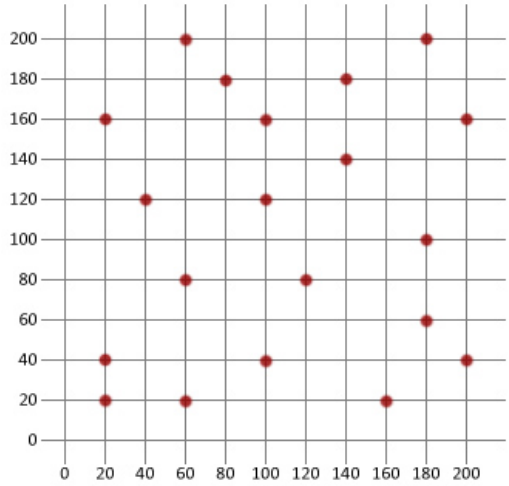
20th August 2012 at 7:58

# Applying a genetic algorithm to the traveling salesman problem




Imagine you are given a map like the one opposite. It contains a total of 20 different cities and it's your job to visit each of these cities. Before you set off on your journey you'll probably first want to plan a route so you can minimize your travel time. As humans we're pretty good at this, we can easily work out a reasonably good route without needing to do much more than glance at the map. But when we've found a route that we believe is optimal how can we test if it really is the optimal route? Well in short, we can't. Well, we can, but not really. To understand the problem here let's consider there are now just 3 cities to visit instead of the original 20. To get a single route, we would first have to choose from a choice of 3 different starting cities, then we'd have a choice of 2 second cities then there is just 1 final city left to pick to complete our route. This would give us 3 x 2 x 1 different routes in total. That's only 6 in this example, so it's pretty simple to check every route for the shortest if there are just 3 cities. If you're good at maths you may have already realized this is a factorial and factorials grow in size remarkably quick. A factorial of 10 is 3628800 and a factorial of 20 would be 2432902008176640000. So if we wanted to find the shortest route in our 20 city example we would have to test 2432902008176640000 different routes! Even with modern computing power this is unpractical and with bigger problems, it's close to impossible.

## Finding a solution

Although it isn't practical to find the best solution for a problem like ours, we have algorithms that let us discover close to optimum solutions such as the nearest neighbor algorithm and swarm optimization. These algorithms can find 'good-enough' solutions to the travelling salesman problem surprisingly quickly. In this tutorial we will be looking at using a genetic algorithm to find a solution to the travelling sales man problem. If you're not familiar with what a genetic algorithm is and how they work then please have a look at the introductory tutorial below:
Creating a genetic algorithm for beginners

Finding a solution to the travelling salesman problem requires we set up our genetic algorithm in a certain specialized way. For instance, a valid solution would need to represent a route where every city is included at least once and only once. This would mean adjusting our mutation function so it doesn't just add a random city to the route, possibility causing a duplicate. The crossover function will also need to be similarly altered. One type of mutation we could use to prevent us evolving invalid solutions is swap mutation. In swap mutation if we have a set of objects we select two objects at random then simply swap their positions. Because we are only swapping objects around we don't risk causing duplicate objects within our solution.

## Twitter Feed

"@garcosc Heh. That's probably my web host then. Thanks, @justhost"
30th May 2014, 21:41:00 | Link

"@garcosc That's weird, can you access, http://t.co/TsekiRxX3K ?"
30th May 2014, 21:36:14 | Link

"@nvrqt Eh, that's probably my awful web host. I tested on my friends Galaxy S3 earlier."
30th May 2014, 21:27:56 | Link

"I built a little HTML5 multiplayer quiz app. I'd love to hear any feedback, http://t.co/9684m3LNdf"
30th May 2014, 21:21:26 | Link

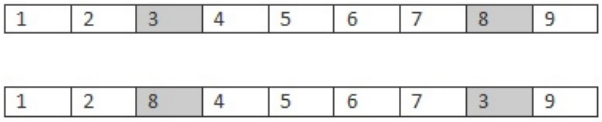
"I wish I was younger. I wish I was smarter."
22nd May 2014, 20:15:22 | Link

""I don't want to earn my living, I want to live""
20th May 2014, 17:36:12 | Link

