Μελέτη της μεθόδου Διαφορικής Εξέλιξης
σε προβλήματα συνδυαστικής βελτιστοποίησης
Εφαρμογή στο πρόβλημα του Sudoku

Σεργκέι Κουρνόσενκοβ, B.Sc., ser.corum@gmail.com

Επιβλέπων Καθηγητής: Σταύρος Π. Αδάμ

Τ.Ε.Ι. Ηπείρου

Σχολή Τεχνολογικών Εφαρμογών

Τμήμα Μηχανικών Πληροφορικής

ΑΡΤΑ 2015

Περίληψη / Abstract

Ο σκοπός της παρούσα εργασίας είναι η παρουσίαση της μεθόδου βελτιστοποίησης Διαφορικής Εξέλιξης, με έμφαση σε προβλήματα συνδυαστικής βελτιστοποίησης σε πεπερασμένα σύνολα. Θα γίνει μια εισαγωγή στην γενικότερη έννοια της βελτιστοποίησης συναρτήσεων και την αναζήτηση λύσεων σε πεπερασμένο χώρο, στους εξελικτικούς αλγορίθμους και στην σύλληψη της ιδέας του διαφοροεξελικτικού αλγορίθμου. Έπειτα θα παρουσιασθεί ο κλασσικός Αλγόριθμος Διαφορικής Εξέλιξης (ΑΔΕ) και η λειτουργία του. Θα εξετασθεί σε συνδυαστικά προβλήματα και ειδικότερα στο πρόβλημα του Sukoku. Μετά θα εξεταστούν δύο παραλλαγές του ΑΔΕ, συγκεκριμένα οι προσεγγίσεις του Πίνακα Αντιμετάθεσης και του Σχετικού Δείκτη Θέσης, όπως και μια προσέγγιση που συνδυάζει τις δύο προηγούμενες. Για την άντληση των αποτελεσμάτων η εφαρμογή υλοποιήθηκε στο MATLAB. Τέλος παρουσιάζονται τα αποτελέσματα από την εκτέλεση των προσεγγίσεων για την λύση του προβλήματος και τα συμπεράσματα από την μελέτη αυτών.

Λέξεις Κλειδιά / Keywords: Διαφορική Εξέλιξη, Differential Evolution (DE), Συνδυαστική Βελτιστοποίηση, Combinatorial Optimization, Πίνακας Αντιμετάθεσης, Permutation Matrix (PM), Σχετικός Δείκτης Θέσης, Relative Position Indexing (RPI), Sudoku

Περιεχόμενα

1. Εισαγωγή

Ο σκοπός αυτής της εργασίας είναι να αναλυθεί ο αλγόριθμος της διαφορικής εξέλιξης των Storn και Price σε προβλήματα στον διακριτό χώρο και να γίνει έρευνα για το κατά πόσο μπορεί να εφαρμοσθεί σε τέτοιου είδους προβλήματα και συγκεκριμένα στο πρόβλημα επίλυσης του παιχνιδιού γρίφου Sudoku. Η προσπάθεια επίλυσης γίνεται με δύο προϋπάρχουσες προσεγγίσεις, αυτή του πίνακα αντιμετάθεσης και αυτή με σχετικό δείκτη θέσης, όπως και μια πρωτότυπη που συνδυάζει τις δύο προηγούμενες.

Στο πρώτο κεφάλαιο θα γίνει μία αναφορά για την προέλευση του όρου βελτιστοποίηση συναρτήσεων, τι σημαίνει και πως ορίζεται. Στη συνέχεια θα γίνει μια αναφορά στους γενετικούς αλγόριθμους γενικότερα, μια σύντομη περιγραφή του τρόπου λειτουργείας τους και γιατί ο αλγόριθμος της Διαφορικής Εξέλιξης (ΑΔΕ) κατατάσσεται σε αυτούς.

Στο δεύτερο κεφάλαιο θα γίνει αναφορά στην σύλληψη της ιδέας του ΑΔΕ, τα πλεονεκτήματα και τα μειονεκτήματά του όπως και γιατί η επιστημονική κοινότητα χρειάζεται πιο εξελιγμένους και βελτιωμένους αλγόριθμους βελτιστοποίησης.

Ακολουθεί ο ορισμός του βασικού ΑΔΕ, δηλαδή ποιους τελεστές χρησιμοποιεί και πως αυτοί αρχικοποιούνται. Περιγράφονται αναλυτικά τα βήματα που ακολουθούνται για να παραχθούν οι υποψήφιες λύσεις μέσα από τις οποίες ανευρίσκεται η βέλτιστη λύση. Μετά δίνονται τα δύο σχήματα που έχουν προταθεί από τους Storn και Price για τον ορισμό των τελεστών του αλγορίθμου και τέλος παρουσιάζεται ο τρόπος με τον οποίο συμβολίζεται ο ΑΔΕ.

Στο τρίτο κεφάλαιο εξετάζεται πως ο ΑΔΕ μπορεί να εφαρμοσθεί σε διακριτά συνδυαστικά προβλήματα, ποιές μέθοδοι και προσεγγίσεις έχουν χρησιμοποιηθεί και ποιά είναι η απόδοσή τους.

Μετά ορίζεται ο ΑΔΕ για το πρόβλημα του Sudoku. Γίνεται περιγραφή του Sudoku, πως παίζεται, ποιοι είναι οι κανόνες του παιχνιδιού και πώς μεταφέρεται στο μαθηματικό διακριτό χώρο. Δίνονται οι τροποποιήσεις ή οι αλλαγές που έχουν γίνει στους τελεστές του αλγορίθμου ώστε να προσαρμοσθεί στο Sudoku και ποιες τιμές έχουν ορισθεί σε αυτούς.

Στο τέταρτο κεφάλαιο περιγράφονται αναλυτικά οι μέθοδοι που προσεγγίζονται για την επίλυση του Sudoku και η μεταφορά τους στο MATLAB. Οι εκτελέσιμοι κώδικες για κάθε μια από τις προσεγγίσεις παρατίθεται στο παράρτημα.

Τέλος, στο πέμπτο κεφάλαιο παρουσιάζονται τα αποτελέσματα της εφαρμογής κάθε προσέγγισης και ποια συμπεράσματα απορρέουν από αυτά.

1.1 Βελτιστοποίηση Συναρτήσεων

Με τον όρο βελτιστοποίηση (Wikipedia, 2014) εννοείται η επιλογή ενός ή περισσοτέρων βέλτιστων (καλύτερων) στοιχείων από ένα, συνήθως περίπλοκο, σύστημα με βάση κάποια κριτήρια. Ένα απλό πρόβλημα βελτιστοποίησης αποτελείται, κατά κύριο λόγο, από ένα σύνολο στοιχείων ή καταστάσεων και την ελαχιστοποίηση (ή μεγιστοποίηση) μιας πραγματικής συνάρτησης f. Η συνάρτηση f είναι μια σχέση ανάμεσα στις μεταβλητές του συστήματος. Δέχεται ως είσοδο μέλη του συνόλου. Υπολογίζεται η τιμή της και ανάλογα (για να βρεθεί η βέλτιστη τιμή της συνάρτησης αυτής και επομένως και το βέλτιστο μέλος.) Εάν με την αντικειμενική συνάρτηση (objective function) f αναζητείται το ελάχιστο σε ένα πρόβλημα συνηθίζεται να ονομάζεται συνάρτηση κόστους (cost function) ενώ εάν αναζητείται το μέγιστο, συνάρτηση καταλληλόλητας (fitness function).

Ένα πρόβλημα βελτιστοποίησης μπορεί να αναπαρασταθεί με τον ακόλουθο τρόπο:

> *Δεδομένα*: μια συνάρτηση $f: A \rightarrow R$ από ένα σύνολο A πραγματικών αριθμών. *Ζητούμενα*: ένα στοιχείο x_0 στο A τέτοιο ώστε $f(x_0) \leq f(x)$ για όλα τα x στο A (“ελαχιστοποίηση”) ή $f(x_0) \geq f(x)$ για όλα τα x στο A (“μεγιστοποίηση”).

Ο πρώτος όρος για τη βελτιστοποίηση ήταν ο γραμμικός προγραμματισμός ο οποίος αναπτύχθηκε από τον Leonid Kantorovich το 1939 και χρησιμοποιήθηκε κατά τη διάρκεια του Β’ Παγκοσμίου Πολέμου για τον υπολογισμό και τον προγραμματισμό των δαπανών προκειμένου να μειωθεί το κόστος του στρατού και να αυξηθούν οι εχθρικές απώλειες. Η μέθοδος παρέμεινε κρυφή μέχρι την δημοσίευση της μεθόδου simplex από τον George B. Dantzig το 1947 και την ανάπτυξη της θεωρίας δυαδικότητας από τον John von Neumann. Έκτοτε έχει χρησιμοποιηθεί από πολλές βιομηχανίες για τον ημερήσιο προγραμματισμό τους.

Ο βασικότερος διαχωρισμός των μεθόδων βελτιστοποίησης είναι σε δύο κατηγορίες, τις διακριτές και τις συνεχείς (Κόλιας, 2012). Ο διαχωρισμός αυτός προέρχεται από τη μορφή του πεδίου ορισμού της προς βελτιστοποίηση αντικειμενικής συνάρτησης. Όταν το πεδίο ορισμού της αντικειμενικής συνάρτησης είναι συνεχές αυτό σημαίνει ότι οι μεταβλητές της αντικειμενικής συνάρτησης μπορούν να πάρουν, θεωρητικά, άπειρες τιμές. Ένα παράδειγμα τέτοιας συνάρτησης είναι η κυματομορφή ενός αναλογικού σήματος, όπου για κάθε δεδομένη χρονική στιγμή η τιμή της ποσότητας που περιγράφει μπορεί να είναι οποιαδήποτε. Όταν το πεδίο ορισμού είναι διακριτό αυτό σημαίνει ότι οι μεταβλητές της αντικειμενικής συνάρτησης μπορούν να πάρουν συγκεκριμένες, αριθμήσιμες και πεπερασμένες τιμές. Ένα παράδειγμα διακριτών μεταβλητών είναι οι μαθητές ανά τάξη σε ένα σχολείο.

Ένας ακόμα διαχωρισμός μεθόδων βελτιστοποίησης είναι οι αιτιοκρατικές και οι στοχαστικές μέθοδοι. Ο διαχωρισμός αυτός γίνεται με βάση τον τρόπο με τον οποίο αναζητάτε η βέλτιστη λύση.

Οι αιτιοκρατικές μέθοδοι (deterministic methods) στηρίζονται στη βαθμιαία ολίσθηση μιας αρχικής λύσης προς τη βέλτιστη μέσω απλών βηματικών διορθώσεων. Δυστυχώς, κάθε τροποποίηση στη συνάρτηση κόστους ή στη μοντελοποίηση του προβλήματος απαιτεί επαναδιατύπωση του μοντέλου βελτιστοποίησης. Παρότι οι αιτιοκρατικές μέθοδοι συνήθως συγκλίνουν προς το ολικό ακρότατο είναι πολλές φορές δύσκολο να ρυθμιστούν σωστά οι παράμετροί τους και απαιτούν μεγάλη υπολογιστική ισχύ και χρόνο.

Οι στοχαστικές μέθοδοι (stochastic methods) από την άλλη, βασίζονται στην συγκροτημένα τυχαία αναζήτηση νέων λύσεων, καλύτερων ως προς την υπάρχουσα, που τελικά οδηγεί στη βέλτιστη. Βασικό χαρακτηριστικό τους είναι η ταχύτητα και συνήθως η μειωμένη απαίτηση σε υπολογιστική ισχύ. Οι πρώτες στοχαστικές μέθοδοι είχαν το μειονέκτημα ότι παγιδεύονταν σε τοπικά ακρότατα αλλά έκτοτε το πρόβλημα αυτό έχει εξαλειφθεί. Λόγο της εύκολης εφαρμογής τους σε μεγάλο πλήθος προβλημάτων χρησιμοποιούνται σε πολλά συνεχή και συνδυαστικά προβλήματα βελτιστοποίησης.

