάροδο τον γενεών.

- Οι Joines και Houck πρότειναν μια τεχνική στην οποία άτομα υπολογίζονται, στην γενιά *t* χρησιμοποιώντας την παρακάτω σχέση, υποθέτοντας πως πρόκειται για πρόβλημα ελαχιστοποίησης (Joines and Houck, 1994) :

$$fitness(\vec{x}) = f(\vec{x}) + (C \times t)^{a} + SVC(\beta, \vec{x}) \qquad (2.13)$$

Όπου C, α και β είναι σταθερές οι οποίες ορίζονται από τον χρήστη. Οι συγγραφείς χρησιμοποίησαν C=0.5 , α=1 ή 2, β=1 ή 2), και SVC(β, $x^{\rightarrow}$) ορίζεται ως εξής:

$$SVC(\beta, \vec{x}) = \sum_{i=1}^{n} D^{\beta}{}_{i}(\vec{x}) + \sum_{i=1}^{n} D_i(\vec{x}) \quad (2.14)$$

$$D^{\beta}{}_{i}(\vec{x}) = \begin{cases} 0 & g_i(\vec{x}) \leq 0 \\ |g_i(\vec{x})| & otherwise \end{cases} \quad 1 \leq i \leq n \quad (2.15)$$

$$D_j(\vec{x}) = \begin{cases} 0 & -\varepsilon \leq h_j(\vec{x}) \leq \varepsilon \\ |h_j(\vec{x})| & otherwise \end{cases} \quad 1 \leq j \leq p \quad (2.16)$$

Όπου g_i οι περιορισμοί ανισότητας, όπου h_j ο περιορισμός ισότητας και το *ε* είναι ο μικρός αριθμός που είδαμε ότι χαλαρώνει τον περιορισμό ισότητας, μετατρέποντάς τον σε περιορισμό ανισότητας.

- Οι Kazarlis και Petridis δοκίμασαν μια άλλη προσέγγιση δυναμικής συνάρτησης ποινής, η οποία έχει την παρακάτω μορφή (Kazarlis and Petridis, 1998):

$$fitness(\vec{x}) = f(\vec{x}) + V(g) \times \left(A \sum_{i=1}^{m} (\delta_i \cdot w_i \cdot \Phi(d_i(S))) + B\right) \times \delta_s \quad (2.17)$$

Όπου *A* είναι ένας συντελεστής «δριμύτητας», *m* είναι ο συνολικός αριθμός περιορισμών, δ_s είναι 1 εάν ο περιορισμός *i* παραβιάζεται και 0 σε αντίθετη περίπτωση, w_i είναι ένας συντελεστής βαρύτητας για τον περιορισμό *i*, $d_i(S)$ είναι ένα μέτρο παραβίασης του περιορισμού *i* από την λύση *S*, *Φ*() είναι μια συνάρτηση αυτού το μέτρου, *B* είναι ένας συντελεστής ποινής που δουλεύει ως κάτω όριο, δ_s είναι ένας δυαδικός συντελεστής ίσος με 1 αν η λύση *S* είναι μη εφικτή και 0 σε αντίθετη περίπτωση, *V(g)* είναι μια αυξανόμενη συνάρτηση της *g* (τρέχουσας γενιάς) στο εύρος 0-1.

Χρησιμοποιώντας ως δοκιμαστικές συναρτήσεις για την παραπάνω μέθοδο το cutting stock problem και unit commitment problem, Kazarlis και Petridis δοκίμασαν διάφορες μορφές της *V(g)* , καταλήγοντας πως η καλύτερη απόδοση προκύτει από τη χρήση μιας συνάρτησης της μορφής:

$$\mathrm{V(g)} = \left(\frac{g}{G}\right)^2 \quad (2.18)$$

Όπου *G* ο συνολικός αριθμός γενεών που έχει οριστεί.

2.4.1.3 *Ποινές ανόπτησης (Annealing penalties)*

Οι Michalewicz και Attia (Michalewicz and Attia, 1994) επινοήσαν μια μέθοδο βασισμένη στην ιδέα της ανόπτησης (simulated annealing) (Kirkpatrick and Vecchi, 1983): οι συντελεστές ποινής αλλάζουν μια φορά μετά από κάποιες γενιές (αφού ο αλγόριθμος έχει παγιδευτεί σε τοπικά βέλτιστα). Μόνο οι ενεργοί περιορισμοί λαμβάνονται υπόψιν σε κάθε επανάληψη, και η ποινή αυξάνεται με τις επαναλήψεις, όπως μειώνεται αντίστοιχα η θερμοκρασία κατά την ανόπτηση, έτσι ώστε μη εφικτές λύσεις να επιβαρύνονται με μεγάλες ποινές στις τελευταίες επαναλήψεις.

- Η μέθοδος των Michalewicz και Attia, απαιτεί οι περιορισμοί να χωρίζονται σε τέσσερις κατηγορίες: γραμμικές ισότητες, γραμμικές ανισότητες, μη-γραμμικές ισότητες και μη-γραμμικές ανισότητες. Επίσης μια ομάδα ενεργών περιορισμών *A* πρέπει να δημιουργείται, και όλες οι μη-γραμμικές ισότητες μαζί με όλες τις παραβιασμένες μη-ανισότητες θα πρέπει να συμπεριλαμβάνονται σε αυτή την ομάδα. Ο πληθυσμός εξελίσσεται σύμφωνα με την παρακάτω εξίσωση (Michalewicz, 1995):

$$fitness(\vec{x}) = f(x^{\rightarrow}) + \frac{1}{2\tau} \sum_{\iota \in A} \varphi_i^2 (\vec{x}) \tag{2.19}$$

Όπου τ είναι η μεταβλητή η οποία ρυθμίζει το πρόγραμμα «ψύξης».

$$\varphi_i(\vec{x}) = \begin{cases} \max[0, g_i(\vec{x})] & if\ 1 \leq i \leq n \\ |h_i(\vec{x})| & if\ n+1 \leq i \leq m \end{cases} \tag{2.20}$$

2.4.1.4 *Προσαρμοσμένες ποινές (Adaptive penalties)*

Οι Bean και Hadj-Alouane (Bean and Hadj-Alouane, 1993) ανέπτυξαν μια μέθοδο η οποία χρησιμοποιεί συνάρτηση ποινής η οποία χρησιμοποιεί μηχανισμό ανάδρασης από την εξελικτική διαδικασία αναζήτησης λύσης. Η συνάρτηση υπολογισμού της καταλληλότητας κάθε ατόμου είναι:

$$fitness(\vec{x}) = f(\vec{x}) + \lambda(t) \left[\sum_{i=1}^{n} g_i^2(\vec{x}) + \sum_{j=1}^{p} \left| h_j(\vec{x}) \right| \right] \tag{2.21}$$

Όπου η συνάρτηση $\lambda(t)$ ανανεώνεται κάθε γενιά t με τον εξής τρόπο:

$$\lambda(t+1) = \begin{cases} \left(\frac{1}{\beta_1}\right) \cdot \lambda(t), & if\ case\ 1 \\ \beta_2 \cdot \lambda(t), & if\ case\ 2 \\ \lambda(t), & otherwise \end{cases} \quad (2.22)$$

Όπου οι περιπτώσεις 1 και 2 αναφέρονται στις περιπτώσεις όπου το πιο κατάλληλο άτομο των τελευταίων k γενιών ήταν πάντα (περίπτωση 1) ή δεν ήταν ποτέ (περίπτωση 2) εφικτή λύση (feasible). Επίσης $\beta_1\ \beta_2 > 1$, $\beta_1 > \beta_2$, $\beta_1 \neq \beta_2$. Με άλλα λόγια, ο συντελεστής ποινής $\lambda(t+1)$ για την γενιά $t+1$ μειώνεται αν όλα τα καλύτερα άτομα τις τελευταίες k γενιές ήταν εφικτές λύσεις, και αυξάνεται αν ήταν όλα μη εφικτές λύσεις. Εάν είναι κάποια άτομα εφικτές λύσεις και άλλα όχι τότε η ποινή δεν αλλάζει.

2.4.1.5 *Συν-εξελικτικές ποινές*

Ο Coello (Coello, 2000), πρότεινε την χρήση μιας συνάρτησης της μορφής:

$$fitness(\vec{x}) = f(\vec{x}) - (coef\ \times w_1 + viol\ \times\ w_2) \quad (2.23)$$

Όπου $f(\vec{x})$ είναι η τιμή της αντικειμενικής συνάρτησης για ένα άτομο, w_1 και w_2 είναι συντελεστές ποινής (ακέραιοι), $coef$ είναι το άθροισμα όλων των παραβιάσεων των περιορισμών (μόνο περιορισμοί ανισότητας είχαν θεωρηθεί):

$$coef = \sum_{i=1}^{b} g_i(\vec{x}) \ \ \forall g_i(\vec{x}) > 0 \quad (2.24)$$

$viol$ είναι ένας ακέραιος συντελεστής, ο οποίος θέτεται αρχικά ίσος με 0 και αυξάνεται κατά ένα για κάθε ένα περιορισμό του προβλήματος ο οποίος παραβιάζεται, ανεξάρτητα από το μέγεθος αυτής της παραβίασης.

Στην προσέγγιση του Coello, η ποινή χωρίζεται σε δύο μέρη, $coef$ και $viol$, έτσι ώστε ο EA να έχει αρκετή πληροφορία όχι μόνο για το πόσοι περιορισμοί παραβιάζονται, αλλά και για το κατά πόσο παραβιάζονται.

2.4.1.6 *Διαχωρισμένος γενετικός αλγόριθμος*

Οι Le Riche, Knopf-Lenoir και Haftka (Le Riche et al., 1995) σχεδίασαν έναν (διαχωρισμένο) γενετικό αλγόριθμο, ο οποίος χρησιμοποιεί δύο παραμέτρους ποινής, (για κάθε περιορισμό) αντί για μια. Αυτές οι δύο τιμές προσπαθούν να επιτύχουν την εξισορρόπηση μεταξύ υψηλών και μέτριων ποινών, διατηρώντας δύο υπό-πληθυσμούς από άτομα αντί για έναν. Ακόμη και όταν άτομα από τους δύο πληθυσμούς αναμιγνύονται, χωρίζονται με την έννοια της ικανοποίησης ενός συγκεκριμένου περιορισμού.

Η διαδικασία είναι η εξής: Ένας πληθυσμός μεγέθους $2 \times m$ δημιουργείται. Κάθε άτομο υπολογίζεται σύμφωνα με δύο συναρτήσεις ποινής, από τις οποίες η μία λειτουργεί με βαριές ποινές και η άλλη με πιο ήπιες. Δύο λίστες με κατάταξη δημιουργούνται και συγχωνεύονται. Μόνο *m* άτομα επιλέγονται από την νέα λίστα για να εφαρμοστούν οι γενετικοί συντελεστές. Τα καλύτερα άτομα από τις δύο αρχικές λίστες (πριν την συγχώνευση) επιλέγονται να γίνουν γονείς για την επόμενη γενιά. Αυτό αποσκοπεί στο να συνδυαστούν άτομα feasible και infeasible, και να βοηθήσει τον αλγόριθμο να μένει μακριά από τοπικά ελάχιστα.

Ακόμη μια σημαντική διαφορά αυτής της προσέγγισης σε σχέση με τους παραδοσιακούς γενετικούς αλγόριθμους είναι ότι εάν οι δύο ποινές έχουν την ίδια τιμή, οι *m* απόγονοι παράγονται μετά από εφαρμογή γενετικών συντελεστών και αναμιγνύονται με τους *m* γονείς τους. Τότε τα καλύτερα *m* άτομα από αυτή την λίστα επιλέγονται για περεταίρω επεξεργασία. Αυτή η στρατηγική αντικατάστασης (η οποία ονομάζεται “super elitism” από τους (Le Riche et al., 1995)), βρέθηκε από τις εξελικτικές στρατηγικές (Schwefel, 1981) και επιτρέπει να εξισορροπείται η επιρροή από τους δύο συντελεστές ποινής οι οποίοι χρησιμοποιούνται.

2.4.1.7 *Θανατική ποινή (Death penalty)*

Η απόρριψη των μη εφικτών λύσεων (επίσης καλείται “death penalty”) είναι μάλλον η πιο εύκολη μέθοδος στον χειρισμό περιορισμών και είναι επίσης υπολογιστικά αποτελεσματική, επειδή όταν μια συγκεκριμένη τιμή παραβιάζει έναν περιορισμό, της δίνει τιμή fitness ίση με 0. Ως εκ τούτου, δεν χρειάζονται άλλοι υπολογισμοί ώστε να υπολογιστεί ο βαθμός της παραβίασης μιας τέτοιας λύσης.

2.4.2 **Ειδική αναπαράσταση και συντελεστές (Special representations and operators)**

Μερικοί μελετητές αποφάσισαν να αναπτύξουν μια μέθοδο ειδικής αναπαράστασης ώστε να αντιμετωπίσουν ένα συγκεκριμένο πρόβλημα για το οποίο το γενικό πλαίσιο αναπαράστασης (πχ δυαδική αναπαράσταση η οποία χρησιμοποιείται στους κλασσικούς γενετικούς αλγόριθμους) μπορεί να μην είναι κατάλληλο. Εξαιτίας της αλλαγής της αναπαράστασης, είναι απαραίτητο να σχεδιαστούν ειδικοί συντελεστές οι οποίοι θα δουλεύουν όπως και οι παραδοσιακοί συντελεστές οι οποίοι χρησιμοποιούνται στη δυαδική αναπαράσταση.

2.4.2.1 *Davis' applications*

Το *Handbook of Genetic Algorithms,* (Davis, 1991) περιλαμβάνει αρκετά παραδείγματα από ΕΑ οι οποίοι χρησιμοποιούν ειδική αναπαράσταση και συντελεστές, ώστε να επιλύσουν πολύπλοκα προβλήματα του πραγματικού κόσμου. Για παράδειγμα, ο Davidor χρησιμοποίησε διάφορου μεγέθους γενετικούς αλγόριθμους ώστε να δημιουργήσει ρομποτικές τροχιές, και καθόρισε έναν ειδικό συντελεστής διασταύρωσης ο οποίος ονομαζόταν *analogous crossover* (Davidor, 1989)

2.4.2.2 *Τυχαία κλειδιά (Random keys)*

Ο Bean πρότεινε μια ειδική αναπαράσταση "random keys encoding" η οποία (σε αντίθεση με τις προσεγγίσεις στο βιβλίο του Davis) χρησιμοποιείται ώστε να εξαλείψει την ανάγκη για ειδικούς συντελεστές διασταύρωσης και μετάλλαξης, σε συγκεκριμένα προβλήματα αλληλουχίας και βελτιστοποίησης (πχ job shop scheduling, parallel machine tool scheduling and facility layout).

Η ιδέα είναι να κωδικοποιείται μια λύση με τυχαίους αριθμούς. Τέτοιοι τυχαίοι αριθμοί χρησιμοποιούνται σαν κλειδιά για κωδικοποίηση της λύσης. Για παράδειγμα για να αναπαριστήσουμε ένα πρόβλημα δρομολόγησης n-εργασιών σε m-μηχανές χρησιμοποιώντας αυτή την προσέγγιση, κάθε αλληλουχία είναι ένας πραγματικός αριθμός στον οποίο το ακέραιο μέρος ανήκει στη σειρά {1,2,…,m}, όπου το δεκαδικό μέρος δημιουργείται τυχαία μέσα στο διάστημα (0,1). Το ακέραιο μέρος του αριθμού ερμηνεύεται ως η ανάθεση της μηχανής για εκείνη την εργασία, όπου το ταξινομημένο κλασματικό μέρος παρέχει την ακολουθία εργασίας σε κάθε μηχανή (Bean and Norman, 1993).

2.4.2.3 *GENOCOP*

Η μέθοδος GENOCOP (GEnetic algorithm for Numerical Optimization for COnstrained Problems) αναπτύχθηκε από τον Michalewicz (Michalewicz, 1992). Αυτό που κάνει είναι να εξαλείφει περιορισμούς ισότητας μαζί με ίσο αριθμό μεταβλητών του προβλήματος. Αυτό αφαιρεί το κομμάτι του χώρου στον οποίο γίνεται αναζήτηση και απλοποιεί το πρόβλημα για τον ΕΑ. Οι περιορισμοί που παραμένουν είναι γραμμικές ανισότητες, οι οποίες σχηματίζουν ένα κυρτό σύνολο το οποίο θα πρέπει να εξερευνηθεί από τον ΕΑ. Ο GENOCOP προσπαθεί να εντοπίσει μία εφικτή λύση, παίρνοντας δείγμα από την εφικτή περιοχή λύσεων (feasible region). Αν δεν το πετύχει αυτό μετά από έναν συγκεκριμένο αριθμό δοκιμών, ο χρήστης ζητείται να παρέχει ένα τέτοιο σημείο εκκίνησης. Ο αρχικός πληθυσμός θα αποτελείται τότε από πανομοιότυπα αντίγραφα αυτού του σημείου εκκίνησης. Οι γενετικοί συντελεστές εκτελούν γραμμικούς συνδυασμούς από άτομα ώστε να εξασφαλίσουν ότι οι απόγονοι θα είναι και αυτοί εφικτές λύσεις.

2.4.2.4 *Συνέπεια περιορισμών (Constraint consistency)*

Ο Kowalczyk (Kowalczyk, 1997) πρότεινε την χρήση συνέπειας περιορισμών, ώστε να μειώνεται το πεδίο αναζήτησης, αποτρέποντας μεταβλητές οι οποίες δεν είναι συνεπής με τους περιορισμούς. Για παράδειγμα εξασφαλίζοντας ότι οι μεταβλητές παράγουν μόνο εφικτές λύσεις.

Ο Kowalczyk χρησιμοποιούσε αναπαράσταση πραγματικών αριθμών και καθόρισε ειδικούς γενετικούς συντελεστές και μία ειδική διαδικασία αρχικοποίησης η οποία ενσωμάτωνε την ιδέα της συνέπειας περιορισμών (constraint consistency). Επίσης έδειξε πως η προσέγγισή του μπορεί να χρησιμοποιηθεί σε συνδυασμό με οποιαδήποτε άλλη CHT, και ήταν ενήμερος πως σε πολλές περιπτώσεις, μερικώς εφικτές (partially feasible) λύσεις είναι προτιμότερες, καθώς μπορούν να κατευθύνουν την αναζήτησης σε καταλληλότερο δρόμο, ή επειδή είναι πολύ πιο εύκολο να βρεθούν.

2.4.2.5 Βρίσκοντας το όριο της εφικτής περιοχής (Locating the boundary of the feasible region)

Η κύρια ιδέα πίσω από αυτή την τεχνική είναι να ψάχνει λύσεις κοντά στο όρια της εφικτής περιοχής. Αφού σε πολλά μη-γραμμικά προβλήματα βελτιστοποίησης, τουλάχιστον κάποιοι περιορισμοί είναι ενεργοί στο ολικό βέλτιστο, είναι πλήρως τεκμηριωμένο να εστιάσει η έρευνα στα όρια μεταξύ της εφικτής και μη-εφικτής περιοχής.

2.4.2.6 *Αποκωδικοποιητές (Decoders)*

Σε αυτή την περίπτωση, ένα χρωμόσωμά «δίνει οδηγίες» στο πώς να χτιστεί μια εφικτή λύση. Κάθε αποκωδικοποιητής επιβάλλει μια σχέση T μεταξύ μιας εφικτής λύσης και μια από-κωδικοποιημένης λύσης (Dasgupta and Michalewicz, 1999).

2.4.3 **Αλγόριθμοι επιδιόρθωσης (Repair algorithms)**

Σε πολλά προβλήματα συνδυαστικής (για παράδειγμα το πρόβλημα του περιοδεύοντος πωλητή, το πρόβλημα του σακιδίου κλπ.) είναι σχετικά εύκολο να «επιδιορθωθεί» μία μη εφικτή λύση, δηλαδή να μετατρέψουμε ένα infeasible άτομο σε feasible. Μια τέτοια μέθοδος επιδιόρθωσης μπορεί να χρησιμοποιηθεί για υπολογισμό μόνο, ή μπορεί και να αντικαθιστά, με κάποια πιθανότητα, τα άτομα στον πληθυσμό.

Οι Liepins και Vose (Liepins and Vose, 1990) έδειξαν, μέσα από πρακτικές δοκιμές σε διάφορα προβλήματα συνδυαστικής, ότι ένας αλγόριθμος «διορθωτής» είναι ικανός να υπερβεί άλλες προσεγγίσεις και σε ταχύτητα και σε απόδοση.

2.4.4 **Διαχωρισμός αντικειμενικής συνάρτηση και περιορισμών (Separation of objectives and constraints)**

Υπάρχουν αρκετές προσεγγίσεις οι οποίες χειρίζονται περιορισμούς και αντικειμενική συνάρτηση ξεχωριστά. Μερικές από αυτές είναι:

2.4.4.1 *Co-evolution*

Ο Paredis πρότεινε μια τεχνική βασισμένη στο συν-εξελικτικό μοντέλο στο οποίο υπάρχουν δύο πληθυσμοί. Ο πρώτος αφορά τους περιορισμούς, και ο δεύτερος αφορά πιθανές λύσεις οι οποίες λύνουν το πρόβλημα. Λύσεις και περιορισμοί έχουν συναντώνται, και άτομα τα οποία ανήκουν και στους δύο πληθυσμούς υπολογίζονται.

2.4.4.2 *Ανωτερότητα εφικτών σημείων*

Οι Powell και Skolnick (Powell and Skolnick, 1993) σκέφτηκαν έναν ευρετικό κανόνα για να επεξεργάζονται μη εφικτές λύσεις: Υπολογισμοί των εφικτών λύσεων κατατάσσονται στο διάστημα (–∞,1) ενώ μη εφικτές λύσεις στο διάστημα (1,∞). Η συνάρτηση υπολογισμού της fitness είναι:

$$fitness(\vec{x}) = \begin{cases} f(\vec{x}), & if\ feasible \\ 1 + r \cdot \left(\sum_{i=1}^{n} g_i(\vec{x}) + \sum_{j=1}^{p} h_j(\vec{x})\right), & otherwise \end{cases} \quad (2.25)$$

Η $f(\vec{x})$ κατατάσσεται στο διάστημα (–∞,1), ενώ οι $g_i(\vec{x})$ και $h_j(\vec{x})$ στο διάστημα (1,∞), και η r είναι σταθερά. Ενδιαφέρον παρουσιάζει πως σε αυτή την μέθοδο, η αντικειμενική συνάρτηση και το μέγεθος της παραβίασης των περιορισμών δεν συνδυάζονται όταν ένα άτομο είναι μη εφικτό (όπως γίνεται στις συναρτήσεις ποινής).

2.4.4.3 *Μνήμη συμπεριφοράς (Behavioural memory)*

Οι Schoenauer και Xanthakis πρότειναν μια επέκταση της τεχνικής behavioural memory, η οποία προτάθηκε αρχικά για βελτιστοποίηση σε προβλήματα χωρίς περιορισμούς (Schoenauer and Xanthakis, 1993). Η κύρια ιδέα αυτής της προσέγγισης είναι ότι οι περιορισμοί χειρίζονται σε μια συγκεκριμένη σειρά. Ο αλγόριθμος είναι ο παρακάτω:

- Εκκίνηση με τυχαίο πληθυσμό ατόμων
- Θέτει j=1 (j είναι ο μετρητής περιορισμών)
- Εξελίσσει τον πληθυσμό ώστε να ελαχιστοποιήσει την παράβαση του j-οστού περιορισμού, μέχρι ένα συγκεκριμένο ποσοστό του πληθυσμού (ονομάζεται κάτω όριο Φ) να είναι εφικτό για τον περιορισμό. Σε αυτή την περίπτωση:

  $$fitness(\vec{x}) = M - g_j(\vec{x}) \quad (2.26)$$

  Όπου M είναι ένας μεγάλος θετικός αριθμός ο οποίος δυναμικά προσαρμόζεται σε κάθε γενιά.

- j = j + 1
- Ο τρέχον πληθυσμός είναι το σημείο εκκίνησης για την επόμενη φάση της εξέλιξης, ελαχιστοποιώντας τον j-οστό περιορισμό.

  $$fitness(\vec{x}) = M - g_j(\vec{x}) \quad (2.27)$$

Σε αυτή τη φάση, σημεία τα οποία δεν ικανοποιούν τουλάχιστον έναν από τους 1^o, 2^o ,….$(j-1)^o$ περιορισμό αποκλείονται από τον πληθυσμό. Η κατάσταση για να σταματήσει αυτό το στάδιο του αλγόριθμου είναι ξανά η ικανοποίηση του j-οστού αλλάζοντας το ποσοστό κάτω ορίου Φ του πληθυσμού.

- Αν $j < m$, επανέλαβε τα τελευταία δύο βήματα, αλλιώς βελτιστοποίησε την αντικειμενική συνάρτηση απορρίπτοντας μη εφικτά άτομα.

Η ιδέα αυτής της τεχνικής είναι να ικανοποιεί διαδοχικά (έναν έναν) τους περιορισμούς. Αυτή η προσέγγιση είναι παρόμοια με αυτή που καλείται «λεξικογραφική ταξινόμηση» (lexicographic ordering), η οποία εφαρμόζεται στην βελτιστοποίηση πολλών στόχων.

2.4.5 **Υβριδικές μέθοδοι (Hybrid methods)**

Σε αυτή την κατηγορία συμπεριλαμβάνονται προσεγγίσεις οι οποίες συνδυάζονται με κάποια άλλη τεχνική, συνήθως αριθμητικής βελτιστοποίησης, ώστε να χειριστούν τους περιορισμούς σε έναν ΕΑ. Θα αναφέρουμε επιγραμματικά κάποιες από αυτές:

- Langrangian multipliers
- Constrained optimization by random evolution
- Fuzzy logic
- Immune system
- Cultural algorithms

3 Μαθηματικό μοντέλο των αλγορίθμων που μελετήθηκαν

3.1 Σύνοψη κεφαλαίου

Στα πλαίσια αυτής της διπλωματικής μοντελοποιήθηκε ένας αλγόριθμος εμπνευσμένος από την εργασία των Li, Yang και Nguyen "*A Self-Learning Particle Swarm Optimizer for Global Optimization Problems*", ή αλλιώς απλά SLPSO (Self-Learning Particle Swarm Optimization) (Li et al., 2012). Ο αλγόριθμος αυτός είναι μια παραλλαγή του κλασσικού PSO, προσθέτοντας σε αυτόν τη δυνατότητα για διαφορετικούς τρόπους εκμάθησης για το κάθε άτομο-σωματίδιο. Αναπτύχθηκε λοιπόν ο SLPSO, και προστέθηκε σε αυτόν ένας στοχαστικός μηχανισμός νέφους απογόνων και επίσης ένας μηχανισμός αναδιάταξης του σμήνους. Η CHT που χρησιμοποιήσαμε, ώστε να αντιμετωπίσουμε προβλήματα με περιορισμούς είναι συνάρτηση ποινής. Τέλος, ο αλγόριθμος μοντελοποιήθκε σε λογισμικό MATLAB ®.

Ο SLPSO δοκιμάστηκε και συγκρίθηκε πάνω σε δοκιμαστικές συναρτήσεις (test functions), από τον διαγωνισμό του CEC 2010 (Commerce and Enterprise Computing) πάνω στο θέμα "*Competition on Constrained Real Parameter Optimization"* (Mallipeddi and Suganthan, 2010). Ο αλγόριθμος δοκιμάστηκε πάνω σε αυτές τις συναρτήσεις και αργότερα συγκρίθηκαν τα αποτελέσματα με αυτά των νικητών του διαγωνισμού, των Takahama και Sakai, με τον αλγόριθμο εDEag (ε constrained Differential Evolution with an Archive and Gradient-Based Mutation) (Takahama and Sakai, 2010).

Επιπλέον ο SLPSO δοκιμάστηκε πάνω σε δύο δοκιμαστικές συναρτήσεις από την μελέτη των Runarsson και Yao, και συγκεκριμένα από την μελέτη "Stochastic Ranking for Constrained Evolutionary Optimization", SRES (Runarsson and Yao, 2000).

Σε αυτό το κεφάλαιο θα αναλύσουμε το μαθηματικό μοντέλο του SLPSO με τον μηχανισμό του νέφους απογόνων, τον αλγόριθμο εDEag των Takahama και Sakai και επίσης την μέθοδο των Runarsson και Yao, SRES.

3.2 Η μέθοδος Self-Learning Particle Swarm Optimization

3.2.1 **Μαθηματικό μοντέλο Particle Swarm Optimization**

Η μέθοδος SLPSO προέκυψε σαν εξέλιξη της μεθόδου PSO την οποία αναφέραμε στο πρώτο κεφάλαιο. Όπως είπαμε οι Kennedy και Eberhart (Kennedy and Eberhart, 1995) εισήγαγαν μια μέθοδο βελτιστοποίησης, η οποία προσπαθεί να μιμηθεί τα σμήνη ή κοπάδια των ζώων, και τους μηχανισμούς επικοινωνίας τους οποίους χρησιμοποιούνε αυτά. Η βασική ιδέα είναι ότι τα άτομα του πληθυσμού μπορούν να επωφεληθούν από τις προσωπικές γνώσεις αλλά και από τις γνώσεις του υπόλοιπου πληθυσμού.

Ο αλγόριθμος χρησιμοποιεί, όπως μας προϊδεάζει και το όνομα, ένα σμήνος σωματιδίων, το οποίο εξελίσσεται και ψάχνει για μια λύση. Θα μπορούσαμε να πούμε ότι τα σωματίδια κινούνται μέσα στο χώρο ψάχνοντας για λύση. Επίσης τα σωματίδια αποθηκεύουν την καλύτερη λύση που έχουν βρει μέχρι στιγμής. Μετά επικοινωνούν με τα υπόλοιπα σωματίδια και βρίσκουν ποιο από αυτά έχει ανακαλύψει την καλύτερη λύση σε όλο το σμήνος. Έπειτα, τα σωματίδια μετακινούνται χρησιμοποιώντας τις πληροφορίες που έχουν, για το δικό τους βέλτιστο και το ολικό βέλτιστο, και η διαδικασία επαναλαμβάνεται.

Τα σωματίδια χρησιμοποιούν ένα διάνυσμα θέσης $\vec{x}$, το οποίο δηλώνει την θέση τους μέσα στον χώρο και χρησιμοποιείται για να υπολογιστεί η αντικειμενική τους αξία. Επιπλέον τα σωματίδια χρησιμοποιούν και ένα διάνυσμα ταχύτητας $\vec{v}$, το οποίο αντιπροσωπεύει τη μετακίνηση του σωματιδίου στον χώρο. Το διάνυσμα ταχύτητας $\vec{v}$ χρησιμοποιείται ώστε να ανανεωθεί η θέση του σωματιδίου στην επόμενη επανάληψη. Τα διανύσματα θέσης και ταχύτητας των σωματιδίων ανανεώνονται σύμφωνα με τις επόμενες δύο εξισώσεις για κάθε σωματίδιο *j*:

$$v_j{}^{d\prime} = v_j{}^d + \varphi 1 \cdot rnd \cdot \left(x_{personal}{}^d - x_j{}^d\right) + \varphi 2 \cdot rnd \cdot \left(x_{global}{}^d - x_j{}^d\right) \qquad (3.1)$$

$$x_j{}^{d\prime} = x_j{}^d + v_j{}^{d\prime} \qquad (3.2)$$

Στις παραπάνω εξισώσεις τα $x_j{}^d$, $x_j{}^{d\prime}$ είναι η προηγούμενη και η τωρινή τιμή της διάστασης *d* το διανύσματος θέσης του σωματιδίου *j* αντίστοιχα, όπου v_j, $v_j{}'$ είναι η προηγούμενη και η τωρινή τιμή της διάστασης d του διανύσματος ταχύτητας του σωματιδίου j αντίστοιχα. ${x_{personal}}^d$ είναι η τιμή της διάστασης d για το διάνυσμα το οποίο πέτυχε την καλύτερη επίδοση για το συγκεκριμένο σωματίδιο j (προσωπικό βέλτιστο-personal best), ${x_{global}}^d$ είναι η τιμή της διάστασης *d* για το διάνυσμα το οποίο πέτυχε την καλύτερη επίδοση όλων των σωματιδίων ως τώρα. *φ1* και *φ2* είναι οι ρυθμοί εκμάθησης που αναφέρονται στις συνιστώσες cognition (επίγνωσης) και social (κοινωνικής συναναστροφής) αντίστοιχα. Ο συντελεστής *φ1* είναι πολλαπλασιαστής της επίδρασης που ασκεί στο νέο διάνυσμα ταχύτητας η προσωπική συνιστώσα, δηλαδή το διάνυσμα $\overrightarrow{x_{personal}}$ και ο συντελεστής *φ2* είναι αντίστοιχα αυτός της κοινωνικής συναναστροφής καθώς εκπροσωπεί την συνιστώσα του ολικού βέλτιστου, δηλαδή την επικοινωνία με το υπόλοιπο σμήνος μέσω του διανύσματος $\overrightarrow{x_{global}}$. Οι συντελεστές *rnd* είναι τυχαίοι αριθμοί στο διάστημα [0,1]. (Στις εξισώσεις δεν έχουμε διανύσματα καθώς μιλάμε για τις τιμές των συγκεκριμένων διαστάσεων. Θα μπορούσαν επίσης να ξανά γραφτούν οι εξισώσεις για διανύσματα.)

