PROJECT REPORT

EVOLUTIONARY ALGORITHM FOR DATABASE GRID OPTIMIZATION

Submitted By:
Suraj Singh (2012JE0817)
Ayush Agrawal (2012JE0632)
B.Tech, Final Year, CSE
Indian School of Mines, Dhanbad

CERTIFICATE

This is to certify that Ayush Agrawal, Suraj Singh, Department of Computer Science And Engineering, Indian School of Mines, Dhanbad have successfully completed their assignment. They have successfully completed all tasks assigned to them along with successful demonstration of the working model.

Annavarapu Chandra Sekhara Rao
Assistant Professor
Department of CSE
Indian School of Mines

Acknowledgement

We would like to express our deep sense of gratitude and respect to our guide, A.C.S Rao, Assistant Professor, Department of Computer Science and Engineering, Indian School of Mines, Dhanbad. We are very grateful for the generosity, expertise and guidance we have received from him while working on the project.
Expressing our sincere thanks to our guide, A.C.S Rao, we would like to convey our sincere regards and gratitude for encouraging us to undertake this assignment and to work on it at time and again, it would not have been possible without him, we have all but enjoyed working on this project under his wings and we shall keep so for the times to come.

Table of Contents

Introduction

Evolution strategies use natural problem-dependent representations, and primarily mutation and selection, as search operators. In common with evolutionary algorithms, the operators are applied in a loop. An iteration of the loop is called a generation. The sequence of generations is continued until a termination criterion is met.

As far as real-valued search spaces are concerned, mutation is normally performed by adding a normally distributed random value to each vector component. The step size or mutation strength (i.e. the standard deviation of the normal distribution) is often governed by self-adaptation. Individual step sizes for each coordinate or correlations between coordinates are either governed by self-adaptation or by covariance matrix adaptation.

Evolutionary Programming is a Global Optimization algorithm and is an instance of an Evolutionary Algorithm from the field of Evolutionary Computation. The approach is a sibling of other Evolutionary Algorithms such as the Genetic Algorithm, and Learning Classifier Systems. It is sometimes confused with Genetic Programming given the similarity in name, and more recently it shows a strong functional similarity to Evolution Strategies.

The objective of the Evolutionary Programming algorithm is to maximize the suitability of a collection of candidate solutions in the context of an objective function from the domain. This objective is pursued by using an adaptive model with surrogates for the processes of evolution, specifically hereditary (reproduction with variation) under competition. The representation used for candidate solutions is directly assessable by a cost or objective function from the domain.

Here our goal was to achieve Database Grid Optimization using Evolutionary Algorithms. So to complete our objective, we have the combined the above mentioned techniques, Evolutionary Strategies and Evolutionary Programming.

