12TH FEBRUARY 2012 AT 11:19 · BY LEE JACOBSON

Creating a genetic algorithm for beginners

Introduction

A genetic algorithm (GA) is great for finding solutions to complex search problems. They're often used in fields such as engineering to create incredibly high quality products thanks to their ability to search a through a huge combination of parameters to find the best match. For example, they can search through different combinations of materials and designs to find the perfect combination of both which could result in a stronger, lighter and overall, better final product. They can also be used to design computer algorithms, to schedule tasks, and to solve other optimization problems. Genetic algorithms are based on the process of evolution by natural selection which has been observed in nature. They essentially replicate the way in which life uses evolution to find solutions to real world problems. Surprisingly although genetic algorithms can be used to find solutions to incredibly complicated problems, they are themselves pretty simple to use and understand.

How they work

As we now know they're based on the process of natural selection, this means they take the fundamental properties of natural selection and apply them to whatever problem it is we're trying to solve.

The basic process for a genetic algorithm is:

1. Initialization - Create an initial population. This population is usually randomly generated and can be any desired size, from only a few individuals to thousands.
2. Evaluation - Each member of the population is then evaluated and we calculate a 'fitness' for that individual. The fitness value is calculated by how well it fits with our desired requirements. These requirements could be simple, 'faster algorithms are better', or more complex, 'stronger materials are better but they shouldn't be too heavy'.
3. Selection - We want to be constantly improving our populations overall fitness. Selection helps us to do this by discarding the bad designs and only keeping the best individuals in the population. There are a few different selection methods but the basic idea is the same, make it more likely that fitter individuals will be selected for our next generation.
4. Crossover - During crossover we create new individuals by combining aspects of our

selected individuals. We can think of this as mimicking how sex works in nature. The hope is that by combining certain traits from two or more individuals we will create an even 'fitter' offspring which will inherit the best traits from each of it's parents.

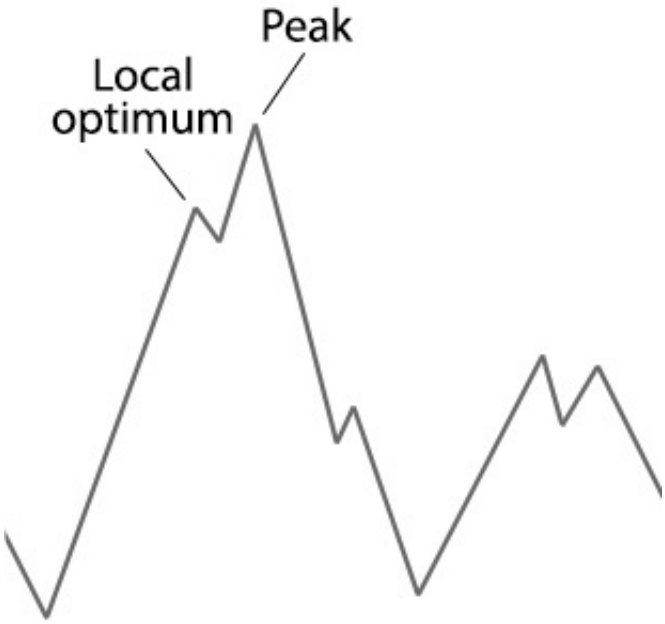
5. Mutation - We need to add a little bit randomness into our populations' genetics otherwise every combination of solutions we can create would be in our initial population. Mutation typically works by making very small changes at random to an individuals genome.
6. And repeat! - Now we have our next generation we can start again from step two until we reach a termination condition.

Termination

There are a few reasons why you would want to terminate your genetic algorithm from continuing it's search for a solution. The most likely reason is that your algorithm has found a solution which is good enough and meets a predefined minimum criteria. Offer reasons for terminating could be constraints such as time or money.

Limitations

Imagine you were told to wear a blindfold then you were placed at the bottom of a hill with the instruction to find your way to the peak. You're only option is to set off climbing the hill until you notice you're no longer ascending anymore. At this point you might declare you've found the peak, but how would you know? In this situation because of your blindfolded you couldn't see if you're actually at the peak or just at the peak of smaller section of the hill. We call this a local optimum. Below is an example of how this local optimum might look:



Unlike in our blindfolded hill climber, genetic algorithms can often escape from these local optimums if they are shallow enough. Although like our example we are often never able to guarantee that our genetic algorithm has found the global optimum solution to our problem. For

more complex problems it is usually an unreasonable exception to find a global optimum, the best we can do is hope for is a close approximation of the optimal solution.