Αρχικά τα διανύσματα ταχυτήτων δημιουργούνται τυχαία στο διάστημα [$-V_{max}$,V_{max}], όπου V_{max} είναι η μέγιστη επιτρεπόμενη τιμή που μπορεί να λάβει η ταχύτητα ενός σωματιδίου. Ο λόγος για τον οποίο θέτουμε τέτοια όρια στις ταχύτητες είναι επειδή υψηλές ταχύτητες μπορούν να ελαττώσουν την απόδοση του PSO. Επίσης το διάνυσμα θέσης κάθε σωματιδίου αρχικοποιείται και αυτό τυχαία μέσα στο πεδίο ορισμού για κάθε διάσταση του προβλήματος.

Βάση της εξίσωσης (3.1) ο Kennedy (Kennedy, 1997) αναγνωρίζει 4 διαφορετικά μοντέλα PSO ανάλογα με τις τιμές των συντελεστών φ1 και φ2, οι οποίες είναι οι παρακάτω:

- Πλήρες μοντέλο, όταν φ1 και φ2 > 0
- Επίγνωση μόνο, όταν φ1>0 και φ2=0
- Κοινωνική συναναστροφή μόνο φ1=0 και φ2>0
- Selfless, όταν φ1=0 και φ2>0 και πάντα ${x_{global}}^d \neq {x_{personal}}^d$ το οποίο ισχύει για κάθε σωματίδιο. Δηλαδή κάθε σωματίδιο αναγνωρίζει το βέλτιστο των υπολοίπων

Αυτός είναι ο αλγόριθμος Particle Swarm Optimization όπως προτάθηκε αρχικά. Από τότε έχουν ακολουθήσει πάρα πολλές παραλλαγές. Μία από τις τελευταίες είναι ο SLPSO, τον οποίο θα αναλύσουμε τώρα.

3.2.2 **Μαθηματικό μοντέλο Self-Learning Particle Swarm Optimizer**

3.2.2.1 *Ο Αλγόριθμος Self-Learning Particle Swarm Optimizer*

Όπως αναφέραμε η μέθοδος Self-Learning Particle Swarm Optimization αναπτύχθηκε το 2012 από τους Li, Yang και Nguyen (Li et al., 2012). Η μέθοδος αυτή εστίασε στην μονοτονία του κλασσικού PSO όσον αφορά τον τρόπο εκμάθησης (ουσιαστικά ο τρόπος ανανέωσης της ταχύτητας), ο οποίος είναι ίδιος για όλα τα σωματίδια. Αυτό που προτάθηκε είναι η χρήση τεσσάρων διαφορετικών στρατηγικών εκμάθησης, οι οποίες θα επιλέγονται από το κάθε σωματίδιο σε κάθε επανάληψη, δίνοντας έτσι την ευκαιρία για ποικιλία στο σμήνος. Οι στρατηγικές αυτές είχαν αναφερθεί για πρώτη φορά στην μέθοδο Adaptive Particle Swarm Optimization (Zhan et al., 2009).

Εμπνευσμένο από την ιδέα του καταμερισμού της εργασίας, μπορούμε να αναθέσουμε διαφορετικούς ρόλους σε κάθε σωματίδιο. Για παράδειγμα κάποια σωματίδια θα προσπαθούν να συγκλίνουν στο ολικό βέλτιστο (global best), άλλα θα εκμεταλλεύονται το προσωπικό τους βέλτιστο, άλλα θα εξερευνούν νέες υποσχόμενες περιοχές, ενώ άλλες θα προσπαθούν να ξεφύγουν από τοπικά βέλτιστα. Έτσι προκύπτουν τέσσερις διαφορετικές στρατηγικές οι οποίες μπορούν να βοηθήσουν σε διαφορετικές καταστάσεις. Πρώτον, για προβλήματα στα οποία η αντικειμενική συνάρτηση έχει τοπολογία με κορυφές οι οποίες είναι μακριά η μία από την άλλη, για αποτελεσματική αναζήτηση στις τοπικές κορυφές, η καλύτερη στρατηγική εκμάθησης είναι για όλα τα σωματίδια να μαθαίνουν από τα γειτονικά βέλτιστα σωματίδια. Δεύτερον, σε περίπτωση που ένα σωματίδιο βρίσκεται σε πλαγιά, η καλύτερη στρατηγική είναι να μάθει από την προσωπική του βέλτιστη θέση. Τρίτον, για τοπίο στο οποίο έχουμε πολλά τοπικά ελάχιστα ομοιόμορφα κατανεμημένα, η εκμάθηση από γειτονικά σωματίδια είναι η καλύτερη στρατηγική καθώς έτσι τα σωματίδια εξερευνούν τον χώρο. Τέλος, για σωματίδια τα οποία έχουν συγκλίνει, ο μόνος τρόπος για να ξεφύγουν από το τοπικό ελάχιστο είναι η μετάλλαξη.

3.2.2.2 *Ανανέωση Ταχυτήτων στον SLPSO*

Σύμφωνα με τα παραπάνω, οι πληροφορίες εκμάθησης για τις διάφορες στρατηγικές στον SLPSO είναι η αποθηκευμένη βέλτιστη θέση *abest* του *gbest* σωματιδίου, το οποίο είναι το υπερ-σωματίδιο (super particle) που αναφέραμε. Επίσης χρησιμοποιείται η καλύτερη θέση του κάθε σωματιδίου *pbest*, η καλύτερα θέση *pbest* ενός τυχαίου σωματιδίου $pbest_{rand}$ που όμως έχει καλύτερη *pbest* από το σωματίδιο στο οποίο καλείται να εφαρμοστεί, και τέλος μια τυχαία θέση στην κοντινή περιοχή *prand*. Οι τέσσερις στρατηγικές παίζουν τους ρόλους της σύγκλισης (convergence), εκμετάλλευσης (exploitation), εξερεύνησης (exploration) , και «διαφυγή» από τοπικά βέλτιστα (jumping out of local optima). Έτσι λέμε πως στον SLPSO έχουμε τέσσερις διαφορετικούς διαχειριστές (operators) για την εκμάθηση των σωματιδίων. Για κάθε σωματίδιο k οι εξισώσεις εκμάθησης που αντιστοιχούν στους τέσσερις διαχειριστές είναι οι παρακάτω:

1. Διαχειριστής α: Εκμάθηση από την *pbest* θέση - Εκμετάλλευση (Exploitation)

$$v_k{}^{d\prime} = \omega \cdot\ v_k{}^d + \eta \cdot r_k{}^d \left(pbest_k{}^d -\ x_k{}^d\right) \tag{3.3}$$

2. Διαχειριστής β: Εκμάθηση από μια τυχαία κοντινή θέση – «Διαφυγή» (Jumping out)

$$x_k{}^{d\prime} =\ x_k{}^d + v_{avg}{}^d \cdot N(0,1) \tag{3.4}$$

3. Διαχειριστής γ: Εκμάθηση από την *pbest* θέση ενός τυχαίου σωματιδίου-Εξερεύνηση (Exploration)

$$v_k{}^{d\prime} = \omega \cdot\ v_k{}^d + \eta \cdot r_k{}^d \left(pbest_{rand}{}^d -\ x_k{}^d\right) \tag{3.5}$$

4. Διαχειριστής δ: Εκμάθηση από την *abest* θέση- Σύγκλιση (Convergence)

$$v_k{}^{d'} = \omega \cdot\ v_k{}^d + \eta \cdot r_k{}^d (abest^d -\ x_k{}^d) \tag{3.6}$$

Στις παραπάνω εξισώσεις $pbest_{rand}$ είναι η θέση *pbest* ενός τυχαίου σωματιδίου, που όμως είναι καλύτερη από την θέση του ίδιου σωματιδίου $pbest_k$. Η ταχύτητα $v_{avg}{}^d$ είναι η μέση ταχύτητα όλων των σωματιδίων για κάθε διάσταση *d*, και υπολογίζεται από την εξίσωση $v_{avg}{}^d$ = $\sum_{k=1}^{N} |v_k{}^d|/N,$ όπου *N* είναι το μέγεθος του σμήνους (πληθυσμού). *N(0,1)* είναι τυχαίος αριθμός ο οποίος παράγεται από κανονική κατανομή με μέση τιμή 0 και διακύμανση ίση με 1. Η θέση *abest* είναι η καλύτερη θέση την οποία έχει βρει ο SLPSO ως τώρα.

Παρακάτω στο Σχήμα (3.1) βλέπουμε τον ψευδο-κώδικα του μέρους του SLPSO που περιγράψαμε, ο οποίος περιλαμβάνει την ανανέωση των ταχυτήτων των σωματιδίων, σύμφωνα με τους τέσσερις διαχειριστές που αναφέραμε. Οι συγγραφείς ονομάζουν αυτό το μέρος Algorithm1 και επίσης βλέπουμε πως στον διαχειριστή γ ότι αν το τυχαίο σωματίδιο που επιλέχτηκε έχει χειρότερη αξία αντικειμενικής, τότε το σωματίδιο ανανεώνει την ταχύτητά του σύμφωνα με τον διαχειριστή α:

Algorithm 1 Update(operator i, particle k, fes)

if $i = a$ **then**
  Update the velocity and position of particle k using operator a and (2);
else if $i = b$ **then**
  Update the position of particle k using operator b;
  else if $i = c$ **then**
  Choose a random particle j that is not particle k;
  if $f(\vec{x}_{pbest_j}) < f(\vec{x}_{pbest_k})$ **then**
    Update the velocity and position of particle k using operator c and (2);
  else
    Update the velocity and position of particle j using operator c and (2);
    $k := j$;
  end if
else
  Update the velocity and position of particle k using operator d and (2);
end if
fes++; where fes is the current number of fitness evaluations.

Σχήμα 3.1: Ο Ψευδο-κώδικας εκμάθησης των σωματιδίων (Li et al., 2012)

Να διευκρινίσουμε σε αυτό το σημείο ότι η θέση *abest* είναι το αρχείο της θέσεις που πέτυχε το σωματίδιο *gbest*, και αργότερα συγκρίνεται το *abest* με το καινούργιο *gbest*, και αν το νέο *gbest* έχει καλύτερη τιμή του *abest*, του δίνει την τιμή του. Η θέση *abest* δεν χρησιμοποιεί κανέναν από τους τέσσερις διαχειριστές για να ανανεωθεί, αλλά ανανεώνεται με το κομμάτι Algorithm2 το οποίο θα εξηγήσουμε σε λίγο.

Η επιλογή του ποιος διαχειριστής εκμάθησης ταιριάζει σε κάθε σωματίδιο εξαρτάται από την τοπολογία της αντικειμενικής συνάρτησης και το που είναι τοποθετημένο το κάθε σωματίδιο εκείνη τη στιγμή. Όμως, συνήθως δεν μπορούμε να γνωρίζουμε τι μορφή έχει η τοπολογία της αντικειμενικής συνάρτησης. Για αυτό τον λόγο προσπαθούμε κάθε σωματίδιο να ανιχνεύει το τι μορφή έχει η αντικειμενική συνάρτηση στη γειτονιά του.

3.2.2.3 *Προσαρμοζόμενος Μηχανισμός Εκμάθησης*

Για να πετύχουμε την ανάθεση του κάθε σωματιδίου στον διαχειριστή που του ταιριάζει χρησιμοποιούμε έναν λόγο επιλογής (selection ratio) για κάθε σωματίδιο και διαχειριστή. Ο λόγος επιλογής κάθε διαχειριστή αρχικά θέτεται ίσος με ¼ για κάθε σωματίδιο και έπειτα ανανεώνεται σύμφωνα με την σχετική τους απόδοση. Η προσέγγιση αυτή δουλεύει υποθέτοντας πως ένας πετυχημένος διαχειριστής για τις τελευταίες επαναλήψεις, θα συνεχίσει να είναι πετυχημένος για μερικές ακόμη επαναλήψεις (για κάποιο σωματίδιο).

Για κάθε σωματίδιο, ένας από τους τέσσερις διαχειριστές επιλέγεται βάση τον λόγο επιλογής τους. Ο διαχειριστής ο οποίος θα έχει καλύτερη απόδοση, το οποίο υπολογίζεται με έναν συνδυασμό της παραγόμενης καταλληλότητας (offspring fitness), τον τρέχων λόγο επιλογής (current selection ratio), και τον πρώην λόγο διαλογής (previous selection ratio), θα αυξήσει και τον λόγο επιλογής του. Σταδιακά, οι καταλληλότεροι διαχειριστές θα επιλέγονται αυτόματα και θα ελέγχουν την εκμάθηση του κάθε σωματιδίου σε διάφορα στάδια της εξελικτικές διαδικασίας. Κατά την περίοδο ανανέωσης των λόγων επιλογής, ο συντελεστής προόδου και η επιβράβευσης του κάθε διαχειριστή υπολογίζονται σύμφωνα με τις παρακάτω εξισώσεις.

Η τιμή της προόδου για τον διαχειριστή *i* του σωματιδίου *k* κατά την επανάληψη *t* είναι $p_i{}^k(t)$ και υπολογίζεται σύμφωνα με την παρακάτω εξίσωση:

$$p_i{}^k(t)=\begin{cases} \left|f\left(\overrightarrow{x_k}(t)\right)-\overrightarrow{f(x_k}(t-1))\right|, if\ operator\ i\ is\ chosen\ by\ \overrightarrow{x_k}(t) \\ \quad and\ \overrightarrow{x_k}(t)\ is\ better\ than\ \overrightarrow{x_k}(t-1) \\ \quad 0,\ otherwise \end{cases} \quad (3.7)$$

Η τιμή της επιβράβευσης $r_i{}^k(t)$ έχει τρείς συνιστώσες, οι οποίες είναι η κανονικοποιημένη τιμή προόδου (progress rate $p_i{}^k$), ο συντελεστής επιτυχίας (success rate) και ο πρώην λόγος επιλογής (previous selection rate).

$$r_i{}^k(t) = \frac{p_i{}^k(t)}{\sum_{j=1}^{R} p_i{}^k(t)}\,\alpha + \frac{g_i{}^k(t)}{G_i{}^k(t)}\,(1-\alpha) + c_i{}^k s_i{}^k(t) \tag{3.8}$$

Όπου $g_i{}^k$ είναι ο μετρητής που καταγράφει τον αριθμό των επιτυχημένων εκμαθήσεων του σωματιδίου *k*, στις οποίες ο απόγονος είναι καταλληλότερος από το σωματίδιο *k* εφαρμόζοντας τον διαχειριστή *i*, από την τελευταία ανανέωση του λόγου επιλογής. $G_i{}^k$ είναι ο συνολικός αριθμός επαναλήψεων όπου έχει επιλεγεί ο διαχειριστής *i* από το σωματίδιο *k*, από την τελευταία ανανέωση του λόγου επιλογής. $\frac{g_i{}^k(t)}{G_i{}^k(t)}$ είναι ο λόγος επιτυχίας του διαχειριστή *i* για το σωματίδιο *k*. α είναι ένας τυχαίος συντελεστής βαρύτητας μεταξύ 0.0 και 1.0. *R* είναι ο αριθμός των διαχειριστών (δηλαδή 4), $c_i{}^k$ είναι ο συντελεστής ποινής για τον διαχειριστή *i* του σωματιδίου *k*, ο οποίος καθορίζεται ως εξής:

$$c_i{}^k = \begin{cases} 0.9, & if\ g_i{}^k = 0\ and\ s_i{}^k(t) = max_{j=1}{}^R(s_j{}^k(t)) \\ & 1, otherwise \end{cases} \tag{3.9}$$

Και $s_i{}^k(t)$ είναι ο λόγος επιλογής του διαχειριστή *i* για το σωματίδιο *k* την συγκεκριμένη επανάληψη *t*. Αν κανένας από τους διαχειριστές δεν έχει βελτιώσει το σωματίδιο *k* από την τελευταία ανανέωση του λόγου επιλογής, τότε $\sum_{j=1}^{R} p_i{}^k(t)$ είναι ίσο με 0. Σε αυτή την περίπτωση, μόνο η τρίτη συνιστώσα θα $c_i{}^k s_i{}^k(t)$ αποδοθεί στο $r_i{}^k(t)$.

Τέλος, ο λόγος επιλογής του διαχειριστή *i* για το σωματίδιο *k* για την επόμενη επανάληψη *t+1* ανανεώνεται σύμφωνα με την παρακάτω εξίσωση:

$$s_i{}^k(t+1) = \frac{r_i{}^k(t)}{\sum_{j=1}^{R} r_i{}^k(t)}(1 - R * \gamma) + \gamma \tag{3.10}$$

Όπου γ είναι ο ελάχιστος λόγος επιλογής για κάθε διαχειριστή, ο οποίος θέτεται συνήθως ίσος με 0.001.

Σύμφωνα με τους παραπάνω ορισμούς, γνωρίζουμε ότι πάντα θα υπάρχει ένας διαχειριστής ο οποίος έχει τον υψηλότερο λόγο επιλογής για κάθε σωματίδιο. Αυτός ο διαχειριστής θα πρέπει να είναι ο καλύτερος σε σχέση με τους άλλους διαχειριστές εκείνη τη χρονική στιγμή. Όμως, όταν ένα σωματίδιο συγκλίνει ή κινείται προς μια νέα περιοχή της οποίας η ιδιότητα διαφέρει με αυτήν της προηγούμενης, τότε ο καλύτερος διαχειριστής δεν ωφελεί πλέον το σωματίδιο. Όταν συμβεί αυτό, τότε σύμφωνα με τον μηχανισμό ποινής (3.9), ο λόγος επιλογής του διαχειριστή θα μειωθεί, ενώ ο λόγος επιλογής των υπόλοιπων διαχειριστών θα αυξηθεί. Έτσι, ένας νέος κατάλληλος διαχειριστής θα ανατεθεί σταδιακά στο σωματίδιο, και ο παλιός διαχειριστής θα χάσει την κυριαρχία του σε αυτό το σωματίδιο.

Επίσης, πρέπει να αναφερθεί ότι οι λόγοι επιλογής των τεσσάρων διαχειριστών ανανεώνονται ταυτόχρονα και όχι σε κάθε επανάληψη. Αντί να μετράμε τον αριθμό των διαδοχικών επαναλήψεων, για την μεταβλητή U_f η οποία καλείται συχνότητα ανανέωσης (update frequency), απλά καταγράφουμε τον αριθμό τον διαδοχικών ανεπιτυχών επαναλήψεων εκμάθησης m_k για κάθε σωματίδιο k. Εάν κάποιο σωματίδιο βελτιωθεί από κάποιον διαχειριστή πριν ο μετρητής m_k να φτάσει την μέγιστη τιμή του U_f, τότε m_k θα επαναφερθεί στην τιμή 0. Αυτή η μέθοδος μειώνει τον κίνδυνο να τιμωρήσουμε τον καλύτερο διαχειριστή λόγω της προσωρινά κακής του απόδοσης. Αφού οι λόγοι επιλογής των τεσσάρων διαχειριστών ανανεωθούν, όλες οι πληροφορίες, εκτός του λόγου επιλογής του κάθε διαχειριστή, θέτονται ίσες με μηδέν.

3.2.2.4 *Ανανέωση πληροφοριών για την θέση abest*

Στους περισσότερους αλγορίθμους που χειρίζονται πληθυσμούς, από την στιγμή που τα άτομα ανανεωθούν ή αντικατασταθούν, όλες οι πληροφορίες σε όλες τις διαστάσεις αντικαθίστανται από καινούργιες. Αυτός ο μηχανισμός έχει το μειονέκτημα ότι μπορούν να χαθούν χρήσιμες πληροφορίες. Για παράδειγμα ας σκεφτούμε το παράδειγμα ένα άτομο να έχει χρήσιμες πληροφορίες σε μια διάσταση. Αυτή η πληροφορία θα χανόταν λόγω της συνολικής κακής απόδοσης, η οποία θα οφειλόταν στις άχρηστες πληροφορίες στις υπόλοιπες διαστάσεις.

Για να το αποφύγουμε αυτό, ο SLPSO έχει έναν μηχανισμό να ελέγχονται διαστάσεις σωματιδίων και ποιες από αυτές είναι χρήσιμες στο σωματίδιο *abest*, το super-particle, πριν αυτές ανανεωθούν. Γενικά, είναι δύσκολο να εφαρμόσουμε την ιδέα αυτή καθώς αν ελέγξουμε όλες τις διαστάσεις των σωματιδίων, θα έχουμε πολύ υψηλό υπολογιστικό κόστος για αυτό. Για να μειώσουμε αυτό το κόστος χρησιμοποιείται μια πιθανότητα εκμάθησης (learning probability) P_l για κάθε διάσταση στη θέση *abest* να μάθει από τα βελτιωμένα σωματίδια. Υπάρχουν δύο πλεονεκτήματα στην εισαγωγή της πιθανότητας εκμάθησης: Πρώτον, ο αλγόριθμος θα κερδίσει πολύ υπολογιστικό κόστος, και δεύτερον, ο αλγόριθμος θα μειώσει την πιθανότητα να μάθει πιθανόν άχρηστες πληροφορίες για την θέση *abest*, έστω και αν αυτό μειώνει επίσης την πιθανότητα να μάθε χρήσιμες πληροφορίες.

Η παραπάνω διαδικασία που περιγράψαμε φαίνεται στο Σχήμα (3.2) στον ψευδο-κώδικα του Algorithm2, ο οποίος είναι αυτός ο οποίος ψάχνει στις διαστάσεις των βελτιωμένων σωματιδίων, με σκοπό να βελτιωθεί η θέση *abest*:

Algorithm 2 UpdateAbest(particle k, fes)

for each dimension d of $abest$ **do**
  if $rand() < P_l^k$ **then**
    $\vec{x}_{t\_abest} := \vec{x}_{abest}$;
    $\vec{x}_{t\_abest}[d] := \vec{x}_k[d]$;
    Evaluate $\vec{x}_{t\_abest}$;
    fes++;
    if $f(\vec{x}_{t\_abest}) < f(\vec{x}_{abest})$ **then**
      $\vec{x}_{abest}[d] := \vec{x}_{t\_abest}[d]$;
    end if
  end if
end for

Σχήμα 3.2: Ψευδο-κώδικας βελτίωσης της θέσης *abest* (Li et al., 2012)

Η εκμάθηση των σωματιδίων από την θέση *abest* έχει σκοπό την επιτάχυνση της σύγκλισης. Για τα σωματίδια τα οποία βρίσκονται κοντά στην θέση *abest*, η έλξη που δημιουργεί η θέση *abest* είναι πολύ ισχυρή, ώστε να έχουν ευκαιρία οι υπόλοιποι διαχειριστές. Για να εκμεταλλευτούμε πλήρως τον μηχανισμό εκμάθησης, είναι ανάγκη να ελέγξουμε τον αριθμό των σωματιδίων τα οποία μαθαίνουν από την θέση *abest*. Υπάρχουν δύο λόγοι για αυτό:

- Πρώτον, ο βέλτιστος αριθμός σωματιδίων τα οποία μαθαίνουν από την θέση *abest*, εξαρτάται από τις ιδιότητες του προβλήματος που λύνουμε. Σε κάποια προβλήματα πρέπει να επιτρέπουμε σε περισσότερα σωματίδια να χρησιμοποιούν αυτόν τον μηχανισμό εκμάθησης ώστε να έχουμε γρήγορη σύγκλιση, και σε άλλα λιγότερα.
- Δεύτερον, δεν είναι δίκαιο για τους υπόλοιπους διαχειριστές να ανταγωνίζονται τον διαχειριστή σύγκλισης, καθώς όλοι τους συνεισφέρουν στον διαχειριστή σύγκλισης. Όταν ένα σωματίδιο βελτιώνεται μέσω οποιουδήποτε από τους τρεις άλλους διαχειριστές, ο διαχειριστής σύγκλισης επωφελείται είτε άμεσα είτε έμμεσα. Άμεσα,

όταν το βελτιωμένο σωματίδιο γίνει η θέση *abest,* και έμμεσα όταν χρήσιμες πληροφορίες εξάγονται από το σωματίδιο.

3.2.2.5 *Ελέγχοντας τα σωματίδια τα οποία μαθαίνουν από την θέση abest*

Σύμφωνα με τις παραπάνω πληροφορίες, πρέπει να ελέγχουμε τον αριθμό των σωματιδίων που χρησιμοποιούν τον διαχειριστή σύγκλισης. Παρόλα αυτά, είναι δύσκολο να γνωρίζουμε το ποια σωματίδια είναι κατάλληλα να χρησιμοποιήσουν τον διαχειριστή σύγκλισης. Για να ξεπεράσουμε το πρόβλημα αυτό, θέτουμε έναν συγκεκριμένο αριθμό σωματιδίων τα οποία θα χρησιμοποιούν τον διαχειριστή σύγκλισης σε κάθε επανάληψη. Για να εφαρμόσουμε αυτή την ιδέα, πρέπει να ανανεώσουμε κάποιες πληροφορίες των σωματιδίων, τα οποία χρησιμοποιούν και μετά δεν χρησιμοποιούν ή το αντίστροφο, τον διαχειριστή σύγκλισης, μεταξύ δύο διαδοχικών επαναλήψεων. Οι πληροφορίες οι οποίες θα ανανεωθούν σε αυτή την περίπτωση είναι η τιμή προόδου, η τιμή επιβράβευσης, ο λόγος επιτυχίας και ο λόγος επιλογής.

Αν η προαναφερθείσα αλλαγή ανάμεσα στις δύο καταστάσεις συμβεί, τότε έχουμε δύο περιπτώσεις για την ανανέωση των πληροφοριών. Η πρώτη περίπτωση είναι το σωματίδιο να χρησιμοποίησε τον διαχειριστή σύγκλισης στην προηγούμενη επανάληψη και να μην τον χρησιμοποιεί στην τρέχουσα επανάληψη, και η δεύτερη περίπτωση είναι να μην χρησιμοποίησε τον διαχειριστή σύγκλισης στην προηγούμενη επανάληψη, αλλά να τον χρησιμοποιεί στην τρέχουσα. Στην πρώτη περίπτωση, θα πρέπει να αφαιρέσουμε την πηγή εκμάθησης της θέσης *abest* και να κανονικοποιήσουμε τους λόγους επιλογής των υπόλοιπων τριών διαχειριστών. Στην δεύτερη περίπτωση, όλες οι πληροφορίες αρχικοποιούνται: Οι τιμές προόδου, επιβράβευσης και οι λόγοι επιτυχίας θέτονται όλα ίσα με μηδέν, και οι λόγοι επιλογής των τεσσάρων διαχειριστών θέτονται ίσοι με ¼. Η διαδικασία ανανέωσης που αναφέραμε φαίνεται στο Σχήμα (3.3).

Algorithm 3 UpdateLearningOpt(particle k)

if $CF_k != true$ && $PF_k = true$ **then**
  $sum := \sum_{j=1}^{3} s_j^k$;
  for $j := 1$ to 3 **do**
    $s_j^k := s_j^k / sum$;
  end for
  $s_4^k := 0$;
end if
if $CF_k = true$ && $PF_k != true$ **then**
  for $j := 1$ to 4 **do**
    $p_j^k := 0; g_j^k := 0; G_j^k := 0; s_j^k := 1/4$
  end for
end if

where CF_k and PF_k are used to record whether particle k uses the convergence operator or not at the current and previous iteration, respectively

Σχήμα 3.3: Ψεοδ-κώδικας ελέγχου των σωματιδίων που χρησιμοποιούν τον διαχειριστή σύγκλισης (Li et al., 2012)

3.2.2.6 *Σκελετός του SLPSO*

Ο δεύτερος τρόπος ελέγχου των σωματιδίων που χρησιμοποιούν τον διαχειριστή σύγκλισης είναι η μεταβλητή *M,* η οποία είναι ο μέγιστος αριθμός σωματιδίων τα οποία μπορούν να χρησιμοποιήσουν τον διαχειριστή σύγκλισης. Η μεταβλητή αυτή αλλάζει με το πέρας της εξελικτικής διαδικασίας και ακολουθεί την παρακάτω εξίσωση:

$$M(fes) = \mathrm{N}\,(1 - \exp\left(-100\left(\frac{fes}{Tfes}\right)^3\right)) \tag{3.11}$$

Όπου fes είναι ο αριθμός των υπολογισμών της αντικειμενικής συνάρτησης ως τώρα, και όπου $Tfes$ είναι ο συνολικός αριθμός των υπολογισμών της αντικειμενικής συνάρτησης που επιτρέπονται για το τρέξιμο του SLPSO.

Μερικές ακόμη παράμετροι που χρησιμοποιεί ο SLPSO είναι οι παρακάτω:

Η παράμετρος V_{max} η οποία χρησιμοποιείται για να ελέγξει τις μέγιστες ταχύτητες τις οποίες αναπτύσσουν τα σωματίδια και η οποία ορίζεται ως το μισό του εύρους αναζήτησης της συγκεκριμένης διάστασης. Όταν έχουμε κάποιο σωματίδιο εκτός του εύρους αναζήτησης, τότε πριν ανανεώσουμε την θέση του, πρώτα ανατρέχουμε στην τιμή της θέσης $x^d(t-1)$ της προηγούμενης επανάληψης και έπειτα υπολογίζουμε μια προσωρινή τιμή x_t με τον Algorithm1 και τελικά, ανανεώνουμε την θέση $x^d(t)$ ως εξής:

$$x^d(t) = \begin{cases} R\left({X_{min}}^d, x^d(t-1)\right), & if\ x_t < {X_{min}}^d \\ R\left(x^d(t-1), {X_{max}}^d\right), & if\ x_t > {X_{max}}^d \\ x_t, & else \end{cases} \tag{3.12}$$

Όπου $R(a, b)$ επιστρέφει έναν ομοιόμορφα κατανεμημένο αριθμό στο διάστημα $[a, b]$ και $[{X_{min}}^d, {X_{max}}^d]$ είναι το εύρος αναζήτησης στην d-οστή διάσταση του προβλήματος.

Η μεταβλητή η οποία καλείται συχνότητα ανανέωσης (update frequency) ορίζεται ως εξής:

$$U_f{}^k = \max(10 * \exp(-\left(1.6 \cdot \frac{k}{N}\right)^4, 1) \tag{3.13}$$

Όπου N είναι το μέγεθος του σμήνους, $U_f{}^k$ είναι ο συχνότητα ανανέωσης του σωματιδίου k. Με βάση την παραπάνω εξίσωση, οι τιμές της συχνότητας ανανέωσης για όλα τα σωματίδια κυμαίνεται από 1 ως 10.

Για την πιθανότητα εκμάθησης P_l χρησιμοποιούμε ίδια μέθοδο με την ανανέωση συχνότητας, όπου οι πιθανότητες εκμάθησης κυμαίνονται μεταξύ 0.05 και 1.00. Ο τρόπος υπολογισμού είναι ο εξής:

$$P_l{}^k = \max(1 - \exp(-\left(1.6 \cdot \frac{k}{N}\right)^4, 0.05) \tag{3.14}$$

Για να μειωθεί ο κίνδυνος να χρησιμοποιηθούν ακατάλληλες τιμές των U_f και P_l για κάποιο σωματίδιο, παράγεται ένας δείκτης μετάθεσης των αριθμών των σωματιδίων σε κάθε επανάληψη και μετά ανανεώνονται οι τιμές των U_f και P_l για κάθε σωματίδιο.