1.2 Οι Εξελικτικοί Αλγόριθμοι

Με τον όρο Εξελικτικοί Αλγόριθμοι (ΕΑ) χαρακτηρίζονται όλοι οι αλγόριθμοι που είναι βασισμένοι στους μηχανισμούς της βιολογικής εξέλιξης, όπως η αναπαραγωγή, η μετάλλαξη, ο ανασυνδυασμός, ο επανασυνδυασμός, η φυσική επιλογή και η επιβίωση του καταλληλότερου. (Αδαμίδης, 1999). Οι ΕΑ μιμούνται τις διαδικασίες βιολογικής εξέλιξης με την υλοποίηση των ιδεών της φυσικής επιλογής και της επικράτησης του ισχυρότερου, έτσι ώστε να παρέχουν αποτελεσματικές λύσεις σε προβλήματα αναζήτησης και βελτιστοποίησης.

Οι ΕΑ διαχωρίζονται και σε επιμέρους εξελικτικά υπολογιστικά μοντέλα όπως σε Γενετικούς Αλγόριθμους, Γενετικό Προγραμματισμό, Εξελικτικές Στρατηγικές και Εξελικτικό Προγραμματισμό, τα οποία όμως βασίζονται στις ίδιες αρχές, δηλ. στις αρχές προσομοίωσης της εξέλιξης μέσω των διαδικασιών της επιλογής, της αναπαραγωγής και της μετάλλαξης.

Η βασική δομή ενός ΕΑ μπορεί να περιγραφεί ως εξής (Παρσόπουλος, 2005):

> Έστω P ένα σύνολο από στοιχεία, τα οποία είναι μέλη ενός πληθυσμού, καθένα από τα οποία αποτελεί μια πιθανή λύση του προβλήματος για μια δεδομένη γενιά g.
>
> Η γενιά g είναι ένας ακέραιος αριθμός που υποδηλώνει την επανάληψη στην οποία βρίσκεται ο αλγόριθμος.

$$P = \{x_1(g), x_2(g), x_3(g), \dots, x_n(g)\}.$$

f η αντικειμενική συνάρτηση, δηλαδή μια συνάρτηση που δείχνει τη σχέση μεταξύ των στοιχείων και το ζητούμενο του προβλήματος.

$$F = \{f(x_1(g)), f(x_2(g)), f(x_3(g)), \dots , f(x_n(g))\}.$$

Αρχικά υπολογίζεται η αντικειμενική συνάρτηση για κάθε μέλος του πληθυσμού. Μετά παράγονται απόγονοι με έναν ή και περισσότερους τρόπους εξέλιξης, αναπαραγωγή, μετάλλαξη, ανασυνδυασμός ή φυσική επιλογή. Αξιολογούνται τα νέα μέλη, και αν δεν ισχύουν τα κριτήρια τερματισμού ο αλγόριθμος επαναλαμβάνεται.

1.2.1 Γενετικοί Αλγόριθμοι

Οι γενετικοί αλγόριθμοι (ΓΑ) προτάθηκαν από τον John Holland στα τέλη του 1960. Από τότε έχουν μελετηθεί από πάρα πολλούς επιστήμονες και έχουν χρησιμοποιηθεί σε μια πληθώρα εφαρμογών σε διάφορα πεδία επιστημών. Λόγω της μεγάλης απήχησής τους εξελίχθηκαν ραγδαία και ως εκ τούτου δημιουργήθηκαν πολλές παραλλαγές αυτών.

Η βασική ιδέα των ΓΑ πηγάζει από τους μηχανισμούς εξέλιξης στην φύση. Οι μηχανισμοί αυτοί είναι η μετάλλαξη, η φυσική επιλογή και η διασταύρωση. Για να υλοποιηθεί ένας ΓΑ θα πρέπει οι παράμετροι του συστήματος να κωδικοποιηθούνε με τέτοιον τρόπο ώστε να μπορούν να αναπαρασταθούν από μια σειρά διακριτών χαρακτηριστικών. Μια συχνά χρησιμοποιούμενη αναπαράσταση είναι κάθε μέλος να αποτελείται από μια σειρά δυαδικών ψηφίων (bits). Έτσι οι βασικοί όροι είναι οι παρακάτω (Γούδος, 2011):

- *Γονίδιο*: είναι η μεταβλητή της οποίας αναζητείται η βέλτιστη τιμή και επομένως και η λύση στο πρόβλημα.
- *Χρωμόσωμα*: εάν για να βρεθεί η λύση στο πρόβλημα χρειάζονται περισσότερες από μια μεταβλητές, τότε μια σειρά από γονίδια αποτελεί το χρωμόσωμα.
- *Πληθυσμός*: το σύνολο των πιθανών λύσεων. Όλα τα γονίδια ή χρωμοσώματα μέσα από τα οποία ο αλγόριθμος αναζητά την βέλτιστη λύση ονομάζεται πληθυσμός.
- *Γενιά*: η επανάληψη στην οποία βρίσκεται ο αλγόριθμος. Σε κάθε γενιά δημιουργούνται νέα μέλη του πληθυσμού με μηχανισμούς εξέλιξης.
- *Γονείς*: όλα τα μέλη του πληθυσμού κατά την τρέχουσα γενιά. Οι γονείς διασταυρώνονται με κάποια στοχαστικά κριτήρια για να παράξουν απογόνους.
- *Απόγονοι:* τα μέλη που παράγονται από την διασταύρωση των γονέων που, ανάλογα με τα κριτήρια βελτιστοποίησης, θα περάσουν στην επόμενη γενιά ή θα απορριφθούν.

- *Διασταύρωση:* ένας μηχανισμός ο οποίος ανταλλάσσει χαρακτηριστικά ανάμεσα σε δύο μέλη του πληθυσμού. Ο συνδυασμός των γενετικών πληροφοριών περνά από τους γονείς στους απογόνους προσδίδοντάς τους νέα χαρακτηριστικά. Η διασταύρωση μπορεί να γίνει σε ένα σημείο ή και περισσότερα. Παράδειγμα διασταύρωσης ενός και δύο σημείων σε δυαδική αναπαράσταση μεταβλητών:

Διασταύρωση	*Ενός σημείου*	*Δύο σημείων*
Γονέας #1	101101 | 101001	101 | 1011010 | 01
Γονέας #2	101001 | 011101	101 | 0010111 | 01
Απόγονος #1	101101 | 011101	101 | 0010111 | 01
Απόγονος #2	101001 | 101001	101 | 1011010 | 01

- *Μετάλλαξη:* μετά από κάθε διασταύρωση υπάρχει μια πολύ μικρή πιθανότητα ο απόγονος να μεταλλαχθεί. Η μετάλλαξη μπορεί να είναι σημειακή ή και πολυγονιδιακή. Παράδειγμα μετάλλαξης σε δυαδική αναπαράσταση μεταβλητών:

Μετάλλαξη	*Ενός σημείου*	*Δύο σημείων*
Απόγονος #1	101101101001	101101101001
Μετάλλαξη #1	101001011101	101001101011
Απόγονος #2	101001011101	101001011101
Μετάλλαξη #2	101001011001	001001111101

- *Επιλογή:* η επιλογή των γονέων προς διασταύρωση μέσα από όλο το σύνολο των μελών. Η επιλογή μπορεί να είναι απολύτως τυχαία, αλλά μπορεί να βασίζεται και στον μηχανισμό επιβίωσης του ισχυρότερου από την φυσική εξέλιξη. Με την απόλυτα τυχαία επιλογή ο χώρος αναζήτησης του αλγορίθμου διευρύνεται ενώ με τον μηχανισμό επιβίωσης του ισχυρότερου ο αλγόριθμος συγκλείνει προς μια βέλτιστη λύσει. Η επιλογή, σε κάθε περίπτωση, παραμένει βασισμένη στην πιθανότητα δίνοντας έτσι την δυνατότητα ακόμα και σε μέλη που δεν είναι ισχυρά να παράξουν απογόνους.

Τα βήματα ενός ΓΑ:

1. Αρχικοποίηση των μελών του πληθυσμού. Η αρχικοποίηση μπορεί να περιέχει και μετατροπή των μεταβλητών σε μορφή κατάλληλη για τον αλγόριθμο.
2. Υπολογισμός της τιμής της συνάρτησης καταλληλότητας, με βάση το γονίδιο ή το χρωμόσωμα, για κάθε μέλος.

3. Δημιουργία απογόνων.
    a. Επιλογή δύο γονέων από τον πληθυσμό ανάλογα με την τιμή της συνάρτησης καταλληλότητάς τους.
    b. Διασταύρωση των επιλεγμένων γονέων για να παραχθούν δύο απόγονοι.
    c. Μετάλλαξη των απογόνων.
    d. Αποδοχή απογόνων στον νέο πληθυσμό. Αυτό το βήμα είναι προαιρετικό. Η αποδοχή μπορεί να γίνει με ποικίλους τρόπους.
4. Επανάληψη του βήματος 3 μέχρις ότου να συμπληρωθεί ο νέος πληθυσμός.
5. Υπολογισμός της τιμής της συνάρτησης καταλληλότητας για κάθε μέλος του δημιουργηθέντος πληθυσμού.
6. Έλεγχος των κριτηρίων τερματισμού. Εάν έχει βρεθεί η βέλτιστη λύση ή ισχύει κάποιο άλλο κριτήριο τερματισμού τερματισμός του αλγορίθμου, αλλιώς επιστροφή στο βήμα 3.

1.2.2 Γενετικός Προγραμματισμός

Ο Γενετικός Προγραμματισμός (ΓΠ) είναι μια αυτοματοποιημένη μέθοδος, εμπνευσμένη και αυτή από την φυσική εξέλιξη, που καλείται να βρει το βέλτιστο υπολογιστικό πρόγραμμα από την εξέλιξη άλλων προγραμμάτων. Ο ΓΠ χρησιμοποιεί την αναπαραγωγή, την διασταύρωση, την μετάλλαξη και την λειτουργία μετατροπής αρχιτεκτονικής για να δημιουργήσει νέα τροποποιημένα εκτελέσιμα προγράμματα, ή κομμάτια κώδικα, ώστε αυτά με τη σειρά τους να αναζητήσουν την βέλτιστη λύση.

1. Δημιουργία τυχαίου αρχικού πληθυσμού από προγράμματα υπολογιστή που σχηματίζονται από διαθέσιμες συναρτήσεις.

2. Επανάληψη των παρακάτω επιμέρους βημάτων:

α. Εκτέλεση κάθε προγράμματος του πληθυσμού και εξακρίβωση της καταλληλότητάς του.

β. Επιλογή ενός ή δύο προγραμμάτων από τον πληθυσμό με μια πιθανότητα που βασίζεται στην καταλληλότητα για να συμμετέχουν στο επόμενο βήμα.

γ. Δημιουργία νέων προγραμμάτων για τον πληθυσμό εφαρμόζοντας τους ακόλουθους γενετικούς τελεστές, με προκαθορισμένες τιμές πιθανότητας:

• Αναπαραγωγή: Αντιγραφή επιλεγμένων προγραμμάτων στον νέο πληθυσμό.

• Διασταύρωση: Δημιουργία απογόνων για τον νέο πληθυσμό, ανασυνδυάζοντας τυχαία επιλεγμένα τμήματα από τα δύο επιλεγμένα προγράμματα.

• Μετάλλαξη: Διαφοροποίηση ενός απογόνου για το νέο πληθυσμό, μεταλλάσσοντας ένα τυχαία επιλεγμένο τμήμα του με ένα από τα επιλεγμένα μέλη.

• Λειτουργίες μετατροπής αρχιτεκτονικής: Επιλογή μίας λειτουργίας μετατροπής αρχιτεκτονικής από τη διαθέσιμη συλλογή τέτοιων λειτουργιών και εφαρμογή στον επιλεγμένο απόγονο.