"@R_D_Vincent I wrote quite a bit. A little about myself, a little about the human race in general and some challenges for the near future."
28th April 2014, 12:50:00 | Link

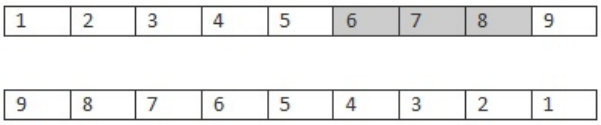
"I am the universe expressing itself as a human for a while."
28th April 2014, 12:46:30 | Link

@leejacobson_

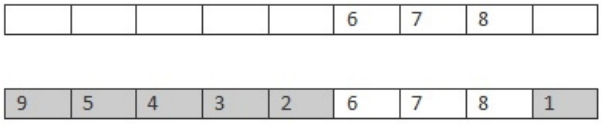



Now we've dealt with our mutation function we need to sort out our crossover function. One algorithm we could use to produce valid offspring for our next generation is ordered crossover. In this algorithm we select a subset from our first parent, then add that subset to our child. Finally we add the objects which are not yet in our child to our child in the second parent's order.

**Parents**




**Child**




## Creating our GA

Now let's look at the code of our GA. The first step is to create a class that can encode the tour cities.

**City.java**

```
/*
* City.java
* Models a city
*/

package tsp;

public class City {
    int x;
    int y;

    // Constructs a randomly placed city
    public City(){
        this.x = (int)(Math.random()*200);
        this.y = (int)(Math.random()*200);
    }

    // Constructs a city at chosen x, y location
    public City(int x, int y){
        this.x = x;
        this.y = y;
    }

    // Gets city's x coordinate
    public int getX(){
        return this.x;
    }

    // Gets city's y coordinate
    public int getY(){
        return this.y;
    }

    // Gets the distance to given city
    public double distanceTo(City city){
        int xDistance = Math.abs(getX() - city.getX());
        int yDistance = Math.abs(getY() - city.getY());
        double distance = Math.sqrt( (xDistance*xDistance) + (yDistance*yDistance) );

        return distance;
    }

    @Override
    public String toString(){
        return getX()+", "+getY();
    }
```

```
}
```

Now we can create a class that holds all of our destination cities for our tour

**TourManager.java**

```
/*
* TourManager.java
* Holds the cities of a tour
*/

package tsp;

import java.util.ArrayList;

public class TourManager {

    // Holds our cities
    private static ArrayList destinationCities = new ArrayList<City>();

    // Adds a destination city
    public static void addCity(City city) {
        destinationCities.add(city);
    }

    // Get a city
    public static City getCity(int index){
        return (City)destinationCities.get(index);
    }

    // Get the number of destination cities
    public static int numberOfCities(){
        return destinationCities.size();
    }
}
```

Next we need a class that can encode our routes, these are generally referred to as tours so we'll stick to the convention.

**Tour.java**

```
/*
* Tour.java
* Stores a candidate tour
*/

package tsp;

import java.util.ArrayList;
import java.util.Collections;

public class Tour{

    // Holds our tour of cities
    private ArrayList tour = new ArrayList<City>();
    // Cache
    private double fitness = 0;
    private int distance = 0;

    // Constructs a blank tour
    public Tour(){
        for (int i = 0; i < TourManager.numberOfCities(); i++) {
            tour.add(null);
        }
    }

    public Tour(ArrayList tour){
        this.tour = tour;
    }

    // Creates a random individual
    public void generateIndividual() {
        // Loop through all our destination cities and add them to our tour
        for (int cityIndex = 0; cityIndex < TourManager.numberOfCities(); cityIndex++) {
          setCity(cityIndex, TourManager.getCity(cityIndex));
        }
        // Randomly reorder the tour
        Collections.shuffle(tour);
    }
```

```
    // Gets a city from the tour
    public City getCity(int tourPosition) {
        return (City)tour.get(tourPosition);
    }

    // Sets a city in a certain position within a tour
    public void setCity(int tourPosition, City city) {
        tour.set(tourPosition, city);
        // If the tours been altered we need to reset the fitness and distance
        fitness = 0;
        distance = 0;
    }

    // Gets the tours fitness
    public double getFitness() {
        if (fitness == 0) {
            fitness = 1/(double)getDistance();
        }
        return fitness;
    }

    // Gets the total distance of the tour
    public int getDistance(){
        if (distance == 0) {
            int tourDistance = 0;
            // Loop through our tour's cities
            for (int cityIndex=0; cityIndex < tourSize(); cityIndex++) {
                // Get city we're travelling from
                City fromCity = getCity(cityIndex);
                // City we're travelling to
                City destinationCity;
                // Check we're not on our tour's last city, if we are set our
                // tour's final destination city to our starting city
                if(cityIndex+1 < tourSize()){
                    destinationCity = getCity(cityIndex+1);
                }
                else{
                    destinationCity = getCity(0);
                }
                // Get the distance between the two cities
                tourDistance += fromCity.distanceTo(destinationCity);
            }
            distance = tourDistance;
        }
        return distance;
    }

    // Get number of cities on our tour
    public int tourSize() {
        return tour.size();
    }

    // Check if the tour contains a city
    public boolean containsCity(City city){
        return tour.contains(city);
    }

    @Override
    public String toString() {
        String geneString = "|";
        for (int i = 0; i < tourSize(); i++) {
            geneString += getCity(i)+"|";
        }
        return geneString;
    }
}
```