Ο συντελεστής βαρύτητας (inertia weight) ο οποίος χρησιμοποιείται στις εξισώσεις ανανέωσης ταχυτήτων υπολογίζεται σύμφωνα με την παρακάτω εξίσωση, και όπως φαίνεται μειώνεται γραμμικά από 0.9 σε 0.4.

$$\omega(fes) = 0.9 - 0.5 * \frac{fes}{Tfes} \tag{3.15}$$

Ο ψευδο-κώδικας ο οποίος πραγματοποιεί την μετάθεση δεικτών φαίνεται στο Σχήμα (3.4) και ο ψευδο-κώδικας ολόκληρου του αλγορίθμου το SLPSO φαίνεται στο Σχήμα (3.5). Μέσα στον ψευδο-κώδικα αυτό καλούνται και τα υπόλοιπα μέρη του αλγόριθμου τα οποία αναφέραμε ως τώρα:

Algorithm 5 UpdatePar()

1: Create a permutation of index number;
2: Update U_f for each particle by (14);
3: Update P_l for each particle by (15);
4: Calculate the number of particles using the convergence operator by (16);
5: Update related information of the four operators for each particle by Algorithm 3;
6: Calculate the inertia weight ω by (17);

Σχήμα 3.4: Ψευδο-κώδικας μετάθεσης δεικτών και ανανέωσης U_f, P_f (Li et al., 2012)

Algorithm 4 The SLPSO Algorithm

Generate initial swarm and set up parameters for each particle;
Set $fes := 0$, iteration counter for initial swarm $t := 0$;
while $fes < T\_Fes$ **do**
  for each particle k **do**
    Select one learning operator i using the roulette wheel selection rule;
    $Update(i, k, fes)$;
    G_i^k++;
    if $f(\vec{x}_k(t)) < f(\vec{x}_k(t-1)$ **then**
      g_i^k++; $m_k := 0$;
      $p_i^k + = f(\vec{x}_k(t-1)) - f(\vec{x}_k(t))$;
      Perform $UpdateAbest(k, fes)$ for the $abest$ position;
    else
      $m_k := m_k + 1$;
    end if
    if $f(\vec{x}_k(t)) < f(\vec{x}_{pbest_k})$ **then**
      $\vec{x}_{pbest_k} := \vec{x}_k$;
      if $f(\vec{x}_k) < f(\vec{x}_{abest})$ **then**
        $\vec{x}_{abest} := \vec{x}_k$;
      end if
    end if
    if $m_k \geq U_f^k$ **then**
      Update the selection ratios according to (10);
      for each operator j **do**
        $p_j^k := 0$; $g_j^k := 0$; $G_j^k := 0$;
      end for
    end if
  end for
  $UpdatePar()$;
  t++;
end while

Σχήμα 3.5: Ψευδο-κώδικας SLPSO (Li et al., 2012)

3.2.2.7 *Μηχανισμός Αναδιάταξης Σωματιδίων (Reshuffling Swarm)*

Όπως είπαμε εκτός από τον κλασσικό SLPSO χρησιμοποιήσαμε άλλους δύο μηχανισμούς. Αρχικά για να εξηγήσουμε τον μηχανισμό αυτό θα πρέπει να εξηγήσουμε πως ο αλγόριθμος, ή καλύτερα η υπολογιστική διαδικασία (το τρέξιμο του αλγορίθμου) είναι χωρισμένη σε κύκλους. Το πόσες επαναλήψεις συμπληρώνουν έναν κύκλο το ορίζει ο χρήστης. Στην δικιά μας περίπτωση ο αριθμός των επαναλήψεων για έναν κύκλο, που συμβολίζεται με την μεταβλητή *Cycle* είναι 100. Η μεταβλητή που δείχνει τον κύκλο στον οποίο βρισκόμαστε είναι η *ICycle.* Αν για παράδειγμα έχουμε ορίσει ολόκληρη την εξελικτική διαδικασία να είναι ίση με 2000 επαναλήψεις και τον κύκλο ίσο με 100 επαναλήψεις, τότε ο συνολικός αριθμός των κύκλων ισούται με 20.

Στο τέλος κάθε κύκλου, αυτό που κάνουμε είναι να παίρνουμε τα καλύτερα σωματίδια, ίσα σε αριθμό με το μέγεθος του σμήνος, όσον αφορά την αντικειμενική συνάρτηση από την αρχή της εξελικτικής διαδικασίας ως τώρα, και να τοποθετούμε αυτά στο σμήνος. Αυτό προϋποθέτει πως αποθηκεύομε κάπου τα καλύτερα αποτελέσματα κάθε επανάληψης. Αυτό έχει ως στόχο να αναδιατάσσει το σμήνος σε περίπτωση που εκείνη την χρονική στιγμή τα σωματίδια δεν βρίσκονται σε υποσχόμενες θέσεις και γενικότερα να αναδιατάξουμε το σμήνος χρησιμοποιώντας τις καλύτερες θέσεις που έχουμε βρεί ως τώρα. Σε περίπτωση που βρίσκονται τα σωματίδια σε υποσχόμενες θέσεις δεν υπάρχει πρόβλημα, αφού θα ξαναμπούνε στο καινούργιο σμήνος.

Η μεταβλητή *ICycle* ορίζεται ως εξής:

$$ICycle(iteration) = RoundUp(\frac{iteration}{Cycle}) \tag{3.16}$$

Έτσι μόλις περνάμε τις *Cycle* επαναλήψεις, αλλάζουμε κύκλο.

Η διαδικασία είναι η παρακάτω:

- Αρχικά να ελέγχεται αν βρισκόμαστε σε νέο κύκλο σε σχέση με την προηγούμενη επανάληψη.
- Αν έχουμε αλλάξει κύκλο, τότε ταξινομούμε τις αποθηκευμένες τιμές των καλύτερων λύσεων σε κάθε επανάληψη της αντικειμενικής συνάρτησης (ή αλλιώς Penalty function στον κώδικα, επειδή συμπεριλαμβάνει και τους περιορισμούς) μαζί με τους δείκτες (ώστε να μπορούμε να μπορούμε να ανατρέξουμε στο αρχείο των καλύτερων θέσεων κάθε επανάληψης)
- Μετά σε κάθε σωματίδιο στο σμήνος θέτουμε την καλύτερη θέση που έχει προκύψει από την ταξινόμηση. Αν έχουμε pa σωματίδια, τότε θα πάρουμε τις πρώτες pa καλύτερες θέσεις, οι οποίες προκύπτουν από τον δείκτη μετά την ταξινόμηση αντικειμενικής.

3.2.2.8 *Ο Μηχανισμός του Νέφους Απογόνων (Cloud offspring)*

Ο δεύτερος επιπλέον μηχανισμός που χρησιμοποιήσαμε είναι αυτός του νέφους απογόνων (cloud offspring). Ο μηχανισμός αυτός είναι στοχαστικός, και αυτό που κάνει είναι να παίρνει την θέση *gbest* σε κάθε επανάληψη, και να «διασκορπίζει» γύρω από αυτήν ένα νέφος σωματιδίων, χρησιμοποιώντας μια τυπική απόκλιση, ελέγχοντας αν έχει προκύψει κάποιο σωματίδιο με καλύτερη θέση από την *gbest.*

Για τον μηχανισμό αυτό χρησιμοποιούμε αρχικά μια μεταβλητή *elitism,* με την οποία αποφασίζεται πόσα σωματίδια θα συμμετέχουν στο νέφος σωματιδίων. Συγκεκριμένα τα σωματίδια του νέφους προκύπτουν από τον πολλαπλασιασμό των σωματιδίων στον SLPSO με την μεταβλητή *elitism.* Τα σωματίδια στο νέφος είναι *PartCloud=elitism*pa* . Επιπλέον χρησιμοποιείται μια τυπική απόκλιση *stdcloud* η οποία ορίζεται στην αρχή του αλγορίθμου από τον χρήστη, ένας αριθμός *r.rate* (reduction rate) ο οποίος παίρνει ακέραιες τιμές από 1 ως 4, και είναι η μεταβλητή η οποία καθορίζει πόσο γρήγορα θα μειωθεί η τελικά χρησιμοποιούμενη τυπική απόκλιση για τον διασκορπισμό των σωματιδίων, με το πέρας των επαναλήψεων. Επίσης, χρησιμοποιούμε τις μεταβλητές που ορίσαμε πιο πάνω *Cycle* και *ICycle.* Η εξίσωση υπολογισμού της τελικά χρησιμοποιούμενης τυπικής απόκλισης είναι:

$$StdCloud = std * l\ /(Icycle^{r.rate} * Cycle) \tag{3.17}$$

Όπου *l* μια τιμή η οποία ορίζεται εξαρχής, και εμείς την ορίσαμε ίση με το λ τον συντελεστή ποινής, τον οποίο θα εξηγήσουμε λίγο παρακάτω.

Αρχικά υπολογίζουμε την χρησιμοποιούμενη δυναμικά προσαρμοζόμενη στην διάρκεια της εξελικτικής διαδικασίας, τυπική απόκλιση , και μετά βρίσκουμε το νέφος σωματιδίων με βάση την θέση *gbest,* προσθέτοντας το γινόμενο της μέγιστης ταχύτητας, του *stdcloud* και του N(0,1) που είναι ένας τυχαία παραγόμενος αριθμός από κανονική κατανομή με μέση τιμή 0 και τυπική απόκλιση ίση με 1.

3.2.2.9 *Χρησιμοποιούμενη μέθοδος χειρισμού περιορισμών*

Ο SLPSO είναι ένας αλγόριθμος βελτιστοποίησης ο οποίος αντιμετωπίζει προβλήματα χωρίς περιορισμούς, όπως και γενικότερα οι εξελικτικοί αλγόριθμοι. Για να ξεπεράσουμε αυτό το εμπόδιο χρησιμοποιήσαμε την μέθοδο της συνάρτησης ποινής, με έναν απλό συντελεστή ποινής, σταθερό καθ’ όλη την διάρκεια της εξελικτικής διαδικασίας. Πριν την εκκίνηση του αλγορίθμου δίνεται μια τιμή στον συντελεστή ποινής (penalty coefficient) λ, και η εξισώσεις υπολογισμού της νέας αντικειμενικής συνάρτησης την οποίο ονομάζουμε συνάρτηση ποινής (penalty function), δεδομένου πως έχουμε ένα πρόβλημα βελτιστοποίησης όπως δίνεται στις εξισώσεις (2.1)(2.2)(2.3)(2.4)(2.5), είναι οι εξής:

$$\varphi(\vec{x})=\sum_{j=1}^{m} max\{0, g_j(\vec{x})\} \quad (3.18)$$

$$\psi(\vec{x})= \sum_{j=1}^{m} max\{0, h_j(\vec{x})\} \quad (3.19)$$

$$PenaltyFunction(\vec{x}) = f(\vec{x}) + \lambda * (\varphi(\vec{x}) + \psi(\vec{x})) \quad (3.20)$$

Όπου $g_j(\vec{x})$ και $h_i(\vec{x})$ είναι οι περιορισμοί ανισότητας και ισότητας αντίστοιχα, *m* και *p* είναι ο αριθμός των περιορισμών ανισότητας και ισότητας αντίστοιχα, λ είναι όπως είπαμε ο συντελεστής ποινής ο οποίος ορίζεται στην αρχή της εξελικτικής διαδικασίας και παραμένει σταθερός καθ’ όλη την διάρκεια αυτής (πχ ίσος με 1 , 10, 100), $f(\vec{x})$ είναι η αρχική αντικειμενική συνάρτηση, και *PenaltyFunction*$(\vec{x})$ είναι η τελική αντικειμενική συνάρτηση ή αλλιώς συνάρτηση ποινής, με την οποία γίνονται όλες οι πράξεις και οι συγκρίσεις πλέον.

3.3 Μαθηματικό μοντέλο εDEag

3.3.1 **Βασικά χαρακτηριστικά εDEag**

Ο αλγόριθμος εDEag, είναι ένας αλγόριθμος βελτιστοποίησης για προβλήματα υπό περιορισμούς και αποτελείται από τέσσερα μέρη ή αλλιώς τέσσερα ξεχωριστά κομμάτια τα οποία έχουν το κάθε ένα τον δικό τους ρόλο. Η μέθοδος ε constrained είναι μια μέθοδος μετασχηματισμού αλγορίθμου, η οποία μπορεί να μετατρέψει αλγορίθμους για επίλυση προβλημάτων χωρίς περιορισμούς (unconstrained), σε αλγόριθμους για επίλυση προβλημάτων υπό περιορισμούς (constrained), χρησιμοποιώντας το επίπεδο σύγκρισης ε, το οποίο συγκρίνει σημεία βασισμένα στο ζευγάρι της αξίας της αντικειμενικής του συνάρτησης και την παραβίαση των περιορισμών. Ο συνδυασμός της μεθόδου ε constrained και του αλγόριθμου βελτιστοποίησης differential evolution (DE), είναι ο εDE, και έχει αποδειχτεί ότι μπορεί να λύνει γρήγορα και με πολύ ποιοτικά αποτελέσματα προβλήματα. Επίσης προστέθηκε η μέθοδος gradient-based mutation, η οποία χρησιμοποιεί κλίσεις των περιορισμών ώστε να λύσει προβλήματα με δύσκολους περιορισμούς. Ο εDEag χρησιμοποιεί ένα αρχείο για να διατηρεί την ποικιλομορφία των ατόμων και υιοθετεί έναν καινούργιο τρόπο να επιλέγει την παράμετρο ελέγχου *ε* (Takahama and Sakai, 2010).

3.3.2 Η Μέθοδος *ε* Constrained

Υποθέτοντας ότι έχουμε ένα πρόβλημα βελτιστοποίησης σύμφωνα με τις εξισώσεις (2.1)(2.2)(2.3)(2.4)(2.5), ορίζουμε αρχικά την συνάρτηση παραβίασης των περιορισμών η οποία μπορεί να δοθεί είτε από τον μέγιστο όλων των παραβιάσεων είτε από το άθροισμα όλων των παραβιάσεων:

$$\varphi(\vec{x}) = \max\left\{\{0, g_j(\vec{x})\}, \{0, h_j(\vec{x})\}\right\} \tag{3.21}$$

$$\varphi(\vec{x}) = \sum_j \left|max\{0, g_j(\vec{x})\}\right|^p + \sum_j \left|h_j(\vec{x})\right|^p \tag{3.22}$$

Όπου *p* ένας θετικός αριθμός.

Το επίπεδο συγκρίσεων *ε* ορίζεται ως μια σειρά σχέσεων σε ένα ζευγάρι αξίας αντικειμενικής και παραβίαση των περιορισμών $(f(\vec{x}), \varphi(\vec{x}))$. Αν η παραβίαση των περιορισμών του σημείου είναι μεγαλύτερη από 0, τότε το σημείο είναι μη εφικτό και έχει χαμηλή αξία. Το επίπεδο συγκρίσεων *ε* ορίζεται ως μια λεξικογραφική σειρά στην οποία η $\varphi(\vec{x})$ προηγείται της $f(\vec{x})$, επειδή το να είναι εφικτή λύση το $\vec{x}$ είναι σημαντικότερο από την ελαχιστοποίηση της $f(\vec{x})$. Αυτή η προτεραιότητα μπορεί να ελεγχθεί με την παράμετρο *ε*.

Έστω ότι f_1 f_2 και φ_1 φ_2 είναι οι αξίες της αντικειμενικής συνάρτησης και οι παραβιάσεις των περιορισμών σε σημεία x_1 x_2 αντίστοιχα. Τότε για κάθε $\varepsilon \geq 0$, το επίπεδο συγκρίσεων *ε*, $<_\varepsilon$ και $\leq_\varepsilon$ μεταξύ $(f_1\ \varphi_1)$ και $(f_2\ \varphi_2)$ ορίζονται ως εξής:

$$(f_1\ \varphi_1) <_\varepsilon (f_2\ \varphi_2) \Leftrightarrow \begin{cases} f_1 < f_2, & if\ \varphi_1, \varphi_2 \leq \varepsilon \\ f_1 < f_2, & if\ \varphi_1 = \varphi_2 \\ \varphi_1 < \varphi_2, & otherwise \end{cases} \tag{3.23}$$

$$(f_1\ \varphi_1) \leq_\varepsilon (f_2\ \varphi_2) \Leftrightarrow \begin{cases} f_1 \leq f_2, & if\ \varphi_1, \varphi_2 \leq \varepsilon \\ f_1 \leq f_2, & if\ \varphi_1 = \varphi_2 \\ \varphi_1 \leq \varphi_2, & otherwise \end{cases} \quad (3.24)$$

Σε περίπτωση που ε=∞, το επίπεδο συγκρίσεων ε $<_\infty$ και $\leq_\infty$είναι ισοδύναμα με την κανονική σύγκριση < και ≤ μεταξύ αξιών αντικειμενικής συνάρτησης. Επίσης, στη περίπτωση που ε=0, $<_0$ και $\leq_0$ είναι ισοδύναμα στην λεξικογραφική σειρά στην οποία η παραβίαση περιορισμού $\varphi(\vec{x})$ προηγείται της $f(\vec{x})$.

Όπως είδαμε το επίπεδο συγκρίσεων ε μετατρέπει προβλήματα βελτιστοποίησης με περιορισμούς σε προβλήματα βελτιστοποίησης χωρίς περιορισμούς αντικαθιστώντας την σχέση προτεραιότητας. Ένα πρόβλημα βελτιστοποίησης το οποίο λύνεται με την μέθοδο συγκρίσεων ε, μετατρέπεται ως εξής:

$$(P_{\leq_\varepsilon})\ \ minimize_{\leq_\varepsilon}\ \ f(\vec{x}) \quad (3.25)$$

Όπου $minimize_{\leq_\varepsilon}$ αναφέρεται στην ελαχιστοποίηση βάση στη ε επιπέδου σύγκριση $\leq_\varepsilon$. Επίσης, ένα πρόβλημα (P^ε) ορίζεται ώστε ο περιορισμός $\varphi(\vec{x}) = 0$, «χαλαρώνεται» και αντικαθίσταται με τα $\varphi(\vec{x}) \leq \varepsilon$:

$$\begin{array}{c} (P^\varepsilon)\ minimize\ \ f(\vec{x}) \\ subject\ to\ \ \varphi(\vec{x}) \leq \varepsilon \end{array} \quad (3.26)$$

3.3.3 Διαφορική Εξέλιξη (Differential evolution)

Η μέθοδος DE αναπτύχθηκε από τους Storn και Price (Storn and Price, 1997), είναι μια μέθοδος στοχαστικής αναζήτησης και χρησιμοποιεί πληθυσμό λύσεων. Οι διάφορες παραλλαγές του DE ταξινομούνται χρησιμοποιώντας την σήμανση DE/base/num/cross. Το "base" δείχνει την μέθοδο με την οποία επιλέγεται ένα άτομο το οποίο θα σχηματίσει το «βασικό διάνυσμα» (base vector). Για παράδειγμα ο αλγόριθμος DE/rand επιλέγει τυχαία ένα άτομο από τον πληθυσμό ως «βασικό διάνυσμα» , ενώ ο αλγόριθμος DE/best επιλέγει το καταλληλότερο άτομο του πληθυσμού.

Στον αλγόριθμο DE/rand/1, για κάθε άτομο $\boldsymbol{x^{i}}$ επιλέγονται τρία άτομα $\boldsymbol{x^{p1}}$, $\boldsymbol{x^{p2}}$ και $\boldsymbol{x^{p3}}$ από τον πληθυσμό. Ένα νέο διάνυσμα $\boldsymbol{x'}$ δημιουργείται με βάση το βασικό διάνυσμα $\boldsymbol{x^{p1}}$ και το διάνυσμα της διαφοράς $\boldsymbol{x^{p2}}$ - $\boldsymbol{x^{p3}}$ όπου F είναι ο παράγοντας κλιμάκωσης.

$$x' = x^{p1} + F(x^{p2} - x^{p3}) \tag{3.27}$$

Όπου “num” είναι ο αριθμός των διαφορετικών διανυσμάτων που χρησιμοποιούνται για το βασικό διάνυσμα. “cross” είναι ο μηχανισμός διασταύρωσης ο οποίος χρησιμοποιείται για έναν απόγονο. Για παράδειγμα, ‘bin’ δείχνει ότι ο μηχανισμός διασταύρωσης ελέγχεται από διωνυμική διασταύρωση με σταθερό συντελεστή διασταύρωσης και ‘exp’ δείχνει ότι η διασταύρωση ελέγχεται από ένα είδος διασταύρωσης δύο σημείων χρησιμοποιώντας εκθετική κατανομή μειώνοντας την τιμή διασταύρωσης.

Στο σχήμα (3.6) βλέπουμε έναν ψευδο-κώδικα εκθετικής διασταύρωσης. Ένας νέος απόγονος $\boldsymbol{x^{child}}$ δημιουργείται από έναν γονέα $\boldsymbol{x^{i}}$ και έναν διάνυσμα $\boldsymbol{x'}$, όπου CR είναι η τιμή διασταύρωσης.

```
k=1; j=randint(1,n);
do {
    x_j^child=x'_j;
    k=k+1; j=(j+1)%n;
} while(k ≤ n && u(0,1) < CR);
while(k ≤ n) {
    x_j^child=x_j^i;
    k=k+1; j=(j+1)%n;
}
```

Σχήμα 3.6: Εκθετική διασταύρωση, όπου randint(1,n) παράγει τυχαία έναν ακέραιο αριθμό στο διάστημα [1,n] (Takahama and Sakai, 2010).

3.3.4 Έλεγχος του Επιπέδου ε

Συνήθως, το επίπεδο *ε* ελέγχεται σύμφωνα με την εξίσωση (3.29). Το αρχικό επίπεδο *ε*, *ε(0),* είναι η παραβίαση των περιορισμών του καλύτερου *θ*-οστού ατόμου στην αρχική αναζήτηση λύσεων. Το επίπεδο *ε* ανανεώνεται έως ότου ο αριθμός των επαναλήψεων *t* γίνει ίσος με την γενιά ελέγχου T_c. Αν οι επαναλήψεις περάσουν τον αριθμό T_c, το επίπεδο *ε* θέτεται ίσο με 0 ώστε να λαμβάνουμε λύσεις με ελάχιστη παραβίαση περιορισμών.

$$\varepsilon(0) = \varphi(x_\theta) \tag{3.28}$$

$$\varepsilon(t) = \begin{cases} \varepsilon(0)(1-\frac{t}{T_c})^{cp}, 0 < t < T_c \\ 0\,, \quad t \geq T_c \end{cases} \tag{3.29}$$

Όπου x_θ είναι το καλύτερο θ-οστό άτομο. Αν $\theta > 1$, $\varphi(x_\theta) = \theta \times max_x \varphi(x)$.

Στην μελέτη αυτή, οι συγγραφείς προτείνουν έναν απλό τρόπο να θέτουν την τιμή cp. Το επίπεδο ε προσαρμόζεται να είναι μια μικρή τιμή $\varepsilon_\lambda=10^{-5}$ στην γενιά $T_\lambda = 0.95T_c$.

$$\varepsilon(T_\lambda) = \varepsilon(0)\left(1-\frac{T_\lambda}{T_c}\right)^{cp} = \varepsilon_\lambda \tag{3.30}$$

$$cp = \frac{\log \varepsilon_\lambda - \log \varepsilon(0)}{\log\left(1-\frac{T_\lambda}{T_c}\right)} = (-5-\log \varepsilon(0))/\log 0.05 \tag{3.31}$$

Για να αποφύγουμε πολύ μικρές τιμές του cp, αν η τιμή του cp είναι μικρότερη από cp_{min}, θέτουμε το cp ίσο με cp_{min} όπου cp_{min}=3. Επίσης, αφού περάσουμε την γενιά T_λ, με στόχο να αυξήσουμε το επίπεδο ε, για να αυξηθεί η περιοχή ε-εφικτών λύσεων και για να αναζητήσουμε καλύτερες τιμές αντικειμενικής, το cp μειώνεται και ο F αυξάνεται ως εξής:

$$cp = 0.3cp + 0.7cp_{min} \tag{3.32}$$

$$F = 0.3F_0 + 0.7 \tag{3.33}$$

3.3.5 Gradient-based Mutation

Η μέθοδος gradient-based mutation, είναι μια διαδικασία παρόμοια με την μέθοδο επιδιόρθωσης που προτείνουν οι Chootinan και Chen. Το διάνυσμα των συναρτήσεων περιορισμών $C(\vec{x})$, το διάνυσμα των παραβιάσεων των περιορισμών $\Delta C(\vec{x})$ και η αύξηση ενός σημείου $\vec{x}$ ώστε να ικανοποιεί τους περιορισμούς $\overrightarrow{\Delta x}$ καθορίζονται ως εξής:

$$C(\vec{x}) = (g_1(\vec{x}) \ldots g_q(\vec{x}) h_{q+1}(\vec{x}) \ldots h_m(\vec{x}))^T \tag{3.34}$$

$$\Delta C(\vec{x}) = (\Delta g_1(\vec{x}) \ldots \Delta g_q(\vec{x}) h_{q+1}(\vec{x}) \ldots h_m(\vec{x}))^T \tag{3.35}$$

$$\nabla C(\vec{x}) \Delta\vec{x} = -\Delta C(\vec{x}) \tag{3.36}$$

$$\Delta\vec{x} = \nabla C(\vec{x})^{-1}\Delta C(\vec{x}) \tag{3.37}$$

Όπου $\Delta g_j(\vec{x}) = max\{0, g_1(\vec{x})\}$. Και $\nabla C(\vec{x})$ είναι η κλίση του πίνακα $C(\vec{x})$.

$$\nabla C(\vec{x}) = \begin{pmatrix} \frac{\partial g_1(\vec{x})}{\partial x_1} & \frac{\partial g_1(\vec{x})}{\partial x_2} & \cdots & \frac{\partial g_1(\vec{x})}{\partial x_n} \\ \vdots & \vdots & \cdots & \vdots \\ \frac{\partial g_q(\vec{x})}{\partial x_1} & \frac{\partial g_q(\vec{x})}{\partial x_2} & \cdots & \frac{\partial g_q(\vec{x})}{\partial x_n} \\ \frac{\partial g_{q+1}(\vec{x})}{\partial x_1} & \frac{\partial g_{q+1}(\vec{x})}{\partial x_2} & \cdots & \frac{\partial g_{q+1}(\vec{x})}{\partial x_n} \\ \vdots & \vdots & \cdots & \vdots \\ \frac{\partial g_m(\vec{x})}{\partial x_1} & \frac{\partial g_m(\vec{x})}{\partial x_2} & \cdots & \frac{\partial g_m(\vec{x})}{\partial x_n} \end{pmatrix} \tag{3.38}$$

Όταν κάποιο περιορισμοί είναι μη-παραγωγίσιμοι, τότε ο πίνακας $\nabla C(\vec{x})$ μπορεί να ληφθεί αριθμητικά, υπολογίζοντας $C(\vec{x})$ συνεχόμενα αλλάζοντας την τιμή κάθε μεταβλητής απόφασης από x_i σε $x_i + \eta$:

$$\nabla C(\vec{x}) = \frac{1}{\eta}(C(\vec{x} + \eta\overrightarrow{e_1}) - C(\vec{x}), \dots, C(\vec{x} + \eta\overrightarrow{e_n}) - C(\vec{x}) \tag{3.39}$$

Όπου $\overrightarrow{e_1}$ είναι ένα διάνυσμα του οποίου το i-οστό στοιχείο είναι 1 και όλα τα άλλα είναι 0 και η είναι μια πολύ μικρή τιμή.

Αν έχει ληφθεί το $\Delta\vec{x}$, ένα μεταλλαγμένο διάνυσμα μπορεί να ληφθεί ως εξής:

$$\overrightarrow{x^{new}} = \vec{x} + \overrightarrow{\Delta x} \tag{3.40}$$

Αυτή η μετάλλαξη ή αλλιώς διαδικασία διόρθωσης εκτελείται με πιθανότητα P_g.

3.3.6 Ο Αλγόριθμος εDEag

Οι ιδέες οι οποίες εισάγονται στον εDEag είναι πρώτον ένα αρχείο το οποίο διατηρεί την ποικιλομορφία των ατόμων. Δεύτερον όταν ένας γονέας παράγει έναν απόγονο ο οποίος δεν είναι καταλληλότερος από αυτόν, τότε ο γονέας μπορεί να δημιουργήσει και άλλο απόγονο Τρίτον, η μέθοδος σύγκρισης με αυτόματη επιλογή του επιπέδου ε υιοθετείται για να βελτιωθεί η ευχρηστία του εDE, και τέταρτον η μέθοδος gradient based mutation εφαρμόζεται για προβλήματα με δύσκολους περιορισμούς.

Ο αλγόριθμος εDEag φαίνεται παρακάτω:

1. Αρχικοποίηση του αρχείου. Δημιουργία με τυχαίο τρόπο M ατόμων $A = \{x^i, i = 1,2, \dots . M\}$ στον χώρο αναζήτησης S και σχηματισμός αρχείου.
2. Αρχικοποίηση του επιπέδου ε. Ένα αρχικό επίπεδο ε δίνεται από την συνάρτηση ελέγχου $\varepsilon(0)$.
3. Αρχικοποίηση πληθυσμού. Τα καταλληλότερα N άτομα επιλέγονται από το αρχείο A και σχηματίζουν έναν πληθυσμό $P = \{x^i\}$. Η κατάταξη των ατόμων καθορίζεται από την σύγκριση επιπέδου ε.
4. Συνθήκη τερματισμού. Αν ο αριθμός των υπολογισμών της αντικειμενικής συνάρτησης υπερβεί τον μέγιστο αριθμό FE_{max} ο αλγόριθμος τερματίζεται.
5. Διαδικασίες DE (το πολύ δύο φορές). Κάθε άτομο x^i επιλέγεται ως γονέας. Αν όλα τα άτομα επιλέγονται ως γονείς τότε πηγαίνουμε στο βήμα 7. Η DE/rand/1/exp διαδικασία εφαρμόζεται και ένας νέος απόγονος x^{child}. Ένας σταθερός παράγοντας κλιμάκωσης F_0 και μια σταθερή τιμή διασταύρωσης CR_0 χρησιμοποιούνται με πιθανότητα 0.05. Το τρίτο διάνυσμα x^{p3} επιλέγεται από το αρχείο A με πιθανότητα 0.95 και επιλέγεται από τον πληθυσμό P με πιθανότητα 0.05. Αν ο απόγονος είναι καλύτερος του γονέα, με βάση την μέθοδο σύγκρισης επιπέδου ε τότε ο γονέας x^i αντικαθίσταται άμεσα από το διάνυσμα x^{child} και πηγαίνουμε στο βήμα 6. Αλλιώς η ίδια διαδικασία εφαρμόζεται στον γονέα, μόνο μια φορά ακόμη.
6. Εφαρμογή της μεθόδου gradient-based mutation αν x^{child} δεν είναι εφικτή λύση (infeasible) ή $\varphi(x^{child}) > 0$, ο απόγονος αλλάζει με την μέθοδο gradient-based mutation με πιθανότητα P_g μέχρι ο αριθμός των αλλαγών γίνει R_g ή x^{child} γίνει εφικτή

λύση (feasible). Πηγαίνουμε πίσω στο βήμα 5 και επιλέγεται το επόμενο άτομα σαν γονέας.

7. Έλεγχος του επιπέδου *ε*. Το επίπεδο *ε* ανανεώνεται με την συνάρτηση ελέγχου *ε(t).*
8. Πηγαίνουμε πίσω στο βήμα 4.

Παρακάτω στο Σχήμα 3.7 βλέπουμε τον ψευδο-κώδικα του αλγόριθμου εDEag, τον οποίο μόλις περιγράψαμε.

```
εDEag()
{
 F=F_0; CR=CR_0;
// Initialize an archive
 A=M individuals generated randomly in S;
 FE=M;
// Initialize the ε level
 ε=ε(0);
// Initialize a population
 P=Top N individuals {x^i} from A using <_ε;
 A=A-P;
 for(t=1; FE ≤ FE_max; t++) {
  F=F_0;
  if(t > 0.95T_C && t < T_C)
    Modify the ε level and F using Eq.(12),(13);
  if(u(0,1)<0.05)
    F=1+|rand_G(0,0.05)|, F is truncated to 1.1;
  for(i=1; i ≤ N; i++) {
// DE operation
   for(k=1; k ≤ 2; k++) {
     x^p1=Randomly selected from P;
     x^p2=Randomly selected from P;
     if(u(0,1)<0.05)
       x^p3=Randomly selected from P;
     else
       x^p3=Randomly selected from P∪A;
     x'=x^p1+(x^p2 - x^p3);
     x^child=trial vector is generated
          from x^i and x' by exponential crossover;
// Gradient-based mutation
     if(t%n==0 && u(0,1) < P_g)
       for(h=1; h ≤ R_g && φ(x^child)>0; h++) {
         Apply gradient-based mutation to x^child;
         FE=FE+n+1;
       }
     else FE=FE+1;
     if((f(x^child), φ(x^child)) <_ε (f(x^i), φ(x^i))) {
       x^i=x^child;
       break;
     } else {
       x^A=Randomly selected from A;
       A=A+{x^child}-{x^A};
     }
   }
  }
// Control the ε level
  ε=ε(t);
 }
}
```

Σχήμα 3.7: Ψευδο-κώδικας του αλγορίθμου εDEag (Takahama and Sakai, 2010).