3. Επανάληψη του αλγορίθμου έως ότου ικανοποιηθεί το κριτήριο τερματισμού. Το πρόγραμμα με τις καλύτερες επιδόσεις ή αποτελέσματα είναι η μέχρι στιγμής η βέλτιστη λύση.

1.2.3 Εξελικτικές Στρατηγικές

Οι Εξελικτικές Στρατηγικές (ΕΣ) αναπτύχθηκαν από τους Rechenberg και Schwefel (Schwefel, 2001). Χρησιμοποιούνται κατά κύριο λόγο σε εφαρμογές αριθμητικής βελτιστοποίησης. Οι ΕΣ χρησιμοποιούν δύο τελεστές, αυτούς του ανασυνδυασμού και της μετάλλαξης, μεταβάλλοντας ταυτόχρονα τόσο τον πληθυσμό όσο και τις παραμέτρους στρατηγικής. Το πλήθος των απογόνων απαιτείται να είναι μεγαλύτερο από αυτό του πληθυσμού των γονέων, ώστε να διατηρούνται οι θεωρητικές ιδιότητες σύγκλισης υπό συνθήκες. Υπάρχουν δύο γενικές κατηγορίες ΕΣ, οι $(\mu + \lambda)$ και οι (μ , λ). Στις $(\mu + \lambda)$-ΕΣ, τα μ καλύτερα άτομα που θα αποτελέσουν τον πληθυσμό της νέας γενιάς επιλέγονται ανάμεσα σε όλους τους απογόνους και τους γονείς τους, ενώ στις (μ , λ)-ΕΣ, η επιλογή των μ ατόμων του νέου πληθυσμού γίνεται μονάχα μεταξύ των απογόνων. Έτσι, στην περίπτωση των $(\mu + \lambda)$-ΕΣ, τα καλύτερα άτομα που βρέθηκαν διατηρούνται στο νέο πληθυσμό σε κάθε γενιά, ενώ, στην περίπτωση των (μ , λ)-ΕΣ, τα άτομα αυτά μπορεί να χαθούν, γεγονός που κάνει την αναζήτηση πιο αργή αλλά αυξάνει την ποικιλότητα (diversity) του πληθυσμού.

1.2.4 Εξελικτικός Προγραμματισμός

Ο εξελικτικός προγραμματισμός (ΕΠ) χρησιμοποιήθηκε για πρώτη φορά από τον Lawrence J. Fogel στις ΗΠΑ το 1960, προκειμένου να χρησιμοποιήσει προσομοιωμένη εξέλιξη ως μια διαδικασία μάθησης με στόχο να δημιουργήσει τεχνητή νοημοσύνη. Ο Fogel χρησιμοποίησε μηχανές πεπερασμένων καταστάσεων ως προγνωστικούς παράγοντες και τους εξέλιξε. Επί του παρόντος, ο εξελικτικός προγραμματισμός είναι μια μεγάλη διάλεκτος εξελικτικού υπολογισμού χωρίς καθορισμένη δομή ή (αναπαράσταση), σε αντίθεση με ορισμένες από τις άλλες διαλέκτους. Με αποτέλεσμα η διάκριση από τις εξελικτικές στρατηγικές να είναι δύσκολη.

Η βασική ιδέα είναι ίδια με τους προηγούμενους αλγορίθμους, μόνο που στην περίπτωση του ΕΠ η μέθοδος που εφαρμόζεται για διαφοροποίηση των απογόνων είναι μόνο η μετάλλαξη. Έτσι από τους γονείς παράγονται απόγονοι μόνο μέσω της μετάλλαξης και έπειτα αξιολογούνται τα νέα μέλη από μια συνάρτηση καταλληλότητας.

Μια μεγάλη διαφορά από τους υπόλοιπους εξελικτικούς αλγορίθμους είναι η χρήση μόνο της μετάλλαξης για διαφοροποίηση. Λόγω αυτού οι μεταβλητές δεν χρειάζονται κωδικοποίηση και μπορούν να είναι πολλών και ποικίλων διαφορετικών τύπων.

2. Ο Αλγόριθμος Διαφορικής Εξέλιξης

Παρόλο που η ανάπτυξη της μεθόδου της διαφορικής εξέλιξης δεν επινοήθηκε με βάση τις φυσικές διαδικασίες, όπως στην περίπτωση των Εξελικτικών Στρατηγικών, Εξελικτικού Προγραμματισμού και Γενετικών Αλγορίθμων, εντούτοις κατατάσσεται στους Εξελικτικούς Αλγορίθμους λόγω των πολλών ομοιοτήτων των τελεστών του με αυτούς όπως θα δούμε στη συνέχεια.

Προβλήματα που αφορούν την ολική βελτιστοποίηση πάνω σε συνεχείς χώρους υπάρχουν παντού στην επιστημονική κοινότητα. Οι συνεχείς μεταβλητές είναι πραγματικοί ή ακόμα και μιγαδικοί αριθμοί που μπορούν να πάρουν οποιαδήποτε τιμή σε ένα δεδομένο διάστημα. Συνήθως αυτές είναι μεταβλητές που αφορούν μετρήσεις φυσικών ποσοτήτων, όπως οι συντεταγμένες, η θερμοκρασία, η ένταση του ρεύματος, ο χρόνος, το βάρος, κ.τ.λ. Έτσι έχουν δημιουργηθεί πολλές τεχνικές που προσπαθούν να δώσουν λύσεις σε αυτά, συνήθως στοχεύοντας σε όσο το δυνατόν μεγαλύτερο εύρος προβλημάτων. Σε γενικές γραμμές, ο στόχος τους είναι η βελτιστοποίηση ορισμένων ιδιοτήτων ενός συστήματος επιλέγοντας εύστοχα, σε σχέση με το τι θέλουμε να βελτιστοποιήσουμε, τις παραμέτρους του συστήματος. Συνήθως οι παράμετροι ενός συστήματος αναπαρίστανται με ένα διάνυσμα. Μια πρακτική τεχνική βελτιστοποίησης θα πρέπει να πληροί τρείς προϋποθέσεις.

- Πρώτον, η μέθοδος θα πρέπει να βρίσκει το πραγματικό ολικό ελάχιστο (ή μέγιστο), ανεξάρτητα από τις αρχικές τιμές των παραμέτρων του συστήματος.
- Δεύτερον, θα πρέπει να γίνεται ταχεία σύγκλιση προς τη βέλτιστη λύση.
- Και τρίτον, ο αλγόριθμος θα πρέπει να έχει ελάχιστες παραμέτρους ελέγχου, ώστε να είναι εύκολος στη χρήση.

Στην αναζήτησή τους λοιπόν, ο Storn και ο Price, για μια γρήγορη και εύκολη στη χρήση τεχνική, ανέπτυξαν μια μέθοδο η οποία δεν είναι μόνο απλή, αλλά αποδίδει καλά σε μια ευρεία ποικιλία προβλημάτων. Η βασική στρατηγική εφαρμόζει τη διαφορά δύο τυχαία επιλεγμένων διανυσμάτων ως πηγή για διαφοροποίηση ενός τρίτου διανύσματος. Η μέθοδος αυτή ονομάστηκε Διαφορική Εξέλιξη (Storn & Price, 1995).

Τα πλεονεκτήματά του είναι κυρίως τρία:

- Βρίσκει το πραγματικό ελάχιστο ανεξαρτήτως των αρχικών τιμών των παραμέτρων.
- Παρουσιάζει γρήγορη σύγκλιση.
- Κάνει χρήση λίγων παραμέτρων για των έλεγχό του.

Άλλα πλεονεκτήματα είναι η απλότητά του, η ταχύτητα και η ευκολία στην χρήση του.

Τα μειονεκτήματά του είναι:

- Επειδή βασίζεται πολύ στις λίγες παραμέτρους ελέγχου του, για να έχει αποδοτική σύγκλιση, χρειάζεται πολύ προσεκτική επιλογή αυτών.
- Είναι δύσκολο να βρεθούν οι σωστές τιμές για τις παραμέτρους ελέγχου σε όχι καλά ή πλήρως αναλυμένα προβλήματα.
- Στην απλή του μορφή, υπάρχει η περίπτωση να παγιδευτεί σε τοπικά ελάχιστα.

Όσο η επιστήμη και η τεχνολογία εξελίσσονται τα προβλήματα αποκτούν μεγαλύτερη περιπλοκότητα και χώρους αναζήτησης με συναρτήσεις που απαιτούν τεράστια υπολογιστική ισχύ για να επιλυθούν. Για τον λόγο αυτό υπάρχει η ανάγκη για πιο γρήγορους και αποδοτικότερους αλγορίθμους βελτιστοποίησης.

Η μέθοδος της Διαφορικής Εξέλιξης έχει συγκριθεί με πολλές άλλες μεθόδους βελτιστοποίησης σε δοκιμαστικές συναρτήσεις με πολύ καλά αποτελέσματα είτε στην ανεύρεση της βέλτιστης λύσης είτε στην ταχύτητα σύγκλισης, είτε και στα δύο (Price, Storn, & Lampinen, 2005).

Έχει επίσης δοκιμαστεί και εφαρμοστεί σε πολλές πραγματικές εφαρμογές όπως είναι η ρομποτική, η σεισμολογία, η αεροδυναμική, η ανάλυση κυκλωμάτων, η ανάλυση εικόνων, η χημεία, κ.α. με πολύ καλά αποτελέσματα και συνήθως καλύτερες επιδόσεις από άλλες μεθόδους βελτιστοποίησης.

2.1 Ορισμός του Αλγόριθμου Διαφορικής Εξέλιξης (ΑΔΕ)

Αναλυτική περιγραφή του κλασικού αλγόριθμου διαφορικής εξέλιξης. Έστω *NP* το μέγεθος του πληθυσμού και $\vec{X}_{i,G}$, με i=1,2,…,*NP* μία υποψήφια λύση αυτού του πληθυσμού για τη γενιά g. Ο χρήστης ορίζει με βάση τη μορφή του προβλήματος τον αριθμό *D* των μεταβλητών της αντικειμενικής συνάρτησης (προαιρετικό), την παράμετρο ελέγχου διαταραχής Q, τον συντελεστή διασταύρωσης CR καθώς και το μέγεθος του πληθυσμού NP.

Για την πρώτη γενιά (g=0) επιλέγεται με τυχαίο τρόπο πληθυσμός *NP* λύσεων, πραγματοποιείται υπολογισμός των τιμών της συνάρτησης κόστους F($\vec{X}_{i,0}$) για κάθε i=1,2,…,NP και εντοπίζεται η υποψήφια λύση του πληθυσμού με τη μικρότερη συνάρτηση κόστους.
Ακολούθως ξεκινά η εξελικτική διαδικασία του αλγόριθμου για κάθε γενιά του πληθυσμού. Κάθε γενιά αποτελείται από μια εσωτερική επανάληψη που πραγματοποιείται για κάθε μέλος της τρέχουσας γενιάς.

2.2 Τα βασικά βήματα του ΑΔΕ

Τα παρακάτω βήματα είναι για τον βασικό ΑΔΕ στον χώρο των πραγματικών αριθμών. Παρουσιάζονται τα δύο πρώτα σχήματα που πρότειναν οι Storn και Price (Storn & Price, 1995) Η συνδυαστική βελτιστοποίηση σε διακριτό χώρο θα αναλυθεί σε επόμενο κεφάλαιο.

Βήμα 1: Τίθεται *g*=0.

Βήμα 2: Τίθεται *i*=1.

Βήμα 3: Με διαδικασία που θα αναλυθεί αναλυτικότερα παρακάτω, παράγεται το "δοκιμαστικό διάνυσμα-γονέας" $\vec{V}_i$ και το οποίο διασταυρώνεται με το διάνυσμα $\vec{X}_{i,G}$ (τονίζεται το γεγονός ότι το διάνυσμα $\vec{X}_{i,G}$ είναι μέλος του πληθυσμού της G γενιάς, ενώ το $\vec{V}_i$ δεν ανήκει στον τρέχοντα πληθυσμό), παράγοντας έτσι την υποψήφια λύση $\vec{U}_i$ ως εξής:

$$\vec{U}\mathrm{i} = \begin{cases} \vec{V}\mathrm{i} + 1 \text{ αν rand } <= \text{ CR} \\ \vec{X}\mathrm{i} \text{ αν rand } > \text{CR} \end{cases}$$

Όπου το CR (συντελεστής διασταύρωσης) δίνεται από τον χρήστη και είναι ένας αριθμός στο διάστημα [0,1].

