Άσκηση 1: Πρόβλημα Ιεραποστόλων και κανιβάλων

Το κόστος μιας λύσης είναι ο συνολικός αριθμός μετακινήσεων της βάρκας.

Περιεχόμενα:

1) Λίγα λόγια για το πρόβλημα

Το πρόβλημα των κανιβάλων και ιεραποστόλων:

τρεις ιεραπόστολοι και τρεις κανίβαλοι πρέπει να διασχίσουν ένα ποτάμι χρησιμοποιώντας μια βάρκα που μπορεί να μεταφέρει το πολύ δύο άτομα, υπό τον περιορισμό ότι, και για τις δύο μεριές, εάν υπάρχουν ιεραπόστολοι στην όχθη, δεν μπορούν να είναι περισσότεροι από τους κανίβαλους (αν ήταν, οι κανίβαλοι θα έτρωγαν τους ιεραπόστολους). Το σκάφος δεν μπορεί να διασχίσει το ποτάμι μόνο του χωρίς επιβαίνοντες.

Το γενικευμένο πρόβλημα των κανιβάλων και ιεραποστόλων:

- Το πρόβλημα που επιλύσαμε, δέχεται ώς μεταβλητές τις τιμές των κανιβάλων και ιεραποστόλων (N), της χωρητικότητας της βάρκας (M) και του μέγιστου επιτρεπόμενου αριθμού διασχίσεων της βάρκας (K) από τον χρήστη.
- Θεωρούμε ότι στην αρχική κατάσταση έχουμε N ιεραποστόλους στη μία όχθη και τον ίδιο αριθμό (N) κανιβάλων στην ίδια όχθη
- Το πρόγραμμά βρίσκει τη βέλτιστη λύση που δεν υπερβαίνει τις K διασχίσεις, αν υπάρχει τέτοια λύση. Αναφέρετε επίσης πόσο περίπου χρόνο (ανάλογα και με τον υπολογιστή) χρειάζεται το πρόγραμμά για να βρει λύση για τις ξεχωριστές κάθε φορά τιμές των N, M, K, καθώς και τις αντίστοιχες λύσεις που βρίσκει.

2) Αρχιτεκτονική προγράμματος

Το πρόγραμμα αποτελείτε από τις κλάσεις Main, State, Child, Parameters και Searcher

Main:

Στην κλάση Main δίνονται οι εντολές ώστε ο χρήστης να εισάγει στοιχεία στο πρόγραμμα (οι παράμετροι N, M ,K), καλούνται οι απαραίτητες μέθοδοι και

εκτελούνται οι απαραίτητες διαδικασίες που οδηγούν στην επίλυση του προβλήματος.

State:

Η κλάση State επιτρέπει σε ένα αντικείμενο να αλλάξει συμπεριφορά όταν αλλάζει η εσωτερική της κατάσταση. Στην προκειμένη εφαρμογή ως κατάσταση state θεωρούμε την κάθε εναλλαγή της τοποθεσίας της βάρκας (boat: αριστερά:0 ή δεξιά: 1), του αριθμού ιεραποστόλων (m) και του αριθμού κανιβάλων (c).

Για την εναλλαγή των καταστάσεων χρησιμοποιείτε μία μέθοδος move( ). Η μέθοδος move() από την μέθοδο getChildren( ). Η μέθοδος getChildren( ) βρίσκει όλες τα πιθανές τροποποιήσεις που μπορεί να υποστεί μία δεδομένη κατάσταση, τηρουμένων των περιορισμών του προβλήματος. Όλες αυτές οι τροποποιήσεις αποτελούν τα παιδιά της δοσμένης κατάστασης, τα οποία εντάσσονται σε έναν πίνακα καταστάσεων children.