3.4 Η μέθοδος της στοχαστικής ταξινόμησης (Stochastic Ranking)

Στην προσπάθεια ενός συντελεστή ποινής ο οποίος θα ισορροπεί την επίδραση της αντικειμενικής συνάρτησης και της συνάρτησης ποινής, οι Runarsson και Yao ανέπτυξαν την μέθοδο Stochastic Ranking (Runarsson and Yao, 2000). Η μέθοδος αυτή χρησιμοποιεί την βασική ιδέα της ταξινόμησης των ατόμων σε συνδυασμό με την εξισορρόπηση της αντικειμενικής συνάρτησης και της συνάρτησης ποινής σε ένα πληθυσμό. Η διαδικασία αυτή, συνδυάζεται με μία μέθοδο (ES) ώστε να δημιουργηθούν οι νέες γενιές, χρησιμοποιώντας έναν αριθμό των καταλληλότερων ατόμων με βάση την ταξινόμησή τους,

Η ταξινόμηση επιτυγχάνεται μέσα από μια στοχαστική διαδικασία ταξινόμησής, παρόμοια με την bubble sort. Κάθε άτομο συγκρίνεται με το επόμενο στην κατάταξη, και αν αυτό αποδειχτεί καταλληλότερο, τότε ανταλλάζουν θέση. Τα κριτήρια αξιολόγησης των σωματιδίων είναι η αντικειμενική συνάρτηση και η συνάρτηση ποινής, όμως εισάγεται και μία πιθανότητα P_f, η οποία θέτεται από τον χρήστη. Πιο συγκεκριμένα, P_f είναι η πιθανότητα να γίνει η σύγκριση μεταξύ δύο ατόμων, μόνο βάση της αντικειμενικής συνάρτησης. Αυτό γίνεται, ελέγχοντας την πιθανότητα P_f με έναν τυχαίο αριθμό $U(0,1)$. Όπως είπαμε αν η πιθανότητα P_f είναι μεγαλύτερη, τότε η σύγκριση γίνετε με βάση την αντικειμενική συνάρτηση μόνο, αλλιώς συγκρίνετε πρώτα η συνάρτηση ποινής, και θεωρείται καλύτερο το άτομο αυτό με μικρότερη ποινή. Επιπλέον, αν και τα δύο άτομα δεν παραβιάζουν τους περιορισμούς, εννοείται πως η σύγκριση γίνετε και πάλι βάση των τιμών της αντικειμενικής συνάρτησης. Στο Σχήμα (3.8) φαίνεται ο ψευδο-κώδικας αυτής της διαδικασίας.

```
I_j = j ∀ j ∈ {1, . . . , λ}
for i = 1 to N do
    for j = 1 to λ − 1 do
        sample u ∈ U(0, 1)
        if (φ(I_j) = φ(I_{j+1}) = 0) or (u < P_f) then
            if (f(I_j) > f(I_{j+1})) then
                swap(I_j, I_{j+1})
            fi
        else
            if (φ(I_j) > φ(I_{j+1})) then
                swap(I_j, I_{j+1})
            fi
        fi
    od
    if no swap done break fi
od
```

Σχήμα 3.8: Ο ψευδο-κώδικας της μεθόδου Stochastic Ranking (Runarsson and Yao, 2000)

4 Προβλήματα που μελετήθηκαν

4.1 Συναρτήσεις του διαγωνισμού CEC 2010

Κάποιες από τις συναρτήσεις στις οποίες δοκιμάσαμε τον αλγόριθμο SLPSO, και τον συγκρίναμε με άλλους, είναι αυτές του διαγωνισμού του CEC 2010 πάνω στο θέμα Constrained Real-Parameter Optimization. Τα προβλήματα σε αυτό τον διαγωνισμό είναι προβλήματα βελτιστοποίησης με περιορισμούς ανισότητας ή/και ισότητας, όπως έχουν περιγράφει στις εξισώσεις (2.1) (2.2) (2.3) (2.4) (2.5).

Οι συναρτήσεις αυτές περιγράφονται αναλυτικά στο *Παράρτημα Α*, όπου φαίνονται οι τεχνικές λεπτομέρειες και άλλες πληροφορίες σχετικά με τις προδιαγραφές του διαγωνισμού CEC 2010.

4.2 Συναρτήσεις δοκιμής από την εργασία των Runarsson και Yao

Επίσης έγιναν δοκιμές του SLPSO πάνω σε δύο δοκιμαστικές συναρτήσεις από εργασία των Runarsson και Yao "*Stochastic Ranking for Constrained Evolutionary Optimization"* (Runarsson and Yao, 2000). Αναλυτικά παρουσιάζονται οι συναρτήσεις στο *Παράρτημα Α*.

4.3 Το πρόβλημα βέλτιστης ανάθεσης φορτίου (Economic load dispatch problem)

Επίσης μελετήσαμε ένα πρόβλημα βέλτιστης ανάθεσης φορτίου το οποίο μοντελοποιήθηκε και λύθηκε με τον αλγόριθμο SLPSO, στο λογισμικό MATLAB ®, και επιπλέον δοκιμάσαμε το ίδιο πρόβλημα στον αλγόριθμο εDEag, σε γλώσσα προγραμματισμού C, τον οποίο μπορεί να βρει κανείς στην ιστοσελίδα του Takahama. Για να γίνει δυνατή η δοκιμή του εDEag πάνω στο πρόβλημα βέλτιστης ανάθεσης φορτίου, χρειάστηκε να κατεβάσουμε από την ιστοσελίδα του Takahama τα αρχεία του αλγορίθμου, που όπως είπαμε είναι γραμμένα σε C, και μετά να επέμβουμε σε αυτά, ορίζοντας το πρόβλημα βέλτιστης ανάθεσης φορτίου. Έτσι καταφέραμε να δοκιμάσουμε τον SLPSO και τον εDEag πάνω σε αυτό το πρακτικό πρόβλημα, ώστε να συγκρίνουμε τα αποτελέσματά τους.

Ο σκοπός ενός προβλήματος βέλτιστης ανάθεσης φορτίου είναι να προγραμματίσει την παραγωγή ηλεκτρικής ενέργειας από τις δεσμευμένες μονάδες παραγωγής, δηλαδή αυτές που είναι προγραμματισμένες να λειτουργήσουν, με τέτοιο τρόπο ώστε να ικανοποιείται η συνολική ζήτηση της ενέργειας με το ελάχιστο κόστος, ικανοποιώντας όποιους περιορισμούς λειτουργίας υπάρχουν (dos Santos Coelho and Lee, 2000).

Σε μια ελεύθερη αγορά ηλεκτρικού ρεύματος, η βελτιστοποίηση της ανάθεσης φορτίου στις μονάδες παραγωγής έχει οικονομικό ενδιαφέρον για τον διαχειριστή του συστήματος διανομής ενέργειας. Ως εκ τούτου η επίλυση του προβλήματος βέλτιστης ανάθεσης φορτίου και ο αλγόριθμος με τον οποίο θα επιτευχθεί αυτό, έχουν μεγάλη αξία για τον αυτόν.

Τα τελευταία χρόνια, τεχνικές βελτιστοποίησης εμπνευσμένες από την EC και τις τεχνικές που μιμούνται τα σμήνη (πχ PSO) έχουν αποδείξει πως μπορούν να αποτελέσουν εναλλακτική για την βελτιστοποίηση δύσκολων προβλημάτων ανάθεσης φορτίου.

Όπως είπαμε σε ένα πρόβλημα βέλτιστης ανάθεσης φορτίου ψάχνουμε να βρούμε το βέλτιστο πρόγραμμα λειτουργίας των δεσμευμένων μονάδων παραγωγής ενέργειας, με τέτοιο τρόπο ώστε να ικανοποιείται η συνολική ζήτηση ενέργειας και να τηρούνται όλοι οι περιορισμοί. Για να ορίσουμε το μαθηματικό μοντέλο για ένα τέτοιο πρόβλημα θα πρέπει να έχουμε μια συνάρτηση κόστους για την κάθε μονάδα παραγωγής, την ελάχιστη και την μέγιστη δυναμικότητα της κάθε μονάδας παραγωγής ενέργειας και την συνολική ζήτηση της ενέργειας για κάθε ώρα (χρονική μονάδα μπορεί να είναι και διαφορετική, ανάλογα με την βελτιστοποίηση που προσπαθούμε να επιτύχουμε).

Στην δικιά μας περίπτωση, το πρόβλημα αφορά την βελτιστοποίηση του προγράμματος λειτουργίας σε 25 μονάδες παραγωγής ενέργειας, τα οποία πρέπει να ικανοποιούν μια ζήτηση d_i όπου *i=1,2,...,24* και αντιπροσωπεύει τις ώρες της ημέρας. Κάθε μονάδα παραγωγής ενέργειας έχει την δικιά της συνάρτηση κόστους $a * x^2 + b * x + c$ όπου a,b,c σταθερές οι οποίες διαφέρουν σε κάθε εργοστάσιο και όπου x είναι η παραγωγή ενέργειας για μια συγκεκριμένη ώρα σε κάποιο εργοστάσιο. Επίσης σε κάθε μονάδα υπάρχει ένα ελάχιστο και ένα μέγιστο όριο παραγωγής ενέργειας για κάθε ώρα. Έτσι προκύπτει ένα πρόβλημα $25 \times 24 = 600$ μεταβλητών απόφασης. Αναλυτικά το πρόβλημα στην μαθηματική του μορφή έχει ως εξής:

$$Minimize\ f(\vec{x}) = \sum_{i=1}^{25} \sum_{j=1}^{24} (a_i {x_{i,j}}^2 + b_i x_{i,j} + c_i) \tag{4.1}$$

$$g_j(\vec{x}) = \sum_{i=1}^{25}(x_{i,j}) - d_j \geq 0 \text{ , } j = 1,2, \dots 24 \tag{4.2}$$

$$Min_i \leq x_{i,j} \leq Max_i \text{ , j=1,2,…24 , i=1,2,…,25} \tag{4.3}$$

Όπου στις παραπάνω εξισώσεις a_i, b_{i}, c_i είναι οι σταθερές του πολυωνύμου για την κάθε μονάδα παραγωγής, δηλαδή πίνακες (25 x 1), d_j είναι η ζήτηση (demand) για ενέργεια, κάθε ώρα της ημέρας, δηλαδή πίνακας (24 x 1) και όπου Min_i, Max_i είναι η ελάχιστη και η μέγιστη επιτρεπτή ωριαία παραγωγή ενέργειας σε κάθε μονάδα παραγωγής, δηλαδή είναι πίνακες (25 x 1). Η λύση που ψάχνουμε είναι ένας πίνακας (25 x 24). Αν σκεφτούμε γραφικά το πρόβλημα ψάχνουμε έναν πίνακα (25 x 24) και οι περιορισμοί μας είναι το άθροισμα κάθε στήλης (που αντιπροσωπεύει την ώρα της ημέρας) να υπερβαίνει την ζήτηση d_j. Επίσης κάθε μεταβλητή απόφασης έχει ένα πεδίο ορισμού, το οποίο διαφέρει για κάθε γραμμή (μονάδα παραγωγή), και αντιπροσωπεύει το ελάχιστο και το μέγιστο όριο παραγωγής για κάθε μονάδα παραγωγής. Τέλος, το κόστος υπολογίζεται εφαρμόζοντας το πολυώνυμο για κάθε μεταβλητή απόφασης, το οποίο διαφέρει όμως για κάθε γραμμή (μονάδα παραγωγής). Συγκεκριμένα, τα διανύσματα του προβλήματος είναι:

a=(1.8, 0.8, 1.7, 21.9, 20.9, 1.3, 24, 12, 14, 22, 1.5, 2.5, 3.1, 1.5, 19, 1.5, 19, 1.8, 2.3, 27, 21.7, 2.4, 2.8, 1.5)

b=(55.9, 39.2, 290.4, 120.9, 120.9, 33, 53, 32, 300, 138, 147, 35, 45, 120, 128, 116, 120.9, 120.9, 33, 53, 32, 300, 260, 400, 35)

c=(55, 40, 5, 43.3, 43.3,45, 3, 3, 46, 43, 55, 42, 5, 43.3, 43.3, 17, 29, 35, 55, 40, 7, 43.3, 43.3, 51, 42)

d=(700, 750, 468, 432, 580, 590, 610, 670, 760, 790, 920, 1010, 1103, 955, 904, 840, 805, 748, 676, 650, 649, 621, 635, 790)

Min = (10, 12, 15, 13, 23, 22, 26, 18, 29, 30, 16, 12, 15, 25, 26, 14, 16, 17, 22, 13, 16, 12, 27, 20, 26)

Max = (50, 48, 55, 67, 110, 43, 76, 54, 45, 52, 43, 51, 50, 48, 55, 67, 127, 54, 45, 52, 43, 51, 98, 86, 47)

5 Παρουσίαση και ανάλυση των αποτελεσμάτων

5.1 Αποτελέσματα συναρτήσεων του διαγωνισμού CEC 2010

Παρακάτω θα παρουσιάσουμε τα αποτελέσματα του SLPSO στις συναρτήσεις του διαγωνισμού CEC 2010 που μελετήσαμε. Τα αποτελέσματα θα συγκριθούν με αυτά του αλγόριθμου εDEag. Παρουσιάζουμε έναν πίνακα με αποτελέσματα του SLPSO και του εDEag. Συγκεκριμένα παρουσιάζουμε την καλύτερη, την μέση και την χειρότερη λύση της αντικειμενικής συνάρτησης, την τυπική απόκλιση της αντικειμενικής συνάρτησης και τον μέσο όρο της παραβίασης των περιορισμών από τα 25 τρεξίματα. Επίσης παρουσιάζουμε την διάμεση τιμή των 25 παραβιάσεων των περιορισμών για τον SLPSO, επειδή σε κάποιες συναρτήσεις υπάρχουν μεγάλες διακυμάνσεις με πολλές από τις τιμές να είναι κοντά στην εφικτή περιοχή, που όμως δεν θεωρούνται εφικτές λύσεις. Η παραβίαση κάτω από 1.00e-04 θεωρείται ίση με μηδέν. Επιπλέον παρουσιάζουμε για τον αλγόριθμο SLPSO τη βέλτιστη λύση της αντικειμενικής συνάρτησης μαζί με την παραβίαση των περιορισμών που βρέθηκαν, για κάθε ένα από τα 25 τρεξίματα.

Στην παρουσίαση παρακάτω όπου Obj και CV είναι η τιμή της αντικειμενικής συνάρτησης και η παραβίαση των περιορισμών. Να προσθέσουμε ότι ο αλγόριθμος ρυθμίστηκε ώστε να κάνει 2000 επαναλήψεις με 5 σωματίδια και πολλαπλασιαστή νέφους 50.

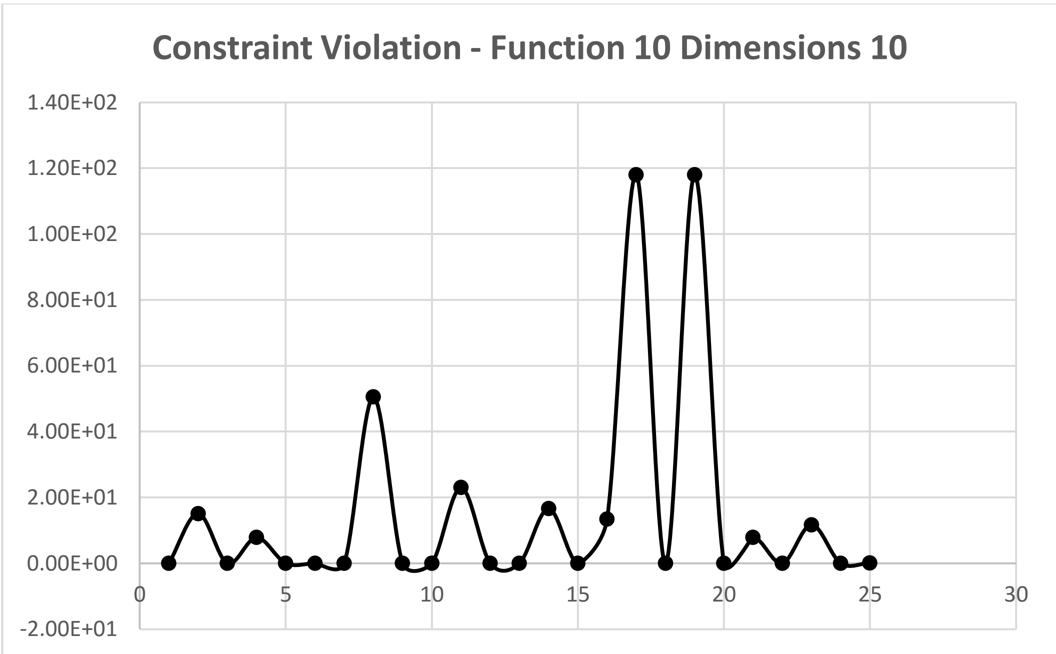
Στις συναρτήσεις στις οποίες έχουμε παραβιάσεις περιορισμών θα παρουσιάζονται τα διαγράμματα παραβίασης περιορισμών για τα 25 τρεξίματα, μιας και αυτό είναι σε αυτή την περίπτωση το κύριο πρόβλημα, και στις συναρτήσεις οι οποίες δεν παρουσιάζουν παραβιάσεις περιορισμών ή παρουσιάζουν πολύ μικρές, θα παρουσιάζονται τα διαγράμματα αντικειμενικής συνάρτησης για τα 25 τρεξίματα.

5.1.1 **Συνάρτηση 10**

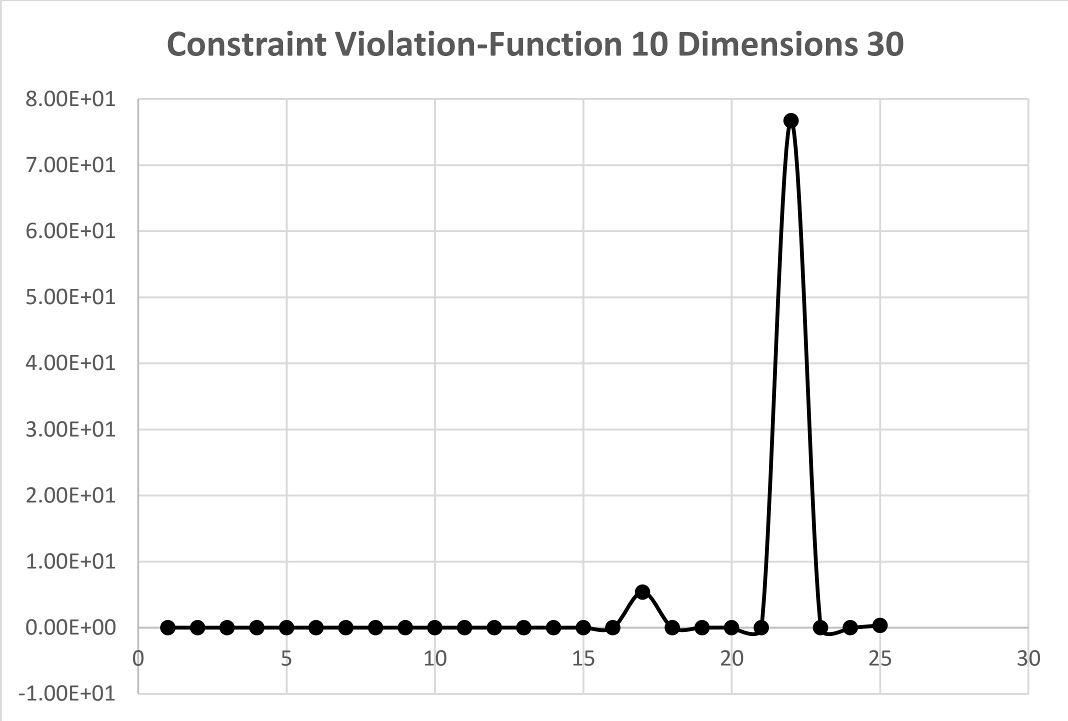
	Function 10			
	10 Dimensions		30 Dimensions	
Run	Obj	CV	Obj	CV
1	5.86E-02	2.12E-03	9.98E+01	5.46E-03
2	5.69E+01	1.51E+01	1.43E+04	1.91E-03
3	1.03E+02	9.26E-03	5.57E+03	5.11E-04

4	2.16E+01	7.83E+00	2.64E+05	5.09E-03
5	9.67E+02	9.25E-03	1.95E+02	4.66E-03
6	7.13E-02	4.21E-03	7.05E+01	3.63E-03
7	8.34E-02	9.29E-03	6.40E+02	8.33E-03
8	4.25E+02	5.05E+01	7.60E+01	0.00E+00
9	4.76E+00	8.47E-03	1.00E+03	3.13E-03
10	1.18E+04	9.71E-03	2.02E+04	1.91E-03
11	1.04E+02	2.30E+01	2.77E+02	4.86E-03
12	2.72E+06	2.89E-03	8.06E+01	4.80E-03
13	6.30E-02	2.47E-04	4.60E+01	7.22E-03
14	6.64E+01	1.66E+01	3.20E+02	2.03E-03
15	5.63E-02	5.91E-04	2.94E+02	3.81E-03
16	9.92E+01	1.34E+01	5.24E+01	2.17E-03
17	1.38E+02	1.18E+02	4.90E+02	5.39E+00
18	3.26E-02	6.22E-04	3.01E+02	5.27E-03
19	1.38E+02	1.18E+02	1.50E+02	7.58E-03
20	6.22E-02	5.38E-03	1.99E+03	7.64E-03
21	2.33E+01	7.82E+00	9.96E+01	1.78E-03
22	4.37E-02	2.51E-03	6.88E+02	7.67E+01
23	4.63E+01	1.17E+01	5.73E+02	9.94E-03
24	4.28E-02	4.02E-03	3.20E+04	9.75E-03
25	6.31E-02	1.32E-01	3.57E+02	3.43E-01

Πίνακας 5.1: Αποτελέσματα SLPSO, Συνάρτηση 10



Σχήμα 5.1: Διάγραμμα παραβίασης περιορισμών για την συνάρτηση 10 σε 10 διαστάσεις για τα 25 τρεξίματα.



Σχήμα 5.2: Διάγραμμα παραβίασης περιορισμών για την συνάρτηση 10 σε 30 διαστάσεις για τα 25 τρεξίματα.

	Function 10			
	10 Dimensions		30 Dimensions	
	SLPSO	εDEag	SLPSO	εDEag
Best Obj	-	0.00E+00	-	3.25E+01
Median Obj	-	0.00E+00	-	3.33E+01
Worst Obj	-	0.00E+00	-	3.46E+01
Mean Obj	-	0.00E+00	-	3.33E+01
Std Obj	-	0.00E+00	-	4.55E-01
Mean CV	1.53E+01	0.00E+00	3.30E+00	0.00E+00

Πίνακας 5.2: Σύγκριση SLPSO και εDEag, Συνάρτηση 10

	Function 10	
	10 D	30 D
Median CV	0.009288	0.004862

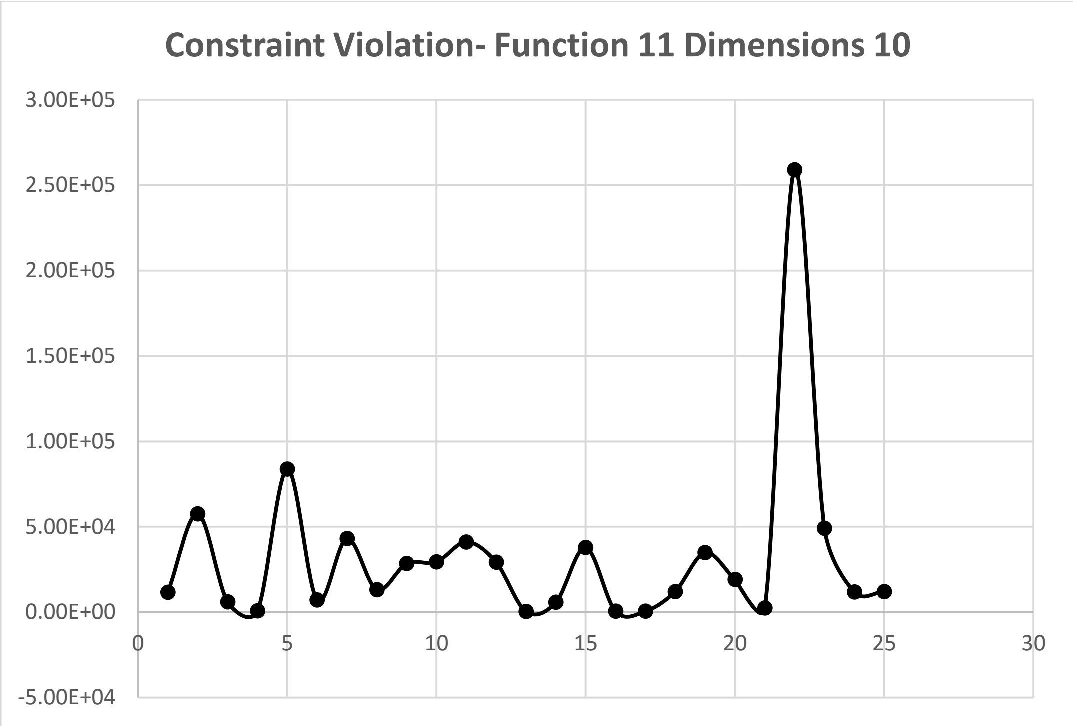
Πίνακας 5.3: Διάμεση τιμή παραβίασης περιορισμών SLPSO, Συνάρτηση 10

Παρατηρήσεις: Από τα αποτελέσματα φαίνεται πως ο αλγόριθμος εDEag απέδωσε πολύ καλύτερα από τον SLPSO. Μάλιστα στο πρόβλημα με τις 10 διαστάσεις ο SLPSO δεν μπόρεσε να βρει ούτε μια λύση εφικτή. Παρόλα αυτά, όπως βλέπουμε από την μέση παραβίαση των περιορισμών από τον SLPSO, ο αλγόριθμος είναι κοντά στην εφικτή περιοχή. Όμως από την άλλη, όπως φαίνεται από τον μέσο όρο των παραβιάσεων, έχουμε κάποιες λύσεις οι οποίες είναι πολύ κακές. Υπάρχει γενικά μεγάλη διακύμανση στην παραβίαση των λύσεων.

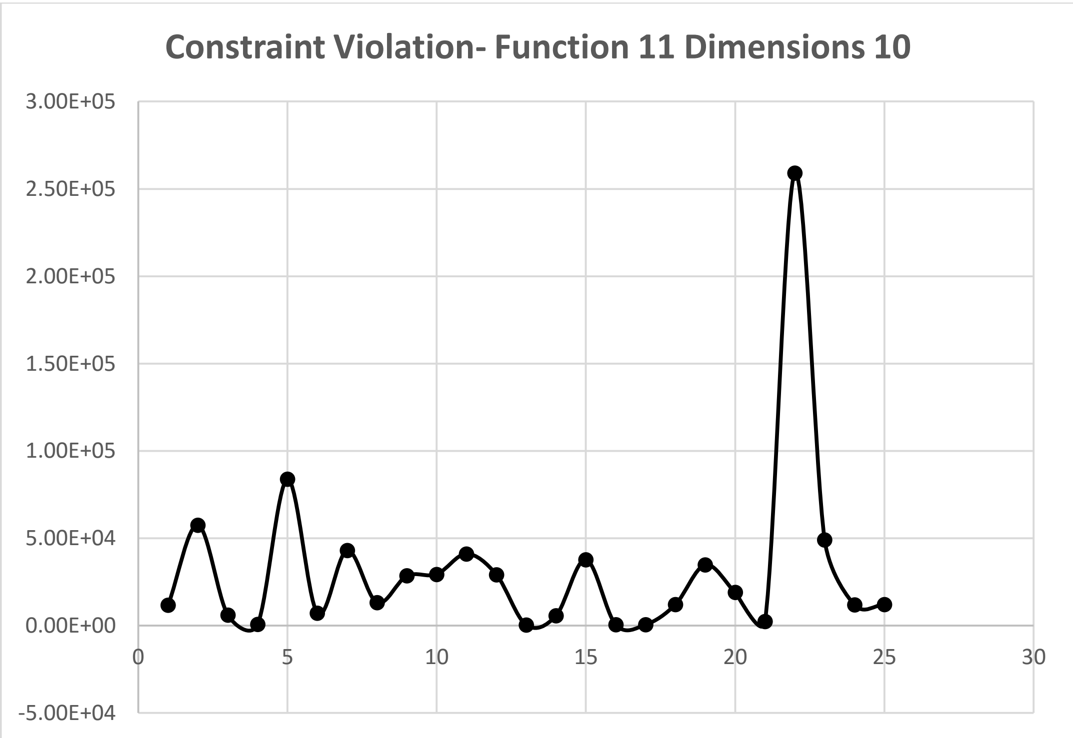
5.1.2 **Συνάρτηση 11**

	Function 11			
	10 Dimensions		30 Dimensions	
Run	Obj	CV	Obj	CV
1	-1.12E+01	1.17E+04	-2.59E-02	5.00E+02
2	9.73E+00	5.76E+04	2.97E-02	5.87E+02
3	-8.31E-01	6.12E+03	2.05E-03	5.62E+02
4	6.91E-03	7.03E+02	2.82E-02	6.09E+02
5	-2.20E+00	8.38E+04	-7.68E-03	5.66E+02
6	-7.01E-01	7.21E+03	-3.06E-02	5.29E+02
7	-1.18E-01	4.31E+04	-1.48E-02	5.83E+02
8	-1.36E+00	1.32E+04	-9.69E-02	5.09E+02
9	9.66E-02	2.85E+04	-1.67E-02	5.04E+02
10	-4.38E-01	2.94E+04	4.85E-02	6.17E+02
11	-2.47E-01	4.11E+04	4.45E-03	5.96E+02
12	-4.47E-01	2.92E+04	-1.87E-02	5.66E+02
13	-1.14E-02	3.92E+02	-2.92E-02	5.43E+02
14	-4.78E-01	5.76E+03	4.85E-02	6.17E+02
15	-1.81E+00	3.78E+04	-2.73E-02	5.60E+02
16	-8.19E-03	5.37E+02	1.83E-02	5.97E+02
17	-2.17E-02	5.46E+02	4.85E-02	6.17E+02
18	9.13E-01	1.21E+04	2.24E-02	6.15E+02
19	-1.01E+00	3.48E+04	-5.04E-02	5.34E+02
20	8.85E-01	1.91E+04	-2.35E-02	5.30E+02
21	3.04E-01	2.39E+03	4.85E-02	6.17E+02
22	3.98E+00	2.59E+05	3.53E-02	6.08E+02
23	5.01E-02	4.91E+04	-8.07E-03	5.50E+02
24	-2.74E-01	1.19E+04	1.30E-02	5.88E+02
25	1.08E+00	1.21E+04	-4.74E-02	5.61E+02

Πίνακας 5.4: Αποτελέσματα SLPSO, Συνάρτηση 11



Σχήμα 5.3: Διάγραμμα παραβίασης περιορισμών για την συνάρτηση 11 σε 10 διαστάσεις για τα 25 τρεξίματα.



Σχήμα 5.4: Διάγραμμα παραβίασης περιορισμών για την συνάρτηση 11 σε 30 διαστάσεις για τα 25 τρεξίματα.

	Function 11			
	10 Dimensions		30 Dimensions	
	SLPSO	εDEag	SLPSO	εDEag
Best Obj	-	-1.52E-03	-	-3.27E-04
Median Obj	-	-1.52E-03	-	-2.84E-04
Worst Obj	-	-1.52E-03	-	-2.24E-04
Mean Obj	-	0.00E+00	-	-2.86E-04
Std Obj	-	0.00E+00	-	2.71E-05
Mean CV	3.19E+04	0.00E+00	3.66E-02	0.00E+00

Πίνακας 5.5: Σύγκριση SLPSO και εDEag, Συνάρτηση 11

	Function 11	
	10 D	30 D
Median CV	13228.92	5.66E+02

Πίνακας 5.6: Διάμεση τιμή παραβίασης περιορισμών SLPSO, Συνάρτηση 11

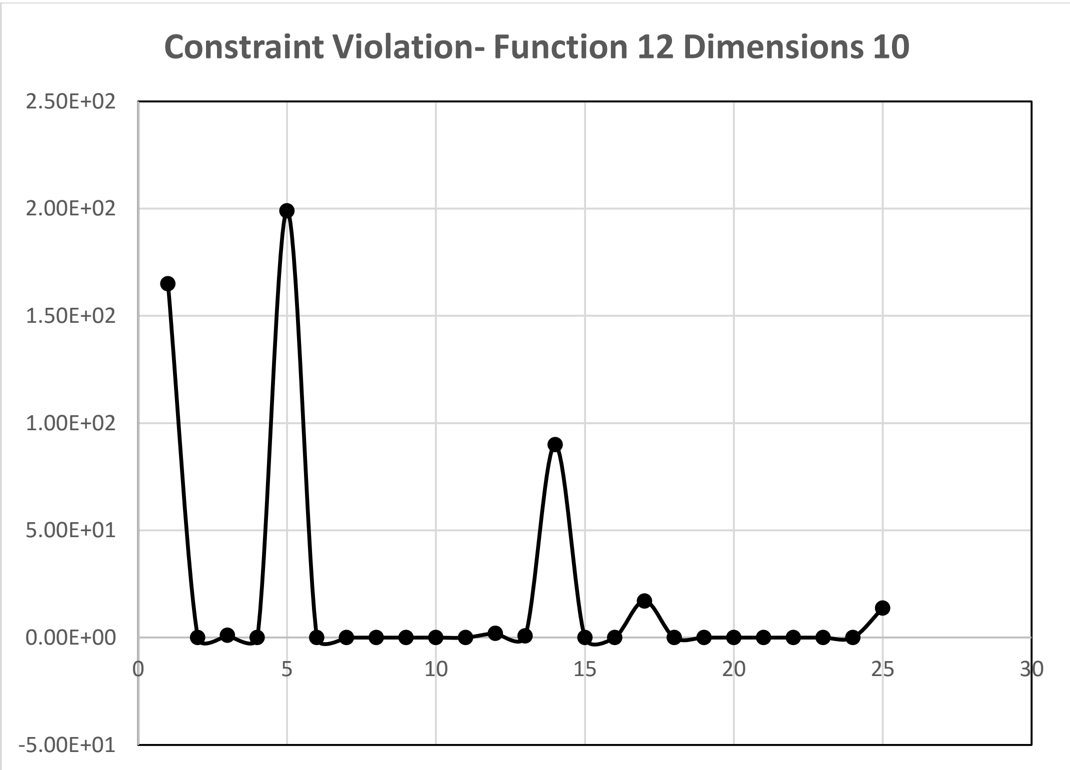
Παρατηρήσεις: Όπως βλέπουμε στα αποτελέσματα ο αλγόριθμος SLPSO δεν μπόρεσε να βρει εφικτή λύση ούτε στις 10 ούτε στις 30 διαστάσεις. Επίσης φαίνεται και από τον μέσο όρο και από τον μέσο των παραβιάσεων ότι βρισκόμαστε στην μη εφικτή περιοχή και ότι ο αλγόριθμος δεν μπορεί να κατευθυνθεί σωστά. Στην εργασία των Liang, Zhigang και Zhihui οι οποίοι ανέπτυξαν επίσης μια παραλλαγή του PSO και συμμετείχαν στον διαγωνισμό CEC 2010 (Liang et al., 2010), ο αλγόριθμος τους επίσης δεν μπόρεσε να βρει καμία εφικτή λύση σε αυτή την συνάρτηση, παρόλο που ο αλγόριθμος ήταν αποτελεσματικός στις υπόλοιπες συναρτήσεις.