Overview

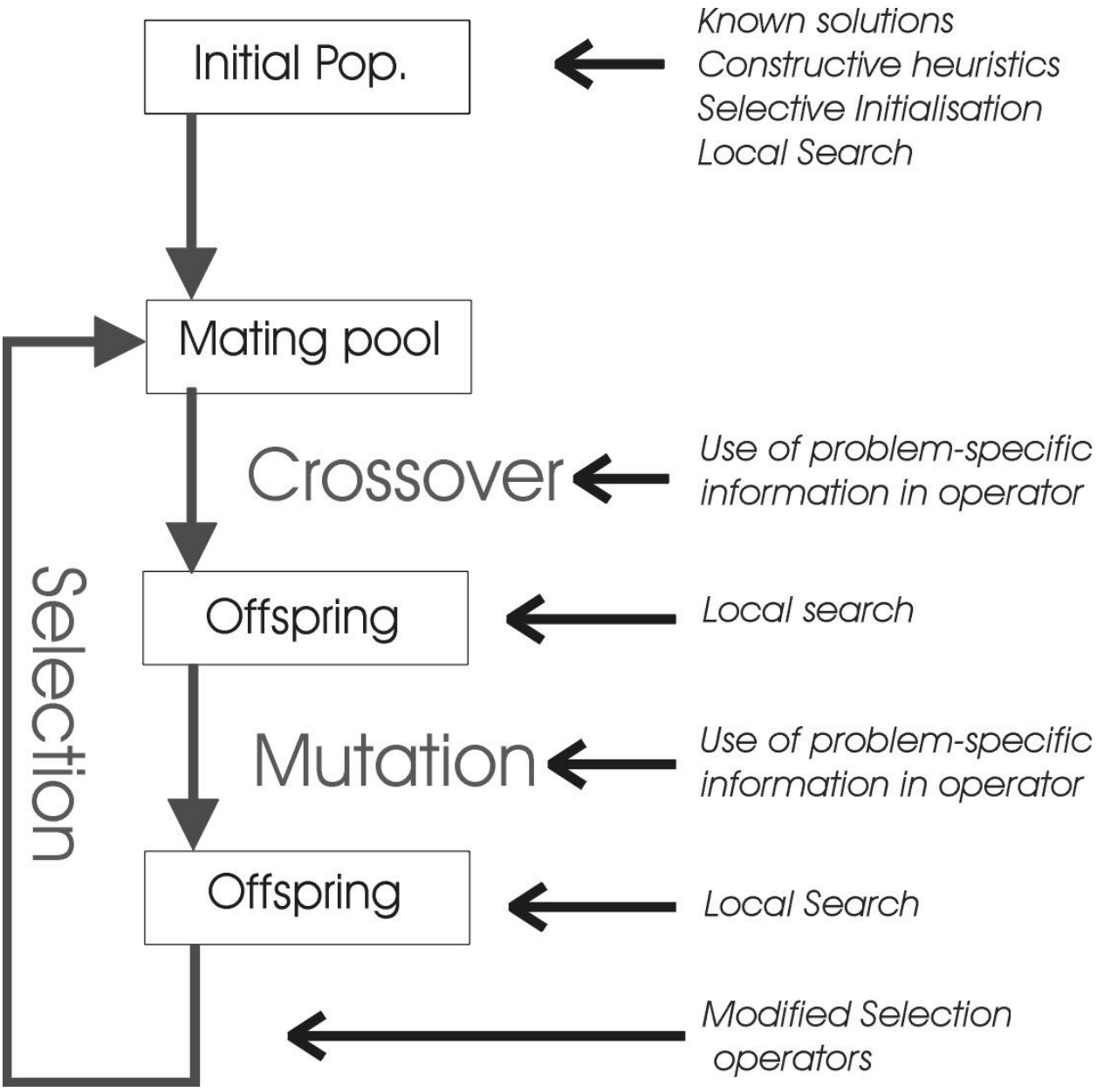

Fig : Flow Chart

Algorithm

1. Take the **initial permutation** input from the user.
2. Take the number of **iterations** from the user.
3. Initialize **ParentA = initial permutation**.
4. Initialize i = 0
5. While (i < iterations)
    5.1. Randomly generate **ParentB.**
    5.2. Do crossover and generate **ChildA** and **ChildB.**
    5.3. Calculate cost of ParentA, ParentB, ChildA, ChildB.
    5.4. Find **minCost** among them along with its corresponding permutation.
    5.5. Do mutation on last obtained permutation and find **mutCost.**
    5.6. If (mutCost < minCost)
        5.6.1. minCost = mutCost
        5.6.2. make mutated permutation as ParentA.
    5.7. Update i = i + 1
6. Print the final permutation along with its minCost.

Output

```
Enter the five distinct numbers (range 1 to 2000)  :  962 247 1430 156
810
Enter maximum number of iterations : 5
The Initial Permutation was 962 247 1430 156 810
Iteration Number :: 1

Parent A : 962 247 1430 156 810
Parent A Cost : 6403
Parent B : 247 1430 810 962 156
Parent B Cost : 6264
Child A : 962 247 1430 810 156
Cost of Child A : 4857
Child B : 247 1430 810 962 156
Cost of Child B : 6264

After cross- over the premutation we got : 962 247 1430 810 156
After Mutation the permutation : 962 247 810 1430 156
Mutated cost 6249
Final, survival chromosome (permutation) after 1 iteration : 962 247 1430 810 15
6
Final Cost ::4857
********************************************************************************


Iteration Number :: 2

Parent A : 962 247 1430 810 156
Parent A Cost : 4857
Parent B : 156 962 247 1430 810
Parent B Cost : 5092
Child A : 962 247 1430 156 810
Cost of Child A : 6403
Child B : 156 962 247 1430 810
Cost of Child B : 5092

After cross- over the premutation we got : 962 247 1430 810 156
After Mutation the permutation : 962 1430 247 810 156
Mutated cost 4140
Final, survival chromosome (permutation) after 2 iteration : 962 1430 247 810 15
6
Final Cost ::4140
********************************************************************************
```