Για την εύρεση των παραγόμενων παιδιών κάθε κατάστασης, οι μέθοδοι getChildren() και move() καλούν ορισμένες μεθόδους ελέγχου που επιστρέφουν true or false αποτελέσματα. Η getChildren() καλέι την μέδοθο evaluate() η οποία εκτελέι όλους τους απαραίτητους γενικούς ελέγχους του προβλήματος, ενώ η move() καλεί τις μεθόδους check_left() και check_right() που εκτελούν όλους τους ελέγχους που επηρεάζονται από την θέση της βάρκας boat. Ο διαχωρισμός των ελέγχων σε διαφορετικές περιπτώσεις γίνεται για αποφευχθούν οι περιττοί έλεγχοι.

Στην κλάση State βρίσκονται επίσης οι μέθοδοι υπολογισμού ευρετικών συναρτήσεων h1() και h2(). Έπειτα στην μέθοδο getHeuristic() βρίσκουμε την βέλτιστη από τις δύο ευρετικές που είναι η ευρετική με την ελάχιστην τιμή (h = min(h1(), h2())). Η τιμή που παράγεται από την getHeuristic() καλείτε στην συνέχεια στον αλγόριθμο A*

Τέλος, η κλάση State περιλαμβάνει κι άλλες μεθόδους που αξιοποιούνται στην κλάση Searcher. Αυτές είναι η equals() που ελέγχοι αν δύο δοσμένες καταστάσεις είναι ίδιες, και η inside(), η οποία ελέγχει αν μία δοσμένη κατάσταση βρίσκεται μέσα σε έναν δοσμένο πίνακα καταστάσεων

Child:

Η κλάση Child επεκτείνει την κλάση State, δηλαδή έχει λαμβάνει όλες τις ιδιότητες της κλάσης State καθώς και ορισμένες καινούριες, όπως η ιδιότητα weight που δηλώνει το βάρος κάθε παιδιού και η ιδιότητα του Parent state.

Parameters:

Η κλάση Parameters επιτυγχάνει την σύνδεση μεταξύ της main() και της κλάσης State. Οι τιμές εισόδου του χρήστη, εισάγονται με μορφή παραμέτρων στην κλάση Parameters με χρήση της set method. Έπειτα, οι παράμετροι καλούνται στην κλάση State με χρήση της get method.

Searcher:

Στην κλάση Searcher βρίσκεται η μέθοδος A_star(), που υλοποιεί τον προσαρμοσμένο στο συγκεκριμένο πρόβλημα αλγόριθμο A* και κάνει επεξεργασία των αντικειμένων της κλάσης State καθώς και των μεθόδων του. Ο αλγόριθμος A* επιστρέφει είτε την κατάσταση που αποτελεί την λύση του προβλήματος έχοντας εκτελέσει όλες τις απαραίτητες ενέργειες μέχρι να φτάσει σε αυτήν, είτε την τιμή null, σε περίπτωση που δεν βρέθηκε η τελική κατάσταση.

Εκτός από τον αλγόριθμο A*, η κλάση Searcher περιλαμβάνει και άλλες μεθόδους που υποβοηθούν την υλοποίηση του αλγορίθμου και την εμφάνιση των αποτελεσμάτων του. Η μέθοδος sort() δέχεται έναν πίνακα καταστάσεων Arraylist<State> και τον ταξινομεί με βάση τις τιμές της f της κάθε κατάστασης. Ο σκοπός χρήσης της συγκεκριμένης μεθόδου στον A* είναι να βρίσκεται σε κάθε προσπέλαση η κατάσταση με την ελάχιστη f στον πινακα καταστάσεων frontier. ‘Επειτα, η μέθοδος printPath() εκτυπώνει το μονοπάτι καταστάσεων το οποίο ακολούθησε ο A* ώστε να φτάσει από την αρχική κατάσταση στην τελική.

3) Μέθοδοι Τεχνητής Νοημοσύνης που χρησιμοποιεί

Για την εύρεση της βέλτιστης λύσης του προβλήματος γίνεται χρήση του αλγορίθμου A* με κλειστό σύνολο.