5.1.3 **Συνάρτηση 12**

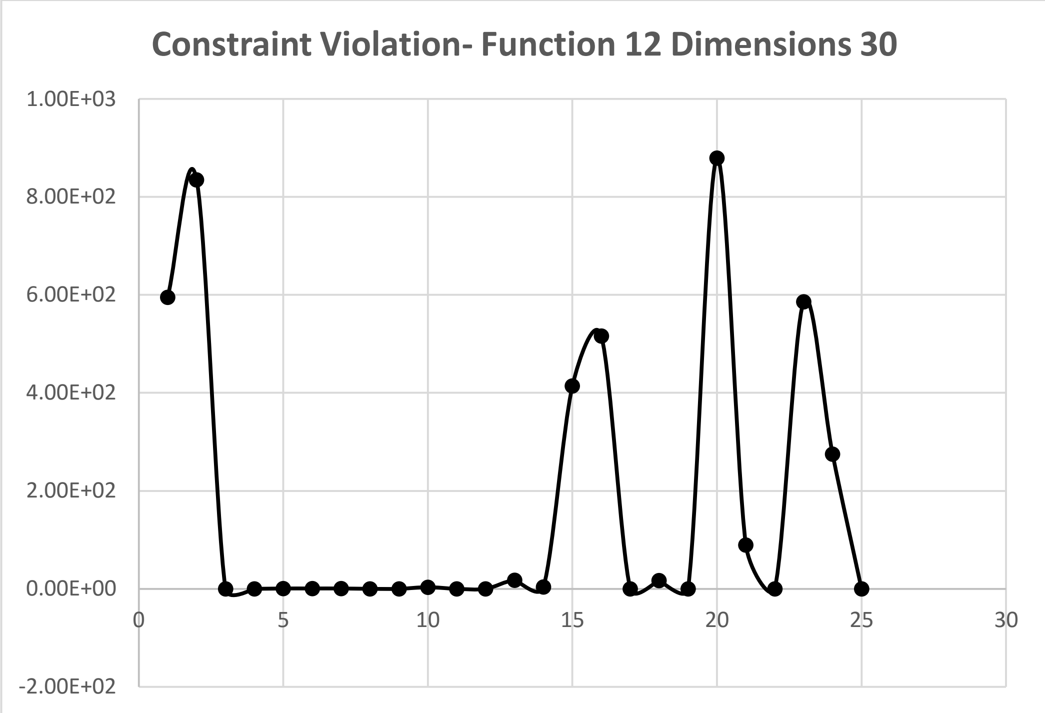
	Function 12			
	10 Dimensions		30 Dimensions	
Run	Obj	CV	Obj	CV
1	-2.72E+02	1.65E+02	-3.78E+02	5.95E+02
2	-2.78E-01	9.97E-03	-4.61E+02	8.35E+02
3	-3.08E+02	1.15E+00	-2.47E-01	3.18E-03
4	-1.28E+01	9.86E-03	-2.48E-01	3.23E-03
5	2.07E+02	1.99E+02	-1.10E+02	7.12E-01

6	-2.84E-01	9.99E-03	-3.28E+02	7.30E-01
7	-1.16E+02	8.97E-03	-1.19E+02	7.23E-01
8	-2.82E-01	9.81E-03	-2.47E-01	3.10E-03
9	-1.16E+02	9.72E-03	-2.45E-01	2.97E-03
10	-2.76E-01	8.72E-03	1.01E+02	3.39E+00
11	-2.73E-01	8.86E-03	-2.46E-01	3.01E-03
12	-1.93E+01	1.98E+00	-2.43E-01	2.79E-03
13	-1.18E+02	7.59E-01	-3.19E+02	1.72E+01
14	-7.40E+00	9.00E+01	-3.05E+02	3.89E+00
15	-3.02E+02	8.99E-03	4.33E+02	4.14E+02
16	-1.16E+02	9.77E-03	-3.22E+02	5.16E+02
17	-3.19E+02	1.71E+01	-2.46E-01	3.05E-03
18	-2.82E-01	9.38E-03	-2.94E+02	1.67E+01
19	-2.78E-01	9.51E-03	-2.47E-01	3.16E-03
20	-2.80E-01	9.89E-03	-8.42E+02	8.79E+02
21	-2.79E-01	9.62E-03	-5.66E+00	8.96E+01
22	-2.77E-01	9.12E-03	-2.46E-01	3.02E-03
23	-1.16E+02	9.35E-03	-4.45E+02	5.86E+02
24	-1.28E+01	9.65E-03	-1.99E+02	2.75E+02
25	-1.14E+01	1.37E+01	-2.46E-01	3.07E-03

Πίνακας 5.7: Αποτελέσματα SLPSO, Συνάρτηση 12



Σχήμα 5.5: Διάγραμμα παραβίασης περιορισμών για την συνάρτηση 12 σε 10 διαστάσεις για τα 25 τρεξίματα.



Σχήμα 5.6: Διάγραμμα παραβίασης περιορισμών για την συνάρτηση 12 σε 30 διαστάσεις για τα 25 τρεξίματα.

	Function 12			
	10 Dimensions		30 Dimensions	
	SLPSO	εDEag	SLPSO	εDEag
Best Obj	-	-5.70E+02	-	-1.99E-01
Median Obj	-	-4.23E+02	-	5.34E+02
Worst Obj	-	-1.99E-01	-	5.46E+00
Mean Obj	-	-3.37E+02	-	3.56E+02
Std Obj	-	1.78E+02	-	2.89E+02
Mean CV	19.55362	0.00E+00	3.66E-02	3.24E-01

Πίνακας 5.8: Σύγκριση SLPSO και εDEag, Συνάρτηση 12

	Function 12	
	10 D	30 D
Median CV	0.009809	7.30E-01

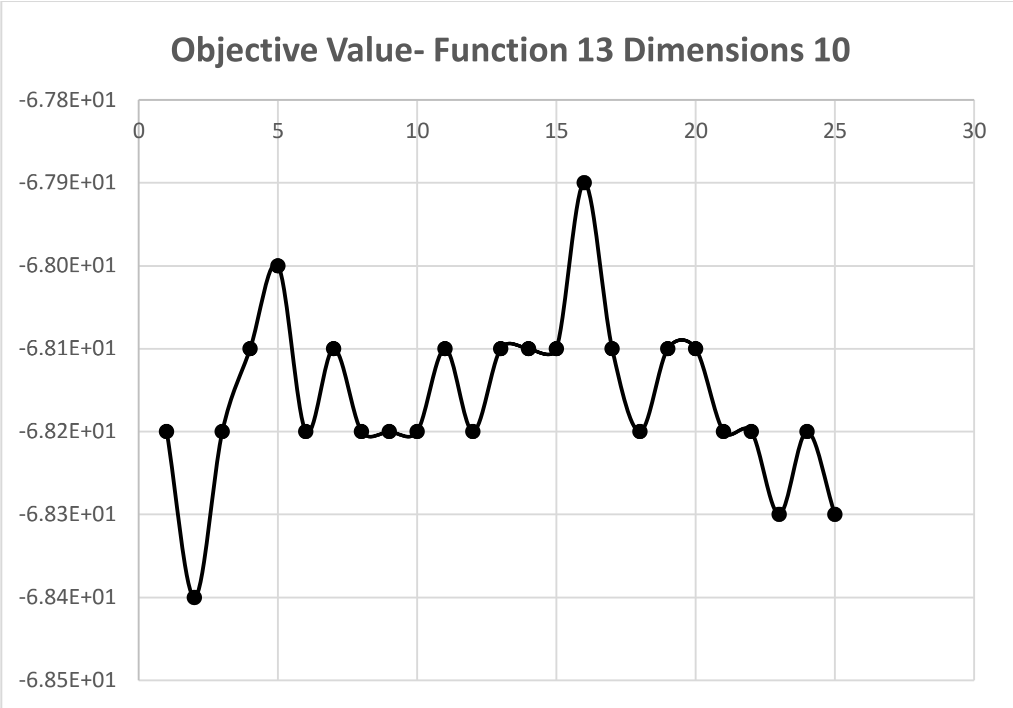
Πίνακας 5.9: Διάμεση τιμή παραβίασης περιορισμών SLPSO, Συνάρτηση 12

Παρατηρήσεις: Στην συνάρτηση 12 βλέπουμε ότι ο SLPSO δεν πετυχαίνει καλά αποτελέσματα. Καμία από τις λύσεις τις οποίες βρίσκει δεν είναι εφικτή. Στις 10 διαστάσεις η παραβίαση των περιορισμών είναι σχετικά μικρή, βλέπουμε ότι η μέση είναι της τάξεως του 1.00e-02, ενώ στις 30 διαστάσεις έχουμε υψηλότερες. Βρίσκονται κάποιες λύσεις κοντά στην εφικτή περιοχή όμως και πολλές μακριά από αυτήν.

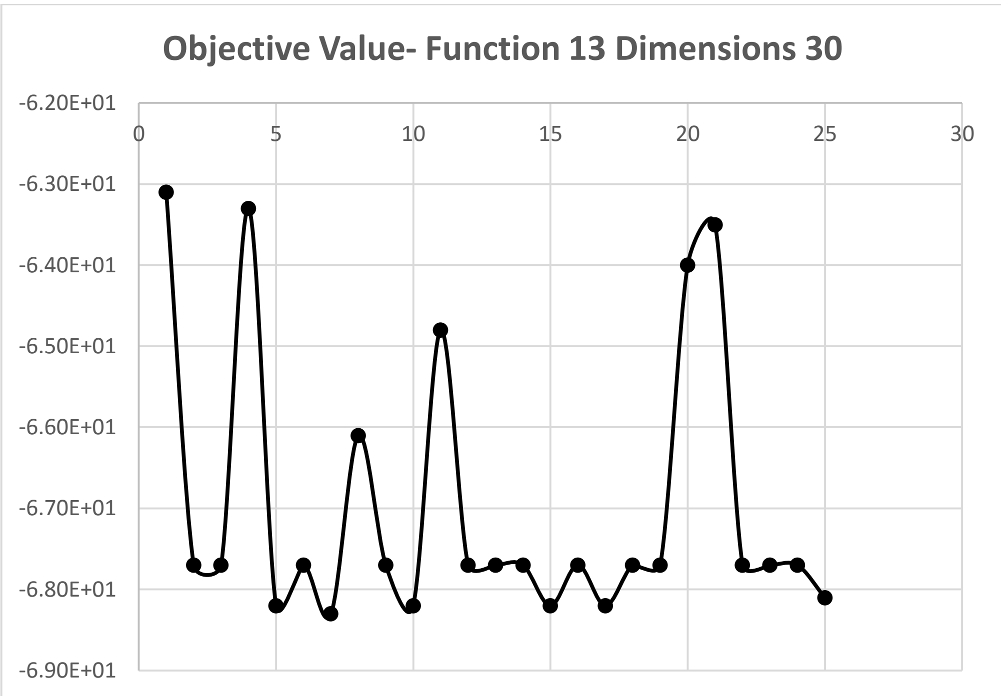
5.1.4 **Συνάρτηση 13**

	Function 13			
	10 Dimensions		30 Dimensions	
Run	Obj	CV	Obj	CV
1	-6.82E+01	0.00E+00	-6.31E+01	0.00E+00
2	-6.84E+01	0.00E+00	-6.77E+01	0.00E+00
3	-6.82E+01	0.00E+00	-6.77E+01	0.00E+00
4	-6.81E+01	0.00E+00	-6.33E+01	1.60E-03
5	-6.80E+01	0.00E+00	-6.82E+01	0.00E+00
6	-6.82E+01	0.00E+00	-6.77E+01	0.00E+00
7	-6.81E+01	0.00E+00	-6.83E+01	0.00E+00
8	-6.82E+01	0.00E+00	-6.61E+01	7.29E-01
9	-6.82E+01	0.00E+00	-6.77E+01	0.00E+00
10	-6.82E+01	0.00E+00	-6.82E+01	0.00E+00
11	-6.81E+01	0.00E+00	-6.48E+01	7.17E-04
12	-6.82E+01	0.00E+00	-6.77E+01	0.00E+00
13	-6.81E+01	0.00E+00	-6.77E+01	0.00E+00
14	-6.81E+01	0.00E+00	-6.77E+01	0.00E+00
15	-6.81E+01	0.00E+00	-6.82E+01	0.00E+00
16	-6.79E+01	0.00E+00	-6.77E+01	9.26E-04
17	-6.81E+01	0.00E+00	-6.82E+01	0.00E+00
18	-6.82E+01	0.00E+00	-6.77E+01	9.02E-03
19	-6.81E+01	0.00E+00	-6.77E+01	5.60E-03
20	-6.81E+01	0.00E+00	-6.40E+01	1.60E-03
21	-6.82E+01	0.00E+00	-6.35E+01	0.00E+00
22	-6.82E+01	0.00E+00	-6.77E+01	2.16E-03
23	-6.83E+01	0.00E+00	-6.77E+01	0.00E+00
24	-6.82E+01	0.00E+00	-6.77E+01	0.00E+00
25	-6.83E+01	0.00E+00	-6.81E+01	0.00E+00

Πίνακας 5.10: Αποτελέσματα SLPSO, Συνάρτηση 13



Σχήμα 5.7: Διάγραμμα αντικειμενικής συνάρτησης για την συνάρτηση 13 σε 10 διαστάσεις για τα 25 τρεξίματα.



Σχήμα 5.8: Διάγραμμα αντικειμενικής συνάρτησης για την συνάρτηση 13 σε 30 διαστάσεις για τα 25 τρεξίματα.

	Function 13			
	10 Dimensions		30 Dimensions	
	SLPSO	εDEag	SLPSO	εDEag
Best Obj	-6.84E+01	-6.84E+01	-6.83E+01	-6.64E+01
Median Obj	-6.82E+01	-6.84E+01	-6.77E+01	-6.53E+01
Worst Obj	-6.79E+01	-6.84E+01	-6.31E+01	-6.43E+01
Mean Obj	-6.82E+01	-6.84E+01	6.70E+01	-6.54E+01
Std Obj	9.61E-02	1.03E-06	1.71E+00	5.73E-01
Mean CV	0.00E+00	0.00E+00	3.00E-02	0.00E+00

Πίνακας 5.11: Σύγκριση SLPSO και εDEag, Συνάρτηση 13

	Function 13	
	10 D	30 D
Median CV	0.00E+00	0.00E+00

Πίνακας 5.12: Διάμεση τιμή παραβίασης περιορισμών SLPSO, Συνάρτηση 13

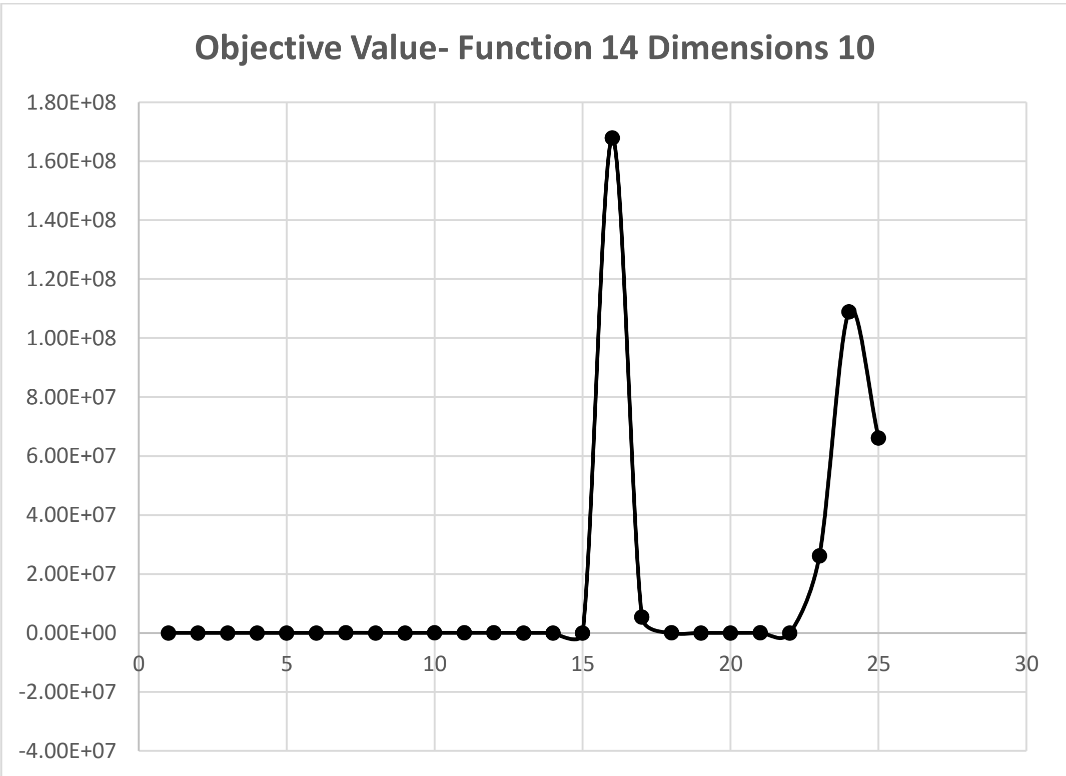
Παρατηρήσεις: Στη συνάρτηση 13 βλέπουμε ότι ο SLPSO είχε πολύ καλά αποτελέσματα περίπου στο ίδιο επίπεδο με αυτά του εDEag. Στις 10 διαστάσεις δεν έχουμε καμία παραβίαση περιορισμών ενώ στις 30 διαστάσεις έχουμε μερικές παραβιάσεις σε κάποια τρεξίματα αλλά πολύ μικρές.

5.1.5 **Συνάρτηση 14**

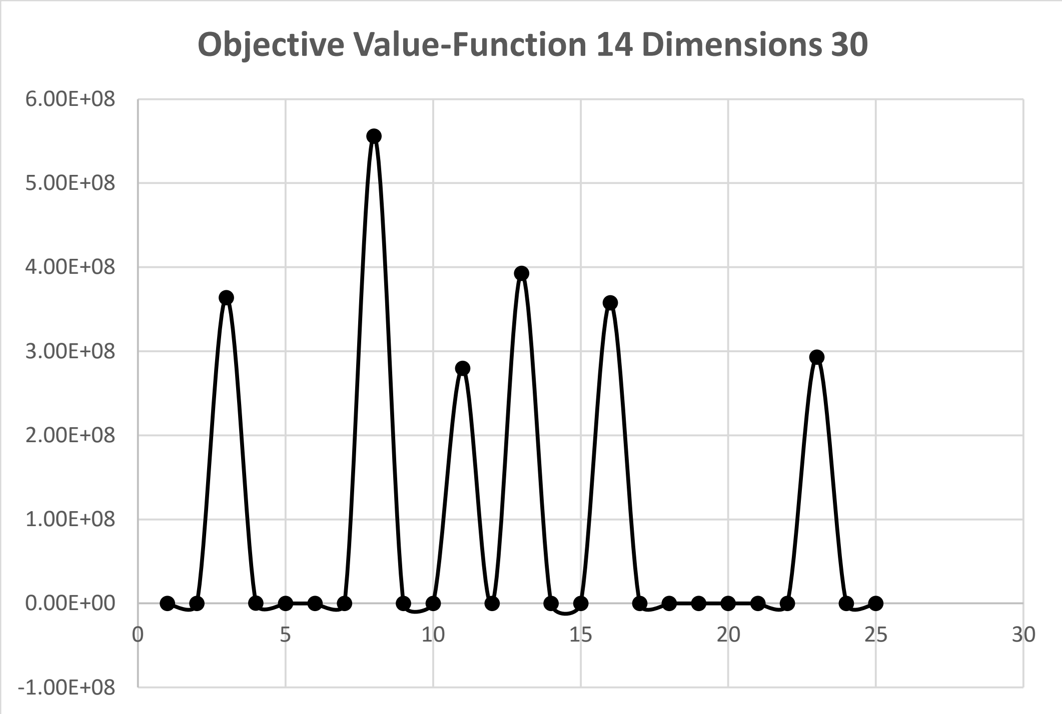
	Function 14			
	10 Dimensions		30 Dimensions	
Run	Obj	CV	Obj	CV
1	4.88E-04	0.00E+00	1.92E+01	0.00E+00
2	1.98E+02	0.00E+00	3.00E+04	0.00E+00
3	4.74E-04	0.00E+00	3.64E+08	0.00E+00
4	3.20E+02	0.00E+00	6.38E+05	0.00E+00
5	2.73E-04	0.00E+00	4.58E+02	0.00E+00
6	2.35E-04	0.00E+00	1.64E+01	0.00E+00
7	3.01E+04	0.00E+00	9.02E+04	0.00E+00
8	6.18E+03	0.00E+00	5.56E+08	0.00E+00

9	4.94E+01	0.00E+00	5.71E+02	0.00E+00
10	1.19E+04	0.00E+00	7.69E+02	0.00E+00
11	1.19E+04	0.00E+00	2.80E+08	0.00E+00
12	1.19E+04	6.06E-03	5.03E+03	0.00E+00
13	2.53E-04	0.00E+00	3.93E+08	0.00E+00
14	3.21E-04	0.00E+00	3.00E+04	0.00E+00
15	4.23E+02	0.00E+00	5.22E+02	0.00E+00
16	1.68E+08	0.00E+00	3.58E+08	0.00E+00
17	5.36E+06	0.00E+00	1.98E+04	0.00E+00
18	5.62E+04	0.00E+00	1.62E+01	0.00E+00
19	4.95E+01	0.00E+00	6.77E+03	0.00E+00
20	3.05E+02	0.00E+00	2.15E+02	0.00E+00
21	1.19E+04	0.00E+00	1.79E+01	0.00E+00
22	2.72E-04	0.00E+00	1.63E+02	0.00E+00
23	2.61E+07	0.00E+00	2.93E+08	0.00E+00
24	1.09E+08	0.00E+00	1.28E+02	0.00E+00
25	6.61E+07	0.00E+00	1.78E+01	0.00E+00

Πίνακας 5.13: Αποτελέσματα SLPSO, Συνάρτηση 14



Σχήμα 5.9: Διάγραμμα αντικειμενικής συνάρτησης για την συνάρτηση 14 σε 10 διαστάσεις για τα 25 τρεξίματα.



Σχήμα 5.10: Διάγραμμα αντικειμενικής συνάρτησης για την συνάρτηση 14 σε 30 διαστάσεις για τα 25 τρεξίματα.

	Function 14			
	10 Dimensions		30 Dimensions	
	SLPSO	εDEag	SLPSO	εDEag
Best Obj	2.35E-04	0.00E+00	1.62E+01	5.02E-14
Median Obj	4.23E+02	0.00E+00	5.03E+03	1.36E-13
Worst Obj	1.68E+08	0.00E+00	5.56E+08	2.92E-12
Mean Obj	1.50E+07	0.00E+00	8.97E+07	3.09E-13
Std Obj	4.06E+07	0.00E+00	1.69E+08	5.61E-13
Mean CV	2.43E-04	0.00E+00	0.00E+00	0.00E+00

Πίνακας 5.14: Σύγκριση SLPSO και εDEag, Συνάρτηση 14

	Function 14	
	10 D	30 D
Median CV	0.00E+00	0.00E+00

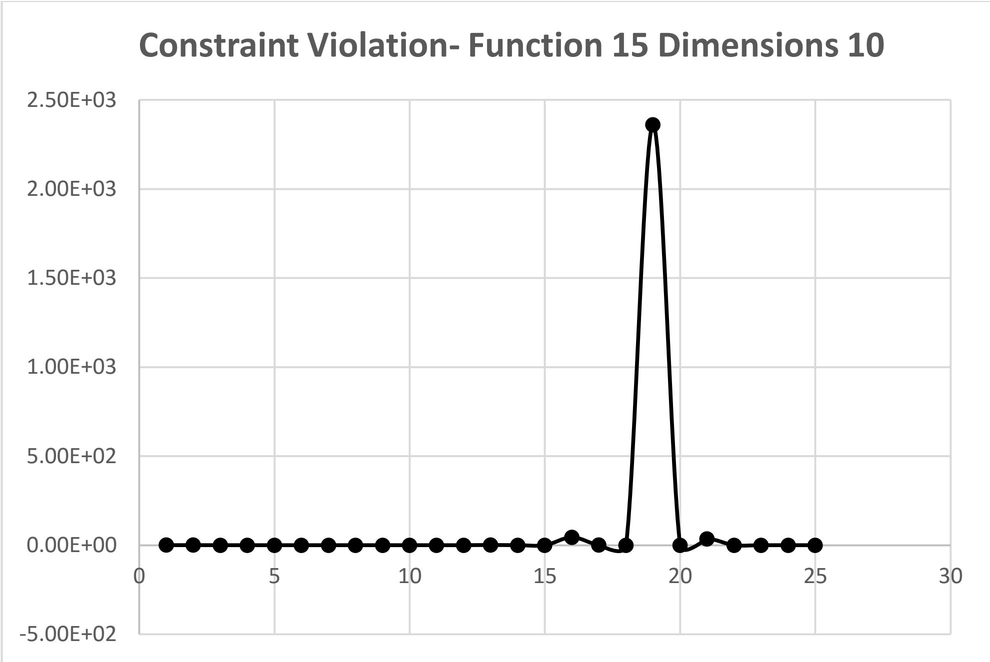
Πίνακας 5.15: Διάμεση τιμή παραβίασης Περιορισμών SLPSO, Συνάρτηση 14

Παρατηρήσεις: Ο SLPSO βρίσκει λύσεις μέσα στην εφικτή περιοχή και δεν παραβιάζει τους περιορισμούς σχεδόν πουθενά. Οι λύσεις όμως παρουσιάζουν τεράστια διακύμανση, και επίσης είναι πολύ χειρότερες από αυτές που βρίσκει ο αλγόριθμος eDEag.

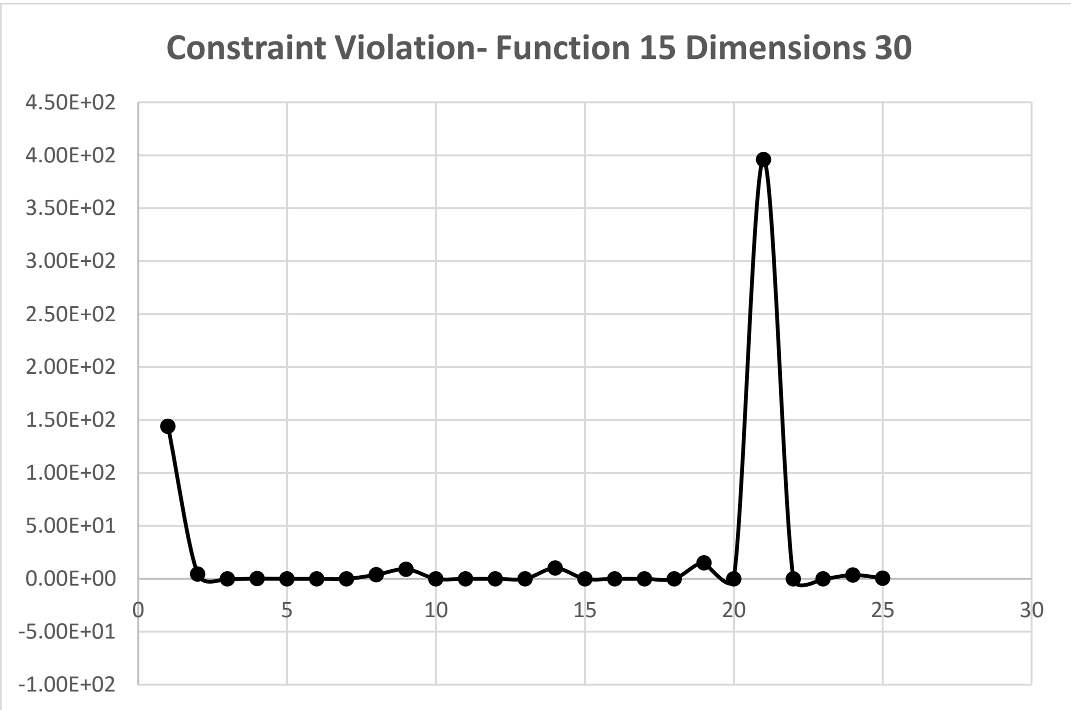
5.1.6 **Συνάρτηση 15**

	Function 15			
	10 Dimensions		30 Dimensions	
Run	Obj	CV	Obj	CV
1	4.11E+02	3.49E-02	1.89E+05	1.44E+02
2	2.92E+04	2.85E-01	2.07E+05	4.49E+00
3	1.18E+03	1.78E-03	1.60E+05	1.54E-03
4	1.36E+03	2.28E-04	1.49E+04	2.30E-01
5	3.16E+03	8.76E-03	3.18E+04	4.15E-04
6	6.94E+04	6.65E-03	6.06E+05	2.14E-03
7	9.30E+02	1.38E-04	2.42E+04	2.32E-04
8	4.11E+03	4.64E-03	1.41E+04	3.86E+00
9	1.33E+04	9.13E-03	2.14E+05	9.07E+00
10	7.37E+02	5.63E-03	7.85E+05	3.08E-03
11	1.12E+05	3.86E-03	1.48E+05	3.35E-03
12	3.24E+03	2.98E-03	4.15E+04	1.31E-03
13	2.41E+04	1.02E+00	2.60E+05	5.09E-03
14	7.52E+02	7.23E-03	3.34E+04	1.01E+01
15	1.47E+04	2.04E-03	9.93E+03	5.35E-04
16	2.01E+04	4.44E+01	7.98E+05	6.18E-03
17	9.92E+01	4.85E-01	3.50E+05	5.02E-03
18	7.14E+04	8.16E-03	1.00E+05	2.90E-03
19	8.15E+04	2.36E+03	3.47E+05	1.51E+01
20	1.48E+04	6.96E-03	1.43E+05	8.57E-04
21	2.81E+04	3.57E+01	1.22E+05	3.96E+02
22	4.22E+02	3.52E-03	9.61E+05	8.71E-03
23	4.97E+03	9.95E-03	3.71E+05	7.11E-03
24	6.42E+02	7.65E-03	1.54E+04	3.59E+00
25	1.01E+05	1.91E-03	7.52E+04	6.53E-01

Πίνακας 5.16: Αποτελέσματα SLPSO, Συνάρτηση 15



Σχήμα 5.11: Διάγραμμα παραβίασης περιορισμών για την συνάρτηση 15 σε 10 διαστάσεις για τα 25 τρεξίματα.



Σχήμα 5.12: Διάγραμμα παραβίασης περιορισμών για την συνάρτηση 15 σε 30 διαστάσεις για τα 25 τρεξίματα.

	Function 15			
	10 Dimensions		30 Dimensions	
	SLPSO	εDEag	SLPSO	εDEag
Best Obj	-	0.00E+00	-	2.16E+01
Median Obj	-	0.00E+00	-	2.16E+01
Worst Obj	-	4.50E+00	-	2.16E+01
Mean Obj	-	1.80E-01	-	2.16E+01
Std Obj	-	8.81E-01	-	1.10E-04
Mean CV	9.76E+01	0.00E+00	2.35E+01	0.00E+00

Πίνακας 5.17: Σύγκριση SLPSO και εDEag, Συνάρτηση 15

	Function 15	
	10 D	30 D
Median CV	7.23E-03	6.18E-03

Πίνακας 5.18: Διάμεση τιμή παραβίασης περιορισμών SLPSO, Συνάρτηση 15

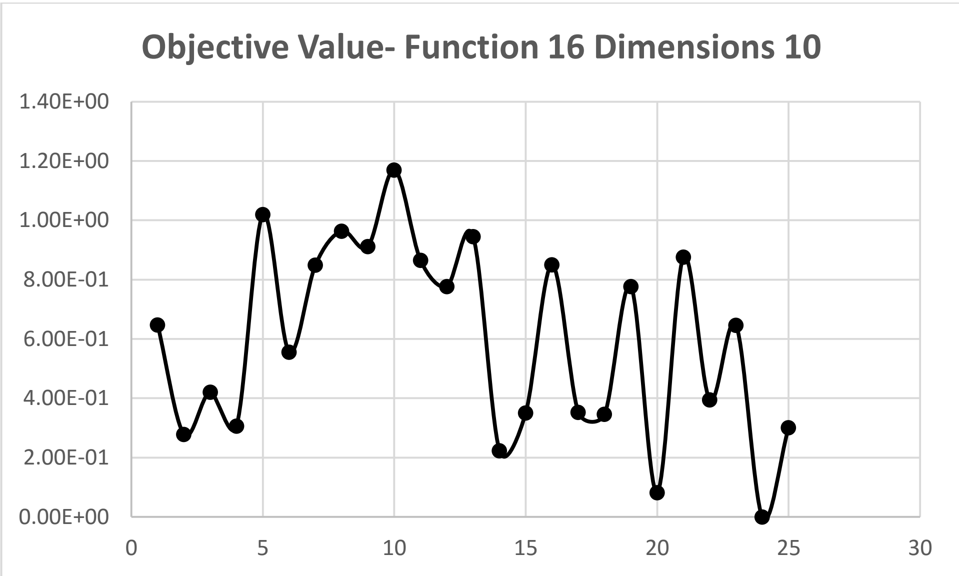
Παρατηρήσεις: Στα αποτελέσματα βλέπουμε ότι ο SLPSO δεν βρίσκει εφικτές λύσεις. Παρόλα αυτά, στον αριθμητικό μέσο της παραβίασης των περιορισμών και στα αναλυτικά αποτελέσματα των 25 τρεξιμάτων βλέπουμε ότι οι παραβιάσεις, με λίγες εξαιρέσεις οι οποίες είναι υψηλές, είναι σχετικά μικρές.

