Solving the Building Scale Problem with GA

Mh. Samadi

Abstract— **In this essay a solution is suggested for the building scale problem using the GA algorithm. The solution is ran with different parameters and the results are shared in this essay.**

I. INTRODUCTION

The Quadratic Assignment Problem (QAP) is a fundamental combinatorial optimization problem in the branch of optimization and operations research. The problem statement resembles that of the assignment problem, except that the cost function is expressed in terms of quadratic inequalities, hence the name.
There are 25 places in a certain distance from each other and there are 25 facilities with a certain flow to each other. The problem demands an assignment of these facilities to each of these places in a way that minimizes the total sum of $flow \times distance$. To solve this problem this essay suggests a type of GA with focus on the crossover operation and a new type of selection called steady-state selection.

II. IMPLEMENTATION

A QAP problem is represented by 3 elements. One is the size of the problem. Other is the distance matrix. The last one is the flow matrix. This representation of the problem is saved in a .dat file. The file is read and fed to the GA algorithm to solve for a number of iterations. The important details of the used GA function is mentioned in their own subsections. The parts that are not mentioned are standard implementations. For example as for the the selection method, roulette wheel algorithm is used.

A. Crossover

We use two point crossovers for implementation of GA. For making child 1 and child 2, two points are randomly selected from 1 to 5 and between these points is hold fixed. Then, child 1 is created according to orders of parent 2 and fixed components of parent 1, as well as child 2 is produced according to order of parent 1 and fixed components of parent 2. In this way we'll be able to do crossovers without having to worry about duplicate assignments of the facilities.

B. Steady-State Selection

Often GAs advance by using generations. In steady-state GA, each time 2 offsprings are generated and they'll replace the individual with the worst fitness. So in each iteration of the algorithm size of the crowd is steady and there's only a chance of replacing two individuals each time.
The pseudocode for the steady-state selection GA is:

```
1. Generate initial population of size N.
2. Evaluate each solutions' fitness/goodness.
3. Select 2 solutions as parents without repetition.
4. Do Crossover, Mutation and Inversion and create 2 offsprings.
5. If offspring is duplicated, then go to step 3.
6. If Not, then evaluate the fitness of offspring.
7. If offspring are better than the worst solutions then replace the worst individuals with the offspring such that population size 8. remains constant.
9.Check for convergence criteria. If convergence criteria are met, then terminate the program else continue with step 3.
```

III. RESULTS

First the normal GA is ran in an attempt to solve the problem. After taking an hour for 10k iterations of the algorithm, best answer has a fitness of 3946. Here are the results of out steady-state version of the algorithm ran with different parameters:

population	best fitness
10	4036
25	4012
50	3898
100	3862
200	3808
500	4204
1000	4446
2000	4592
3000	4674
5000	4774

population size variation - pc = 0.8, pm = 0.2, 10k iterations
(the algorithm is ran once per population size hence the variations)

An ongoing growth in the fitness value (which is not ideal) can be observed in the data as the initial population size grows. If that's caused by the fact that each iteration only 2 offsprings are generated so for a bigger population the algorithm should be ran for more iterations for the steady-state selection to cover the whole population and cause growth, then if the population size is 5000 and the algorithm is ran for 250k iterations, it should yield better results than the time it was ran with initial population size of 200. Which is true. The algorithm yields 3802 as the final solutions fitness ([13 19 1 15 22 10 7 6 21 20 23 3 16 9 8 4 24 14 25 11 12 17 18 2 5]).
The next criteria is the probability of crossover:

population	best fitness
0.9	3794
0.8	3808
0.2	4538

pc variation - pop size = 200, pm = 0.2, 10k iterations

0.8 is a already a high probability. If pc is reduced the results get worse. If it's increased the algorithm yields the best result so far which is [13 4 21 24 12 23 7 14 3 17 19 10 6 16 18 1 9 8 25 11 22 15 20 2 5] with a fitness of 3794.
Next is the probability of mutation:

population	best fitness
0.1	3960
0.2	3808
0.5	3892

pm variation - pop size = 200, pc = 0.8, 10k iterations

IV. Conclusion

In some problems moving with precision matters. In such problems it's better to use a steady-state selection instead of generating a new generation which is messy and careless about the getting worse than the worst already existing individual. By using such an algorithm and fine-tuning the parameters a very close results to the best answer can be achieved within seconds.