Implementing a basic binary genetic algorithm in Java

These examples are build in Java. If you don't have Java installed and you want to follow along please head over to the Java downloads page, http://www.oracle.com/technetwork/java/javase/downloads/index.html

Let's take a look at the classes we're going to create for our GA:

- Population - Manages all individuals of a population
- Individual - Manages an individuals
- Algorithm - Manages our evolution algorithms such as crossover and mutation
- FitnessCalc - Allows us set a candidate solution and calculate an individual's fitness

Population.java

```
package simpleGa;

public class Population {

    Individual[] individuals;

    /*
     * Constructors
     */
    // Create a population
    public Population(int populationSize, boolean initialise) {
        individuals = new Individual[populationSize];
        // Initialise population
        if (initialise) {
            // Loop and create individuals
            for (int i = 0; i < size(); i++) {
                Individual newIndividual = new Individual();
                newIndividual.generateIndividual();
                saveIndividual(i, newIndividual);
            }
        }
    }

    /* Getters */
    public Individual getIndividual(int index) {
        return individuals[index];
    }

    public Individual getFittest() {
        Individual fittest = individuals[0];
        // Loop through individuals to find fittest
        for (int i = 0; i < size(); i++) {
            if (fittest.getFitness() <= getIndividual(i).getFitness()) {
                fittest = getIndividual(i);
            }
        }
        return fittest;
```

```
    }

    /* Public methods */
    // Get population size
    public int size() {
        return individuals.length;
    }

    // Save individual
    public void saveIndividual(int index, Individual indiv) {
        individuals[index] = indiv;
    }
}
```

Individual.java

```
package simpleGa;

public class Individual {

    static int defaultGeneLength = 64;
    private byte[] genes = new byte[defaultGeneLength];
    // Cache
    private int fitness = 0;

    // Create a random individual
    public void generateIndividual() {
        for (int i = 0; i < size(); i++) {
            byte gene = (byte) Math.round(Math.random());
            genes[i] = gene;
        }
    }

    /* Getters and setters */
    // Use this if you want to create individuals with different gene lengths
    public static void setDefaultGeneLength(int length) {
        defaultGeneLength = length;
    }

    public byte getGene(int index) {
        return genes[index];
    }

    public void setGene(int index, byte value) {
        genes[index] = value;
        fitness = 0;
    }

    /* Public methods */
    public int size() {
        return genes.length;
    }

    public int getFitness() {
        if (fitness == 0) {
            fitness = FitnessCalc.getFitness(this);
        }
        return fitness;
    }

    @Override
    public String toString() {
```

```
        String geneString = "";
        for (int i = 0; i < size(); i++) {
            geneString += getGene(i);
        }
        return geneString;
    }
}
```

Algorithm.java

```
package simpleGa;

public class Algorithm {
```

```
    // Evolve a population
    public static Population evolvePopulation(Population pop) {
        Population newPopulation = new Population(pop.size(), false);

        // Keep our best individual
        if (elitism) {
            newPopulation.saveIndividual(0, pop.getFittest());
        }

        // Crossover population
        int elitismOffset;
        if (elitism) {
            elitismOffset = 1;
        } else {
            elitismOffset = 0;
        }
        // Loop over the population size and create new individuals with
        // crossover
        for (int i = elitismOffset; i < pop.size(); i++) {
            Individual indiv1 = tournamentSelection(pop);
            Individual indiv2 = tournamentSelection(pop);
            Individual newIndiv = crossover(indiv1, indiv2);
            newPopulation.saveIndividual(i, newIndiv);
        }

        // Mutate population
        for (int i = elitismOffset; i < newPopulation.size(); i++) {
            mutate(newPopulation.getIndividual(i));
        }

        return newPopulation;
    }

    // Crossover individuals
    private static Individual crossover(Individual indiv1, Individual indiv2) {
        Individual newSol = new Individual();
        // Loop through genes
        for (int i = 0; i < indiv1.size(); i++) {
```

```
            // Crossover
            if (Math.random() <= uniformRate) {
                newSol.setGene(i, indiv1.getGene(i));
            } else {
                newSol.setGene(i, indiv2.getGene(i));
            }
        }
        return newSol;
    }

    // Mutate an individual
    private static void mutate(Individual indiv) {
        // Loop through genes
        for (int i = 0; i < indiv.size(); i++) {
            if (Math.random() <= mutationRate) {
                // Create random gene
                byte gene = (byte) Math.round(Math.random());
                indiv.setGene(i, gene);
            }
        }
    }

    // Select individuals for crossover
    private static Individual tournamentSelection(Population pop) {
        // Create a tournament population
        Population tournament = new Population(tournamentSize, false);
        // For each place in the tournament get a random individual
        for (int i = 0; i < tournamentSize; i++) {
            int randomId = (int) (Math.random() * pop.size());
            tournament.saveIndividual(i, pop.getIndividual(randomId));
        }
        // Get the fittest
        Individual fittest = tournament.getFittest();
        return fittest;
    }
}
```