Βήμα 4: Αξιολογείται η υποψήφια λύση $\vec{U}_i$ με υπολογισμό της τιμής της συνάρτησης κόστους $F(\vec{U}_i)$. Αν $F(\vec{U}_i) < F(\vec{X}_{i,G})$, τότε η υποψήφια λύση $\vec{U}_i$ παίρνει τη θέση της λύσης $\vec{X}_{i,G}$ στον πληθυσμό, αλλιώς η λύση $\vec{U}_i$ απορρίπτεται και η λύση $\vec{X}_{i,G}$ διατηρεί τη θέση της στον πληθυσμό.

Βήμα 5: Αυξάνεται η τιμή της παραμέτρου *i* κατά 1 (δηλαδή i=i+1) και επαναλαμβάνεται η διαδικασία από το βήμα 3 έως το βήμα 5 για όλα τα μέλη του πληθυσμού, δηλαδή μέχρι να πάρει το *i* την τιμή *i=NP*, όπου και συνεχίζει ο αλγόριθμος πηγαίνοντας στο βήμα 6.

Βήμα 6: Εντοπίζεται η υποψήφια λύση με τη μικρότερη τιμή της συνάρτησης κόστους για το πρόβλημα βελτιστοποίησης και εφαρμόζεται το κριτήριο σύγκλισης. Εάν δε θεωρηθεί ότι η μέθοδος παρουσιάζει σύγκληση, προχωράει ο αλγόριθμος στην επόμενη γενιά, επαναλαμβάνοντας τα βήματα 1 έως 6, μέχρι να ικανοποιηθεί το κριτήριο σύγκλισης της μεθόδου (ή η συνθήκη επανάληψης).

Τελεστής Παραγωγής Δοκιμαστικού Διανύσματος-Γονέα

Για τη δημιουργία του δοκιμαστικού διανύσματος γονέα, προτάθηκαν από τους Storn και Price τα εξής δύο σχήματα:

Πρώτο Σχήμα

Με τυχαίο τρόπο επιλέγονται 3 μέλη του πληθυσμού, διαφορετικά μεταξύ τους και διαφορετικά από το μέλος του πληθυσμού με το οποίο θα διασταυρωθεί το δοκιμαστικό διάνυσμα. Από αυτά τα 3 διανύσματα, σχηματίζεται το δοκιμαστικό διάνυσμα:

$$\vec{V}\mathrm{i} = \vec{X}\mathrm{r}_{1,G} + Q\cdot(\vec{X}\mathrm{r}_{2,G} - \vec{X}\mathrm{r}_{3,G})$$

με $r_1, r_2, r_3 \in [1,NP]$, ακέραιοι και διάφοροι μεταξύ τους, με $Q>0$ (συνήθως μέχρι 1, σπάνια υπάρχει βελτίωση πάνω από τη μονάδα).

Οι ακέραιοι r_1, r_2 και r_3 έχουν επιλεχθεί με τυχαίο τρόπο, είναι διάφοροι μεταξύ τους και διάφοροι της τιμής i, δηλαδή του μέλος του πληθυσμού με το οποίο θα διασταυρωθεί το δοκιμαστικό μας διάνυσμα. Ο συντελεστής *Q* (συνηθίζεται στη βιβλιογραφία να γράφεται και ως *F)*, ορίζεται από τον χρήστη και έχει σταθερή τιμή διάφορη του μηδενός.

Όσο μεγαλύτερος είναι ο όρος $Q\cdot(\vec{X}\mathrm{r}_{2,G} - \vec{X}\mathrm{r}_{3,G})$ τόσο μεγαλύτερη διασπορά εμφανίζεται στους απογόνους του πληθυσμού, έτσι ο όρος *Q* ονομάζεται συντελεστής διασποράς.

Η διασπορά αυτή όμως μπορεί να είναι επιθυμητή για τις πρώτες γενιές της μεθόδου, στις οποίες πραγματοποιείται ανίχνευση του χώρου των πιθανών λύσεων, δεν είναι όμως το ίδιο επιθυμητή για τις επόμενες γενιές, κατά τις οποίες επιθυμείται περισσότερο η προσέγγιση των ήδη ανιχνευμένων ακροτάτων και λιγότερο η εύρεση νέων ακροτάτων.

Συνηθίζεται, το διάνυσμα $\vec{X}\mathrm{r}_{1,G}$ να ονομάζεται διάνυσμα δότης (donor), καθότι σε αυτό προστίθεται η διαφορά των άλλων δύο διανυσμάτων-λύσεων.

Δεύτερο Σχήμα

Στο δεύτερο σχήμα, κατά τη διαδικασία δημιουργίας του δοκιμαστικού διανύσματος συμμετέχουν 2 τυχαία μέλη του πληθυσμού, το μέλος i του πληθυσμού που θα διασταυρωθεί με το δοκιμαστικό διάνυσμα και το μέλος του πληθυσμού για το οποίο έχουμε υπολογίσει έως τώρα τη μικρότερη τιμή της αντικειμενικής συνάρτησης.

Απαραίτητη προϋπόθεση είναι τα 2 μέλη του πληθυσμού που επιλέγονται με τυχαίο τρόπο να είναι διάφορα μεταξύ τους και να μην είναι κάποιο από αυτά το μέλος i του πληθυσμού.

Συγκεκριμένα, έχουμε:

$$\vec{V}\mathrm{i} = \vec{X}_{\mathrm{i},G} + \lambda\cdot(\vec{X}_{\mathrm{best},G} - \vec{X}_{\mathrm{i},G}) + Q\cdot(\vec{X}\mathrm{r}_{2,G} - \vec{X}\mathrm{r}_{3,G})$$

όπου, $\mathrm{i}, \mathrm{r}_2, \mathrm{r}_3 \in \mathbf{Z}$, $\mathrm{r}_2 \neq \mathrm{r}_3 \neq \mathrm{i}$, $\mathrm{i}, \mathrm{r}_2, \mathrm{r}_3 \in [1,\mathrm{NP}]$, $\lambda>0$ και $Q>0$. Ο συντελεστής λ επίσης ορίζεται εκ των προτέρων από το χρήστη και πρέπει να είναι πάντα μεγαλύτερος του μηδενός, όπως και ο συντελεστής Q που έχει ήδη αναφερθεί από τη θεωρία του πρώτου σχήματος.

Όσο μεγαλύτερος ο συντελεστής λ, τόσο περισσότερο προσεγγίζουν οι επόμενες γενιές την έως εκείνη τη στιγμή βέλτιστη λύση που έχει υπολογιστεί. Κάτι τέτοιο μπορεί να είναι επιθυμητό στις τελευταίες γενιές του αλγόριθμου, λειτουργεί όμως ενάντια στην επιθυμία για διασπορά κατά τις πρώτες γενιές.

Να σημειωθεί ότι ανεξάρτητα της μεθόδου που επιλέγεται για τη δημιουργία του δοκιμαστικού διανύσματος, πρέπει πάντα να γίνεται έλεγχος ότι οι τιμές των ελεύθερων μεταβλητών που προκύπτουν βρίσκονται εντός των ορίων που έχει θέσει ο χρήστης. Σε περίπτωση που υπερβαίνουν κάποιο όριο, μπορούμε είτε να θέσουμε αυτή την ελεύθερη μεταβλητή ίση με το όριο το οποίο ξεπέρασε, είτε την τιμή εντός των αποδεκτών ορίων που να απέχει όμως από το όριο, όσο θα απείχε εάν το είχε ξεπεράσει.

2.3 Συμβολισμοί

Μαζί με τον αλγόριθμο ΔΕ εμφανίστηκε και ένας συμβολισμός για τις διάφορες στρατηγικές της ΔΕ. Ο συμβολισμός παριστάνεται DE/x/y/z όπου:

- x: υποδηλώνει ότι το διάνυσμα μεταλλάξεων επιλέγεται είτε τυχαία (rand) από τον πληθυσμό είτε ως το διάνυσμα με τη μικρότερη τιμή κόστους από τον τρέχοντα πληθυσμό (best)
- y: αντιπροσωπεύει τον αριθμό των διανυσμάτων διαφοράς που χρησιμοποιούνται
- z: δηλώνει τον τρόπο της διασταύρωσης.

Μερικά παραδείγματα συμβολισμών:

DE / rand / 1 / bin

$$\vec{V}_{i,\,G+1} = \vec{X}_{r1,\,G} + F\cdot(\vec{X}_{r2,\,G} - \vec{X}_{r3,\,G})$$

DE / rand-to-best / 2 / bin

$$\vec{V}_{i,\,G+1} = \vec{X}_{i,\,G} + F\cdot(\vec{X}_{best,\,G} - \vec{X}_{i,\,G}) + F\cdot(\vec{X}_{r1,\,G} - \vec{X}_{r2,\,G}) + F\cdot(\vec{X}_{r3,\,G} - \vec{X}_{r4,\,G})$$

DE / rand / 2 / bin

$$\vec{V}_{i,\,G+1} = \vec{X}_{r1,\,G} + F\cdot(\vec{X}_{r2,\,G} - \vec{X}_{r3,\,G}) + F\cdot(\vec{X}_{r4,\,G} - \vec{X}_{r5,\,G})$$

DE / current-to-rand / 1 / bin

$$\vec{V}_{i,\,G+1} = \vec{X}_{i,\,G} + K\cdot(\vec{X}_{r1,\,G} - \vec{X}_{i,\,G}) + F\cdot(\vec{X}_{r2,\,G} - \vec{X}_{r3,\,G})$$

3. Ο ΑΔΕ σε συνδυαστικά προβλήματα

Σε συνδυαστικά προβλήματα οι παράμετροι μπορούν να πάρουν τιμές σε ένα σύνολο με πεπερασμένο αριθμό δηλαδή διακριτών καταστάσεων. Έτσι ο αριθμός των πιθανών διανυσμάτων είναι επίσης πεπερασμένος. Ο όρος *Συνδυαστική βελτιστοποίηση* συχνά εξισώνεται με τον όρο *διακριτή βελτιστοποίηση σε πεπερασμένα σύνολα*.

Είναι πολλές φορές δύσκολο και καμία φορά αδύνατο να μεταφερθεί ή να μετατραπεί ένα πρόβλημα από διακριτό σε συνεχές, έτσι ο ΑΔΕ σε τέτοιου είδους προβλήματα δεν μπορεί να εφαρμοστεί στην κανονική του μορφή. Άλλες προσεγγίσεις μετατρέπουν τα διακριτά στοιχειά του προβλήματος σε συνεχή, εφαρμόζουν τον αλγόριθμο και ξαναμετατρέπουν τους απογόνους στον διακριτό χώρο, άλλες προσεγγίσεις μεταφέρουν τις παραμέτρους ελέγχου του αλγορίθμου απευθείας στο πρόβλημα. Αν και

ο ΑΔΕ είχε καλές επιδόσεις σε κάποιες εφαρμογές σε διακριτά προβλήματα είναι ακόμα ένα θέμα προς μελέτη και συζήτηση.

3.1 Ο ΑΔΕ στο πρόβλημα του Sudoku

Το *Sudoku* είναι ένα παιχνίδι/γρίφος στο οποίο ο παίχτης, για να φτάσει στη νίκη, πρέπει να συμπληρώσει ένα πλέγμα, συνήθως 9x9, με αριθμούς, διατηρώντας τη μοναδικότητά τους σε γραμμές, στήλες και κουτιά. Υπάρχουν πάντα εξ αρχής κάποιοι δεδομένοι αριθμοί ώστε η λύση στο Sudoku να είναι μοναδική. Ανάλογα με το πλήθος αυτών των δεδομένων αριθμών αλλάζει και η δυσκολία του προβλήματος.
Παρακάτω φαίνονται μερικά παραδείγματα από Sudoku με τις λύσεις τους:

Στις εφαρμογές της παρούσας εργασίας θα χρησιμοποιηθεί το πρώτο από τα παρακάτω Sudoku.