Fig : Output 1

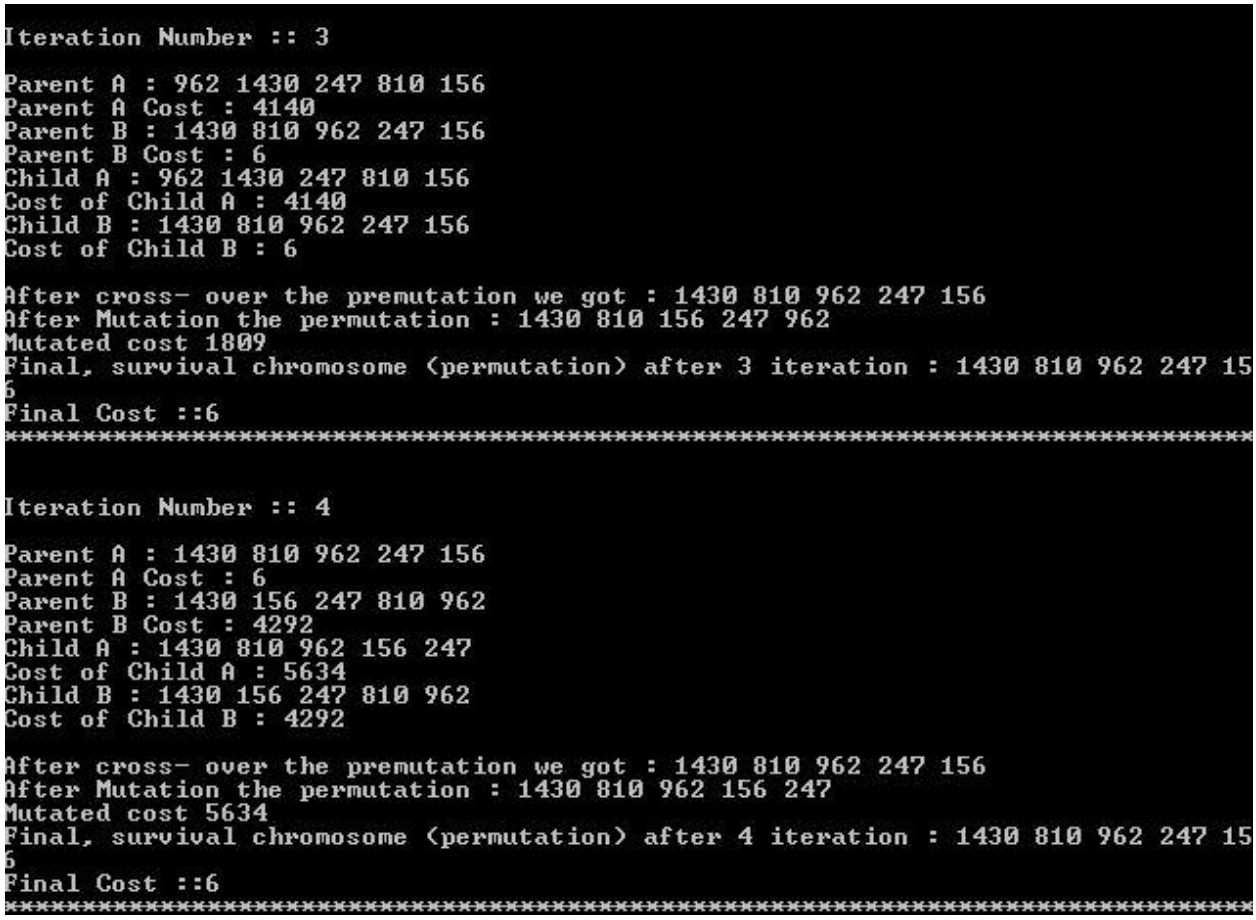

Fig : Output 2

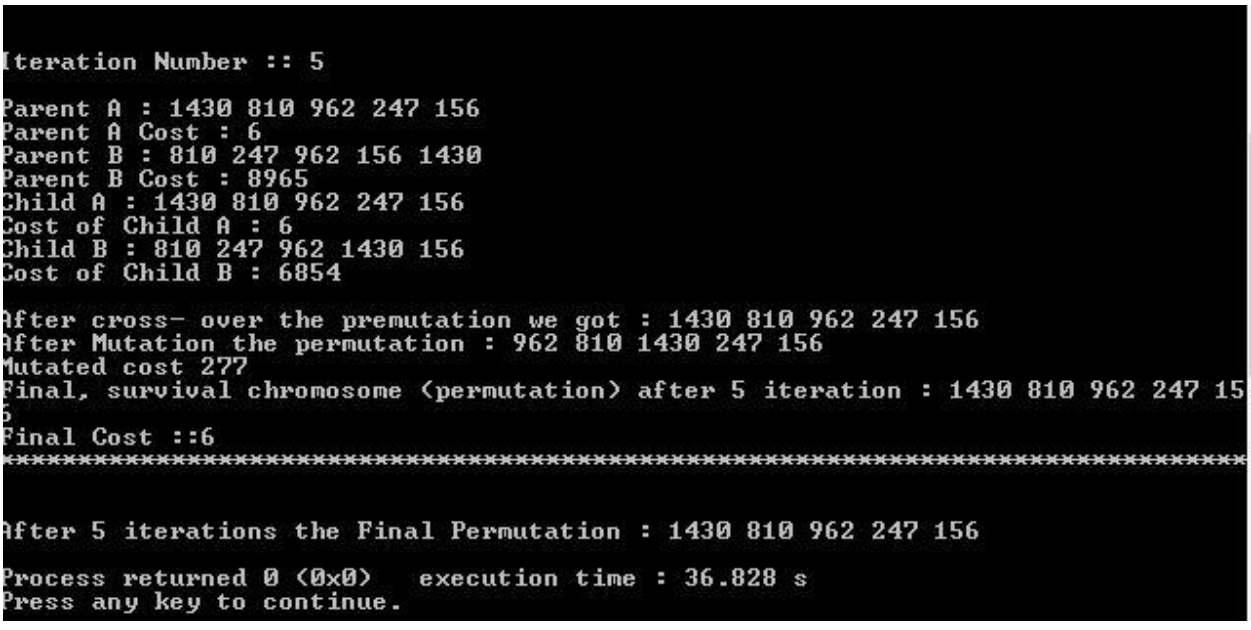

Fig : Output 3