We also need to create a class that can hold a population of candidate tours

**Population.java**

```
/*
* Population.java
* Manages a population of candidate tours
*/

package tsp;

public class Population {

    // Holds population of tours
```

```
    Tour[] tours;

    // Construct a population
    public Population(int populationSize, boolean initialise) {
        tours = new Tour[populationSize];
        // If we need to initialise a population of tours do so
        if (initialise) {
            // Loop and create individuals
            for (int i = 0; i < populationSize(); i++) {
                Tour newTour = new Tour();
                newTour.generateIndividual();
                saveTour(i, newTour);
            }
        }
    }

    // Saves a tour
    public void saveTour(int index, Tour tour) {
        tours[index] = tour;
    }

    // Gets a tour from population
    public Tour getTour(int index) {
        return tours[index];
    }

    // Gets the best tour in the population
    public Tour getFittest() {
        Tour fittest = tours[0];
        // Loop through individuals to find fittest
        for (int i = 1; i < populationSize(); i++) {
            if (fittest.getFitness() <= getTour(i).getFitness()) {
                fittest = getTour(i);
            }
        }
        return fittest;
    }

    // Gets population size
    public int populationSize() {
        return tours.length;
    }
}
```

Our next class will need to evolve our population of solutions

**GA.java**

```
/*
* GA.java
* Manages algorithms for evolving population
*/

package tsp;

public class GA {

    /* GA parameters */
    private static final double mutationRate = 0.015;
    private static final int tournamentSize = 5;
    private static final boolean elitism = true;

    // Evolves a population over one generation
    public static Population evolvePopulation(Population pop) {
        Population newPopulation = new Population(pop.populationSize(), false);

        // Keep our best individual if elitism is enabled
        int elitismOffset = 0;
        if (elitism) {
            newPopulation.saveTour(0, pop.getFittest());
            elitismOffset = 1;
        }

        // Crossover population
        // Loop over the new population's size and create individuals from
        // Current population
        for (int i = elitismOffset; i < newPopulation.populationSize(); i++) {
            // Select parents
            Tour parent1 = tournamentSelection(pop);
            Tour parent2 = tournamentSelection(pop);
            // Crossover parents
            Tour child = crossover(parent1, parent2);
```

```
            // Add child to new population
            newPopulation.saveTour(i, child);
        }

        // Mutate the new population a bit to add some new genetic material
        for (int i = elitismOffset; i < newPopulation.populationSize(); i++) {
            mutate(newPopulation.getTour(i));
        }

        return newPopulation;
    }

    // Applies crossover to a set of parents and creates offspring
    public static Tour crossover(Tour parent1, Tour parent2) {
        // Create new child tour
        Tour child = new Tour();

        // Get start and end sub tour positions for parent1's tour
        int startPos = (int) (Math.random() * parent1.tourSize());
        int endPos = (int) (Math.random() * parent1.tourSize());

        // Loop and add the sub tour from parent1 to our child
        for (int i = 0; i < child.tourSize(); i++) {
            // If our start position is less than the end position
            if (startPos < endPos && i > startPos && i < endPos) {
                child.setCity(i, parent1.getCity(i));
            } // If our start position is larger
            else if (startPos > endPos) {
                if (!(i < startPos && i > endPos)) {
                    child.setCity(i, parent1.getCity(i));
                }
            }
        }

        // Loop through parent2's city tour
        for (int i = 0; i < parent2.tourSize(); i++) {
            // If child doesn't have the city add it
            if (!child.containsCity(parent2.getCity(i))) {
                // Loop to find a spare position in the child's tour