Αλγόριθμος A*:
Ο Αλγόριθμος A* είναι μία μίξη των αλγορίθμων lowest-cost-first και best-first search. Αντιμετωπίζει τον frontier ως ουρά προτεραιότητας με βάση την f(p) = g(p) + h(p). Ο αλγόριθμος επιλέγει πάντα μία κατάσταση στον frontier με την μικρότερη υπολογισμένη συνολική απόσταση από τη ρίζα ως κάποια τελική κατάσταση. Όπου, g(p): το κόστος από τη ρίζα ως τον p και h(p): ευρετική συνάρτηση

Ο αλγόριθμος A* είναι:
Πλήρης, αν το b είναι πεπερασμένο και το κόστος κάθε μετάβασης είναι > 0. – Αποκλείεται να παγιδευτούμε σε άπειρα κλαδιά, γιατί όσο προχωράμε σε βάθος αυξάνεται το g(n).
Βέλτιστος, αν η h είναι αποδεκτή (η απόδειξη ακολουθεί). – Αποδεκτή ευρετική συνάρτηση: h(n) ≤ C(n)*. – C(n)*: το πραγματικό κόστος του βέλτιστου μονοπατιού από τον n σε έναν κόμβο τελικής κατάστασης. – Π.χ. η ευθεία απόσταση είναι πάντα μικρότερη από την οδική.
Πολυπλοκότητα χώρου και χρόνου: – Εκθετική στη χειρότερη περίπτωση, όπως στον BFS, αλλά στην πράξη μια καλή ευρετική μπορεί να μειώσει πολύ τον απαιτούμενο χρόνο και χώρο.

A* και κλειστό σύνολο

• Με συνεπή h, ο A* παραμένει βέλτιστος και όταν χρησιμοποιούμε κλειστό σύνολο.

• Το κλειστό σύνολο ίσως μας αναγκάσει να διακόψουμε την εξερεύνηση ενός μονοπατιού κάτω από έναν κόμβο κατάστασης A την οποία έχουμε ξανασυναντήσει.

• Μήπως χάνουμε το βέλτιστο μονοπάτι; Όχι, γιατί αφού η h είναι συνεπής, το προηγούμενο μονοπάτι με το οποίο είχαμε φτάσει σε A ήταν το συντομότερο μέχρι A. − Και τα δύο υποδέντρα των A είναι ίδια.

Επινόηση ευρετικών συναρτήσεων για το πρόβλημα ιεραποστόλων και κανιβάλων

Θεωρούμε σε αυτό το παράδειγμα μοναδιαίο κόστος για κάθε διάσχιση της βάρκας από την μία μεριά στην άλλη.

(α) Η ευρετική συνάρτηση h1( ): Αφαιρούμε τον περιορισμό που δεν επιτρέπει ο αριθμός των κανιβάλων σε κάποια όχθη να υπερβεί τον αριθμό των ιεραποστόλων στην ίδια όχθη.

(α1) Ποιος είναι (ως συνάρτηση του n) ο ελάχιστος αριθμός κινήσεων μέχρι το στόχο (την τελική κατάσταση), αν βρισκόμαστε σε κατάσταση όπου η βάρκα βρίσκεται δεξιά (επομένως υπάρχει τουλάχιστον ένας άνθρωπος στη δεξιά όχθη) και υπάρχουν συνολικά n > 0 άνθρωποι στην αριστερή όχθη; Εξηγήστε τον υπολογισμό σας.

Στην περίπτωση αυτή, η βέλτιστη ακολουθία κινήσεων είναι να πάει ένας άνθρωπος που βρίσκεται στα δεξιά τη βάρκα αριστερά, να επιστρέψει δεξιά με δύο ανθρώπους κ.ο.κ. μέχρι να έχουν έρθει όλοι οι άνθρωποι στα δεξιά, κάτι που απαιτεί 2 · n κινήσεις (διασχίσεις).

(α2) Ποιος είναι ο ελάχιστος αριθμός κινήσεων μέχρι το στόχο, αν βρισκόμαστε σε κατάσταση όπου η βάρκα βρίσκεται αριστερά και υπάρχουν συνολικά n > 0 άνθρωποι στην αριστερή όχθη; Δώστε δύο ξεχωριστές απαντήσεις για n = 1 και n > 1; Εξηγήστε τον υπολογισμό σας.