9	**2**		**6**	**3**			**4**	
6					**4**			**3**
		4				**5**		
		9	**8**		**6**	**1**		
8				**1**				**6**
1			**7**		**5**			**2**
		7		**5**		**6**	**1**	
4					**7**			**8**
	3	**6**		**4**			**2**	

9	**2**	5	**6**	**3**	1	8	**4**	7
6	1	8	5	7	**4**	2	9	**3**
3	7	**4**	9	8	2	**5**	6	1
7	4	**9**	**8**	2	**6**	**1**	3	5
8	5	2	4	**1**	3	9	7	**6**
1	6	3	**7**	9	**5**	4	8	**2**
2	8	**7**	3	**5**	9	**6**	**1**	4
4	9	1	2	6	**7**	3	5	**8**
5	**3**	**6**	1	**4**	8	7	**2**	9

4			
	2		
		1	
3			

4	3	2	1
1	**2**	3	4
2	4	**1**	3
3	1	4	2

Για να μπορέσει να εφαρμοστεί ο ΑΔΕ στο Sudoku θα πρέπει να οριστεί το πρόβλημα έτσι ώστε ο αλγόριθμος να μπορέσει να ψάξει την βέλτιστη λύση σε ένα πεπερασμένο σύνολο καταστάσεων.

Στην εργασία αυτή το Sudoku ορίζεται ως εξής:

Ως ένα τυχαίο μέλος του συνόλου, δηλαδή μια υποψήφια λύση, ορίζεται ένα πλέγμα πλήρως συμπληρωμένο με τους δεδομένους αριθμούς και τα κενά κελιά συμπληρωμένα με τυχαίους αριθμούς μέσα από το πεδίο ορισμού του προβλήματος.

Τα μέλη του συνόλου μετά από πειράματα έχουν τεθεί στα 200, ώστε ο αλγόριθμος να έχει καλά αποτελέσματα και ταχύτητα χωρίς να χρειάζεται υπερβολική υπολογιστική ισχύς.

Κάθε μέλος του συνόλου των υποψήφιων λύσεων είναι ένα ολόκληρο Sudoku που αρχικά ορίζεται με τυχαία ομοιόμορφη κατανομή.

Ο συντελεστής διασποράς Q έχει οριστεί στην τιμή 0.85 μετά από δοκιμές.

Ο συντελεστής μετάλλαξης d έχει οριστεί στην τιμή 0.5.

Έχει εισαχθεί ένας μετρητής count για ανακατανομή του πληθυσμού σε περίπτωση παγίδευσης σε τοπικό ελάχιστο.

Η αντικειμενική συνάρτηση (συνάρτηση κόστους) ορίζεται ως εξής:

F = F1 + F2 όπου F1: ο αριθμός των μη μοναδικών στοιχείων σε κάθε στήλη επί 50
και F2: ο αριθμός των μη μοναδικών στοιχείων σε κάθε κουτί επί 50

Το δοκιμαστικό διάνυσμα γονέας βγαίνει είτε από τον τύπο: $V_{if} = X_{r3f} + Q(X_{r1f} - X_{r2f})$ όπου X_{r1f}, X_{r2f}, X_{r3f} τυχαία μέλη του πληθυσμού που έχουν μεταφερθεί στον χώρο των πραγματικών. Είτε από τον τύπο: $x_{r1} = P \cdot x_{r2}$ όπου το P είναι ένας πίνακας αντιμεταθέσεων και το x_{r2} ένα τυχαίο μέλος του πληθυσμού.

Στην συνδυασμένη προσέγγιση χρησιμοποιούνται και οι δύο τύποι.

Τέλος γίνεται διόρθωση των παραγόμενων μελών σε σχέση με τους δεδομένους αριθμούς. Η απόρριψη μελών που δεν τηρούν την διάταξη των δεδομένων αριθμών έχει μεγάλο αντίκτυπο στην απόδοση των αλγορίθμων για τον λόγο αυτό προτιμήθηκε η διόρθωση των μελών αυτών.

4. Εφαρμογή του ΑΔΕ στο MATLAB για το Sudoku

Για τον σκοπό αυτής της εργασίας έχει υλοποιηθεί μια εφαρμογή στο MATLAB ακολουθώντας τις προσεγγίσεις των Price, Storn και Lampinen (Price, Storn, & Lampinen, Differential Evolution A Practical Approach to Global Optimization, 2005) με Πίνακα Αντιμετάθεσης και με Σχετικό Δείκτη Θέσης όπως και μια πρωτότυπη που συνδυάζει τις δύο αυτές προσεγγίσεις.

Προτιμήθηκε το λογισμικό MATLAB για την υλοποίηση της εφαρμογής λόγω των δυνατοτήτων του στην διαχείριση πινάκων.

Η συνάρτηση κόστους και για τις τρείς προσεγγίσεις ορίσθηκε ως εξής:

Το άθροισμα του αθροίσματος του πληθάριθμου των κοινών, μη μοναδικών, στοιχείων σε κάθε στήλη και κουτί επί 50.

Στο MATLAB: `f1 = f1 + abs(numel(unique(X(:,i)))-9),` παρομοίως η `f2` και για τα κουτιά.

Και τελικά η συνάρτηση κόστους: `F = f1 * 50 + f2 * 50`

4.1 Προσέγγιση με Πίνακα Αντιμετάθεσης (Permutation Matrix)

Η προσέγγιση με Πίνακα Αντιμετάθεσης (Permutation Matrix) προτάθηκε από τους Price και Storn για εφαρμογή του ΑΔΕ στον διακριτό χώρο και μελετήθηκε περεταίρω από τους (Onwubolu & Davendra, 2009). Η ιδέα της διαφοράς διανυσμάτων του ΑΔΕ στον χώρο των πραγματικών μεταφέρθηκε στην ομάδα των αντιμεταθέσεων. Όπως η διαφορά δυο διανυσμάτων στον πραγματικό χώρο δίνει και πάλι ένα διάνυσμα έτσι και δύο αντιμεταθέσεις ορίζουν έναν "χάρτη" για μια αντιμετάθεση. Έτσι ένας πίνακας P χαρτογραφεί τη μετάλλαξη ενός διανύσματος x_{r2} σε x_{r1} όπως φαίνεται στη σχέση: $x_{r1}=P\cdot x_{r2}$

π.χ. για τα διανύσματα $x_{r2} = \begin{pmatrix}1\\4\\3\\5\\2\end{pmatrix}$ και $P = \begin{pmatrix}1\\3\\2\\4\\5\end{pmatrix}$ το P μετατρέπεται ως εξής: $P = \begin{pmatrix}1&0&0&0&0\\0&0&1&0&0\\0&1&0&0&0\\0&0&0&1&0\\0&0&0&0&1\end{pmatrix}$.

Το γινόμενο $P\cdot x_{r2}$ θα μας δώσει: $P\cdot x_{r2} = \begin{pmatrix}1&0&0&0&0\\0&0&1&0&0\\0&1&0&0&0\\0&0&0&1&0\\0&0&0&0&1\end{pmatrix}\begin{pmatrix}1\\4\\3\\5\\2\end{pmatrix} = \begin{pmatrix}1\\3\\4\\5\\2\end{pmatrix} = x_{r1}$

Δηλαδή δημιουργείται ένα καινούριο διάνυσμα x_{r1} το οποίο προήλθε από την αντιμετάθεση των στοιχείων του x_{r2} με βάση τα στοιχεία του P.

Επειδή οι αντιμεταθέσεις γίνονται σε διανύσματα τα οποία είναι ήδη υποψήφιες λύσεις παράγονται διανύσματα τα οποία και αυτά με τη σειρά τους είναι πάντα έγκυρες υποψήφιες λύσεις.

Επιπλέων αντί του συντελεστή διασταύρωσης εισάγεται ο συντελεστής μετάλλαξης d ο οποίος μεταλλάσσει τον P για να κλιμακώσει την επίδρασή του πάνω στο διάνυσμα αντιμετάθεσης. Ο αλγόριθμος για την εφαρμογή του συντελεστή d σε MATLAB φαίνεται παρακάτω (Price & Storn, International Computer Science Institute, 2014):

```
for i=1:N //ψάξε όλες τις στήλες του P
    if(P(i,i) == 0) //το 1 δεν είναι στην κύρια διαγώνιο
        if(rand > d) //εάν ένας τυχαίος αριθμός είναι μεγαλύτερος του d
            j = 1; //βρες τη γραμμή όπου το P(j,i) = 1
            while(P(j,i) ~= 1)
                j = j + 1;
            end
            P([i j],:) = P([j i],:); //αντάλλαξε τις γραμμές
        end
    end
end
```

Η προσέγγιση με πίνακα αντιμετάθεσης τείνει να παγιδεύεται σε τοπικά ελάχιστα επειδή οι υποψήφιες λύσεις πού παράγονται σε κάθε γενιά δεν είναι πάντα αποδοτικές. Για να αποφευχθεί αυτή η παγίδευση μετά από την υλοποίηση της PM δοκιμάζεται και μια τροποποιημένη μορφή. Η τροποποιημένη μορφή μετά από έναν αριθμό γενεών χωρίς καμία βελτίωση κάνει ανακατανομή του πληθυσμού εκτός του βέλτιστου μέλους.

Ο κώδικας MATLAB για την PM παρατίθεται στο παράρτημα.

4.2 Προσέγγιση με Σχετικό Δείκτη Θέσης (Relative Position Indexing)

Η μέθοδος με Σχετικό Δείκτη Θέσης (Relative Position Indexing) ήταν μια διαφορετική προσέγγιση που επίσης προτάθηκε από τους Storn και Price και μελετήθηκε από τους (Onwubolu & Davendra, 2009) για διακριτό χώρο. Και αυτή η προσέγγιση κάνει αντιμετάθεση των διακριτών τιμών σε διανύσματα αλλά εδώ πρώτα μετατρέπονται σε διανύσματα πραγματικών στο διάστημα [0, 1], εφαρμόζεται η διαφορική μετάλλαξη και έπειτα ξαναμετατρέπονται σε διακριτά. Η RPI είναι πιο κοντά στον ΑΔΕ επειδή χρησιμοποιεί την διαφορά διανυσμάτων όπως ο κλασικός αλγόριθμος. Για να μετατραπούν τα στοιχεία των διανυσμάτων για το διάστημα [0, 1] διαιρούνται με το μέγιστο στοιχείο, υπολογίζεται η διαφορά των διανυσμάτων δοτών, υπολογίζεται η αντικειμενική συνάρτηση για το προκύπτον δοκιμαστικό διάνυσμα με τον κλασικό τύπο του ΑΔΕ και μετατρέπονται τα στοιχεία πίσω στον διακριτό χώρο ως εξής: το στοιχείο με τη μικρότερη τιμή παίρνει την τιμή 1, το επόμενο μικρότερο την τιμή 2 κ.ο.κ. ως ότου τελειώσουν όλα τα στοιχεία, τέλος αξιολογείται η αντικειμενική συνάρτηση του νέου δοκιμαστικού διανύσματος και αναλόγως αντικαθιστά το μέλος της τρέχουσας γενιάς ή απορρίπτεται.