                for (int ii = 0; ii < child.tourSize(); ii++) {
                    // Spare position found, add city
                    if (child.getCity(ii) == null) {
                        child.setCity(ii, parent2.getCity(i));
                        break;
                    }
                }
            }
        }
        return child;
    }

    // Mutate a tour using swap mutation
    private static void mutate(Tour tour) {
        // Loop through tour cities
        for(int tourPos1=0; tourPos1 < tour.tourSize(); tourPos1++){
            // Apply mutation rate
            if(Math.random() < mutationRate){
                // Get a second random position in the tour
                int tourPos2 = (int) (tour.tourSize() * Math.random());

                // Get the cities at target position in tour
                City city1 = tour.getCity(tourPos1);
                City city2 = tour.getCity(tourPos2);

                // Swap them around
                tour.setCity(tourPos2, city1);
                tour.setCity(tourPos1, city2);
            }
        }
    }

    // Selects candidate tour for crossover
    private static Tour tournamentSelection(Population pop) {
        // Create a tournament population
        Population tournament = new Population(tournamentSize, false);
        // For each place in the tournament get a random candidate tour and
        // add it
        for (int i = 0; i < tournamentSize; i++) {
            int randomId = (int) (Math.random() * pop.populationSize());
            tournament.saveTour(i, pop.getTour(randomId));
        }
        // Get the fittest tour
        Tour fittest = tournament.getFittest();
        return fittest;
```

```
    }
}
```

Now we can create our main method, add our cities and evolve a route for our travelling salesman problem.

**TSP_GA.java**

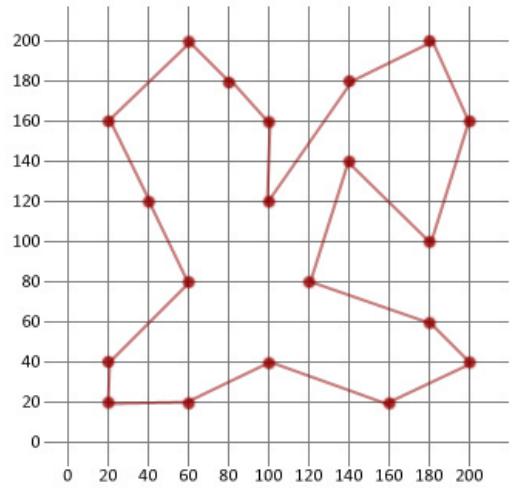
```
/*
* TSP_GA.java
* Create a tour and evolve a solution
*/

package tsp;

public class TSP_GA {

    public static void main(String[] args) {

        // Create and add our cities
        City city = new City(60, 200);
        TourManager.addCity(city);
        City city2 = new City(180, 200);
        TourManager.addCity(city2);
        City city3 = new City(80, 180);
        TourManager.addCity(city3);
        City city4 = new City(140, 180);
        TourManager.addCity(city4);
        City city5 = new City(20, 160);
        TourManager.addCity(city5);
        City city6 = new City(100, 160);
        TourManager.addCity(city6);
        City city7 = new City(200, 160);
        TourManager.addCity(city7);
        City city8 = new City(140, 140);
        TourManager.addCity(city8);
        City city9 = new City(40, 120);
        TourManager.addCity(city9);
        City city10 = new City(100, 120);
        TourManager.addCity(city10);
        City city11 = new City(180, 100);
        TourManager.addCity(city11);
        City city12 = new City(60, 80);
        TourManager.addCity(city12);
        City city13 = new City(120, 80);
        TourManager.addCity(city13);
        City city14 = new City(180, 60);
        TourManager.addCity(city14);
        City city15 = new City(20, 40);
        TourManager.addCity(city15);
        City city16 = new City(100, 40);
        TourManager.addCity(city16);
        City city17 = new City(200, 40);
        TourManager.addCity(city17);
        City city18 = new City(20, 20);
        TourManager.addCity(city18);
        City city19 = new City(60, 20);
        TourManager.addCity(city19);
        City city20 = new City(160, 20);
        TourManager.addCity(city20);

        // Initialize population
        Population pop = new Population(50, true);
        System.out.println("Initial distance: " + pop.getFittest().getDistance());

        // Evolve population for 100 generations
        pop = GA.evolvePopulation(pop);
        for (int i = 0; i < 100; i++) {
            pop = GA.evolvePopulation(pop);
        }

        // Print final results
        System.out.println("Finished");
        System.out.println("Final distance: " + pop.getFittest().getDistance());
        System.out.println("Solution:");
        System.out.println(pop.getFittest());
    }
}
```