FitnessCalc.java

```
package simpleGa;

public class FitnessCalc {

    static byte[] solution = new byte[64];

    /* Public methods */
    // Set a candidate solution as a byte array
    public static void setSolution(byte[] newSolution) {
        solution = newSolution;
    }

    // To make it easier we can use this method to set our candidate solution
    // with string of 0s and 1s
    static void setSolution(String newSolution) {
        solution = new byte[newSolution.length()];
        // Loop through each character of our string and save it in our byte
        // array
        for (int i = 0; i < newSolution.length(); i++) {
            String character = newSolution.substring(i, i + 1);
            if (character.contains("0") || character.contains("1")) {
                solution[i] = Byte.parseByte(character);
            } else {
```

```
                solution[i] = 0;
            }
        }
    }

    // Calculate inidividuals fittness by comparing it to our candidate solution
    static int getFitness(Individual individual) {
        int fitness = 0;
        // Loop through our individuals genes and compare them to our cadidates
        for (int i = 0; i < individual.size() && i < solution.length; i++) {
            if (individual.getGene(i) == solution[i]) {
                fitness++;
            }
        }
        return fitness;
    }

    // Get optimum fitness
    static int getMaxFitness() {
        int maxFitness = solution.length;
        return maxFitness;
    }
}
```

Now let's create our main class.

First we need to set a candidate solution (feel free to change this if you want to).

```
FitnessCalc.setSolution("1111000000000000000000000000000000000000000000000000000000001111");
```

Now we'll create our initial population, a population of 50 should be fine.

```
Population myPop = new Population(50,true);
```

Now we can evolve our population until we reach our optimum fitness

```
int generationCount = 0;
while(myPop.getFittest().getFitness() < FitnessCalc.getMaxFitness()){
  generationCount++;
  System.out.println("Generation: "+generationCount+" Fittest: "+myPop.getFittest().getFitness
  myPop = Algorithm.evolvePopulation(myPop);
}
System.out.println("Solution found!");
System.out.println("Generation: "+generationCount);
System.out.println("Genes:");
System.out.println(myPop.getFittest());
```

Here's the complete code for our main class:

GA.java

```
package simpleGa;
```

```
public class GA {

    public static void main(String[] args) {

        // Set a candidate solution
        FitnessCalc.setSolution("1111000000000000000000000000000000000000000000000000000011

        // Create an initial population
        Population myPop = new Population(50, true);

        // Evolve our population until we reach an optimum solution
        int generationCount = 0;
        while (myPop.getFittest().getFitness() < FitnessCalc.getMaxFitness()) {
            generationCount++;
            System.out.println("Generation: " + generationCount + " Fittest: " + myPop.getFitte
            myPop = Algorithm.evolvePopulation(myPop);
        }
        System.out.println("Solution found!");
        System.out.println("Generation: " + generationCount);
        System.out.println("Genes:");
        System.out.println(myPop.getFittest());

    }
}
```

If everything's right, you should get an output similar to the following:

```
Generation: 1 Fittest: 40
Generation: 2 Fittest: 43
Generation: 3 Fittest: 50
Generation: 4 Fittest: 50
Generation: 5 Fittest: 52
Generation: 6 Fittest: 59
Generation: 7 Fittest: 59
Generation: 8 Fittest: 61
Generation: 9 Fittest: 61
Generation: 10 Fittest: 61
Generation: 11 Fittest: 63
Generation: 12 Fittest: 63
Generation: 13 Fittest: 63
Generation: 14 Fittest: 63
Generation: 15 Fittest: 63
Solution found!
Generation: 15
Genes:
1111000000000000000000000000000000000000000000000000000001111
```

Remember you're output isn't going to be exactly the same as above because of the inherent characteristics of a genetic algorithm.