π.χ. για τα διανύσματα $x_{r1} = \begin{pmatrix}1\\3\\4\\5\\2\end{pmatrix}$, $x_{r2} = \begin{pmatrix}1\\4\\3\\5\\2\end{pmatrix}$ και $x_{r3} = \begin{pmatrix}1\\3\\2\\4\\5\end{pmatrix}$ θα έχουμε:

$x_{r1f} = x_{r1} \div 5 = \begin{pmatrix}0.2\\0.6\\0.8\\1\\0.4\end{pmatrix}$, $x_{r2f} = x_{r2} \div 5 = \begin{pmatrix}0.2\\0.8\\0.6\\1\\0.4\end{pmatrix}$ και $x_{r3f} = x_{r3} \div 5 = \begin{pmatrix}0.2\\0.6\\0.4\\0.8\\1\end{pmatrix}$

$v_{if} = x_{r3f} + F(x_{r1f} - x_{r2f}) = \begin{pmatrix}0.2\\0.43\\0.57\\0,8\\1\end{pmatrix}$ με F = 0.85 και $v_i = \begin{pmatrix}1\\2\\3\\4\\5\end{pmatrix}$

Επειδή στην προσέγγιση αυτή είναι δυνατόν να προκύψουν λύσεις με ίδια στοιχεία μέσα στο διάνυσμα θα πρέπει να δοθεί προσοχή ώστε αυτά να απορρίπτονται ή να επιδιορθώνονται.

Στην RPI ο συντελεστής διασταύρωσης CR καταργείται επειδή για ένα διάνυσμα αντιμεταθέσεων πρέπει όλα τα στοιχεία να είναι μοναδικά.
Η προσέγγιση με Σχετικό Δείκτη Θέσης τείνει και αυτή να παγιδεύεται σε τοπικά ελάχιστα για τον λόγο ότι τα δοκιμαστικά διανύσματα-γονείς που προκύπτουν μετά την μετατροπή πίσω στον διακριτό χώρο παράγονται τυχαία. Δοκιμάζεται και εδώ μια τροποποιημένη μορφή που κάνει ανακατανομή του πληθυσμού όπως και στην PM.

Ο κώδικας MATLAB για την RPI παρατίθεται στο παράρτημα.

4.3 Μέθοδος που συνδυάζει RPI και PM (RPPM)

Τελευταία δοκιμάζεται μια μέθοδος που συνδυάζει την προσέγγιση με Σχετικό Δείκτη Θέσης και την προσέγγιση με Πίνακα Αντιμετάθεσης (Relative Position Permutation Matrix).

Εδώ πρώτα δημιουργείται ένα δοκιμαστικό διάνυσμα-γονέας με RPI και στη συνέχεια, εάν για έναν προκαθορισμένο αριθμό γενεών δεν υπάρχει βελτίωση ως προς μια βέλτιστη λύση, το ίδιο αυτό διάνυσμα χρησιμοποιείται ως πίνακας αντιμετάθεσης, για να δημιουργήσει το τελικό δοκιμαστικό

διάνυσμα. Αυτή η προσέγγιση δείχνει να μεταλλάσσει περισσότερο τα δοκιμαστικά διανύσματα ώστε να αποφευχθεί η παγίδευση σε τοπικά ελάχιστα.

Η πειραματική αυτή μέθοδος έχει πολύ καλύτερα αποτελέσματα από την RPI, όπως και η PM άλλωστε, αλλά περίπου ίδια με την PM. Συγκριτικά με την PM αρχίζει να έχει αποτυχίες νωρίτερα αλλά ο μέσος όρος της συνάρτησης κόστους αυξάνει με μικρότερο ρυθμό.

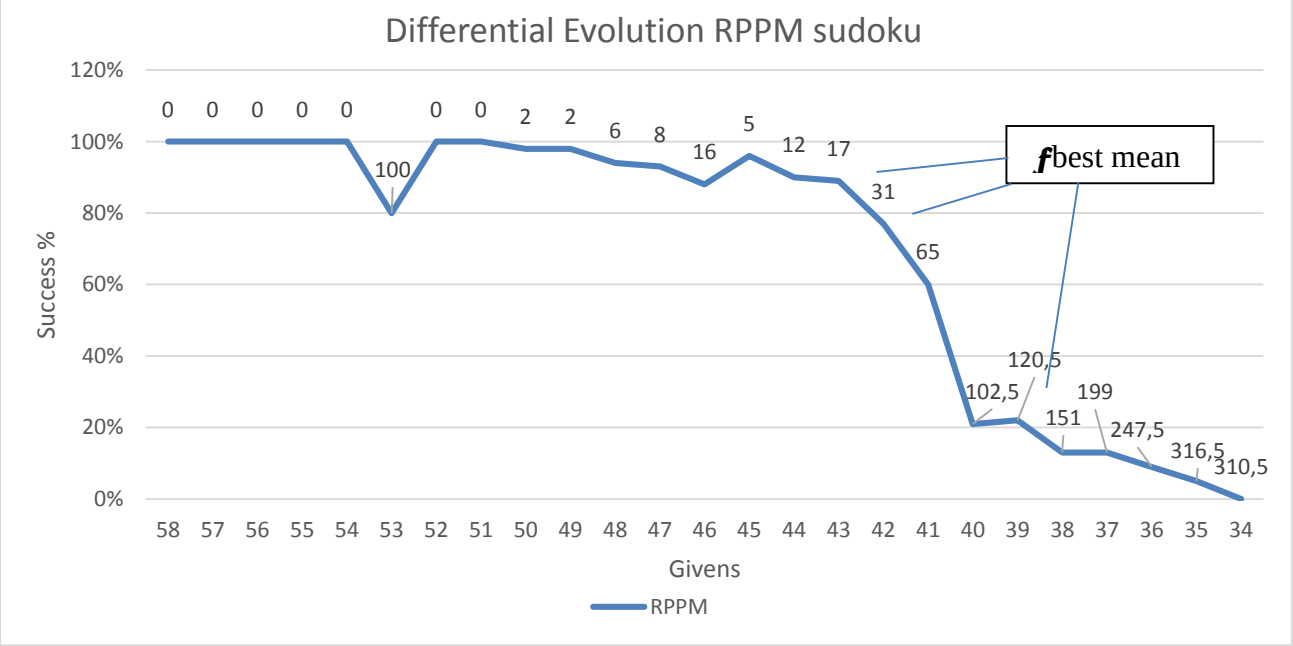
Φυσικά και αυτή όπως και οι προηγούμενες μέθοδοι, επειδή κάνουν αντιμετάθεση των στοιχείων ενός διανύσματος, δεν παράγουν πάντα αποδοτικές λύσεις. Δηλαδή βασίζονται πιο πολύ στον παράγοντα της τύχης όπως οι γενετικοί αλγόριθμοι.

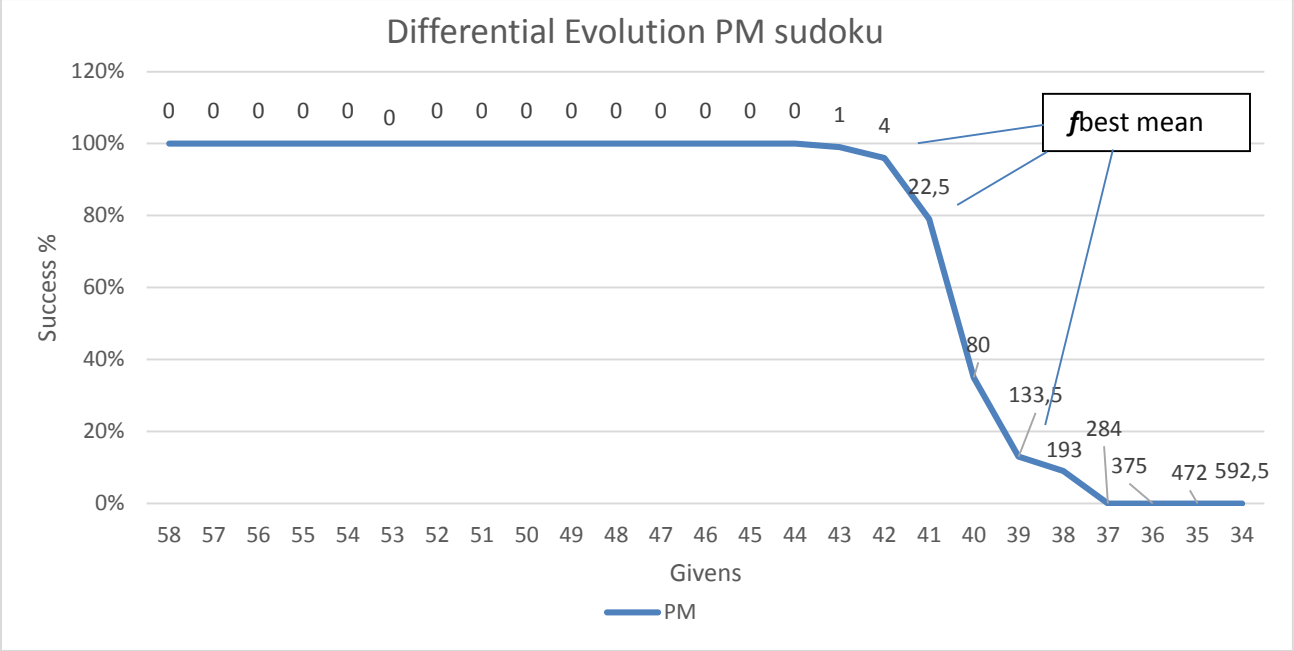
Ο κώδικας MATLAB για την RPPM παρατίθεται στο παράρτημα

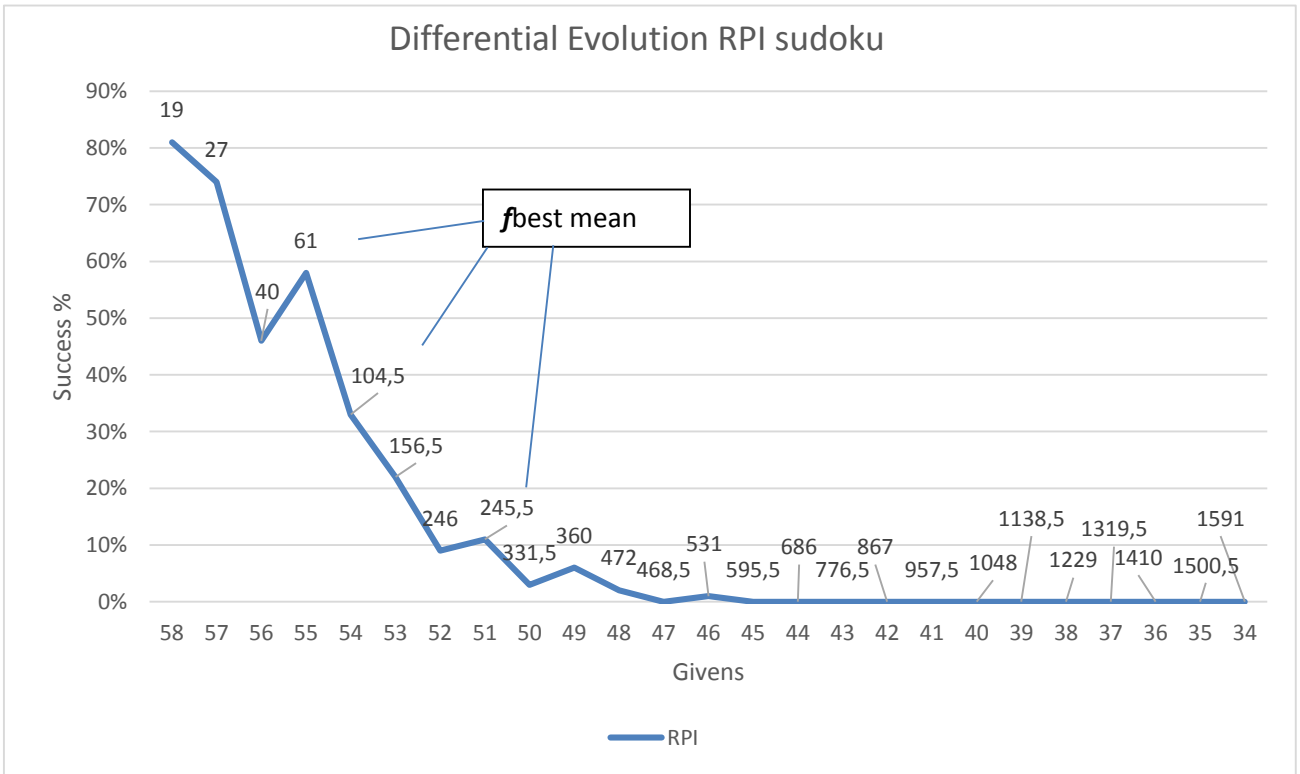
5. Πειραματικά αποτελέσματα και συμπεράσματα

Για την άντληση των αποτελεσμάτων της κάθε προσέγγισης η εφαρμογή έτρεξε τον αλγόριθμο 100 φορές για κάθε πλήθος δεδομένων αριθμών. Ο αριθμός των γενεών για την κάθε μέθοδο έχει τεθεί στις 1000.