**Output:**

```
Initial distance: 1996
Finished
Final distance: 940
Solution:
|60, 200|20, 160|40, 120|60, 80|20, 40|20, 20|60, 20|100, 40|160, 20|200, 40|180, 60|120, 80|
```

As you can see in just 100 generations we were able to find a route just over twice as good as our original and probably pretty close to optimum.

**Final Results:**




If you liked this tutorial you might also enjoy, Simulated Annealing for beginners

## Social Links



## Tags

genetic-algorithms artificial-intelligence java algorithm tsp

## Related Articles

Simulated Annealing for beginners
Introduction to Artificial Neural Networks - Part 1
Creating a genetic algorithm for beginners
Autobots and Beyond
Consciousness, coming to a machine near you.

## Comments

**Riccardo** • 11 months ago

Hi! I have some problem with compilation:

javac -Xlint *.java
Tour.java:22: warning: [unchecked] unchecked call to add(E) as a member of the raw type java.util.ArrayList
tour.add(null);
^
Tour.java:47: warning: [unchecked] unchecked call to set(int,E) as a member of the raw type java.util.ArrayList
tour.set(tourPosition, city);
^
TourManager.java:17: warning: [unchecked] unchecked call to add(E) as a member of the raw type java.util.ArrayList
destinationCities.add(city);

Where do I'm wrong?
Thanks!

4 • Reply • Share ›

**basteln** • 6 months ago

Hi, next year ill have to give a presentation about genetics algorithms and the travelling salesman problem. So I c&p your code and programmed a gui to illustrate the tsp to my audience.

Now I try to understand your code. I understood everything except this number 100 in this loop here:

Population pop = new Population(50, true);

the 50 is the number of new populations

// Evolve population for 50 generations
pop = GA.evolvePopulation(pop);
for (int i = 0; i < 100; i++) << ist this just a random number, or is it populationsize * 2

Would it be the following for 100 populations?

Population pop = new Population(100, true);

// Evolve population for 50 generations
pop = GA.evolvePopulation(pop);
for (int i = 0; i < 200; i++)

Thanks a lot!
Greatz from Germany (:

1 • Reply • Share ›

**Lee Jacobson** Mod ➚ basteln • 6 months ago

That's my mistake.
The comment should have read 100 generations, not 50. Thanks for pointing that out, I've updated it now. Sorry for the confusion!

1 • Reply • Share ›

**Yuriy Chernyshov** • 8 months ago

Good article but a little bit pure explanation.
Suppose I have 25 cities, following Your idea I do initialize they and set into TourManager.
Now, what I do need is to get exact answer for the question: the minimum cost of a traveling salesman tour for this instance.
Could You explain in more details about Population and GA classes. Running Your code every time I get different results.

1 • Reply • Share ›

**turbofart** • 2 years ago

Hi Lee, I've found your article & code interesting, and decided to rewrite your code to Pyton:
http://goo.gl/xtKrr

1 • Reply • Share ›

**Lee Jacobson** Mod ➔ turbofart • 2 years ago

Thanks for sharing!

• Reply • Share ›

**Michał Błażej** • 3 days ago

Hi guys i want to make a simple presentation in java for that algorithm. Anyone know where i can find a good tutorial for that?

• Reply • Share ›

**Exclusive** • 21 days ago

Hi,
Your article about the implementation of Genetic Algorithm Tsp question and got
a very good example. How Memetics algorithms developed for this problem?