Αν n = 1, τότε προφανώς χρειάζεται μόνο μία διάσχιση.

Αν n > 1, τότε χρειάζονται 2 · (n – 1) – 1 = 2n – 3 διασχίσεις·

για n = 2, προφανώς χρειάζεται πάλι μόνο μία διάσχιση, όσες προβλέπει σωστά ο τύπος·

για n = 3, η βάρκα πρέπει να πάει δεξιά με δύο ανθρώπους, να γυρίσει αριστερά με έναν και να ξαναπάει δεξιά πάλι με δύο, συνολικά 3 διασχίσεις, όσες προβλέπει ο τύπος κ.ο.κ.

(β) Η ευρετική συνάρτηση h2( ): Αφαιρούμε τώρα και τον περιορισμό ότι στη βάρκα χωράνε το πολύ δύο άτομα.

(β1) Ποιος είναι ο ελάχιστος αριθμός κινήσεων μέχρι το στόχο, αν βρισκόμαστε σε κατάσταση όπου η βάρκα βρίσκεται δεξιά και υπάρχουν συνολικά n > 0 άνθρωποι στην αριστερή όχθη; Εξηγήστε τον υπολογισμό σας.

Στην περίπτωση αυτή πρέπει η βάρκα να πάει αριστερά με έναν επιβάτη (ή και περισσότερους) και κατόπιν να γυρίσει δεξιά με όλους τους ανθρώπους της αριστερής όχθης, επομένως δύο κινήσεις.

(β2) Ποιος είναι ο ελάχιστος αριθμός κινήσεων μέχρι το στόχο, αν βρισκόμαστε σε κατάσταση όπου η βάρκα βρίσκεται αριστερά και υπάρχουν συνολικά n > 0 άνθρωποι στην αριστερή όχθη; Εξηγήστε τον υπολογισμό σας.

Προφανώς αρκεί μία διάσχιση.

(γ) Αφαίρεση περιορισμών: Επιστρέφουμε στο αρχικό πρόβλημα, όπου δεν έχει αφαιρεθεί κανένας περιορισμός. Βασιζόμενοι στις απαντήσεις που δώσατε στα σκέλη (α) και (β), γράψτε τους τύπους δύο αποδεκτών ευρετικών συναρτήσεων h1(s) και h2(s), για κάθε δυνατή κατάσταση s του αρχικού προβλήματος. Εξηγήστε πώς προέκυψαν οι δύο ευρετικές και γιατί είναι αποδεκτές.

Ως h1(s) και h2(s) θα χρησιμοποιήσουμε τα ακριβή κόστη των βέλτιστων λύσεων από την s ως την τελική κατάσταση στα απλοποιημένα προβλήματα των σκελών (α) και (β) αντίστοιχα. Γνωρίζουμε ότι το ακριβές βέλτιστο κόστος λύσης σε ένα απλοποιημένο πρόβλημα (που έχει προκύψει με αφαίρεση περιορισμών) είναι αποδεκτή ευρετική του αρχικού προβλήματος (αφού στο αρχικό πρόβλημα απαιτούνται περισσότερες ή το πολύ ίσες κινήσεις).

Έστω n ο συνολικός αριθμός ανθρώπων που στην κατάσταση s βρίσκονται στην αριστερή όχθη. Τότε:

h1(s) = 2 · n, αν η βάρκα βρίσκεται δεξιά στην s και n > 0,

1, αν η βάρκα βρίσκεται αριστερά στην s και n = 1,

2· n – 3, αν η βάρκα βρίσκεται αριστερά στην s και n > 1,

0, αν n = 0.

h2(s) = 2, αν η βάρκα βρίσκεται δεξιά στην s και n > 0,

1, αν η βάρκα βρίσκεται αριστερά στην s και n > 0,

0, αν n = 0.