5.1.7 **Συνάρτηση 16**

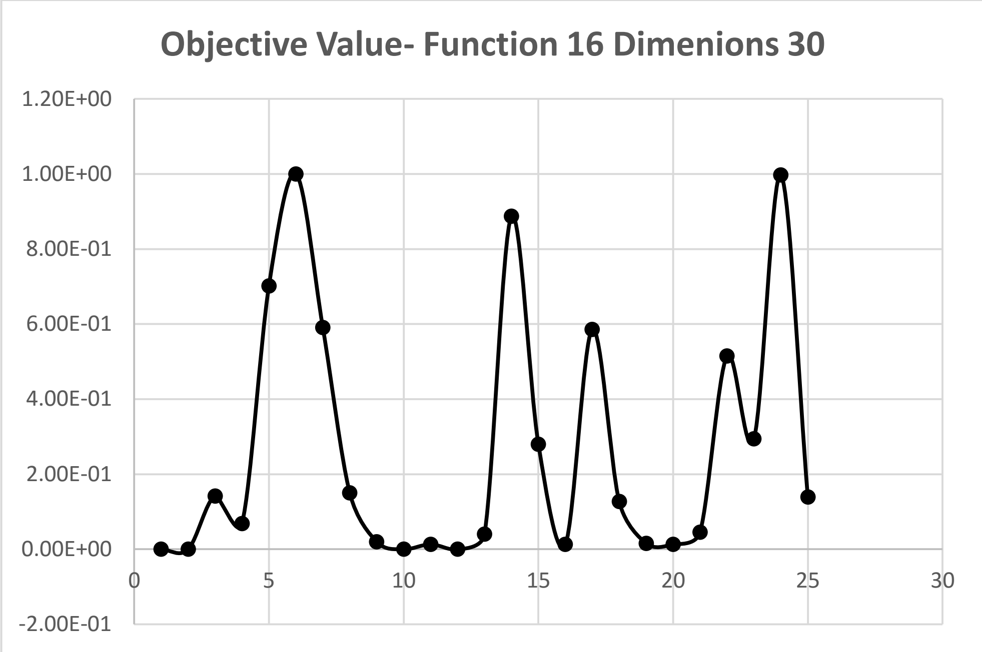
	Function 16			
	10 Dimensions		30 Dimensions	
Run	Obj	CV	Obj	CV
1	6.47E-01	0.00E+00	3.44E-07	0.00E+00
2	2.79E-01	0.00E+00	2.70E-07	0.00E+00
3	4.21E-01	0.00E+00	1.42E-01	1.01E-04
4	3.07E-01	0.00E+00	6.89E-02	0.00E+00
5	1.02E+00	0.00E+00	7.02E-01	0.00E+00
6	5.56E-01	0.00E+00	1.00E+00	0.00E+00
7	8.49E-01	0.00E+00	5.91E-01	0.00E+00
8	9.64E-01	0.00E+00	1.50E-01	0.00E+00
9	9.12E-01	0.00E+00	2.01E-02	0.00E+00
10	1.17E+00	0.00E+00	3.46E-07	0.00E+00
11	8.65E-01	0.00E+00	1.32E-02	0.00E+00
12	7.77E-01	0.00E+00	9.61E-07	0.00E+00
13	9.45E-01	0.00E+00	4.00E-02	0.00E+00
14	2.24E-01	0.00E+00	8.87E-01	0.00E+00
15	3.51E-01	0.00E+00	2.80E-01	1.16E-04
16	8.50E-01	0.00E+00	1.30E-02	1.01E-04

17	3.53E-01	0.00E+00	5.86E-01	0.00E+00
18	3.46E-01	0.00E+00	1.27E-01	0.00E+00
19	7.77E-01	0.00E+00	1.60E-02	0.00E+00
20	8.22E-02	0.00E+00	1.32E-02	0.00E+00
21	8.76E-01	0.00E+00	4.56E-02	0.00E+00
22	3.95E-01	0.00E+00	5.15E-01	9.61E-04
23	6.46E-01	0.00E+00	2.94E-01	2.24E-04
24	5.09E-13	0.00E+00	9.97E-01	0.00E+00
25	3.01E-01	0.00E+00	1.39E-01	0.00E+00

Πίνακας 5.19: Αποτελέσματα SLPSO, Συνάρτηση 16



Σχήμα 5.13: Διάγραμμα αντικειμενικής συνάρτησης για την συνάρτηση 16 σε 10 διαστάσεις για τα 25 τρεξίματα.



Σχήμα 5.14: Διάγραμμα αντικειμενικής συνάρτησης για την συνάρτηση 16 σε 30 διαστάσεις για τα 25 τρεξίματα.

	Function 16			
	10 Dimensions		30 Dimensions	
	SLPSO	εDEag	SLPSO	εDEag
Best Obj	5.09E-13	0.00E+00	2.70E-07	0.00E+00
Median Obj	6.46E-01	2.82E-01	1.27E-01	0.00E+00
Worst Obj	1.17E+00	1.02E+00	1.00E+00	5.42E-20
Mean Obj	5.96E-01	3.70E-01	2.66E-01	2.17E-21
Std Obj	3.23E-01	3.71E-01	3.37E-01	1.06E-20
Mean CV	0.00E+00	0.00E+00	6.01E-05	0.00E+00

Πίνακας 5.20: Σύγκριση SLPSO και εDEag, Συνάρτηση 16

	Function 16	
	10 D	30 D
Median CV	0.00E+00	0.00E+00

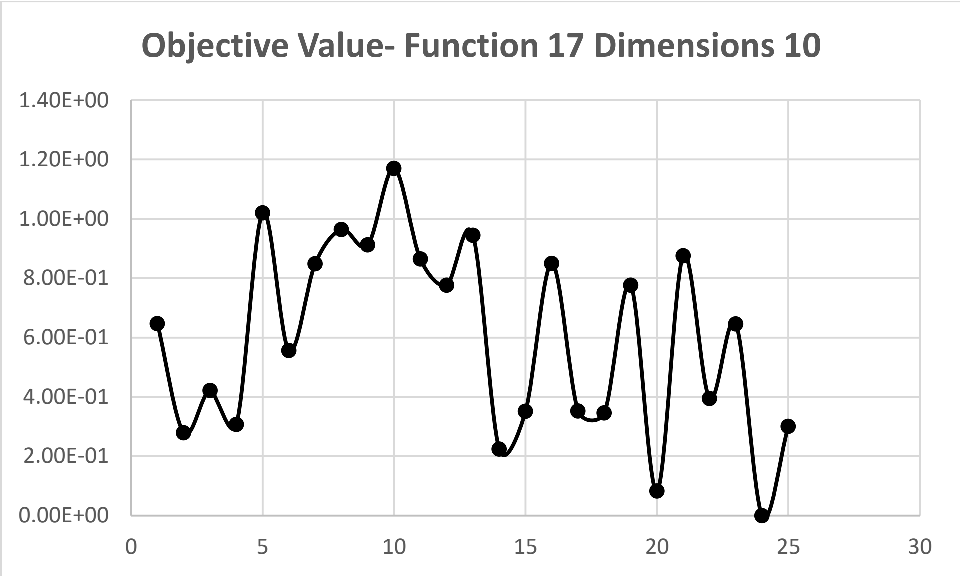
Πίνακας 5.21: Διάμεση τιμή παραβίασης περιορισμών SLPSO, Συνάρτηση 16

Παρατηρήσεις: Στα αποτελέσματα βλέπουμε ότι ο SLPSO καταφέρνει να βρει λύσεις εφικτές και στις 10 και στις 30 διαστάσεις, όμως υπάρχει πολύ μεγάλη διακύμανση στην τιμή της αντικειμενικής συνάρτησης, ενώ βλέπουμε ότι ο αλγόριθμος eDEag καταφέρνει να βρει πολύ καλύτερες λύσεις. Πιθανότητα ο SLPSO να εγκλωβίζεται σε τοπικά βέλτιστα της συνάρτησης.

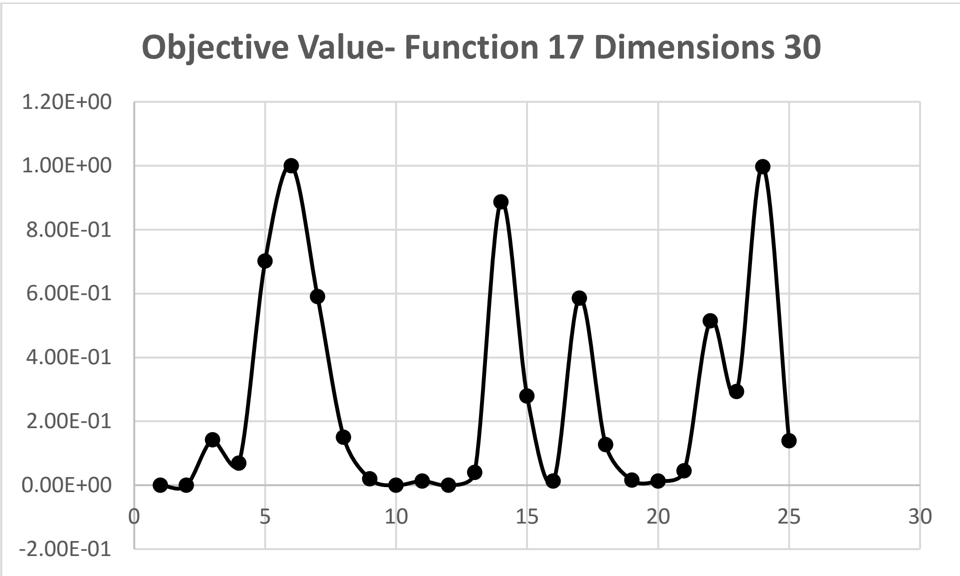
5.1.8 **Συνάρτηση 17**

	Function 17			
	10 Dimensions		30 Dimensions	
Run	Obj	CV	Obj	CV
1	6.47E-01	0.00E+00	3.44E-07	0.00E+00
2	2.79E-01	0.00E+00	2.70E-07	0.00E+00
3	4.21E-01	0.00E+00	1.42E-01	1.01E-04
4	3.07E-01	0.00E+00	6.89E-02	0.00E+00
5	1.02E+00	0.00E+00	7.02E-01	0.00E+00
6	5.56E-01	0.00E+00	1.00E+00	0.00E+00
7	8.49E-01	0.00E+00	5.91E-01	0.00E+00
8	9.64E-01	0.00E+00	1.50E-01	0.00E+00
9	9.12E-01	0.00E+00	2.01E-02	0.00E+00
10	1.17E+00	0.00E+00	3.46E-07	0.00E+00
11	8.65E-01	0.00E+00	1.32E-02	0.00E+00
12	7.77E-01	0.00E+00	9.61E-07	0.00E+00
13	9.45E-01	0.00E+00	4.00E-02	0.00E+00
14	2.24E-01	0.00E+00	8.87E-01	0.00E+00
15	3.51E-01	0.00E+00	2.80E-01	1.16E-04
16	8.50E-01	0.00E+00	1.30E-02	1.01E-04
17	3.53E-01	0.00E+00	5.86E-01	0.00E+00
18	3.46E-01	0.00E+00	1.27E-01	0.00E+00
19	7.77E-01	0.00E+00	1.60E-02	0.00E+00
20	8.22E-02	0.00E+00	1.32E-02	0.00E+00
21	8.76E-01	0.00E+00	4.56E-02	0.00E+00
22	3.95E-01	0.00E+00	5.15E-01	9.61E-04
23	6.46E-01	0.00E+00	2.94E-01	2.24E-04
24	5.09E-13	0.00E+00	9.97E-01	0.00E+00
25	3.01E-01	0.00E+00	1.39E-01	0.00E+00

Πίνακας 5.22: Αποτελέσματα SLPSO, Συνάρτηση 17



Σχήμα 5.15: Διάγραμμα αντικειμενικής συνάρτησης για την συνάρτηση 17 σε 10 διαστάσεις για τα 25 τρεξίματα.



Σχήμα 5.16: Διάγραμμα αντικειμενικής συνάρτησης για την συνάρτηση 17 σε 30 διαστάσεις για τα 25 τρεξίματα.

	Function 17			
	10 Dimensions		30 Dimensions	
	SLPSO	εDEag	SLPSO	εDEag
Best Obj	5.09E-13	1.46E-17	2.70E-07	2.17E-01
Median Obj	6.46E-01	5.65E-03	1.27E-01	5.32E+00
Worst Obj	1.17E+00	7.30E-01	1.00E+00	1.89E+01

Mean Obj	5.96E-01	1.25E-01	2.66E-01	6.33E+00
Std Obj	3.23E-01	1.94E-01	3.37E-01	4.99E+00
Mean CV	0.00E+00	0.00E+00	6.01E-05	0.00E+00

Πίνακας 5.23: Σύγκριση SLPSO και εDEag, Συνάρτηση 17

	Function 17	
	10 D	30 D
Median CV	0.00E+00	0.00E+00

Πίνακας 5.24: Διάμεση τιμή παραβίασης περιορισμών SLPSO, Συνάρτηση 17

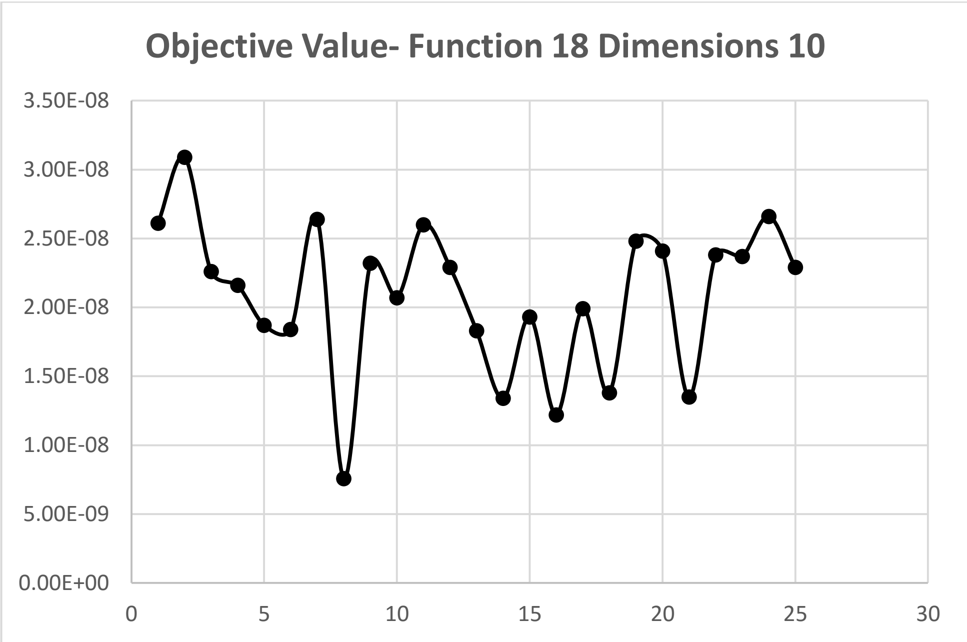
Παρατηρήσεις: Στα αποτελέσματα της συνάρτησης 17 βλέπουμε καταρχήν ότι ο SLPSO καταφέρνει και βρίσκει εφικτές λύσεις και στις 10 και στις 30 διαστάσεις. Έπειτα, βλέπουμε ότι στις 10 διαστάσεις ο εDEag είναι καλύτερος από τον SLPSO ενώ στις 30 διαστάσεις ο SLPSO αποδίδει καλύτερα από τον εDEag.

5.1.9 **Συνάρτηση 18**

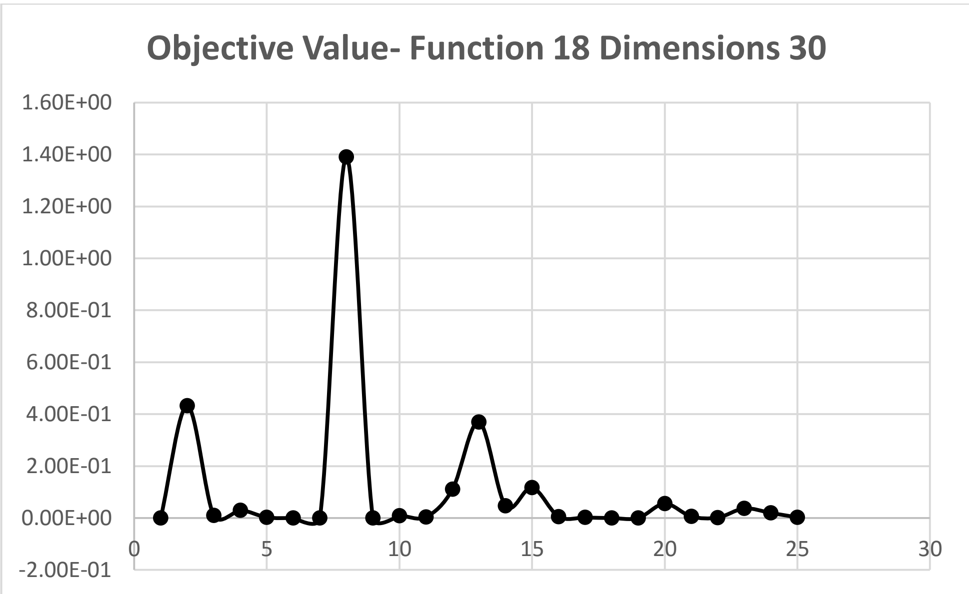
	Function 18			
	10 Dimensions		30 Dimensions	
Run	Obj	CV	Obj	CV
1	2.61E-08	0.00E+00	6.49E-07	0.00E+00
2	3.09E-08	0.00E+00	4.32E-01	0.00E+00
3	2.26E-08	0.00E+00	9.74E-03	0.00E+00
4	2.16E-08	0.00E+00	2.99E-02	0.00E+00
5	1.87E-08	0.00E+00	2.53E-03	0.00E+00
6	1.84E-08	0.00E+00	5.40E-06	0.00E+00
7	2.64E-08	0.00E+00	3.33E-06	0.00E+00
8	7.58E-09	0.00E+00	1.39E+00	0.00E+00
9	2.32E-08	0.00E+00	3.21E-06	0.00E+00
10	2.07E-08	0.00E+00	9.06E-03	0.00E+00
11	2.60E-08	0.00E+00	4.08E-03	0.00E+00
12	2.29E-08	0.00E+00	1.11E-01	0.00E+00
13	1.83E-08	0.00E+00	3.70E-01	0.00E+00
14	1.34E-08	0.00E+00	4.68E-02	0.00E+00
15	1.93E-08	0.00E+00	1.17E-01	0.00E+00
16	1.22E-08	0.00E+00	5.44E-03	0.00E+00
17	1.99E-08	0.00E+00	1.97E-03	0.00E+00
18	1.38E-08	0.00E+00	5.92E-05	0.00E+00
19	2.48E-08	0.00E+00	5.17E-07	0.00E+00
20	2.41E-08	0.00E+00	5.52E-02	0.00E+00
21	1.35E-08	0.00E+00	6.03E-03	0.00E+00
22	2.38E-08	0.00E+00	1.78E-03	0.00E+00

23	2.37E-08	0.00E+00	3.68E-02	0.00E+00
24	2.66E-08	0.00E+00	1.96E-02	0.00E+00
25	2.29E-08	0.00E+00	2.51E-03	0.00E+00

Πίνακας 5.25: Αποτελέσματα SLPSO, Συνάρτηση 18



Σχήμα 5.17: Διάγραμμα αντικειμενικής συνάρτησης για την συνάρτηση 18 σε 10 διαστάσεις για τα 25 τρεξίματα.



Σχήμα 5.18: Διάγραμμα αντικειμενικής συνάρτησης για την συνάρτηση 18 σε 10 διαστάσεις για τα 25 τρεξίματα.

	Function 18			
	10 Dimensions		30 Dimensions	
	SLPSO	εDEag	SLPSO	εDEag
Best Obj	7.58E-09	3.73E-20	5.17E-07	1.23E+00
Median Obj	2.26E-08	4.10E-19	6.03E-03	2.68E+01
Worst Obj	3.09E-08	9.23E-18	1.39E+00	7.38E+02
Mean Obj	2.09E-08	9.68E-19	1.06E-01	8.75E+01
Std Obj	5.45E-09	1.81E-18	2.89E-01	1.66E+02
Mean CV	0.00E+00	0.00E+00	0.00E+00	0.00E+00

Πίνακας 5.26: Σύγκριση SLPSO και εDEag, Συνάρτηση 18

	Function 18	
	10 D	30 D
Median CV	0.00E+00	0.00E+00

Πίνακας 5.27: Διάμεση τιμή παραβίασης περιορισμών SLPSO, Συνάρτηση 18

Παρατηρήσεις: Στην συνάρτηση 18 βλέπουμε ο SLPSO να βρίσκει εφικτές λύσεις οι οποίες είναι χειρότερες από αυτές του εDEag στις 10 διαστάσεις όμως είναι καλύτερες από αυτές του εDEag στις 30 διαστάσεις, ότι συνέβη και στην συνάρτηση 17.

5.1.10 **Ανάλυση αποτελεσμάτων**

Παραπάνω είδαμε όλα τα αποτελέσματα του SLPSO στις συναρτήσεις 10 ως 18 του διαγωνισμού CEC 2010, και επίσης και μια σύγκριση με αυτά του εDEag. Παρακάτω, θα κάνουμε μια ανάλυση των αποτελεσμάτων αυτών. Έχοντας τις ιδιότητες της κάθε συνάρτησης, μπορούμε να βαθμολογήσουμε την απόδοση του SLPSO σε αυτές, και να διακρίνουμε πιθανές δυνάμεις και αδυναμίες, ανάλογα με το τι είδος συνάρτησης και περιορισμών έχουμε να κάνουμε. Επιπλέον να προσθέσουμε ότι οι ρυθμίσεις των μεταβλητών για τον SLPSO προέκυψαν μετά από δοκιμές, και επιλέξαμε τις καταλληλότερες για κάθε συνάρτηση στα 25 τρεξίματα που παρουσιάσαμε.

Για την ανάλυση των αποτελεσμάτων διακρίναμε 5 διαφορετικές κατηγορίες στις οποίες μπορεί να ανήκει μια συνάρτηση ανάλογα με τα αποτελέσματα που βρήκαμε με τον SLPSO σε σύγκριση με αυτά που βρίσκει ο αλγόριθμος εDEag. Στις παρακάτω κατηγορίες, ο αριθμός της κάθε μιας αντιστοιχεί και σε μια βαθμολόγηση, όπου 1 είναι ο χειρότερος βαθμός και 5 είναι το άριστα. Αυτές οι πέντε διαφορετικές κατηγορίες είναι:

1. O SLPSO δεν μπορεί να βρει εφικτές λύσεις και η παραβίαση των περιορισμών είναι σταθερά μεγαλύτερη του 1. Σε αυτή την περίπτωση βρισκόμαστε μακριά από την εφικτή περιοχή.
2. O SLPSO δεν μπορεί να βρει εφικτές λύσεις και η παραβίαση των περιορισμών βρίσκεται στο διάστημα [1.00e-04 , 1.00e+00]. Αρκεί η διάμεση τιμή των παραβιάσεων από τις 25 επαναλήψεις να βρίσκεται σε αυτό το διάστημα. Σε αυτή την περίπτωση βρισκόμαστε κοντά στην εφικτή περιοχή.
3. O SLPSO βρίσκει εφικτές λύσεις αλλά οι τιμές της αντικειμενικής συνάρτησης που βρίσκει είναι αρκετά χειρότερες, πιο συγκεκριμένα τουλάχιστον 10 φορές χειρότερες, από αυτές που βρίσκει ο εDEag.
4. O SLPSO βρίσκει εφικτές λύσεις και οι τιμές της αντικειμενικής συνάρτησης που βρίσκει είναι το ίδιο καλές με αυτές που βρίσκει ο εDEag.
5. O SLPSO βρίσκει εφικτές λύσεις και οι τιμές της αντικειμενικής συνάρτησης που βρίσκει είναι καλύτερες από αυτές που βρίσκει ο εDEag

Ο πίνακας που προκύπτει από την παραπάνω ανάλυση είναι ο εξής. Το επόμενος βήμα είναι να συμπεριλάβουμε στον πίνακα αυτό, τα χαρακτηριστικά της κάθε συνάρτησης.

Function	Βαθμολογία	
	10 Dimensions	30 Dimensions
10	2	2
11	1	1
12	2	2
13	4	4
14	3	3
15	2	2
16	4	3
17	4	5
18	3	5

Πίνακας 5.28: Πίνακας βαθμολόγησης των συναρτήσεων

Function	Βαθμολογία		Type of Objective	Number of Constraints	
	10 Dimensions	30 Dimensions		Euqality	Inequality
10	2	2	Non Seperable	1 Rotated	0
11	1	1	Rotated	1 Non Seperable	0
12	2	2	Seperable	2 Non Seperable	1 Seperable
13	4	4	Seperable	0	2 Seperable, 1 Non Seperable
14	3	3	Non Seperable	0	3 Seperable
15	2	2	Non Seperable	0	3 Rotated
16	4	3	Non Seperable	2 Seperable	1 Seperable, 1 Non Seperable
17	4	5	Non Seperable	1 Seperable	2 Non Seperable
18	3	5	Non Seperable	1 Seperable	1 Seperable

Πίνακας 5.29: Πίνακας βαθμολόγησης των συναρτήσεων μαζί με τα χαρακτηριστικά τους

Στον παραπάνω πίνακα βλέπουμε την βαθμολόγηση που έχουμε δώσει στην κάθε συνάρτηση, σύμφωνα με τα κριτήρια που ορίσαμε παραπάνω και επίσης βλέπουμε από δίπλα τα χαρακτηριστικά της κάθε συνάρτησης, όπως δίνονται στον διαγωνισμό. Τώρα μπορούμε να εξάγουμε κάποια συμπεράσματα όσον αφορά την απόδοση του SLPSO:

- Ο SLPSO είχε την χειρότερή του απόδοση, αυτή που κωδικοποιήσαμε με 1, στην συνάρτηση 11, η οποία είναι η μόνη συνάρτηση στην οποία η αντικειμενική συνάρτηση είναι «περιεστρεμμένη» (rotated).
- Στις συναρτήσεις 10 και 15 χρησιμοποιούνται περιεστρεμμένοι περιορισμοί. Συγκεκριμένα στην 10 χρησιμοποιείται ένας τέτοιος περιορισμός ισότητας και στην 15 τρεις ανισότητας. Το αποτέλεσμα είναι και στις δύο να μην μπορούμε να βρούμε εφικτές λύσεις, αλλά να βρισκόμαστε σχετικά κοντά στην εφικτή περιοχή, και η βαθμολόγηση τους είναι 2.

- Μια ακόμη περίπτωση στην οποία έχουμε βαθμολογία 2 είναι η συνάρτηση 12, στην οποία έχουμε μη διαχωρίσιμη αντικειμενική συνάρτηση, έναν διαχωρίσιμο περιορισμό ανισότητας έναν διαχωρίσιμο περιορισμό ισότητας. Σε αυτή την περίπτωση φαίνεται πως ο συνδυασμός αυτός οδηγεί τον SLPSO σε κακά αποτελέσματα. Τα χαρακτηριστικά των συναρτήσεων όμως, όπως φαίνεται από τα υπόλοιπα αποτελέσματα, δεν αποτελούν ικανή συνθήκη για την αποτυχία του αλγορίθμου και πιθανόν θα φταίει ο συγκεκριμένος συνδυασμός.
- Στην συνάρτηση 17 και 18 βλέπουμε πως ενώ στις 10 διαστάσεις έχουμε τις συναρτήσεις αυτές στις κατηγορίες 4 και 3 αντίστοιχα, στις 30 διαστάσεις είναι και οι δύο στην κατηγορία 5. Αυτό μπορεί να οφείλεται σε αδυναμία του εDEag σε πολλές διαστάσεις.
- Γενικότερα οι συναρτήσεις 13, 14, 16, 17, 18 χρησιμοποιούν συνδυασμούς διαχωρίσιμων και μη, αντικειμενικών και περιορισμών ισότητας και ανισότητας και όλες οι συναρτήσεις βρίσκονται στην κατηγορία 3 ή παραπάνω. Μόνο η συνάρτηση 12 που αναφέραμε, βρίσκεται στην κατηγορία 2, από τις συναρτήσεις που χρησιμοποιούν αυτόν τον συνδυασμό. Είναι αυτονόητο βέβαια, πως κάθε συνάρτηση, πέραν του χαρακτηρισμού για το αν έχει διαχωρίσιμη ή μη, έχει και άλλα χαρακτηριστικά τα οποία την καθιστούν εύκολη ή δύσκολη για βελτιστοποίησης.
- Από τα αποτελέσματα φαίνεται πως ο SLPSO δεν αντιμετωπίζει δυσκολίες σε προβλήματα με μη διαχωρίσιμες συναρτήσεις ή περιορισμούς. Σε κάποια προβλήματα με τέτοιες δεν έχουμε καλά αποτελέσματα, όμως από την στιγμή που σε άλλα προβλήματα με μη διαχωρίσιμη αντικειμενική ή περιορισμούς ο SLPSO δεν αντιμετωπίζει δυσκολίες, μπορούμε να συμπεράνουμε πως αιτία για τα κακά αποτελέσματα είναι ο συγκεκριμένος συνδυασμός αντικειμενικής και περιορισμών και όχι μια γενική αδυναμία του SLPSO σε μη διαχωρίσιμες συναρτήσεις και περιορισμούς.

5.2 Αποτελέσματα συναρτήσεων από την εργασία Runarsson & Yao

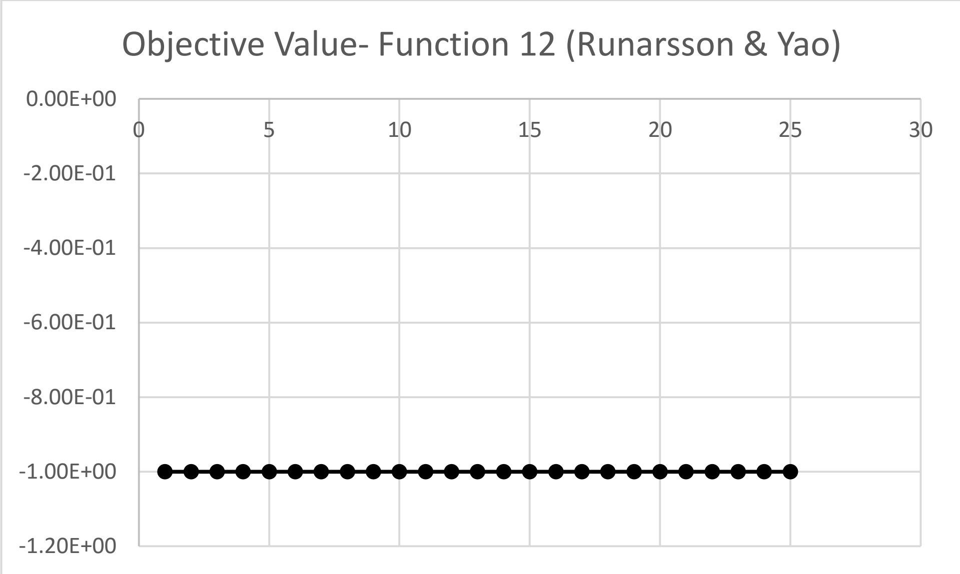
Οι παρακάτω συναρτήσεις βρέθηκαν στην εργασία των Runarsson και Yao (Runarsson and Yao, 2000) και συγκεκριμένα πρόκειται για τις συναρτήσεις 12 και 13, τις οποίες ορίσαμε στο κεφάλαιο 4.