Thanks

• Reply • Share ›

**Karan Dhami** • a month ago

REally good tutorial. Excellent work. Can you help me with just one thing. What will you say is your chromosome for this algorithm or for this problem? Please it is really urgent if you or anyone can reply.

• Reply • Share ›

**Myagmarbat** • 2 months ago

hi

• Reply • Share ›

**teja kondapalli** • 3 months ago

hi,please suggest me,how to create timetable for college by using genetics algorithm,how to implement that algorithm..

• Reply • Share ›

**Ravi Shankar Soni** • 4 months ago

plz tell me how to run all the classes above in netbean.
It shows

Main method not found in class tsp.TSP_GA

• Reply • Share ›

**Mike** • 4 months ago

I've been reading the basic article on GA and this one but still haven't been able to figure out how to apply a similar GA for the Mastermind problem http://en.wikipedia.org/wiki/M...
Some ideas regarding this would be helpful and I would really appreciate the help.
The main class, population and individual are practically the same as in the basic GA article, but i am having some problems with the Algorithm and FitnessCalc classes.
I represented the permutation as "1234" where "1", "2", "3" and "4" represent the color code that a peg has. (4 pegs, 6 colors used).
I am unable to understand how the genetic operators are to be applied and also how do we calculate the fitness of an individual with the above representation.

Thanks and waiting for an answer

• Reply • Share ›

**Reuel Ramos Ribeiro** • 5 months ago

Thank you. Very helpful your article.

• Reply • Share ›

zura • 6 months ago

hello people! this is very good tutorial for me.

But now i have one question, when i run there code in my computer i get different result every running.

for example:

Initial distance: 3184
Finished
Final distance: 2244
Solution:
|898, 20|200, 160|180, 200|140, 180|140, 140|100, 160|80, 180|60, 200|20, 160|40, 120|100, 120|120, 80|60, 80|20, 40|60, 20|100, 40|160, 20|200, 40|180, 60|180, 100|

Initial distance: 3129
Finished
Final distance: 2355
Solution:
|80, 180|60, 200|20, 160|40, 120|100, 160|140, 180|140, 140|120, 80|100, 40|60, 20|20, 40|60, 80|100, 120|180, 100|180, 60|160, 20|200, 40|898, 20|200, 160|180, 200|

why i'm getting different result, can anybody help me?

• Reply • Share ›

Lee Jacobson Mod ➚ zura • 6 months ago

This is to be expected when using genetic algorithms. The solutions are randomly generated so every run will produce different results.

Check out my, "Creating a genetic algorithm for beginners." tutorial if you're unsure about how genetic algorithms work.

• Reply • Share ›

zura ➚ Lee Jacobson • 6 months ago

i don't understand where is random operation used, i see only one place in Cyti(Construqtor) class, where random is used, but when i debug this random operation not work.

I want to create array of City class and then use genetic algorithm to get solution. how to solve my exampl?

• Reply • Share ›

zura ➚ zura • 6 months ago

sory now i see random() operation, why use random? can i remove all random operation from my code? is this good idea?

• Reply • Share ›

Lee Jacobson Mod ➚ zura • 6 months ago

I urge you to read up a bit more on GAs. But no, if you remove the randomness it wouldn't be a genetic algorithm.

For more information on how GAs use randomness to create solutions:
http://www.theprojectspot.com/...

1 • Reply • Share ›

Le Bac • 8 months ago

It works!!! Thank you very much for the tutorial.

• Reply • Share ›

**shenaiata gajbhiye** • 9 months ago

sir could u plz help me i have to take these city as input through user by using applet

• Reply • Share ›

**Chandler** • 10 months ago

In getDistance method in Tour class, you could have written

destinationCity = getCity((cityIndex + 1) % tourSize());

instead of

```
if(cityIndex+1 < tourSize()){
destinationCity = getCity(cityIndex+1);
}else{
destinationCity = getCity(0);
}
```

Just a tip. Still great project though. It helped me a lot. Thanks :)

• Reply • Share ›

**akari** • 11 months ago

what kind a method for your mutation?
inversion mutation?
thx..

• Reply • Share ›

**Bruce** • a year ago

How would you change the fitness function to prevent route crossing?