(δ) Αν χρησιμοποιήσουμε μία από τις δύο ευρετικές του σκέλους (γ) και τον αλγόριθμο A* χωρίς κλειστό σύνολο, είναι σίγουρο ότι θα βρούμε λύση; Γιατί;

Ναι, γιατί το κόστος κάθε κίνησης είναι θετικό (>0) και ο μέγιστος παράγοντας διακλάδωσης (αριθμός δυνατών κινήσεων σε κάθε κατάσταση) είναι πεπερασμένος. Στην περίπτωση αυτή γνωρίζουμε ότι ο A* είναι πλήρης, δηλαδή αν υπάρχει λύση, σίγουρα τη βρίσκει. Και γνωρίζουμε ότι το πρόβλημα έχει λύση.

(ε) Είναι σίγουρο ότι η λύση που θα βρούμε στο σκέλος (δ) θα είναι βέλτιστη; Γιατί; Ναι, γιατί γνωρίζουμε ότι με αποδεκτή ευρετική ο A* είναι βέλτιστος.

(στ) Είναι σίγουρο ότι η λύση που θα βρούμε στο σκέλος (δ) θα είναι βέλτιστη ακόμη κι αν χρησιμοποιήσουμε κλειστό σύνολο; Γιατί;

Ναι, γιατί γνωρίζουμε ότι όταν το κόστος κάθε μετάβασης είναι θετικό, οι ευρετικές που προκύπτουν με αφαίρεση περιορισμών είναι συνεπείς. Και με συνεπή ευρετική, γνωρίζουμε ότι ο A* παραμένει βέλτιστος, ακόμα και όταν χρησιμοποιείται κλειστό σύνολο.

(ζ) Ποια από τις δύο ευρετικές του σκέλους (γ) είναι προτιμότερο να χρησιμοποιήσουμε και γιατί;

Είναι προτιμότερο να χρησιμοποιήσουμε την h1, γιατί είναι μεν και οι δύο αποδεκτές, αλλά η h1 κυριαρχεί επί της h2 (h1(s) >= h2(s), για κάθε s), που σημαίνει ότι δίνει ακριβέστερες προβλέψεις (προβλέπει πιο καλά το ακριβές βέλτιστο κόστος λύσης από την s ως την τελική κατάσταση), κάτι που γνωρίζουμε ότι βοηθά τον A* να εστιαστεί σε μικρότερο τμήμα του χώρου αναζήτησης.

4) Παραδείγματα χρήσης προγράμματος

Παράδειγμα 1:

number of missionaries and the number of cannibals for the initial state: 5

boat capacity: 3

maximum number of boat crossings: 100

Για τις συγκεκριμένες τιμές εισόδου από τον χρήστη, προκύπτει το παρακάτω αποτέλεσμα:

Search time:0.003 sec.

Boat side	Missionaries left	Cannibals left	Missionaries right	Cannibals right
0	5	5	0	0
1	5	3	0	2
0	5	4	0	1
1	5	1	0	4
0	5	2	0	3
1	2	2	3	3
0	3	3	2	2
1	0	3	5	2
0	0	4	5	1
1	0	2	5	3
0	0	3	5	2
1	0	0	5	5

Παράδειγμα 2:

number of missionaries and the number of cannibals for the initial state: 3

boat capacity: 2

maximum number of boat crossings: 80

Για τις συγκεκριμένες τιμές εισόδου από τον χρήστη, προκύπτει το παρακάτω αποτέλεσμα:

Search time:0.002 sec.

Boat side	Missionaries left	Cannibals left	Missionaries right	Cannibals right
0	3	3	0	0
1	3	1	0	2
0	3	2	0	1
1	3	0	0	3
0	3	1	0	2
1	1	1	2	2
0	2	2	1	1
1	0	2	3	1
0	0	3	3	0
1	0	1	3	2
0	0	2	3	1
1	0	0	3	3

5) Πειραματικά αποτελέσματα

Θεωρώ την παράμετρο b, που είναι η χωρητικότητα της βάρκας, δηλαδή, ο ελάχιστος αριθμός ατόμων που μπορεί να χωρέσει η βάρκα ώστε το πρόβλημα να έχει λύση:

b = 2 if n <= 3;

b = 3 if n = 4 or n = 5;

b = 4 if n >= 6; , όπου n, ο αριθμός των ατόμων

Συγκεκριμένα , b = n if n = 2, διαφορετικά b < n.