5.2.1 **Συνάρτηση 12**

Όπως αναφέραμε η βέλτιστη λύση είναι $\overrightarrow{x^*} = (5{,}5{,}5)$ και $f(\overrightarrow{x^*}) = 1$. Από την στιγμή που έχουμε όμως πρόβλημα μεγιστοποίησης, πολλαπλασιάζουμε την συνάρτηση με -1, ώστε να το μετατρέψουμε σε πρόβλημα ελαχιστοποίησης. Η βέλτιστη λύση παραμένει η ίδια, ενώ η αντικειμενική συνάρτηση για αυτήν είναι $f(\overrightarrow{x^*}) = -1$. Για την συνάρτηση αυτή ρυθμίσαμε τον SLPSO να κάνει 500 επαναλήψεις, με 5 σωματίδια, και πολλαπλασιαστή νέφους 50. Ο πίνακας ο οποίος προέκυψε από τα 25 τρεξίματα είναι:

Obj	CNV
-1.00E+00	0.00E+00
-1.00E+00	0.00E+00
-1.00E+00	0.00E+00
-1.00E+00	0.00E+00
-1.00E+00	0.00E+00
-1.00E+00	0.00E+00
-1.00E+00	0.00E+00
-1.00E+00	0.00E+00
-1.00E+00	0.00E+00
-1.00E+00	0.00E+00
-1.00E+00	0.00E+00
-1.00E+00	0.00E+00
-1.00E+00	0.00E+00
-1.00E+00	0.00E+00
-1.00E+00	0.00E+00
-1.00E+00	0.00E+00
-1.00E+00	0.00E+00
-1.00E+00	0.00E+00
-1.00E+00	0.00E+00
-1.00E+00	0.00E+00
-1.00E+00	0.00E+00
-1.00E+00	0.00E+00
-1.00E+00	0.00E+00
-1.00E+00	0.00E+00
-1.00E+00	0.00E+00

Πίνακας 5.30: Αποτελέσματα του SLPSO στις 25 επαναλήψεις για την συνάρτηση 12



Σχήμα 5.19: Διάγραμμα αντικειμενικής συνάρτησης για την συνάρτηση 12 από την εργασία των Runarsson & Yao για τα 25 τρεξίματα.

Best Obj	-1.00E+00
Median Obj	-1.00E+00
Worst Obj	-1.00E+00
Mean Obj	-1.00E+00
Std Obj	0.00E+00
Mean CV	0.00E+00
Median CV	0.00E+00

Πίνακας 5.31: Αποτελέσματα του SLPSO για την συνάρτηση 12

Παρατηρήσεις: Στα αποτελέσματα βλέπουμε ότι ο αλγόριθμος σε όλα τα τρεξίματα βρήκε την βέλτιστη λύση. Αυτό ενισχύει και το επιχείρημά μας ότι ο αλγόριθμος SLPSO μπορεί να αντιμετωπίζει επιτυχώς συναρτήσεις με μη περιορισμούς οι οποίοι έχουν ασύνδετα πεδία εφικτών λύσεων, όπως σε αυτή την περίπτωση.

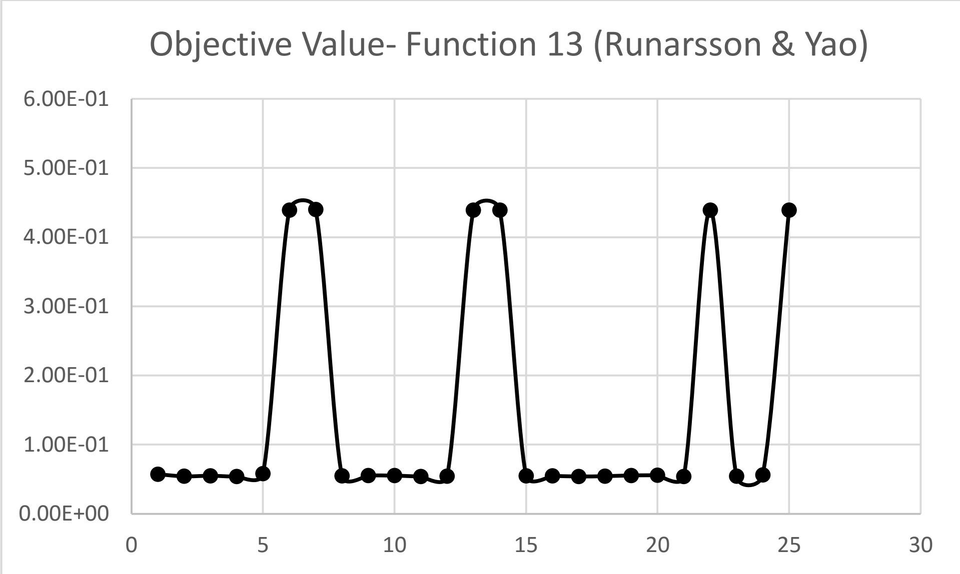
5.2.2 Συνάρτηση 13

Όπως αναφέραμε η βέλτιστη λύση είναι:

$\overrightarrow{x^*} = (-1.717143, 1.595709, 1.827247, -0.7636413, -0.763645)$ και $f(\overrightarrow{x^*}) = 0.0539498$

Obj	CNV
5.72E-02	0.00E+00
5.42E-02	2.15E-04
5.50E-02	1.86E-04
5.41E-02	0.00E+00
5.80E-02	1.03E-04
4.39E-01	0.00E+00
4.40E-01	0.00E+00
5.47E-02	1.43E-04
5.55E-02	1.50E-04
5.52E-02	0.00E+00
5.40E-02	1.17E-04
5.42E-02	0.00E+00
4.39E-01	0.00E+00
4.39E-01	0.00E+00
5.51E-02	0.00E+00
5.48E-02	0.00E+00
5.40E-02	0.00E+00
5.44E-02	0.00E+00
5.55E-02	0.00E+00
5.56E-02	0.00E+00
5.40E-02	2.05E-04
4.39E-01	0.00E+00
5.42E-02	1.03E-04
5.63E-02	0.00E+00
4.39E-01	0.00E+00

Πίνακας 5.32: Αποτελέσματα του SLPSO στις 25 επαναλήψεις για την συνάρτηση 13



Σχήμα 5.20: Διάγραμμα αντικειμενικής συνάρτησης για την συνάρτηση 13 από την εργασία των Runarsson & Yao για τα 25 τρεξίματα.

Best Obj	5.40E-02
Median Obj	5.52E-02
Worst Obj	4.40E-01
Mean Obj	1.47E-01
Std Obj	0.00E+00
Mean CV	4.89E-05
Median CV	0.00E+00

Πίνακας 5.33: Αποτελέσματα του SLPSO για την συνάρτηση 13

Παρατηρήσεις: Από τα αποτελέσματα βλέπουμε ότι ο αλγόριθμος κάποιες φορές βρίσκει την βέλτιστη λύση, άλλες φορές την πλησιάζει και κάποιες φορές έχουμε μικρές παραβιάσεις των περιορισμών.

5.3 Αποτελέσματα στο πρόβλημα βέλτιστης ανάθεσης φορτίου (Economic load dispatch problem)

Σε αυτό το σημείο θα παρουσιάσουμε τα αποτελέσματα του SLPSO και του εDEag στο πρόβλημα βέλτιστης ανάθεσης φορτίου (Economic load dispatch problem), το οποίο ορίσουμε στο κεφάλαιο 4.3. Για να δοκιμάσουμε να λύσουμε αυτό το πρόβλημα με τον αλγόριθμο, αρχικά κατεβάσαμε από το διαδίκτυο, από την ιστοσελίδα του Tetsuyuki Takahama, τον αλγόριθμο εDEag, ο οποίος είναι γραμμένος σε γλώσσα προγραμματισμού C. Έπειτα, ορίσαμε το πρόβλημα ανάθεσης φορτίου, σύμφωνα με την τεχνική με την οποία είχαν ορίσει τα υπόλοιπα προβλήματα σε αυτά τα αρχεία, και επιλύσαμε το πρόβλημα ανάθεσης φορτίου με τον αλγόριθμο εDEag.

Για να προκύψουν τα αποτελέσματα τα οποία θα παρουσιάσουμε, για 5, 10, 20 εκατομμύρια υπολογισμούς της αντικειμενικής συνάρτησης (function evaluations) χρειάστηκε να ρυθμίσουμε τις παραμέτρους τις οποίες χρησιμοποιεί ο αλγόριθμος εDEag, μέχρι να βρούμε τις καταλληλότερες για το συγκεκριμένος πρόβλημα. Στον παρακάτω πίνακα φαίνεται ο πίνακας με τα στατιστικά αποτελέσματα που προέκυψαν από 25 ανεξάρτητα τρεξίματα του κάθε αλγόριθμου, για 5, 10, 20 εκατομμύρια υπολογισμούς αντικειμενικής συνάρτησης (Function Evaluations-FES).

FES	5.00E+06		1.00E+07		2.00E+07	
Algorithm	SLPSO	εDEag	SLPSO	εDEag	SLPSO	εDEag
median	4.914E+06	5.005E+06	4.910E+06	4.954E+06	4.909E+06	4.922E+06
best	4.909E+06	4.972E+06	4.906E+06	4.929E+06	4.906E+06	4.912E+06
worst	4.918E+06	5.046E+06	4.913E+06	4.990E+06	4.914E+06	4.943E+06
std	2.608E+03	1.813E+04	1.781E+03	1.752E+04	1.961E+03	9.039E+03

Πίνακας 5.33: Πίνακας αποτελεσμάτων για το πρόβλημα βέλτιστης ανάθεσης φορτίου

Παρατηρήσεις: Στα αποτελέσματα βλέπουμε πάντα ότι ο SLPSO παρουσιάζει λίγο καλύτερα αποτελέσματα. Πιο συγκεκριμένα, τα αποτελέσματα του εDEag έχουν παραβίαση περιορισμών πάντα 0, ενώ ο SLPSO βρίσκει τα αποτελέσματα αυτά με μια μικρή συνήθως παραβίαση των περιορισμών της τάξεως του 1.00e-03, το οποίο είναι δεκτό από εμάς στο συγκεκριμένο πρόβλημα. Το παραπάνω εξηγεί την μικρή διαφορά που έχουν οι αλγόριθμοι στα αποτελέσματα της αντικειμενικής.

Παρακάτω θα δείξουμε αναλυτικά τα 25 αποτελέσματα του SLPSO στα οποία φαίνονται κάποια ενδιαφέροντα στοιχεία:

FES	5.00E+06		1.00E+07		2.00E+07	
Run/Algorithm	Obj	CNV	Obj	CNV	Obj	CNV
1	4.916E+06	0.000E+00	4.910E+06	8.731E-03	4.906E+06	6.590E-03
2	4.917E+06	0.000E+00	4.910E+06	4.267E-03	4.910E+06	2.943E-03
3	4.915E+06	0.000E+00	4.908E+06	8.246E-03	4.911E+06	8.194E-03
4	4.912E+06	7.981E-03	4.910E+06	5.701E-03	4.910E+06	9.778E-03
5	4.915E+06	0.000E+00	4.911E+06	4.908E-03	4.909E+06	5.498E-03
6	4.913E+06	1.159E-03	4.911E+06	7.081E-03	4.910E+06	5.009E-03
7	4.915E+06	3.771E-03	4.907E+06	6.457E-03	4.914E+06	7.233E-03
8	4.914E+06	9.756E-03	4.910E+06	7.016E-03	4.908E+06	4.334E-03
9	4.914E+06	4.508E-03	4.911E+06	9.181E-03	4.910E+06	6.780E-03
10	4.915E+06	7.761E-03	4.907E+06	9.568E-03	4.907E+06	9.936E-03
11	4.911E+06	9.953E-03	4.909E+06	9.936E-03	4.908E+06	6.582E-03
12	4.916E+06	8.303E-03	4.912E+06	9.175E-03	4.913E+06	7.634E-03
13	4.910E+06	0.000E+00	4.908E+06	7.180E-03	4.912E+06	9.320E-03
14	4.916E+06	0.000E+00	4.909E+06	9.248E-03	4.909E+06	6.850E-03
15	4.910E+06	9.033E-03	4.911E+06	9.811E-03	4.908E+06	9.078E-03
16	4.911E+06	0.000E+00	4.908E+06	8.842E-03	4.909E+06	3.569E-03
17	4.918E+06	0.000E+00	4.910E+06	8.674E-03	4.909E+06	5.093E-03
18	4.913E+06	7.902E-03	4.909E+06	1.611E-03	4.907E+06	4.967E-03
19	**4.909E+06**	**0.000E+00**	4.910E+06	7.830E-03	4.911E+06	7.915E-03
20	4.911E+06	4.001E-03	4.906E+06	8.551E-03	4.909E+06	5.229E-03
21	4.914E+06	3.045E-03	4.912E+06	4.672E-03	4.910E+06	9.944E-03
22	4.918E+06	6.312E-03	4.908E+06	3.569E-03	4.908E+06	9.122E-03
23	4.912E+06	1.108E-03	4.913E+06	8.336E-03	4.912E+06	8.770E-03
24	4.911E+06	0.000E+00	4.913E+06	8.847E-03	4.906E+06	4.876E-03
25	4.916E+06	0.000E+00	4.911E+06	6.955E-03	4.908E+06	6.683E-03

Πίνακας 5.34: Αναλυτικά αποτελέσματα SLPSO στο πρόβλημα βέλτιστης ανάθεσης φορτίου

Στον παραπάνω πίνακα βλέπουμε τα αναλυτικά αποτελέσματα του SLPSO για τα 25 τρεξίματα και για τον εκάστοτε αριθμό FES. Ενδιαφέρον είναι τα παρακάτω:

- Όπως βλέπουμε στις 5.00E+06 FES έχουμε αποτέλεσμα 4.909E+06 με παραβίαση περιορισμών ίση με 0. Η παραπάνω λύση είναι καλύτερη από αυτή που βρήκε ο εDEag σε οποιοδήποτε αριθμό FES. Το αναφέρουμε αυτό γιατί οι λύσεις του εDEag είναι όλες με παραβίαση περιορισμών ίση με 0.
- Στον πίνακα αυτόν φαίνεται αρχικά περίεργο το ότι στις 5.00E+06 FES βρίσκουμε πολλές λύσεις με μηδενική παραβίαση, ενώ στις 10.00E+06 FES και 20.00E+06 FES έχουμε μόνιμος παραβιάσεις (σε όλες τις περιπτώσεις βέβαια πολύ μικρές). Το παραπάνω περίεργο φαινόμενο εξηγείται εύκολα αν σκεφτούμε ότι ο αλγόριθμος δουλεύει με μηχανισμό συνάρτησης ποινής, και πιο συγκεκριμένα συμβαίνει το εξής:

- Στις 5.00E+06 FES ο αλγόριθμος βρίσκει κάποιες λύσεις με μηδενική παραβίαση.
- Αργότερα στις 10.00E+06 FES και 20.00E+06 FES ο αλγόριθμος βρίσκει άλλα σημεία τα οποία έχουν καλύτερη αξία αντικειμενικής, κατά 1000-5000 ας πούμε, αλλά έχουν μια παραβίαση της τάξεως του 0.001.
- Έτσι αυτή η λύση με τον μηχανισμό της συνάρτησης ποινής, και συγκεκριμένα με λ (παράμετρος ποινής-penalty coefficient) το οποίο είχαμε θέσει σε τάξη μεγέθους 1000, αποκτά τελική αξία αντικειμενικής ή αλλιώς αξία συνάρτησης αντικειμενικής μικρότερη κατά 1000-5000 λόγω της αντικειμενικής και μεγαλύτερη κατά 1000*0.001 λόγω της ποινής. Άρα η τελική αξία της συνάρτησης ποινής που βρίσκει ο αλγόριθμος είναι μικρότερη από αυτή που είχε βρει προηγουμένως

- Στα αποτελέσματα φαίνεται ότι ο SLPSO και ο εDEag βρήκαν διαφορετικές λύσεις. Ο SLPSO έβρισκε τις λύσεις εκείνες οι οποίες είχαν μια μικρή παραβίαση περιορισμών και καλύτερη τιμή αντικειμενική συνάρτησης ενώ ο εDEag έβρισκε εκείνες οι οποίες είχαν μηδενική παραβίαση περιορισμών και λίγο χειρότερη τιμή αντικειμενικής συνάρτησης. Αυτό οφείλετε στον διαφορετικό τρόπο των αλγόριθμων αυτών όσον αφορά τον χειρισμό των περιορισμών.
- Επιβεβαιώνεται πάντως αυτό που είδαμε και σε άλλες συναρτήσεις ότι σε μεγάλα προβλήματα με πολλές διαστάσεις ο SLPSO δεν υστερεί έναντι του εDEag.

6 Συμπεράσματα και επίλογος

6.1 Ανακεφαλαίωση

Στόχος στα προβλήματα βελτιστοποίησης είναι η εύρεση της τιμής αυτής, η οποία ικανοποιεί το πρόβλημα με τον βέλτιστο τρόπο. Τέτοια προβλήματα εμφανίζονται παντού στον πραγματικό κόσμο. Για παράδειγμα το πρόβλημα βέλτιστης ανάθεσης φορτίου που είδαμε, στο οποίο προσπαθούμε να ικανοποιήσουμε την συνολική ζήτηση ενέργειας με το ελάχιστο κόστος. Επίσης όπως είπαμε, τα προβλήματα αυτά στον πραγματικό κόσμο συνοδεύονται σχεδόν πάντα από περιορισμούς οι οποίοι πρέπει να ικανοποιηθούν. Έτσι προκύπτουν τα προβλήματα βελτιστοποίησης υπό περιορισμούς. Τα προβλήματα του πραγματικού κόσμου μοντελοποιούνται και εκφράζονται σε μαθηματική μορφή, με στόχο να γίνονται οι κατάλληλοι υπολογισμοί και να μπορούν αυτά να επιλυθούν από αλγορίθμους. Έπειτα, χρησιμοποιούμε τα αποτελέσματα που έχουμε από το πρόβλημα «μοντέλο» στην λήψη αποφάσεων στον πραγματικό κόσμο.

Στην διπλωματική εργασία αυτή μελετήσαμε και συγκρίναμε δύο εξελικτικούς αλγορίθμους, τον SLPSO και τον εDEag, οι οποίοι προτάθηκαν πολύ πρόσφατα και αποτελούν και οι δύο πεδίο έρευνας αυτή την στιγμή, για την εύρεση ενός όσο το δυνατόν «ισχυρότερου» αλγορίθμου βελτιστοποίησης. Ένας τέτοιος αλγόριθμος, ιδανικά θα έχει όσο το δυνατόν λιγότερες παραμέτρους ρύθμισης, δηλαδή θα είναι ένας αλγόριθμος «μαύρο κουτί» για οποιοδήποτε πρόβλημα βελτιστοποίησης χωρίς να ξοδεύουμε υπολογιστικό κόστος για την εύρεση των καταλλήλων παραμέτρων ρύθμισης για το κάθε πρόβλημα. Το μόνο που θα απαιτείται θα είναι η μοντελοποίηση του προβλήματος. Επίσης θα επιλύει γρήγορα τα προβλήματα, το οποίο συνεπάγεται με μικρό υπολογιστικό κόστος.

Η εύρεση ενός τέτοιου αλγορίθμου αποτελεί πεδίο ενασχόλησης για πολλούς ερευνητές και με την πάροδο του χρόνου η εργαλειοθήκη των εξελικτικών αλγορίθμων συνεχώς εμπλουτίζεται με νέες τεχνικές και μεθόδους, οι οποίες προκύπτουν μετά από επίπονη έρευνα και πολλά πειράματα, προσθέτοντας έτσι νέες τεχνικές για την εύρεση αυτού του «ισχυρού» αλγορίθμου. Μια τέτοια προσπάθεια γίνετε και στην σχολή μας, στο εργαστήριο Επιχειρησιακής Έρευνας. Έτσι και αυτή η διπλωματική, έχει ως στόχο, να δοκιμάσει τον αλγόριθμο SLPSO , ο οποίος προτάθηκε το 2012 από τους Li, Yang και Nguyen (Li et al., 2012, σε διάφορα προβλήματα βελτιστοποίησης υπό περιορισμούς, να συγκρίνει τα αποτελέσματα αυτά με τα αποτελέσματα άλλων αλγορίθμων, και να βγάλουμε μέσα από αυτά συμπεράσματα για τις δυνάμεις και τις αδυναμίες του αλγόριθμου. Το επόμενο στάδιο είναι να αξιοποιήσουμε τα συμπεράσματα αυτά ώστε να προτείνουμε και να δοκιμάσουμε αλλαγές πάνω στον αλγόριθμο.

Οι επιπλέον λειτουργίες που προσθέσαμε στον SLPSO σε αυτή την διπλωματική είναι το νέφος απογόνων και η αναδιάταξη των καλύτερων λύσεων, οι οποίες έχουν σχέση με την αναζήτηση του αλγορίθμου. Το νέφος απογόνων προσθέτει σε κάθε επανάληψη έναν στοχαστικό μηχανισμό αναζήτησης γύρω από την θέση του τρέχοντος βέλτιστου σωματιδίου. Η αναδιάταξη των καλύτερων λύσεων επιτρέπει το σμήνος να αναδιαταχτεί ανά κάποιες επαναλήψεις, σε περίπτωση που τα σωματίδια δεν βρίσκονται σε υποσχόμενες θέσεις. Επιπλέον, επειδή ο SLPSO είναι αλγόριθμος βελτιστοποίησης προβλημάτων χωρίς περιορισμούς, και επειδή εμείς μελετάμε προβλήματα βελτιστοποίησης υπό περιορισμούς, τα οποία συναντάμε και στον πραγματικό κόσμο, προσθέσαμε μια μέθοδο χειρισμού περιορισμών, και πιο συγκεκριμένα την συνάρτηση ποινής. Έτσι προέκυψε μια παραλλαγή του κλασσικού SLPSO, η οποία μπορεί να επιλύει προβλήματα βελτιστοποίησης υπό περιορισμούς.

Έπειτα συγκρίναμε τα αποτελέσματα με αυτά του αλγορίθμου εDEag, ο οποίος έχει αναπτυχθεί από τους Takahama και Sakai (Takahama and Sakai, 2010) και αναδείχθηκε νικητής στον διαγωνισμό CEC 2010 πάνω στο θέμα βελτιστοποίηση υπό περιορισμούς. Στην διπλωματική εξηγήσαμε αναλυτικά το πώς λειτουργεί αυτός ο αλγόριθμος.

Πιο συγκεκριμένα επιλύσαμε με τον SLPSO 9 από τα προβλήματα του διαγωνισμού του CEC 2010, 2 από τα προβλήματα από την εργασία των Runarsson και Yao και επίσης ένα μεγάλο πρόβλημα βέλτιστης ανάθεσης φορτίου. Επίσης τροποποιήσαμε τον κώδικα του αλγόριθμου εDEag τον οποίο βρήκαμε στην ιστοσελίδα των συγγραφών, ορίζοντας σε αυτόν το πρόβλημα βέλτιστης ανάθεσης, ώστε να τρέξουμε και με αυτόν τον αλγόριθμο το πρόβλημα αυτό και να συγκρίνουμε τα αποτελέσματα με τον SLPSO. Κάθε δοκιμαστική συνάρτηση που προσπαθούμε να επιλύσουμε έχει διαφορετικά χαρακτηριστικά, τα οποία μπορούν να βρεθούν και σε ένα πρόβλημα στον πραγματικό κόσμο, δίνοντάς μας έτσι πληροφορίες για την συμπεριφορά του αλγορίθμου μας σε προβλήματα με τα αντίστοιχα χαρακτηριστικά. Στην διπλωματική εργασία αυτή παρουσιάσαμε πολλά από τα αποτελέσματα που προέκυψαν από την εφαρμογή του αλγορίθμου στα προβλήματα αυτά και προσπαθήσαμε να συσχετίσουμε αυτά τα αποτελέσματα ώστε να βγάλουμε κάποια συμπεράσματα.

6.2 *Συμπεράσματα*

Όσον αφορά το πρακτικό πρόβλημα που μελετήσαμε, το πρόβλημα βέλτιστης ανάθεσης φορτίου, είδαμε πως και οι δύο αλγόριθμοι, ο SLPSO και ο εDEag, είχαν πολύ καλή απόδοση και έβρισκαν λύσεις πάρα πολύ κοντά στο ολικό βέλτιστο. Το συγκεκριμένο πρόβλημα ήταν αρκετά μεγάλο, με 600 μεταβλητές απόφασης, και είδαμε πως μπορεί αυτό να επιλυθεί αποτελεσματικά με εξελικτικούς αλγορίθμους, και να λαμβάνονται αποφάσεις με βάση τα αποτελέσματα αυτά.

Από τα αποτελέσματα αυτά φαίνεται πως μπορούν να χρησιμοποιηθούν οι συγκεκριμένοι εξελικτικοί αλγόριθμοι για την επίλυση πρακτικών προβλημάτων βέλτιστης ανάθεσης φορτίου και να λαμβάνονται αποφάσεις και να εξοικονομούνται πόροι, με τον βέλτιστο καταμερισμό της ενεργειακής παραγωγής στις μονάδες παραγωγής. Πρακτικά αυτό μπορεί να λειτουργήσει πραγματοποιώντας αρχικά την πρόβλεψη της ενεργειακής ζήτησης με βάση τα ιστορικά δεδομένα (πχ της επόμενης ημέρας) και αργότερα επιλύοντας αυτό το πρόβλημα με κάποιον από τους αλγόριθμους. Έτσι, σχετικά γρήγορα, ο διαχειριστής του συστήματος διανομής ενέργειας θα έχει στα χέρια του ένα σχέδιο παραγωγής ενέργειας, το οποίο θα χρησιμοποιεί τους πόρους του δικτύου με τον βέλτιστο τρόπο και θα εξοικονομεί πολλά χρήματα στην εταιρεία παραγωγής ενέργειας.

Το μόνο μειονέκτημα που έχουν οι παραπάνω αλγόριθμοι είναι πως για κάθε καινούργια ζήτηση ή κάθε αλλαγή στις συναρτήσεις κόστους, απαιτούνται νέες ρυθμίσεις στις παραμέτρους του αλγόριθμου, οι οποίες μπορούν να βρεθούν μόνο με δοκιμές. Δηλαδή κάθε αλλαγή στο μαθηματικά εκφρασμένο πρόβλημα απαιτεί νέες ρυθμίσεις. Όμως, το παραπάνω μειονέκτημα δεν αποτελεί ιδιαίτερο εμπόδιο, αφού οι αλγόριθμοι βρίσκουν πολύ γρήγορα λύσεις, και έτσι μπορούμε σχετικά γρήγορα να βρούμε τις νέες κατάλληλες παραμέτρους ρύθμισης. Όπως έχουμε πει αποτελεί πεδίο έρευνας να βρεθούν αλγόριθμοι με όσο το δυνατόν λιγότερες παραμέτρους ρύθμισης.

Όπως είδαμε και στην παρουσίαση των αποτελεσμάτων σε κάποιες συναρτήσεις ο αλγόριθμος συγκλίνει και βρίσκει το ολικό βέλτιστο, σε άλλες μόνο κάποια τοπικά βέλτιστα και σε άλλες δεν μπορεί να βρει εφικτή λύση. Επίσης, είδαμε ότι υπάρχει κάποια συσχέτιση της δυνατότητας εύρεσης λύσης και το είδος αυτής με το είδος της συνάρτησης και των περιορισμών. Για παράδειγμα, στις περιεστρεμμένες συναρτήσεις και περιορισμούς, ο αλγόριθμος δεν μπορούσε να βρει εφικτή λύση. Αυτό είναι μια αδυναμία του αλγόριθμου, SLPSO που μοντελοποιήσαμε, την οποία άλλοι αλγόριθμοι την έχουν ξεπεράσει (βλέπε εDEag) και μένει να βρούμε κάποια λύση ή αλλιώς κάποια προσθήκη, ώστε να ξεπεραστεί αυτό το εμπόδιο και να μπορεί και ο SLPSO να βρίσκει λύση σε τέτοια προβλήματα.

Τα συμπεράσματα από την πειραματική διαδικασία και την σύγκριση των αποτελεσμάτων με αυτά του αλγόριθμου εDEag είναι τα εξής:

- Ο SLPSO δυσκολεύεται να βρει εφικτές λύσεις σε περιεστρεμμένες συναρτήσεις. Όπως αναφέραμε σε μια άλλη μελέτη που βρήκαμε στην βιβλιογραφία, μια άλλη παραλλαγή του PSO επίσης έχει πολύ κακά αποτελέσματα στην ίδια συνάρτηση, ενώ και εκείνη γενικότερα παρουσιάζει καλά αποτελέσματα.
- Ο SLPSO επίσης δυσκολεύεται να βρει εφικτές λύσεις σε προβλήματα με περιεστρεμμένους περιορισμούς.
- Από τα αποτελέσματα προκύπτει πως ο SLPSO δεν έχει αδυναμία σε προβλήματα στα οποία οι συναρτήσει ή οι περιορισμοί είναι μη διαχωρίσιμες συναρτήσεις. Μπορεί σε κάποια από αυτά τα προβλήματα με μη διαχωρίσιμες αντικειμενικές συναρτήσεις και περιορισμούς να μην είχα καλά αποτελέσματα ο SLPSO, αλλά φαίνεται περισσότερο να φταίει ο συνδυασμός της αντικειμενικής συνάρτησης με τους περιορισμούς, καθώς σε άλλα προβλήματα με μη διαχωρίσιμες συναρτήσεις και περιορισμούς ο SLPSO είχε πολύ καλά αποτελέσματα.
- Επίσης όπως προκύπτει από τα αποτελέσματα το μέγεθος του προβλήματος, οι διαστάσεις του προβλήματος δηλαδή, δεν αποτελεί δυσκολία για τον SLPSO. Σε κάποια προβλήματα είδαμε πως ο SLPSO σε σύγκριση με τον εDEag, είχε καλύτερα αποτελέσματα στις 30 διαστάσεις παρά στις 10. Επίσης στο πρόβλημα βέλτιστης ανάθεσης φορτίου με 600 μεταβλητές ο SLPSO πάλι είχε πολύ καλά αποτελέσματα σε σχέση με τον εDEag.
- Από τα αποτελέσματα που έχουμε για το πρόβλημα βέλτιστης ανάθεσης φορτίου φαίνεται πως ο SLPSO βρίσκει κάποιες λύσεις οι οποίες έχουν καλύτερη τιμή αντικειμενικής συνάρτησης, με κάποια μικρή παραβίαση στους περιορισμούς, ενώ ο εDEag βρίσκει λύσεις οι οποίες έχουν λίγο χειρότερη τιμή αντικειμενικής συνάρτησης με μηδενικής παραβίαση περιορισμών. Αυτό μας δείχνει το πόσο σημαντικό ρόλο παίζει η CHT που χρησιμοποιείται, η οποία αλλάζει τελείως την τοπολογία της αντικειμενικής συνάρτησης. Άρα συμπεραίνουμε πως η μέθοδος χειρισμού των περιορισμών που χρησιμοποιούμε παίζει πολύ σημαντικό ρόλο και η δοκιμή κάποιας άλλης, πέραν από την συνάρτηση ποινής, θα μπορούσε να έχει θετικά αποτελέσματα.

Όπως είδαμε από τα αποτελέσματα και τα συμπεράσματα ο SLPSO σε γενικές γραμμές ήταν αρκετά αποτελεσματικός, αλλά έχει κάποιες αδυναμίες. Οι αλλαγές που μπορούν να γίνουν στον αλγόριθμο γενικά συνοψίζονται σε τρείς κατηγορίες οι οποίες είναι οι παρακάτω:

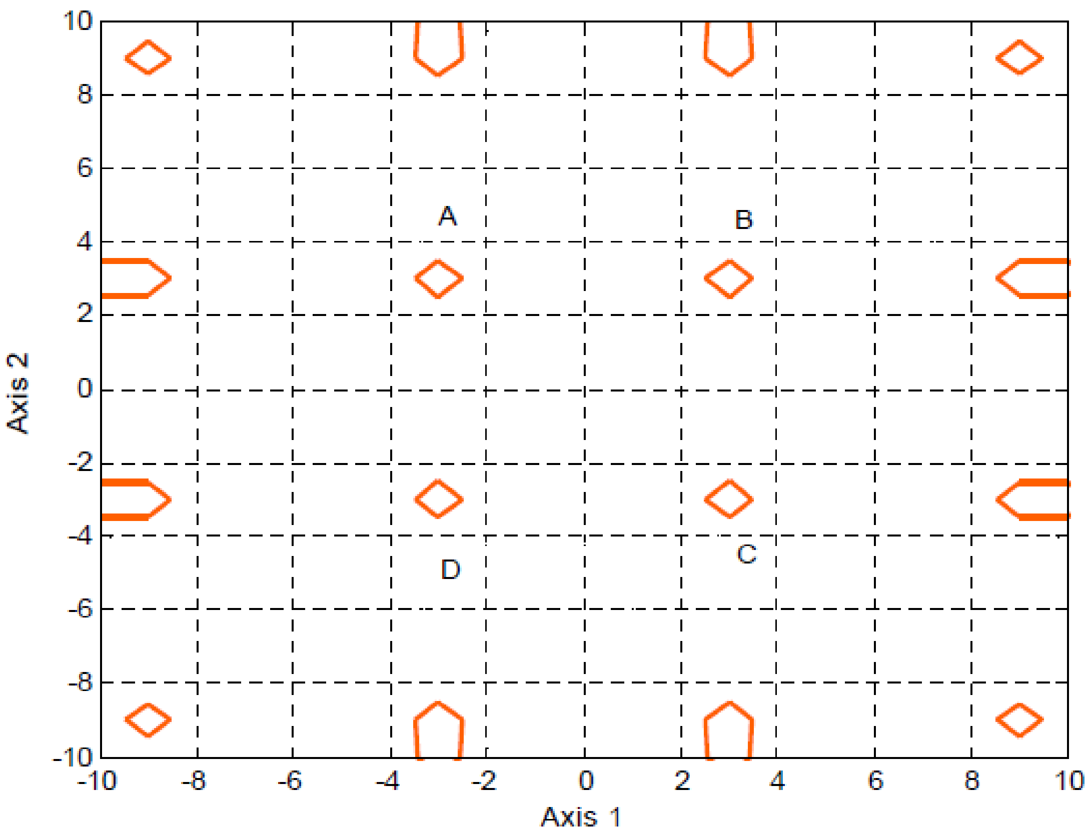
1. Αλλαγές οι οποίες έχουν να κάνουν με τον αλγόριθμο SLPSO καθαυτό. Τέτοιες για παράδειγμα θα ήταν η εισαγωγή μιας νέας μεθόδου επιλογής των διαχειριστών εκμάθησης, η εισαγωγή ενός νέου διαχειριστή εκμάθησης, η αλλαγή στην ανανέωση ταχυτήτων κ.α.
2. Αλλαγές οι οποίες αφορούν κάποια νέα προσθήκη στον αλγόριθμο. Τέτοιες είναι για παράδειγμα οι δύο που ήδη έχουμε προσθέσει στον αλγόριθμο: το νέφος απογόνων και η αναδιάταξη των καλύτερων λύσεων.
3. Στην τρίτη κατηγορία ανήκουν οι αλλαγές οι οποίες έχουν σχέση με τον χειρισμό των περιορισμών. Θα μπορούσαμε να συνδυάσουμε τον SLPSO με μια άλλη CHT, πέραν από την συνάρτηση ποινής. Για παράδειγμα θα μπορούσαμε να χρησιμοποιήσουμε την μέθοδο ελέγχου επιπέδου ε η οποία χρησιμοποιείται στον αλγόριθμο εDEag.