For example, I am using this
(3, 1), (5, 1), (7, 1), (8, 2), (6, 2), (4, 2), (3, 3), (5, 3), (7, 3),
(8, 4), (6, 4), (4, 4), (5, 5), (7, 5), (8, 6), (6, 6), (7, 7), (8, 8).

I want to start at (3,1) and end at (8,8)
The actual problem is linking water pipes.

• Reply • Share ›

**Lee Jacobson** Mod ➚ Bruce • a year ago

What do you mean by route crossing? The route you provided seems optimal to me.

• Reply • Share ›

**Bruce** ➚ Lee Jacobson • a year ago

OK, now I have learned something. The route I started with was not optimal. GA gave me one with a shorter length. But then I realized that I don't care about the length. What I needed was simply a visit to every node (from start to end) and this problem is a Hamiltonian path.

Thanks for the info...

• Reply • Share ›

**bruce** ➚ Lee Jacobson • a year ago

lucky for me, I just noticed that someone translated your code to python... and he did a great job.

Here is an example of what I'm trying to automate.
http://dl.dropboxusercontent.c...



• Reply • Share ›

**Lee Jacobson** Mod ↗ bruce • a year ago

Changing the fitness function to punish a solution that doesn't meet your constraints probably won't give great results. Ideally you're going to want to edit the crossover and mutation functions to ensure the solutions being created by your GA fit the constraints.

• Reply • Share ›

**Bruce** ↗ Lee Jacobson • a year ago

Thanks, I wasn't sure why you were changing the mutation and cross-over but now it makes sense. I will also change getDistance to exclude the tour back to the starting city.

The SA code is conceptually easier, did you compare GA and SA in terms of speed? More important, does one of these methods consistently get a shorter distance?

• Reply • Share ›

**Bruce** ↗ Lee Jacobson • a year ago

It's the end view of a heat exchanger. The co-ordinates are the ends of the tubes that have to be linked to complete a circuit. You can't cross link the pipes because of space. Since each tube must be connected, I thought TSM might give a possible solution. You are right, the order that I entered is probably optimal but it's not always as clear. In the real case, there is a bank of tubes and multiple parallel circuits. Java is not my language but your code looked easy to understand so I'm using it to experiment with GA.

• Reply • Share ›

**lili** • a year ago

Hi, i wanted to know what kind of crossover do you use here, is it the position-based crossover?

• Reply • Share ›

**Lee Jacobson** Mod ↗ lili • a year ago

Ordered crossover is used in the example

• Reply • Share ›

**aiaimanel** • a year ago

Hello, i wanted to congrat you for your amazing program but i have one question..

how do you know that a solution is the optimum?

isn't there a better option than applying a for to 100 generations? why not generate until you get a population with a 90% fitness or something like that?

My regards and sorry for my english.

• Reply • Share ›

**Lee Jacobson** Mod ↗ aiaimanel • a year ago

To say a solution is 90% optimum you would need to know what the optimum solution is to compare it against. To know what the optimum is you would need to try every possible solution which would be practically impossible, also if you knew what the optimum solution was you wouldn't be running the algorithm in the first place.

The best we can do is say, Solution X is better than Solution Y and apply a higher fitness value to it.

1 • Reply • Share ›

**aiaimanel** ↗ Lee Jacobson • a year ago

you're right, this isn't the kind of problem you could know the optimum solution, like 8-puzzle or the magic square..

but can you know if an optimum solution is found before the for ends?
and another question.. by looking to your code (which, once again, is really good) i've the feeling that you calculate the distance using every city, like this: 1 dist 2, 2 dist 3, 3 dist 4, 4 dist 1.. am i right?

• Reply • Share ›

**FreddyJoe** • a year ago

Anyone know how I can imeplement this in Simulink? I allready have the "cities" in an array, and just need to calculate the route, then send it to my vehicle =)

• Reply • Share ›

**mary92** • a year ago

hii....well...i have a problem and it is finding the main action of this code in order to analyze the time and memory orders of this code...I would appreciate you if u can help me on this problem.....thank u in advance....