Άσκηση 2: Πρόβλημα Ν βασίλισσών

Περιεχόμενα:

1) Λίγα λόγια για το πρόβλημα

Το πρόβλημα των βασιλισσών :

Σε ένα board μεγέθους 8*8 πρέπει να τοποθετήσουμε 8 βασίλισσες με τέτοιο τρόπο ώστε καμία βασίλισσα να μην απειλείται.

Το γενικευμένο πρόβλημα των Ν βασιλισσών:

- Το πρόβλημα που επιλύσαμε, δέχεται ως μεταβλητές το μέγεθος της σκακιέρας που είναι ίδιο και με το πλήθος των βασιλισσών
- Θεωρούμε ότι στην αρχική κατάσταση έχουμε τυχαία boards με τυχαία τοποθετημένες επάνω τις βασίλισσές κατά μήκος του άξονα χ.
- Το πρόγραμμά βρίσκει μια από τις λύσεις. Αναφέρετε επίσης πόσο περίπου χρόνο (ανάλογα και με τον υπολογιστή) χρειάζεται το πρόγραμμά για να βρει λύση για τις ξεχωριστές κάθε φορά τιμές του Ν καθώς και εμφανίζει το σωστό board.

2) Αρχιτεκτονική προγράμματος

Το πρόγραμμα αποτελείτε από τις κλάσεις Main, Chromosome και GeneticAlgorihm

Main:

Στην κλάση Main δίνεται εντολή να δώσει ο χρήστης το επιθυμητό Ν. Έπειτα δημιουργείται ένα αντικείμενο τύπου GeneticAlgorihm στο οποίο δίνονται και τα καταλληλά ορίσματα (num και population). Στη συνέχεια καλείται η μέθοδος genetic Search του αντικειμένου με ορίσματα το mutation probability, το num, το population και το minfitness. Τέλος αφού έχει βρεθεί λύση εκτυπώνεται με την μέθοδο print Solution ο χρόνος το board καθώς και οι συντεταγμένες .

Chromosome:

Στην chromosome έχουμε έναν πίνακα genes που κρατάει τις συντεταγμένες x,y των βασιλισσών του chromosome,έχουμε δυο constructors στον πρώτο φτιάχνουμε ένα τυχαιοποιημένο board μεγέθους num και καλούμε και την calculate fitness() να υπολογίσουμε το fitness του chromosome, στον δεύτερο constructor

κατασκευάζουμε ένα chromosome με συγκεκριμένες ,ήδη υπάρχουσες συντεταγμένες δηλαδή έχουμε έναν copy constructor.Έπειτα έχουμε την Calculate fitness() που υπολογίζει με βάση το x,y και της διαγώνιους το fitness με τον τρόπο του εργαστήριου. Έπειτα έχουμε getters(), setters() και print του εργαστήριου.

GeneticAlgorithm:

Σε αυτή την κλάση έχουμε private δεδομένα, treedepth το βάθος του δέντρου, τον πίνακα population τύπου chromosome που είναι τα τυχαιοποιημένα chromosomes το chromosome bestSol και propableSecondbestsol και ο δυσδιάστατος πίνακας doubletree και το goalgen τα generetions που θέλει μέχρι να βρε λύση.Ο contructor έχει ως είσοδο την Num και το population, μέσα αρχικοποιουμαι τον population,υπολογίζουμε το βάθος του δέντρου με βάση το POPULATION που το παίρνουμε από την main,δίνουμε μέγεθος και γεμίζουμε το πίνακα population με τυχαία chromosomes.Στην μέθοδο GeneticSearch() παίρναμε τα ορίσματα num,POPULATION,mutationpropability και minfitness από την main, ξεκινάμε μια while μέχρι να βρει λύση δηλαδή μέχρι το fitness του bestSol να ισούται με το fitness που υπολογίσαμε στην main, καλούμε την μέθοδο tournayselect και την populate() που αναφέρουμε παρακάτω τι κάνουν και αυξάνουμε το goalgen για κάθε επανάληψη δηλαδή για κάθε καινούργια γενιά. Στη μέθοδο TourneySelect() αφού έχουμε καλέσει την createTournamentTree() προκειμένου να κατασκευάσουμε το δέντρο η οποία περιγράφεται ποιο κάτω, βάζουμε στο bestSol το τελευταίο κελί του doubletree και το πρώτο στοιχείο στον πίνακα σε αυτό το κελί, και καλούμε και την setPropableSecondBest() που εξηγούμε παρακάτω τι κάνει. Στην μέθοδο createTournamentTree() βάζουμε στην θέση 0 του δυσδιάστατου πίνακα doubleTree τον πίνακα population,φτιάχνουμε έναν πίνακα currentrow για να μην πειράξουμε τα δεδομένα της population,με μια for ξεκινάμε να δένουνε σε κάθε κελί της doubletree το καινούργιο πίνακα με το μισό μέγεθος που κατασκευάζουμε στον πίνακα nextrow με βάση την σύγκριση του κάθε κόμβου με τον διπλανό του ανά δυο, στην currentrow παίρναμε τον πίνακα που φτιάξαμε από αυτήν την σύγκριση και επαναλαμβάνουμε μέχρι στο doubletree[length-1] να μπει το καλύτερο chromosome.Στην setPropableSecondBestμε μια if else ελέγχουμε αν το propableSecondBest είναι ίδιο με το bestSol και αν δεν είναι τότε βάζουμε το δεύτερο παιδί στην αμέσως προηγουμένη θέση του πίνακα double tree. Στη μέθοδο populate() οπού δεχόμαστε σαν ορίσματα το num καθώς και τα δυο chromosomes που βρήκαμε παραπάνω , διαλέγουμε έναν τυχαίο int και έχοντας μια slice true με την χρήση της μεθόδου System.arraycopy σπάμε το κάθε chromosome στην θέση που μας υποδεικνύει το τυχαίο int και κατασκευάζουμε το καινούργιο chromosome το οποίο έχει από το [0] έως το τυχαίο int τα genes του πρώτου γονέα και από το τυχαίο int μέχρι και το num τα genes του δευτέρου. Έπειτα διαλέγουμε ένα τυχαίο double και αν είναι μικρότερο από το mutation probability τότε κάνουμε mutate αλλάζοντας τυχαία μια θέση βασίλισσας χρησιμοποιώντας τον constructor της chromosome δίνοντας το πίνακα new chromosome που φτιάξαμε και το Num τέλος αλλάζουμε το slice σε κάθε είσοδο στην for για να μπαίνει και στην If και στην else.Στην μέθοδο fitnessFunc(chromosome 1, chromosome 2) παίρνουμε δυο

chromosomes και επιστρέφουμε αυτό με το καλύτερο fitness.Τέλος στην PrintSol() εμφανίζουμε την λύση αν βρούμε και τα generetions.

3) Μέθοδοι Τεχνητής Νοημοσύνης που χρησιμοποιεί

Για την εύρεση της λύσης του προβλήματος γίνεται χρήση του γενετικού αλγορίθμου χωρίς περιορισμούς σε maxsteps, αλλά με περιορισμό στο population.

Γενετικός Αλγόριθμος: Μιμούνται στοιχεία του φυσικού μηχανισμού εξέλιξης. • Πληθυσμός καταστάσεων – Παριστάνονται ως χρωμοσώματα.
• Συνάρτηση καταλληλόλητας: αξιολογεί τις καταστάσεις του πληθυσμού.
• Αναπαραγωγή: Ζεύγη καταστάσεων του πληθυσμού παράγουν απογόνους. – Συνδυάζουν χαρακτηριστικά των προγόνων τους.
• Επιλογή: Οι καταλληλότερες καταστάσεις έχουν περισσότερες πιθανότητες να αναπαραχθούν.
• Σταδιακά υπερισχύουν στον πληθυσμό οι καταστάσεις με τα καλύτερα χαρακτηριστικά.
Κάθε χρωμόσωμα παριστάνει μια κατάσταση.
– Κάθε γονίδιο παριστάνει τη θέση μιας βασίλισσας στη στήλη της.
– Έχουμε απαλλαγεί από καταστάσεις όπου υπάρχουν δύο ή περισσότερες βασίλισσες ανά στήλη, αφού δεν αποτελούν λύσεις

• Μετάλλαξη: Για κάθε γονίδιο του απογόνου, υπάρχει μια μικρή πιθανότητα να πάρει τυχαία τιμή.