And there you have it, that's a very basic binary GA. The great thing about a binary GA is that it is easy to represent any problem, although it might not always be the best way of going about it.

Want to apply a genetic algorithm to a real search problem? Check out the following tutorial,

applying a genetic algorithm to the traveling salesman problem

Author

Hello, I'm Lee.
I'm a developer from the UK who loves technology and business. Here you'll find articles and tutorials about things that interest me. If you want to hire me or know more about me head over to my about me page

Social Links

Tags

genetic-algorithms artificial-intelligence java algorithm

Related Articles

Simulated Annealing for beginners
The Bionic Limb
Applying a genetic algorithm to the traveling salesman problem
Introduction to Artificial Neural Networks - Part 1
Autobots and Beyond

Comments

89 Comments The Project Spot 1 Login

Recommend 9 Share Sort by Best

Join the discussion…

Ganesh Sivalingam • 2 years ago

Hi I've converted your program to python, trying to keep as true to your original as possible. I wanted to make it available to everyone, do you want me to send it to you and you can post it (acknowledging me of course;)), or should I put it on my own blog and just put a link to your site for the full explanation? Please reply by email. Thanks

site for the full explanation? Please reply by email. Thanks

28 ▲ | ▼ • Reply • Share ›

aksiksi • 4 years ago

Just what I needed. I can't seem to follow the code, but I believe I'll get it all as I learn Java in the course I'm currently enrolled in (I'm a Python enthusiast).

Bookmarked.

12 ▲ | ▼ • Reply • Share ›

Jason • 2 years ago

nice explanation, Please let me know if there is any tutorial u did or that u know of to study NSGA-II i need a basic code to get started with my project. Thanks!!

11 ▲ | ▼ • Reply • Share ›

Danny G • 4 years ago

Thanks for the interesting article!
There may be a typo though (unused code) - In the FitnessCalc.setSolution(byte[] newSolution) method, you probably intended to set solution = newSolution and not the other way around.

4 ▲ | ▼ • Reply • Share ›

Lee Jacobson ↗ Danny G • 4 years ago

Thanks! I did get it the wrong way round, corrected it now.
It's unused but I thought it might be useful for anyone experimenting with the code.

1 ▲ | ▼ • Reply • Share ›

shekhar ↗ Lee Jacobson • 2 years ago

pls share more improved code..... or send by mail shekhariet@yahoo.co.in

1 ▲ | ▼ • Reply • Share ›

Guest • 4 years ago

You might want to post the output of running this.

3 ▲ | ▼ • Reply • Share ›

Lee Jacobson ↗ Guest • 4 years ago

Just added the output

▲ | ▼ • Reply • Share ›

JohnDavids • 10 months ago

Thanks for the nice example, I created a variation of this code in PHP here:
http://www.abrandao.com/2015/0...

It uses strings instead of binary 0/1 but otherwise is functionally the same.

1 ▲ | ▼ • Reply • Share ›

1 • Reply • Share ›

Arthur Vinícius • 2 years ago

I implemented it in Python, check it out : https://gist.github.com/arthur...

1 • Reply • Share ›

Patrick Murphy • 3 years ago

Hey thanks for sharing this. I was able to adapt it to what i needed to use it for very easily. It was easy to read and follow your code i didn't even have to read a single comment. Good post and great demonstration. I am stoked i can now use GA's in my side projects now! I remember looking at them before and just didn't feel like sitting down to figure it out. Now that i got some years under my belt is alot easier to understand now.

1 • Reply • Share ›

Lee Jacobson Mod ➦ Patrick Murphy • 3 years ago

That's great to hear!
Good luck your side projects! =)

• Reply • Share ›

vishallthape • 3 years ago

This is fantastic helps understand GA , cool :)

1 • Reply • Share ›

Kazimuth • 4 years ago

In
private static Individual tournamentSelection(Population pop),
int randomId = (int) (Math.random() * pop.size());
is rendering improperly in chrome, after Math.random() the line is grayed out. Just thought you should know.
Great article though, explains it very well

1 • Reply • Share ›

Chinku ➦ Kazimuth • 3 years 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?