• Reply • Share ›

**hamderlat** • a year ago

please am new to the topic and I have been asked to do a research work on how genetic algorithm is used to solve the traveling salesman problem. I would like it if you can write the GA code for the 3 city example you gave in the introduction of the topic for easy understanding. thanks

• Reply • Share ›

**Lee Jacobson** Mod ↗ hamderlat • a year ago

Just edit the code where the cities are initialised. Just keep the first 3 if you want to have 3 cities:

```
City city = new City(60, 200);
TourManager.addCity(city);
City city2 = new City(180, 200);
TourManager.addCity(city2);
City city3 = new City(80, 180);
TourManager.addCity(city3);
```

1 • Reply • Share ›

**devkbsc** • a year ago

Hello, How the genetique algo works with this project. I don't what value to give for the mutation rate and tournamentsize. the following is my data.

<points>

<size>20</size>

<point x="1.0" y="0.5"/>

<point x="0.904508497187" y="0.793892626146"/>

<point x="0.654508497187" y="0.975528258148"/>

<point x="0.345491502813" y="0.975528258148"/>

<point x="0.0954915028125" y="0.793892626146"/>

<point x="0.0" y="0.5"/>

<point x="0.0954915028125" y="0.206107373854"/>

see more

• Reply • Share ›

**Lee Jacobson** Mod ↗ devkbsc • a year ago

There isn't really a correct answer.

You want to use a mutation rate that's not so large that it prevents solutions from converging, and not too small that there isn't enough variation within the population.

With the tournament size you need to pick a size that's big enough to prevent too many weaker individuals being selected, while being small enough that some less optimal solutions can occasionally be selected, providing genetic variation.

• Reply • Share ›

**Odpadnem** • a year ago

Hi, excelent article. I have just one question. Is it ok, that in my implementation results vary each run. I mean, given those 20 cities of yours. Should it vary even before evolving a population?

• Reply • Share ›

**Lee Jacobson** Mod → Odpadnem • a year ago

Yep this is to be expected. The initial population is generated randomly. We do this because we initially have no idea where an optimal solution is within the search space.

• Reply • Share ›

**Deki Satria** • a year ago

ehm, could you please tell me how to make that diagram in java ?

• Reply • Share ›

**That Guy** • a year ago

HI, I know nobody has posted in a while but I guess ill give this a shot. I want to create the traveling salesman problem but I would not like the cities to be predefined. I want in the UI for a person to click on the location on the form they would like the city to be and that a class be automatically created for that city. The problem that occurs is that I don't know how to automate class creation. Click one would create city1, click two would create city2 and so on. If you know the answer to this it would be much appreciated thanks.

• Reply • Share ›

**Lee Jacobson** Mod → That Guy • a year ago

You'd need to create a GUI with a click event that adds a city:

```
City city = new City([x], [y]);
TourManager.addCity(city);
```

• Reply • Share ›

**0729** • 2 years ago

hi, I am using genetic algorithms to do a project which is similar to yours in some point. I have places to visit and also restaurants to visit at the same time. Actually I am created a travel itinerary. My question is how do I included different gene in a chromosome. Put it simple, my itinerary will be TourismSpot1-->TourismSpot2->Restaurant1(Lunch)->TourismSpot2->Restaurant2 ( Dinner ) . How I need to specify things, so the algorithm will know it is a time to included restaurant instead of tourism spot? Looking forward to hear your ideas. Thanks! By the way, this is a very good tutorials.

• Reply • Share ›

**wujtehacjusz** • 2 years ago

Nice article. I think that you might have a bug in the mutation part.

```
// Get the cities at position in tour
City city1 = tour.getCity(tourPos1);
City city2 = tour.getCity(tourPos2);

// Swap them around
tour.setCity(tourPos1, city1);
tour.setCity(tourPos2, city2);
```

This will not swap the cities around...

• Reply • Share ›

**Lee Jacobson** Mod ➚ wujtehacjusz • 2 years ago

Thanks amended.

2 • Reply • Share ›

**Chinku** ➚ Lee Jacobson • a year ago

Can anyone help me how to apply genetic algorithm to detect communities in social network? Suppose, we have one graph and we can easily make adjacency matrix from that graph. Now my query is how to apply genetic algorithm to detect the communities(nodes are dense intra-connected) in that graph?

• Reply • Share ›

Subscribe Add Disqus to your site

**Thanks for visiting!**

Lets keep in touch,

Send me an email or follow me on Google+

Home | About Me | Projects | Articles | Tutorials | Blog

Built by cwpStudios.com

Maintained and updated by Lee Jacobson