Οι αλλαγές οι οποίες ανήκουν στην κατηγορία 1 είναι οι πιο δύσκολες στην υλοποίηση γιατί απαιτούν μια ολική αλλαγή του βασικού σκελετού του SLPSO, ο οποίος έχει δείξει σε γενικές γραμμές πολύ καλά αποτελέσματα. Αντίθετα, οι αλλαγές των κατηγοριών 2 και 3 είναι πιο εύκολες στην υλοποίηση αφού απλά πρόκειται για μια προσθήκη. Με βάση τα αποτελέσματα αυτής της διπλωματικής, μια πρόταση για αλλαγή στον αλγόριθμο θα μπορούσε να είναι η προσθήκη της μεθόδου *gradient based mutation* η οποία χρησιμοποιείται από τον εDEag για να αντιμετωπίζει δύσκολους περιορισμούς. Επίσης, θα μπορούσε να δοκιμαστεί η CHT που χρησιμοποιεί ο εDEag, η σύγκριση επιπέδου ε, η οποία φαίνεται να έχει πολύ καλά αποτελέσματα. Οι δύο παραπάνω προτάσεις θα μπορούσαν να αντιμετωπίσουν το πρόβλημα που συνάντησε ο SLPSO στις συναρτήσεις με τους περιεστρεμμένους περιορισμούς και να οδηγούσε σε μια γενικά καλύτερη απόδοση.

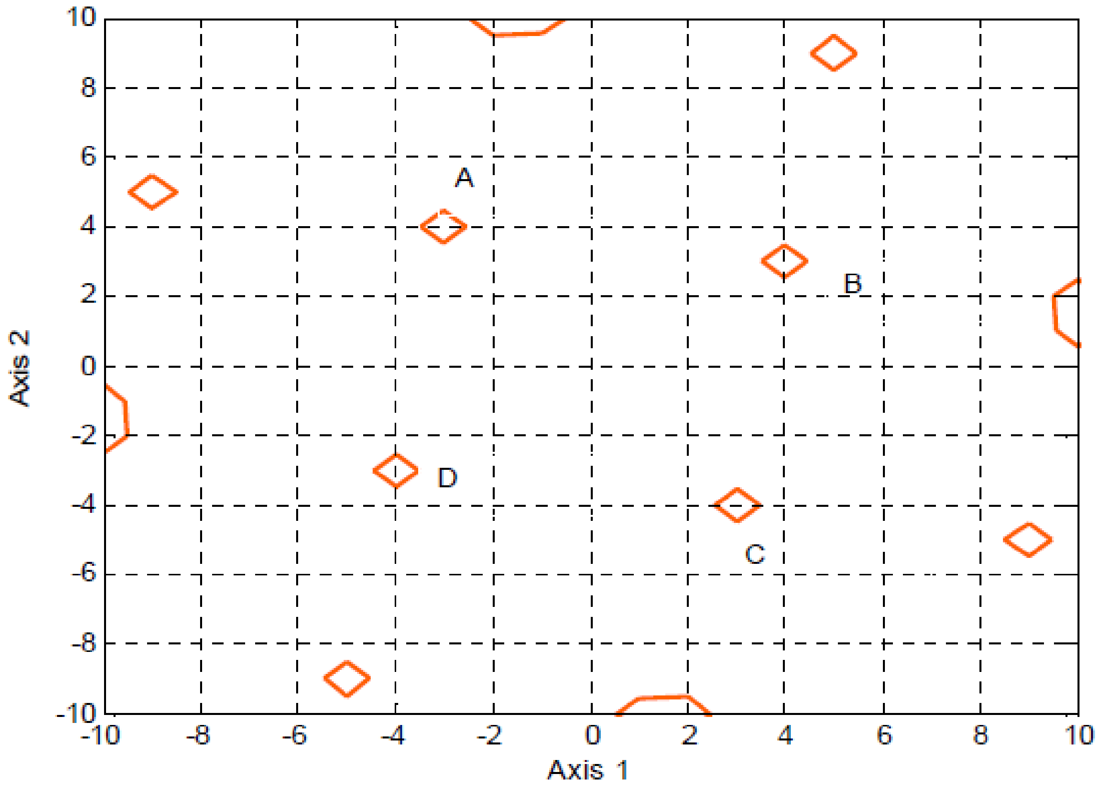
Παράρτημα Α

Συναρτήσεις διαγωνισμού CEC 2010

Επειδή σε ένα πρόβλημα με περιορισμούς, στο οποίο οι εφικτές περιοχές είναι παράλληλες στους άξονες μπορεί να λυθεί εύκολα από κάποιους αλγορίθμους οι οποίοι πραγματοποιούν γραμμική αναζήτηση με δύο ή περισσότερα διανύσματα (όπως ο DE). Έτσι, για να μην έχουν πλεονέκτημα κάποιοι αλγόριθμοι, στα περισσότερα προβλήματα του διαγωνισμού CEC 2010, έχουν περιστραφεί οι περιορισμοί. Στα σχήματα Σχήμα (4.1) και Σχήμα (4.2) φαίνεται πως επηρεάζει η περιστροφή τον περιορισμών το πρόβλημα



Σχήμα Π.1: Χώρος αναζήτησης χωρίς περιστροφή



Σχήμα Π.2: Χώρος αναζήτησης με περιστροφή

Σε αυτή τη διπλωματική θα μελετήσουμε τα προβλήματα αυτού του διαγωνισμού 10 ως 18. Στα προβλήματα αυτά θα δοκιμάσουμε τον αλγόριθμο SLPSO για 10 διαστάσεις και για 30 διαστάσεις, οι οποίες συμβολίζονται με D στα προβλήματα. Επίσης, το διανύσματα με τα οποία περιστρέφονται τα προβλήματα, τα οποία συμβολίζονται με o και M, και είναι διανύσματα τα οποία δίνονται στον ορισμό των συναρτήσεων και μπορούν να βρεθούν αν κατεβάσει κανείς τους κώδικες που δίνονται.

Στο Σχήμα (4.3) βλέπουμε τις ιδιότητες του κάθε προβλήματος. Στην πρώτη στήλη αναγράφεται ο αριθμός του προβλήματος και το πεδίο ορισμού δηλαδή ο χώρος αναζήτησης. Στην δεύτερη στήλη βλέπουμε το είδος της αντικειμενικής συνάρτησης, αν χωρίζεται η όχι και αν έχει περιστραφεί. Στις στήλες τρία και τέσσερα αναγράφονται το πλήθος των περιορισμών ισότητας και ανισότητας και το είδος τους (Number of Constraints). Στις στήλες πέντε και έξι (Feasibility Region) βλέπουμε για το προβλήματα 10 και 30 διαστάσεων αντίστοιχα, τον λόγο της εφικτής περιοχής F (Feasible region) και του συνολικού χώρου αναζήτησης S (Search space) $\rho = \frac{|F|}{|S|}$.

Problem/Search Range	Type of Objective	Number of Constraints		Feasibility Region (ρ)	
		E	I	10D	30D
C10 [-500,500]^D	Non Separable	1 Rotated	0	0.000000	0.000000
C11 [-100,100]^D	Rotated	1 Non Separable	0	0.000000	0.000000
C12 [-1000,1000]^D	Separable	1 Non Separable	1 Separable	0.000000	0.000000
C13 [-500,500]^D	Separable	0	3 2 Separable, 1 Non Separable	0.000000	0.000000
C14 [-1000,1000]^D	Non Separable	0	3 Separable	0.003112	0.006123
C15 [-1000,1000]^D	Non Separable	0	3 Rotated	0.003210	0.006023
C16 [-10,10]^D	Non Separable	2 Separable	2 1 Separable, 1 Non Separable	0.000000	0.000000
C17 [-10,10]^D	Non Separable	1 Separable	2 Non Separable	0.000000	0.000000
C18 [-50,50]^D	Non Separable	1 Separable	1 Separable	0.000010	0.000000

Σχήμα Π.3: Τα χαρακτηριστικά των προβλημάτων του διαγωνισμού CEC 2010

Στα πλαίσια του διαγωνισμού, έπρεπε να παρουσιαστούν αποτελέσματα για μέγιστο αριθμό υπολογισμών της αντικειμενικής συνάρτησης (Max Function Evaluations- FES) 20.000, 100.000 και 200.000 για τα προβλήματα 10 διαστάσεων και 60.000, 300.000 και 600.000 για τα προβλήματα 30 διαστάσεων. Για κάθε συνάρτηση θα παρουσιαστούν τα ακόλουθα αποτελέσματα: καλύτερη, μέση, χειρότερη λύση, μέση τιμή και τυπική απόκλιση των 25 τρεξιμάτων.

Επίσης για την παραβίαση των περιορισμών υπήρχαν τρείς τάξεις παραβιασμών, οι οποίες ήταν πάνω από 1, 0.01 και 0.0001. Έπειτα υπολογίζεται η μέση παραβίαση $\overline{v}$.

$$\overline{v} = \frac{\left(\sum_{i=1}^{p} G_i(X) + \sum_{j=p+1}^{m} H_j(X)\right)}{m} \quad (\Pi.1)$$

Όπου

$$G_i(X) = \begin{cases} g_i(X) & if\ g_i(X) > 0 \\ 0 & if\ g_i(X)\ \leq 0 \end{cases} \quad (\Pi.2)$$

$$H_j(X) = \begin{cases} |h_j(X)| & if\ |h_j(X)| - \varepsilon\ > 0 \\ 0 & if\ |h_j(X)| - \varepsilon\ \leq 0 \end{cases} \quad (\Pi.3)$$

Όπου ε =0.0001

Επίσης παρουσιάζεται ο λόγος εφικτών λύσεων προς αυτόν των συνολικών τρεξιμάτων (Feasibility Rate) ως εξής:

Feasible Run: Ένα τρέξιμο στο οποίο έχει βρεθεί τουλάχιστον μια εφικτή λύση πριν τερματίσουμε στις Max FES.

Για τον λόγο εφικτών λύσεων ισχύει: *Feasibility Rate = (# of feasible runs) / Total run*

Παρακάτω ορίζουμε τα προβλήματα 10 ως 18 όπως δίνονται στον διαγωνισμό CEC2010 (Mallipeddi and Suganthan, 2010).

Συνάρτηση 10

$$Min\ f(x) = \sum_{i=1}^{D-1} (100\ (z_i{}^2 - z_{i+1}{}^2)^2 +\ (z_i{}^2 - 1)^2) \quad (\Pi.4)$$

$$z = (x + 1 - o), y = (x - o)M \quad (\Pi.5)$$

$$h(x) = \sum_{i=1}^{D} (ysin(\sqrt{|y_i|} = 0 \quad (\Pi.6)$$

$$x \in [-500{,}500]^D \quad (\Pi.7)$$

Συνάρτηση 11

$$Min\ f(x) = \frac{1}{D} \sum_{i=1}^{D} (-z_i \cos(2\sqrt{|z_i|})) \quad (\Pi.8)$$

$$z = (x - o)M, y = (x + 1 - o) \tag{Π.9}$$

$$h(x) = \sum_{i=1}^{D} (100\,({y_i}^2 - {y_{i+1}}^2)^2 + \,({y_i}^2 - 1)^2) = 0 \tag{Π.10}$$

$$x \in [-100{,}100]^D \tag{Π.11}$$

Συνάρτηση 12

$$Min\, f(x) = \sum\nolimits_{i=1}^{D} (z_i \sin(\sqrt{|z_i|})) \tag{Π.12}$$

$$z = (x - o) \tag{Π.13}$$

$$h(x) = \sum_{i=1}^{D-1} ({z_i}^2 - {z_{i+1}}^2)^2 = 0 \tag{Π.14}$$

$$g(x) = (z - 100\cos(0.1z) + 10) \leq 0 \tag{Π.15}$$

$$x \in [-1000{,}1000]^D \tag{Π.16}$$

Συνάρτηση 13

$$Min\, f(x) = \frac{1}{D} \sum\nolimits_{i=1}^{D} (-z_i \sin(\sqrt{|z_i|})) \tag{Π.17}$$

$$z = (x - o) \tag{Π.18}$$

$$g_1(x) = -50 + \frac{1}{100D} \sum_{i}^{D} {z_i}^2 \ \leq 0 \tag{Π.19}$$

$$g_2(x) = -\frac{50}{D} + \sum_{i}^{D} \sin(\frac{1}{50}\pi z) \ \leq 0 \tag{Π.20}$$

$$g_3(x) = 75 - 50\left(\sum_{i=1}^{D}\frac{{z_i}^2}{4000} - \prod_{i=1}^{D}\cos\left(\frac{z_i}{\sqrt{i}}\right) + 1\right) \leq 0 \quad (\text{Π}.21)$$

$$x \in [-500{,}500]^D \quad (\text{Π}.22)$$

Συνάρτηση 14

$$Min\ f(x) = \sum_{i=1}^{D-1}(100\ ({z_i}^2 - {z_{i+1}}^2)^2 + \ ({z_i}^2 - 1)^2) \quad (\text{Π}.23)$$

$$z = x + 1 - o, y = x - o \quad (\text{Π}.24)$$

$$g_1(x) = \sum_{i=1}^{D}\left(-y_i \cos\left(\sqrt{|y_i|}\right)\right) - D\ \leq 0 \quad (\text{Π}.25)$$

$$g_2(x) = \sum_{i=1}^{D}\left(y_i \cos\left(\sqrt{|y_i|}\right)\right) - D\ \leq 0 \quad (\text{Π}.26)$$

$$g_3(x) = \sum_{i=1}^{D}\left(-y_i \cos\left(\sqrt{|y_i|}\right)\right) - 10D\ \leq 0 \quad (\text{Π}.27)$$

$$x \in [-1000{,}1000]^D \quad (\text{Π}.28)$$

Συνάρτηση 15

$$Min\ f(x) = \sum_{i=1}^{D-1}(100\ ({z_i}^2 - {z_{i+1}}^2)^2 + \ ({z_i}^2 - 1)^2) \quad (\text{Π}.29)$$

$$z = x + 1 - o, y = (x - o)M \quad (\text{Π}.30)$$

$$g_1(x) = \sum_{i=1}^{D}\left(-y_i \cos\left(\sqrt{|y_i|}\right)\right) - D\ \leq 0 \quad (\text{Π}.31)$$

$$g_2(x) = \sum_{i=1}^{D}\left(y_i \cos\left(\sqrt{|y_i|}\right)\right) - D\ \leq 0 \quad (\text{Π}.32)$$

$$g_3(x) = \sum_{i=1}^{D}\left(-y_i \cos\left(\sqrt{|y_i|}\right)\right) - 10D\ \leq 0 \quad (\text{Π}.33)$$

$$x \in [-1000,1000]^D \quad (Π.34)$$

Συνάρτηση 16

$$Min\ f(x) = \sum_{i=1}^{D} \frac{{z_i}^2}{4000} - \prod_{i=1}^{D} \cos\left(\frac{z_i}{\sqrt{i}}\right) + 1 \quad (Π.35)$$

$$z = x - o \quad (Π.36)$$

$$g_1(x) = \sum_{i=1}^{D} [{z_i}^2 - 100\cos(\pi z_i) + 10\ ] \leq 0 \quad (Π.37)$$

$$g_2(x) = \prod_{i=1}^{D} z_i \leq 0 \quad (Π.38)$$

$$h_1(x) = \left(z_i \sin\left(\sqrt{|z_i|}\right)\right) = 0 \quad (Π.39)$$

$$h_2(x) = \left(-z_i \sin\left(\sqrt{|z_i|}\right)\right) = 0 \quad (Π.40)$$

$$x \in [-10,10]^D \quad (Π.41)$$

Συνάρτηση 17

$$Minf(x) = \sum_{i=1}^{D-1} (z_i - z_{i+1})^2 \quad (Π.42)$$

$$z = x - o \quad (Π.43)$$

$$g_1(x) = \prod_{i=1}^{D} z_i \leq 0 \quad (Π.44)$$

$$g_2(x) = \sum_{i=1}^{D} z_i \leq 0 \quad (Π.45)$$

$$h(x) = \sum_{i=1}^{D} \left(z_i \sin\left(4\sqrt{|z_i|}\right)\right) = 0 \quad (Π.46)$$

$$x \in [-10,10]^D \quad (Π.47)$$

Συνάρτηση 18

$$Min f(x) = \sum_{i=1}^{D-1} (z_i - z_{i+1})^2 \tag{Π.48}$$

$$z = x - o \tag{Π.49}$$

$$g(x) = \frac{1}{D}\sum_{i=1}^{D}\left(-z_i \sin\left(\sqrt{|z_i|}\right)\right) \le 0 \tag{Π.50}$$

$$h(x) = \frac{1}{D}\sum_{i=1}^{D}\left(z_i \sin\left(\sqrt{|z_i|}\right)\right) = 0 \tag{Π.51}$$

$$x \in [-50,50]^D \tag{Π.52}$$

Συναρτήσεις από εργασία Runarsson & Yao

Παρακάτω παρουσιάζονται οι δύο συναρτήσεις από την εργασία των Runarsson και Yao (Runarsson and Yao, 2000)

Συνάρτηση 12

$$Maximize\ f(\vec{x}) = \frac{(100 - (x_1 - 5)^2 - (x_2 - 5)^2 - (x_3 - 5)^2)}{100} \tag{Π.53}$$

$$g(\vec{x}) = (x_1 - p)^2 + (x_2 - q)^2 + (x_3 - r)^2 - 0.0625 \le 0 \tag{Π.54}$$

Όπου $0 \le x_i \le 10$ (i=1,2,3) και $p, q, r = 1,2, \dots, 9$

Η εφικτή περιοχή (feasible region) αποτελείται από τον χώρο αναζήτησης ο οποίος αποτελείται από 9^3 ασύνδετες σφαίρες. Ένα σημείο (x_1, x_2, x_3) είναι εφικτό μόνο αν υπάρχει συνδυασμός p, q, r τέτοιος ώστε να ισχύει η ανισότητα. Η βέλτιστη λύση είναι $\overrightarrow{x^*} = (5,5,5)$ και $f\left(\overrightarrow{x^*}\right) = 1$

Συνάρτηση 13

$$Minimize\ f(\vec{x}) = e^{x_1 x_2 x_3 x_4 x_5} \tag{Π.55}$$

$$h_1(\vec{x}) = {x_1}^2 + {x_2}^2 + {x_3}^2 + {x_4}^2 + {x_5}^2 - 10 = 0 \tag{Π.56}$$

$$h_2(\vec{x}) = x_2 x_3 - 5x_4 x_5 = 0 \tag{Π.57}$$

$$h_3(\vec{x}) = {x_1}^3 + {x_2}^3 + 1 = 0 \tag{Π.58}$$

Όπου $-2.3 \leq x_i \leq 2.3$ (i=1,2) και $-3.2 \leq x_i \leq 3.2$ (i=3,4,5) Η βέλτιστη λύση είναι $\overrightarrow{x^*} = (-1.717143, 1.595709, 1.827247, -0.7636413, -0.763645)$ και $f\left(\overrightarrow{x^*}\right) = 0.0539498$

Βιβλιογραφία

Adleman, L. M., 1994, Molecular computation of solutions to combinatorial problems. *Science, 266*(5187), 1021-1024.

Bäck, T., Hammel, U., & Schwefel, H. P., 1997, Evolutionary computation: Comments on the history and current state. *Evolutionary computation*, *IEEE Transactions on*, *1*(1), 3-17.

Bean, J. C., & ben Hadj-Alouane, A., 1993, *A dual genetic algorithm for bounded integer programs*.

Bean, J. C., & Norman, B. A., 1993, Random keys for job shop scheduling.

Box, G. E., 1957, Evolutionary operation: A method for increasing industrial productivity. *Appl. Statistics, vol. VI, no. 2*, 81-101.

Bremermann, H. J., 1962, Optimization through evolution and recombination.*Self-organizing systems*, *93*, 106.

Carroll, C. W., 1961, The created response surface technique for optimizing nonlinear, restrained systems. *Operations Research*, *9*(2), 169-184.

Coello, C. A. C., 2000, Use of a self-adaptive penalty approach for engineering optimization problems. *Computers in Industry*, *41*(2), 113-127.

Coello, C. A. C., 2002, Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art.*Computer methods in applied mechanics and engineering*, *191*(11), 1245-1287.

Colorni, A., Dorigo, M., & Maniezzo, V., 1991, Distributed optimization by ant colonies. In *Proceedings of the first European conference on artificial life* (Vol. 142, pp. 134-142).

Courant, R., 1943, Variational methods for the solution of problems of equilibrium and vibrations. *Bull. Amer. Math. Soc*, *49*(1), 1-23.

Darwin, C., & Wallace, A., 1858, On the tendency of species to form varieties; and on the perpetuation of varieties and species by natural means of selection.*Journal of the proceedings of the Linnean Society of London. Zoology*, *3*(9), 45-62.

Darwin, C., 1859, The origin of species. Reprint. *New York: Modern Library*.

Dasgupta, D., & Forrest, S., 1999, Artificial immune systems in industrial applications. In *Intelligent Processing and Manufacturing of Materials, 1999. IPMM'99. Proceedings of the Second International Conference on* (Vol. 1, pp. 257-267). IEEE.

Dasgupta, D., & Michalewicz, Z., 1999, Evolutionary Algorithms in Engineering Applications. *International Journal on Evolutionary Optimization*, *1*, 93-94..

Davidor, Y. (1991). A genetic algorithm applied to robot trajectory generation.*Handbook of Genetic Algorithms.*, 144-165.

Davidor, Y., 1989, Analogous crossover. In *Proceedings of the Third International Conference on Genetic Algorithms* (pp. 98-103). Morgan Kaufmann Publishers Inc..

Davis, L., 1991, Handbook of genetic algorithms.

Dhar, V. and Ranganathan, N., 1990, *Integer Programming vs. Expert Systems: An Experimental Comparison,* Communications of the ACM, 33(3), 323-336.

dos Santos Coelho, L., & Lee, C. S., 2008, Solving economic load dispatch problems in power systems using chaotic and Gaussian particle swarm optimization approaches. *International Journal of Electrical Power & Energy Systems, 30(5), 297-307.*

Eiben, A. E., & Smith, J. E., 2003, What is an Evolutionary Algorithm?, *Introduction to evolutionary computing* (Vol. 53). Heidelberg: springer.

Fiacco, A. V., & McCormick, G. P., 1966, Extensions of SUMT for nonlinear programming: equality constraints and extrapolation. *Management Science,12*(11), 816-828.

Fogel, D. B., 1998, *Evolutionary computation: the fossil record*. Wiley-IEEE Press.

Fogel, L. J., Owens, A. J. and Walsh, M. J., 1966, *Artificial Intelligence Through Simulated Evolution.* New York, NY: John Wiley & Sons.

Forrest, S., Perelson, A. S., Allen, L., and Cherukuri, R., 1994, Self-Nonself Discrimination in a Computer. In Proceedings of IEEE Symposium on Research in Security and Privacy, pages 202--212, Oakland, CA

Fraser, A. S., 1957, Simulation of genetic systems by automatic digital computers. I. Introduction. *Australian Journal of Biological Sciences,* 10, 484-491

Friedberg, R. M. (1958). A learning machine: Part I. *IBM Journal of Research and Development*, *2*(1), 2-13.

Glover, F., 1977, Heuristics for integer programming using surrogate constraints. *Decision Sciences, 8*(1), 156-166.

Glover, F., 1986, Future Paths for Integer Programming and Links to Artificial Intelligence. *Computers and Operations Research, vol. 13*, 533-549.

Hillier, M. S., & Hillier, F. S., 2003, Conventional optimization techniques. In*Evolutionary Optimization* (pp. 3-25). Springer US.

Hoffmeister, F., & Sprave, J., 1996, Problem-independent handling of constraints by use of metric penalty functions.

Holland, J. H., 1975, *Adaptation in Natural and Artificial Systems.* Ann Arbor, MI: The University of Michigan Press.

Homaifar, A., Qi, C. X., & Lai, S. H., 1994, Constrained optimization via genetic algorithms. *Simulation*, *62*(4), 242-253.

Joines, J. A., & Houck, C. R., 1994, On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with GA's. In *Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on* (pp. 579-584). IEEE.

Jong, K. A. D., 1975, *An analysis of the behavior of a class of genetic adaptive systems.* PhD thesis, University of Michigan, Ann Arbor

Kazarlis, S., & Petridis, V., 1998, Varying fitness functions in genetic algorithms: Studying the rate of increase of the dynamic penalty terms. In *Parallel Problem Solving from Nature—PPSN V* (pp. 211-220). Springer Berlin Heidelberg.

Kelly Jr, J. D., & Davis, L., 1991, Hybridizing the Genetic Algorithm and the K Nearest Neighbors Classification Algorithm. In *ICGA* (pp. 377-383).

Kennedy, J., & Eberhart, R. C., 1995, Particle swarm optimization. In *Proceedings of IEEE International Conference on Neural Networks* (Vol. 4, pp. 1942-1948).

Kennedy, J., 1997, The particle swarm: social adaptation of knowledge. In *Evolutionary Computation, 1997., IEEE International Conference on* (pp. 303-308). IEEE.

Kirkpatrick, S., & Vecchi, M. P., 1983, Optimization by simmulated annealing.*science*, *220*(4598), 671-680.

Kobti, Z., Reynolds, R. G., & Kohler, T., 2004, Agent-based modeling of cultural change in swarm using cultural algorithms. *Funded by NSF grant BCS-0119981*.

Kowalczyk, R., 1997, Constraint consistent genetic algorithms. In*Evolutionary Computation, 1997., IEEE International Conference on* (pp. 343-348). IEEE.

Koza, 1990, Genetic Programming: A Paradigm for Genetically Breeding Populations of Computer Programs to Solve Problems.

Larranaga, P., & Lozano, J. A., 2002, *Estimation of distribution algorithms: A new tool for evolutionary computation* (Vol. 2). Springer Science & Business Media.

Le Riche, R., Knopf-Lenoir, C., & Haftka, R. T., 1995, A Segregated Genetic Algorithm for Constrained Structural Optimization. In *ICGA* (pp. 558-565).

Li, C., & Yang, S., 2009, An adaptive learning particle swarm optimizer for function optimization. In *Evolutionary Computation, 2009. CEC'09. IEEE Congress on* (pp. 381-388). IEEE.

Li, C., Yang, S., & Nguyen, T. T., 2012, A self-learning particle swarm optimizer for global optimization problems. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, *42*(3), 627-646.

Liang, J. J., Zhigang, S., and Zhihui, L.,2010, Coevolutionary comprehensive learning particle swarm optimizer. In Evolutionary Computation (CEC), 2010 IEEE Congress on, pages 1–8. IEEE.

Liepins, G. E., & Vose, M. D., 1990, Representational issues in genetic optimization. *Journal of Experimental & Theoretical Artificial Intelligence*, *2*(2), 101-115.

Mallipeddi, R., & Suganthan, P. N., 2010, Problem definitions and evaluation criteria for the CEC 2010 competition on constrained real-parameter optimization. *Nanyang Technological University, Singapore*.

Michalewicz, Z., & Attia, N., 1994, Evolutionary optimization of constrained problems. *Proceedings of the 3rd Annual Conference on Evolutionary Programming* (pp. 98-108). World Scientific.

Michalewicz, Z., & Attia, N., 1994, Evolutionary optimization of constrained problems. In *Proceedings of the 3rd annual conference on evolutionary programming* (pp. 98-108). World Scientific Publishing, River Edge, NJ.

Michalewicz, Z., & Janikow, C. Z., 1991, Handling Constraints in Genetic Algorithms. In *ICGA* (pp. 151-157).

Michalewicz, Z., & Nazhiyath, G.,1995, Genocop III: A co-evolutionary algorithm for numerical optimization problems with nonlinear constraints. In *Evolutionary Computation, 1995., IEEE International Conference on* (Vol. 2, pp. 647-651). IEEE.

Michalewicz, Z., & Schmidt, M., 2003, Evolutionary algorithms and constrained optimization. In *Evolutionary Optimization* (pp. 57-86). Springer US.

Michalewicz, Z., & Schoenauer, M., 1996, Evolutionary algorithms for constrained parameter optimization problems. *Evolutionary Computation, vol. 4, issue 1*.

Michalewicz, Z., 1992, *Genetic Algorithms+Data Structures=Evolution Program, Springer-Verlag, AI Series, New York*

Michalewicz, Z., 1995, Genetic algorithms, numerical optimization, and constraints. In *Proceedings of the sixth international conference on genetic algorithms* (Vol. 195, pp. 151-158). Morgan Kaufmann, San Mateo, CA.

Michalewicz, Z., 1995, Heuristic Methods for Evolutionary Computation Techniques. *Journal of Heuristics*, 1(2), 177-206.

Michalewicz, Z.,1993, A hierarchy of evolution programs: An experimental study. *Evolutionary Computation*.

Morales, A. K., & Quezada, C. V., 1998, A universal eclectic genetic algorithm for constrained optimization. In *Proceedings of the 6th European congress on intelligent techniques and soft computing* (Vol. 1, pp. 518-522).

Moscato, P., 1989, On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts: Towards Memetic Algorithms. *Caltech Concurrent Computation Program (report 826)*.

Powell, D., & Skolnick, M. M., 1993, Using genetic algorithms in engineering design optimization with non-linear constraints. In *Proceedings of the 5th International conference on Genetic Algorithms* (pp. 424-431). Morgan Kaufmann Publishers Inc..

Rechenberg, I. Evolutionsstrategie., 1973, *Optimierung Technischer Systeme Nach Prinzipien der Biologischen Evolution," Friedrich Frommann Verlag*.

Reynolds, R. G., 1994, An introduction to cultural algorithms. In*Proceedings of the third annual conference on evolutionary programming* (pp. 131-139). Singapore.

Runarsson, T. P., & Yao, X., 2000, Stochastic ranking for constrained evolutionary optimization. *Evolutionary Computation, IEEE Transactions on,4*(3), 284-294.

Runarsson, T., & Yao, X., 2003, Constrained evolutionary optimization. In *Evolutionary optimization* (pp. 87-113). Springer US.

Schoenauer, M., & Xanthakis, S., 1993, July, Constrained GA optimization. In*ICGA* (pp. 573-580).

Schwefel, H. P., 1965, *Kybernetische Evolution als Strategie der experimentellen Forschung in der Strom ungstechnik.* Diploma thesis, Technical University of Berlin.

Schwefel, H. P., 1975, *Evolutionsstrategie und numerische Optimierung*. Technische Universität Berlin.

Schwefel, H. P., 1981, *Numerical optimization of computer models*. John Wiley & Sons, Inc..

Storn, R., & Price, K., 1997, Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, *11*(4), 341-359.

Takahama, T., & Sakai, S., 2010, Constrained optimization by the ε constrained differential evolution with an archive and gradient-based mutation. In *Evolutionary Computation (CEC), 2010 IEEE Congress on* (pp. 1-9). IEEE.

Xin Yao, 2003, "Evolutionary Computation, A Gentle Introduction" in Evolutionary Optimization, Kluwer Academic Publishers, New York, Boston, Dordrecht, London, Moscow, pp. 27-53.

Zhan, Z. H., Zhang, J., Li, Y., & Chung, H. S. H., 2009, Adaptive particle swarm optimization. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, *39*(6), 1362-1381.