1 • Reply • Share ›

Angel • 3 days ago

Can any1 convert this to be able to work on Arduino? Thanks

• Reply • Share ›

Angel • 5 days ago

Can any1 convert this and make it to work for Arduino uno tnx :D

• Reply • Share ›

juanmf • 2 months ago

hi, Interesting!

I made a generic implementation in java, so, ideally, you just have to implement the problem structure and the Fitness function, and it can start searching.

I'd like your opinion :D https://github.com/juanmf/ga

• Reply • Share ›

Sogomn • 3 months ago

This is by far the worst OOP that I've seen in a long time. Almost every class has methods it should not have.

• Reply • Share ›

Shane Abram Mendez • 4 months ago

The individual and population classes make the code unnecessarily complex. The only thing the individual "knows" are the genes, and its fitness. However it doesnt need to know its fitness, therefore it can be represented as just its genes. Population only knows a list of individuals, therefore it can be an array of individuals. The fitness of each individual can be mapped in a separate array in order of the population array, and the fittest individual can be found when calculating the fitness of each individual.

• Reply • Share ›

Jahaan • 4 months ago

Hey fantastic tutorial i have a question my code reaches fitness 62 and then gets stuck and keeps going for 50000+ generations, is this normal , a local maxima ? or is there a bug in my code

• Reply • Share ›

Kuuboore Marcellinus • 4 months ago

Pls can you help me implement this in Matlab?

• Reply • Share ›

havoknation • 6 months ago

Can anyone pls tell me how to initialise a big data set with about 500 rows and 4 attributes with 1 designated attribute on this JAVA code?

• Reply • Share ›

Rashmi Naik • 7 months ago

I'm Working on a research project called "Dynamic price optimization" using genetic algorithm to optimize product price's in an e-commerce store. I need help in defining a Fitness function to

optimize the products price. Can any one help me on this.!!

• Reply • Share ›

Bil • 7 months ago

Hi Lee , I have converted the code to C# , but compare to your output my output found the solution in around 50000 generation , Is that normal

• Reply • Share ›

Lee Jacobson Mod ➦ Bil • 7 months ago

No, it shouldn't take that long. If your parameters are correct you probably have a bug in your code somewhere.
Genetic algorithms can be quite a pain to debug because of the randomness, but you'll probably want to check your fitness function is returning the right value first as that will give you a predictable answer at least. Then move on to testing the mutation, crossover and selection functions.

• Reply • Share ›

osman • 10 months ago

Please, can you write CMA-ES for beginners in java?
thanks

• Reply • Share ›

negar • a year ago

hi, is it possible to help me to write this code in c#?

• Reply • Share ›

Pam • a year ago

Thanks for the article.. really the code explains better than the theory. If u can pls do code for PSO too.

• Reply • Share ›

Edoardo Rosa • a year ago

how can I implement an algorithm based on Ant System for this type of problem?

• Reply • Share ›

Trần Tiến Anh • a year ago

thank verry much!! Hey thanks for sharing this. I was able to adapt it to what i needed to use it for very easily. Photowonder It was easy to read and follow your code i didn't even have to read a single comment. Good post and great demonstration.just wanted to ask what kind of crossover are you using. It seems to be multiple point crossover i.e. where the random number is less than 0.5. Tai photowonder Normally for one point crossover, you randomly select the index if the individuals are suppose to crossover based on crossover probability.

• Reply • Share ›

hemant • 2 years ago

Dear sir,

I wanted to know How we can apply this algorithm for Intrusion Detection Problem to find anomaly.

• Reply • Share ›

azzouztr • 2 years ago

Please, i want to know how I can customize the genetic algorithms in the selection of web services based on QoS

• Reply • Share ›

Psypher • 2 years ago

Just wanted to ask what kind of crossover are you using. It seems to be multiple point crossover i.e. where the random number is less than 0.5. Normally for one point crossover, you randomly select the index if the individuals are suppose to crossover based on crossover probability.