Τα αποτελέσματα φαίνονται παρακάτω σε πίνακες και διαγράμματα.




Differential Evolution PM sudoku
120%
100%
80%
60%
40%
20%
0%
Success %
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 4 22,5 80 133,5 193 284 375 472 592,5
ƒbest mean
58 57 56 55 54 53 52 51 50 49 48 47 46 45 44 43 42 41 40 39 38 37 36 35 34
Givens
PM


Differential Evolution RPI sudoku
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
Success %
19 27 40 61 104,5 156,5 246 245,5 331,5 360 472 468,5 531 595,5 686 776,5 867 957,5 1048 1138,5 1229 1319,5 1410 1500,5 1591
ƒbest mean
58 57 56 55 54 53 52 51 50 49 48 47 46 45 44 43 42 41 40 39 38 37 36 35 34
Givens
RPI

RPPM		
Givens	Success	*f*best mean
58	100%	0
57	100%	0
56	100%	0
55	100%	0
54	100%	0
53	80%	100
52	100%	0
51	100%	0
50	98%	2
49	98%	2
48	94%	6
47	93%	8
46	88%	16
45	96%	5
44	90%	12
43	89%	17
42	77%	31
41	60%	65
40	21%	102,5
39	22%	120,5
38	13%	151
37	13%	199
36	9%	247,5
35	5%	316,5
34	0%	310,5

PM		
Givens	Success	*f*best mean
58	100%	0
57	100%	0
56	100%	0
55	100%	0
54	100%	0
53	100%	0
52	100%	0
51	100%	0
50	100%	0
49	100%	0
48	100%	0
47	100%	0
46	100%	0
45	100%	0
44	100%	0
43	99%	1
42	96%	4
41	79%	22,5
40	35%	80
39	13%	133,5
38	9%	193
37	0%	284
36	0%	375
35	0%	472
34	0%	592,5

RPI		
Givens	Success	*f*best mean
58	81%	19
57	74%	27
56	46%	40
55	58%	61
54	33%	104,5
53	22%	156,5
52	9%	246
51	11%	245,5
50	3%	331,5
49	6%	360
48	2%	472
47	0%	468,5
46	1%	531
45	0%	595,5
44	0%	686
43	0%	776,5
42	0%	867
41	0%	957,5
40	0%	1048
39	0%	1138,5
38	0%	1229
37	0%	1319,5
36	0%	1410
35	0%	1500,5
34	0%	1591

Από τα αποτελέσματα της εφαρμογής φαίνεται ότι ένα εύκολο πρόβλημα Sudoku μπορεί να λυθεί με τη μέθοδο της διαφορικής εξέλιξης. Η μέθοδος όμως χάνει πολύ γρήγορα την αποδοτικότητά της όσο το πρόβλημα δυσκολεύει, όσο δηλαδή μειώνεται ο αριθμός των δεδομένων αριθμών. Αυτό φαίνεται να οφείλεται στο ότι η μέθοδος της διαφορικής εξέλιξης, ψάχνοντας τον πραγματικό χώρο, όσο συγκλίνει προς την βέλτιστη λύση τόσο αυξάνει και η πιθανότητα τα παραγόμενα μέλη να είναι καλύτερα, κάτι που δεν συμβαίνει στον χώρο των διακριτών. Στον χώρο των διακριτών κάθε φορά που αναζητείται μια υποψήφια λύση η αναζήτηση γίνεται σε όλο το πεδίο ορισμού με αποτέλεσμα όσο ο αλγόριθμος πλησιάζει στην βέλτιστη λύση τόσο πιο δύσκολο είναι να επιτύχει μια καλύτερη λύση ή να βρει την βέλτιστη.

Στο πρόβλημα του Sudoku όσο οι δεδομένοι αριθμοί μειώνονται τόσο αυξάνεται και ο χώρος αναζήτησης για τη βέλτιστη λύση.

Συμπερασματικά από την εργασία αυτή φαίνεται ότι ο αλγόριθμος της διαφορικής εξέλιξης μπορεί να χρησιμοποιηθεί και σε συνδυαστικά προβλήματα στον διακριτό χώρο και συγκεκριμένα στο πρόβλημα του Sudoku αλλά χάνει την αποδοτικότητα που έχει, σε σχέση με άλλους αλγόριθμους και σε σχέση με την εφαρμογή του στον πραγματικό χώρο.

Σίγουρα το Sudoku μπορεί να λυθεί πιο αποδοτικά με ειδικούς αλγόριθμους που έχουν δημιουργηθεί για αυτόν ακριβώς τον σκοπό, όπως ο αλγόριθμος Ωμής Βίας (Brute Force) ή με αλγορίθμους που έχουν δημιουργηθεί για αυτού του είδους προβλήματα, όπως φαίνεται στις ακόλουθες πηγές: (Skiena, 2008), (Chu, 2006), (Norvig, n.d.), δηλαδή αλγόριθμοι που από την αρχή αναζητούν λύση σε διακριτά προβλήματα ή συγκεκριμένα για το Sudoku.

Ο ΑΔΕ έχει δοκιμαστεί στο Sudoku από τους Urszula Boryczka και Przemysław Juszczuk (Boryczka & Juszczuk, 2012) αλλά με μεγάλες μετατροπές στον αλγόριθμο με αποτέλεσμα να ξεφεύγει από τον ΑΔΕ και να πλησιάζει περισσότερο στους Γενετικούς Αλγορίθμους.

Ο ΑΔΕ φαίνεται να μπορεί να βελτιωθεί περαιτέρω με τεχνικές διόρθωσης, αναζήτησης στοιχείων, οπισθοδρόμησης κ.α. αλλά με αυτόν τον τρόπο ξεφεύγει από την απλότητά του και την γενικότητα της χρήσης του.

Παράρτημα

Οι δεδομένοι αριθμοί δίνονται σε πίνακα με την παρακάτω μορφή. Η πρώτη στήλη υποδηλώνει την γραμμή του αριθμού στο πλέγμα, η δεύτερη την στήλη στην οποία βρίσκεται ο αριθμός και η τρίτη την τιμή.

```
B = [1,1,9;%givens 9x9
    1,2,2;
    1,3,5;
    1,4,6;
    1,5,3;
      .
      .
      .
    8,6,7;
    8,8,5;
    8,9,8;
    9,2,3;
    9,3,6;
    9,5,4;
    9,6,8;
    9,7,7;
    9,8,2;
    9,9,9];
```

Κώδικας για την Συνάρτηση κόστους

```
function cost=fDE9opt(X)
f1 = 0;
f2 = 0;

for i=1:9%f1
    f1 = f1 + abs(numel(unique(X(:,i)))-9);
end

for i=0:3:6%f2
    for j=0:3:6
        block = X(i+(1:3),j+(1:3));
        blockVec = block(:);
        f2 = f2 + abs(numel(unique(blockVec))-9);
    end
end

cost = 50*f1 + 50*f2;
end
```

Κώδικας για Πίνακα Αντιμετάθεσης

```
function fbestval=sudokuDEPM(Giv)
clear;
clc;

B = Giv;
```

```
n = 3;
dim = n^2;
pop = 1:dim;
NP = 200;
fest = 0;
X=zeros(dim,dim,NP);
cost=zeros(1,NP);
Givens = zeros(dim,dim);

solved = [9 2 5 6 3 1 8 4 7
    6 1 8 5 7 4 2 9 3
    3 7 4 9 8 2 5 6 1
    7 4 9 8 2 6 1 3 5
    8 5 2 4 1 3 9 7 6
    1 6 3 7 9 5 4 8 2
    2 8 7 3 5 9 6 1 4
    4 9 1 2 6 7 3 5 8
    5 3 6 1 4 8 7 2 9];

for i=1:size(B,1)
    Givens(B(i,1),B(i,2)) = B(i,3);
end

for j=1:NP
    for i=1:dim%X init
        NoSam = nonzeros(Givens(i,:))';
        Sam = setdiff(pop,NoSam);
        RandSam = randsample(Sam,size(find(Givens(i,:)==0),2));
        X(i,Givens(i,:)==0,j) = RandSam;
        X(i,:,j) = X(i,:,j)+Givens(i,:);
    end
    cost(j)=fDE9opt(X(:,:,j));
end

[fbest,minind] = min(cost);
Xbest = X(:,:,minind);
fbestval = fbest;
g = 0;
gmax = 1000;
Q = 0.85;
d = 0.5;
I = eye(dim,dim);
P = zeros(dim,dim,dim);
count = 0;
V=zeros(dim,dim);

while (g<gmax)&&(fbest>fest)
    g = g+1;
    for i = 1:NP
        r1 = randi(NP);
        while(r1==i)
            r1 = randi(NP);
        end
        Xr1 = X(:,:,r1);
        r2 = randi(NP);
        while(r2==i)&&(r2==r1)
```

```
    r2 = randi(NP);
end
Xr2 = X(:,:,r2);

for l=1:dim
    Vt = Xr1(l,:);
    P(:,:,l) = I(Vt,:);
end
for thirdd=1:dim
    for c1=1:dim
        if(P(c1,c1,thirdd) == 0)
            c2 = 1;
            if(rand > d)
                while(P(c2,c1,thirdd) ~= 1)
                    c2 = c2+1;
                end
                P([c1 c2],:,thirdd) = P([c2 c1],:,thirdd);
            end
        end
    end
end

for j=1:dim
    V(j,:) = Xr2(j,:)*P(:,:,j);
end

for irep=1:dim
    for jrep=1:dim
        if((V(irep,jrep)~=Givens(irep,jrep))&&(Givens(irep,jrep)~=0))
            index= V(irep,:)==Givens(irep,jrep);
            V(irep,index)=V(irep,jrep);
            V(irep,jrep)=Givens(irep,jrep);
        end
    end
end

Vcost = fDE9opt(V);

if Vcost < cost(i)
    X(:,:,i) = V;
    count = 0;
end

cost(i)=fDE9opt(pop,X(:,:,i),Givens);
[fbest,minind] = min(cost);
Xbest = X(:,:,minind);
fbestval = fbest;

if count>dim*2
    X=zeros(dim,dim,NP);
    for j=1:NP
        for i2=1:dim%X init
            NoSam = nonzeros(Givens(i2,:))';
            Sam = setdiff(pop,NoSam);
            RandSam = randsample(Sam,size(find(Givens(i2,:)==0),2));
```

```
                    X(i2,Givens(i2,:)==0,j) = RandSam;
                    X(i2,:,j) = X(i2,:,j)+Givens(i2,:);
                end
                cost(j)=fDE9opt(X(:,:,j));
            end
            X(:,:,minind) = Xbest;
            cost(minind) = fbest;
            %cost(minind)=fDE9opt(X(:,:,minind));
            [fbest,minind] = min(cost);
            fbestval = fbest;
        end

    end
    count = count + 1;
    clc
    gens = sprintf('generations: %d, count: %d', g, count);
    disp(gens);
end
disp('solved');
disp(solved);
disp('Xbest');
disp(Xbest);
showGivens = ['# of Givens: ', num2str(size(B,1))];
disp(showGivens);
```