Συνάρτηση καταλληλότητας: συνήθως απεικονίζει κάθε χρωμόσωμα σε έναν πραγματικό Î [0, 1].

• Η τιμή της συνάρτησης καταλληλότητας μπορεί να παριστάνει την αξία του χρωμοσώματος. – Π.χ. κόστος υλοποίησης προγράμματος εξετάσεων.

• Ή μπορεί να αποτελεί ευρετική εκτίμηση της απόστασης από μια τελική κατάσταση. – Π.χ. ποσοστό περιορισμών που παραβιάζονται.

• Μπορεί να είναι προτιμότερη μια λιγότερο ακριβής συνάρτηση καταλληλότητας που μπορεί όμως να υπολογισθεί γρηγορότερα. – Περισσότερες αναπαραγωγές στον ίδιο χρόνο.

• Σε κάθε αναπαραγωγή επιλέγουμε τους δύο γονείς με πιθανότητα ανάλογη της αξίας τους.

• Χρωμοσώματα με μεγάλη αξία έχουν μεγάλη πιθανότητα να ζευγαρώσουν πολλές φορές.

• Χρωμοσώματα με μικρή αξία είναι πιθανότερο να μη ζευγαρώσουν καθόλου.

4)Παραδείγματα χρήσης προγράμματος

Παράδειγμα 1:

N: 8

```
Enter the number of N: 8
Chromosome : |4|6|0|3|1|7|5|2|, Fitness : 28
-------------------------------------
| | |Q| | | | | | |
| | | | |Q| | | | |
| | | | | | | |Q|
| | | |Q| | | | |
|Q| | | | | | | |
| | | | | | |Q| |
| |Q| | | | | | |
| | | | | |Q| | |
-------------------------------------
Generations: 209
That took 0.018 seconds
```

Search time:0.018 sec.

Παράδειγμα 2:

N: 20

```
Enter the number of N: 20
Chromosome : |15|18|0|13|5|12|2|6|16|9|4|1|14|17|19|10|7|3|11|8|, Fitness : 190
-----------------------------------
| | |Q| | | | | | | | | | | | | | | | | |
| | | | | | | | | | |Q| | | | | | | |
| | | | | | |Q| | | | | | | | | | | |
| | | | | | | | | | | | | | | | |Q| | |
| | | | | | | | | |Q| | | | | | | | |
| | | | |Q| | | | | | | | | | | | | |
| | | | | | | |Q| | | | | | | | | | |
| | | | | | | | | | | | | | |Q| | | |
| | | | | | | | | | | | | | | | | |Q|
| | | | | | | | | |Q| | | | | | | | |
| | | | | | | | | | | | | |Q| | | | |
| | | | | | | | | | | | | | | |Q| | |
| | | | |Q| | | | | | | | | | | | | |
| | | |Q| | | | | | | | | | | | | | |
| | | | | | | | | | |Q| | | | | | | |
|Q| | | | | | | | | | | | | | | | | |
| | | | | | | | |Q| | | | | | | | | |
| | | | | | | | | | | |Q| | | | | | |
| |Q| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | |Q| | | | | |
-----------------------------------
Generations: 3891
That took 0.322 seconds
```

Search time:0.322 sec.

Παράδειγμα 3:

N: 100

```
Enter the number of N: 100
Chromosome : |9|47|44|54|25|

56|18|, Fitness : 4950


-----------------------------------
Generations: 4315
That took 29.594 seconds
```

*Το board δεν φαίνεται διότι είναι πάρα πολύ μεγάλο

Search time:29.594sec.