• Reply • Share ›

Abhinav • 2 years ago

Awesome Tut Man!!
Helped me a lot in getting started on GA.... Thanks a lot! :)

• Reply • Share ›

Lee Jacobson Mod ➚ Abhinav • 2 years ago

That's great to hear =)

• Reply • Share ›

Ali • 2 years ago

Hi, Lee! its extremely helpful. I run this project a null pointer exception occur. I just comment as

\\if(initilise)

{
}
At Population Constructor, It works fine. What is the reality about it? I can't able to understand. Thanks!

• Reply • Share ›

Frans • 2 years ago

Hello sir, i'm intrigued to use this code that you made.

Just one question i dont understand is this..

when you try to calculate fitness function why did you use this decision statement? i mean why you didn't calculate again? Is it just for reducing computation time or something?
Thanks!
if (fitness == 0) {

fitness = FitnessCalc.getFitness(this);

}

Trung • 2 years ago

Thanks for this! I am beginer in GA - that's useful for me

Buddy James • 2 years ago

What a great tutorial! The code is great and the subject is conveyed in a clear and concise manner. Keep up the great work.

RahulR • 2 years ago

Hello! thanks for sharing this. I need a little help from you. I am trying to run your code in netbeans but am getting this error:

Exception in thread "main" java.lang.RuntimeException: Uncompilable source code - Erroneous sym type: simplega.FitnessCalc.setSolution

at simplega.GA.main(GA.java:18)

Thanks in advance.

Lee Jacobson Mod ➦ RahulR • 2 years ago

This could help? http://stackoverflow.com/quest...

If you copy and pasted the code fully it should run without any problems.

akari • 2 years ago

can u give me code about scheduling problem with GA in java please?

guest • 3 years ago

Hello i don't understand why you used elitism since it's final which means that your program is always gonna preserve the first individual from getting mutated or crossed over!

Lee Jacobson Mod ➚ guest • 3 years ago

Without the elitism the best individual would need to be run through the crossover and mutation function before being added to the next population. If that happened there would be a strong possibility that it would no longer be the best solution.

1 ^ | v • Reply • Share ›

zakaria • 3 years ago

Hello ! can i just ask you why you added the class fitnessCalc as you could have only used the class individual to calculate the fitness.
Actually i am working on the same algorithm to optimize numerical functions (maximization or minimization) but i've had a problem with the fitness should i use an independant class for that ??
please tell me what you suggest i'am a beginner in java and can't sort it out!

^ | v • Reply • Share ›

Lee Jacobson Mod ➚ zakaria • 3 years ago

The FitnessCalc class holds the target solution. It could have been added in the Individual class but I felt it was cleaner to separate the logic out.

Without looking at your code it's hard for me to see the problem. You're welcome to email me at, lee@cwpstudios.com and I'll take a look.

1 ^ | v • Reply • Share ›

zakaria ➚ Lee Jacobson • 3 years ago

hi lee please take a look at the program i sent you on your e-mail lee@cwpstudios.com and tell me what i've done wrong

^ | v • Reply • Share ›

Lee Jacobson Mod ➚ zakaria • 3 years ago

It doesn't look like your crossover is being implemented correctly. From what I can tell individual 1 is always being used as a parent during crossover.

I'm also not sure if your roulette wheel selection has been implemented correctly but it's kinda hard to tell with the variable names, sorry! It might be easier to implement tournament selection to start with then move over to roulette wheel selection when it's working.

^ | v • Reply • Share ›

zak ➚ Lee Jacobson • 2 years ago

I think you didn't see that there are three functions for the roulette wheel

the first one chooses the parents that will endure crossing over

the second chooses the point where crossing over will occur

the third actually exchanges values btween them.

anyway i've found other things that don't work ! so thank you for your ideas and help.

• Reply • Share ›

Load more comments

ALSO ON **THE PROJECT SPOT**

WHAT'S THIS?

Solving the Traveling Salesman Problem using Google Maps and Genetic

12 comments • 8 months ago

Seno Widya Manggala — hey i'm seno from indonesia.. i'm trouble with api's the webpage says "This page was unable to

Autobots and Beyond

3 comments • 2 years ago

Jonathan Peterson — Hello. would you like to learn a secret of our universe? facebook JonathanPeterson1466. Go to IBMresearch

Simulated Annealing algorithm for beginners

42 comments • 3 years ago

anum eman — i have no words to say you

The day we fought back with a list of names

3 comments • 2 years ago

michaelamie — I've seen the argument

Thanks for visiting!

Lets keep in touch,
Send me an email or follow me on Google+