Κώδικας για Σχετικό Δείκτη Θέσης

```
function fbestval=sudokuDERPI9()
clear;
clc;

B = Giv;
n = 3;
dim = n^2;
pop = 1:dim;
NP = 20;
fest = 0;
X=zeros(dim,dim,NP);
cost=zeros(1,NP);
Givens = zeros(dim,dim);

solved = [9 2 5 6 3 1 8 4 7
    6 1 8 5 7 4 2 9 3
    3 7 4 9 8 2 5 6 1
    7 4 9 8 2 6 1 3 5
    8 5 2 4 1 3 9 7 6
    1 6 3 7 9 5 4 8 2
    2 8 7 3 5 9 6 1 4
    4 9 1 2 6 7 3 5 8
    5 3 6 1 4 8 7 2 9];

for i=1:size(B)
```

```
    Givens(B(i,1),B(i,2)) = B(i,3);
end

for j=1:NP
    for i=1:dim%X init
        NoSam = nonzeros(Givens(i,:))';
        Sam = setdiff(pop,NoSam);
        RandSam = randsample(Sam,size(find(Givens(i,:)==0),2));
        X(i,Givens(i,:)==0,j) = RandSam;
        X(i,:,j) = X(i,:,j)+Givens(i,:);
    end
    cost(j)=fDE9opt(X(:,:,j));
end

[fbest,minind] = min(cost);
Xbest = X(:,:,minind);
fbestval = fbest;
g = 0;
gmax = 1000;
Q = 0.85;
count = 0;
V=zeros(dim,dim);

while (g<gmax)&&(fbest>fest)
    g = g+1;
    for i = 1:NP
        r1 = randi(NP);
        while(r1==i)
            r1 = randi(NP);
        end
        Xr1f = X(:,:,r1)/dim;

        r2 = randi(NP);
        while(r2==i)||(r2==r1)
            r2 = randi(NP);
        end
        Xr2f = X(:,:,r2)/dim;

        r3 = randi(NP);
        while(r3==i)||(r3==r1)||(r3==r2)
            r3 = randi(NP);
        end
        Xr3f = X(:,:,r3)/dim;

        Vf = Xr3f + Q*(Xr1f-Xr2f);%dokimastiko dianusma-goneas
        for k=1:dim
            [~,V(k,:)] = sort(Vf(k,:));
            [~,V(k,:)] = sort(V(k,:));
        end

        Vcost = fDE9opt(V);

        for irep=1:dim
                for jrep=1:dim
```

```
if((V(irep,jrep)~=Givens(irep,jrep))&&(Givens(irep,jrep)~=0))
                        index= V(irep,:)==Givens(irep,jrep);
                        V(irep,index)=V(irep,jrep);
                        V(irep,jrep)=Givens(irep,jrep);
                    end
                end
        end

        if Vcost < cost(i)
            X(:,:,i) = V;
            count = 0;
        end
        cost(i)=fDE9opt(X(:,:,i));
        [fbest,minind] = min(cost);
        Xbest = X(:,:,minind);
        fbestval = fbest;
    end%end for

    if count>dim*2
        X=zeros(dim,dim,NP);
        for j=1:NP
            for i2=1:dim%X init
                NoSam = nonzeros(Givens(i2,:))';
                Sam = setdiff(pop,NoSam);
                RandSam = randsample(Sam,size(find(Givens(i2,:)==0),2));
                X(i2,Givens(i2,:)==0,j) = RandSam;
                X(i2,:,j) = X(i2,:,j)+Givens(i2,:);
            end
            cost(j)=fDE9opt(X(:,:,j));
        end
        X(:,:,minind) = Xbest;
        cost(minind) = fbest;
        %cost(minind)=fDE9opt(pop,X(:,:,minind),Givens);
        [fbest,minind] = min(cost);
        fbestval = fbest;
    end

    count = count+1;
    clc
    gens = sprintf('gens: %d  fbest: %d', g, fbest);
    disp(gens);

end%end while

disp('solved');
disp(solved);
disp('Xbest');
disp(Xbest);
showGivens = ['# of Givens: ', num2str(size(B,1))];
disp(showGivens);

figure;hold on;axis off;axis equal % prepare to draw
rectangle('Position',[0 0 9 9],'LineWidth',3,'Clipping','off') % outside border
rectangle('Position',[3,0,3,9],'LineWidth',2) % heavy vertical lines
rectangle('Position',[0,3,9,3],'LineWidth',2) % heavy horizontal lines
```

```
rectangle('Position',[0,1,9,1],'LineWidth',1) % minor horizontal lines
rectangle('Position',[0,4,9,1],'LineWidth',1)
rectangle('Position',[0,7,9,1],'LineWidth',1)
rectangle('Position',[1,0,1,9],'LineWidth',1) % minor vertical lines
rectangle('Position',[4,0,1,9],'LineWidth',1)
rectangle('Position',[7,0,1,9],'LineWidth',1)

if size(Xbest,2) == 9 % 9 columns
   [SM,SN] = meshgrid(1:9); % make i,j entries
   Xbest = [SN(:),SM(:),Xbest(:)]; % i,j,k rows
end

for ii = 1:size(Xbest,1)
   text(Xbest(ii,2)-0.5,9.5-Xbest(ii,1),num2str(Xbest(ii,3)))
end

hold off
```

Κώδικας για RPPM(συνδυασμός Σχετικού Δείκτη Θέσης & Πίνακα Αντιμετάθεσης)

```
function fbestval=sudokuDERPPM()
clear;
clc;

B = Giv;
n = 3;
dim = n^2;
pop = 1:dim;
NP = 200;
fest = 0;
X=zeros(dim,dim,NP);
cost=zeros(1,NP);
Givens = zeros(dim,dim);

solved = [9 2 5 6 3 1 8 4 7
    6 1 8 5 7 4 2 9 3
    3 7 4 9 8 2 5 6 1
    7 4 9 8 2 6 1 3 5
    8 5 2 4 1 3 9 7 6
    1 6 3 7 9 5 4 8 2
    2 8 7 3 5 9 6 1 4
    4 9 1 2 6 7 3 5 8
    5 3 6 1 4 8 7 2 9];

for i=1:size(B,1)
    Givens(B(i,1),B(i,2)) = B(i,3);
end

for j=1:NP
    for i=1:dim%X init
        NoSam = nonzeros(Givens(i,:))';
        Sam = setdiff(pop,NoSam);
        RandSam = randsample(Sam,size(find(Givens(i,:)==0),2));
        X(i,Givens(i,:)==0,j) = RandSam;
```

```
        X(i,:,j) = X(i,:,j)+Givens(i,:);
    end
    cost(j)=fDE9opt(pop,X(:,:,j),Givens);
end

[fbest,minind] = min(cost);
Xbest = X(:,:,minind);
fbestval = fbest;
g = 0;
gmax = 1000;
Q = 0.85;
d = 0.5;
I = eye(dim,dim);
P = zeros(dim,dim,dim);
count = 0;
V=zeros(dim,dim);

while (g<gmax)&&(fbest>fest)

    g = g+1;

    for i = 1:NP
        r1 = randi(NP);
        while(r1==i)
            r1 = randi(NP);
        end
        Xr1f = X(:,:,r1)/dim;
        r2 = randi(NP);
        while(r2==i)&&(r2==r1)
            r2 = randi(NP);
        end
        Xr2f = X(:,:,r2)/dim;
        r3 = randi(NP);
        while(r3==i)&&(r3==r1)&&(r3==r2)
            r3 = randi(NP);
        end
        Xr3f = X(:,:,r3)/dim;

        Vf = Xr3f + Q*(Xr1f - Xr2f);

        for k=1:dim
            [~,V(k,:)] = sort(Vf(k,:));
            [~,V(k,:)] = sort(V(k,:));
        end

        if count > dim * 2
            for l=1:dim
                Vt = V(l,:);
                P(:,:,l) = I(Vt,:);
            end
            for thirdd=1:dim
                for c1=1:dim
                    if(P(c1,c1,thirdd) == 0)
                        c2 = 1;
                        if(rand > d)
```

```
                            while(P(c2,c1,thirdd) ~= 1)
                                c2 = c2+1;
                            end
                            P([c1 c2],:,thirdd) = P([c2 c1],:,thirdd);
                        end
                    end
                end
            end

            for j=1:dim
                V(j,:) = X(j,:,i)*P(:,:,j);
            end
            count = 0;
        end

        for irep=1:dim
                for jrep=1:dim

if((V(irep,jrep)~=Givens(irep,jrep))&&(Givens(irep,jrep)~=0))
                        index= V(irep,:)==Givens(irep,jrep);
                        V(irep,index)=V(irep,jrep);
                        V(irep,jrep)=Givens(irep,jrep);
                    end
                end
        end

        Vcost = fDE9opt(pop,V,Givens);

        if Vcost < cost(i)
            X(:,:,i) = V;
            count = 0;
        end
        cost(i)=fDE9opt(pop,X(:,:,i),Givens);
        [fbest,minind] = min(cost);
        Xbest = X(:,:,minind);
        fbestval = fbest;

    end
    count = count + 1;
    clc
    gens = sprintf('generations: %d, count: %d', g, count);
    disp(gens);
end
disp('solved');
disp(solved);
disp('Xbest');
disp(Xbest);
showGivens = ['# of Givens: ', num2str(size(B,1))];
disp(showGivens);
```

Βιβλιογραφία

Boryczka, U., & Juszczuk, P. (2012). *SOLVING THE SUDOKU WITH THE DIFFERENTIAL*. Sosnowiec: Bialystok University of Technology.

Chu, J. (2006). *A Sudoku Solver using Knuth's Dancing Links Algorithm.* berkeley.

Norvig, P. (n.d.). *http://norvig.com/sudoku.html*. Retrieved from http://norvig.com/sudoku.html: http://norvig.com/sudoku.html

Onwubolu, C. G., & Davendra, D. (2009). *Differential Evolution: A Handbook for Global Permutation-Based Combinatorial Optimization.* Springer Berlin Heidelberg.

Price, K., & Storn, R. (2014, February 7). *International Computer Science Institute*. Ανάκτηση από ICSI site: http://www1.icsi.berkeley.edu/~storn/code.html#matl

Price, K., Storn, R., & Lampinen, J. (2005). *Differential Evolution A Practical Approach to Global Optimization.* Heidelberg: Springer.

Schwefel, H.-P. (2001, Oktober). *Technische Universität Dortmund*. Ανάκτηση από http://ls11-www.cs.uni-dortmund.de/: http://ls11-www.cs.uni-dortmund.de/

Skiena, S. S. (2008). *The Algorithm Design Manual.* New York: Springer-Verlag.

Storn, R., & Price, K. (1995). *Differential Evolution - A simple and efficient adaptive scheme for global optimization over continuous spaces.* Technical Report, Berkeley.

Wikipedia. (2014, June 13). Ανάκτηση June 18, 2014, από Wikipedia: http://en.wikipedia.org/wiki/Mathematical_optimization

Αδαμίδης, Π. (1999, 5). Ανάκτηση 2 2014, από http://hydra.it.teithe.gr/~adamidis/IntelSys/EA_notes.pdf

Γούδος, Σ. Κ. (2011). *Σχεδίαση και βελτιστοποίηση στοιχειοκεραιών με χρήση διαφορικής εξέλιξης.* Θεσσαλονίκη: Αριστοτέλειο πανεπιστήμιο Θεσσαλονίκης, πολυτεχνική σχολή-τμήμα ηλεκτρολόγων μηχανικών ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ.

Κόλιας, Ι. Α. (2012). *Η Συνεχής Συζυγής Μέθοδος για τη Βελτιστοποίηση Τοπολογίας στη Μηχανική των Ρευστών Προγραμματισμός Μετεπεξεργαστών και Εφαρμογές σε Προβλήματα της Αυτοκινητοβιομηχανίας.* Αθήνα,: Εθνικό Μετσόβιο Πολυτεχνείο Σχολή Μηχανολόγων Μηχανικών.

Παρσόπουλος, Κ. (2005). *Αλγόριθμοι υπολογιστικής νοημοσύνης για αριθμητική βελτιστοποίηση.* Πάτρα: Συγγραφέας.