COMP2051.AIM Coursework Report

Memetic Algorithm for MKP

Self-report

School of Computer Science
University of Nottingham Ningbo China

Group Member

Jingyu Ma - 20031510

Supervisor

Ruibin Bai

April 2020

1 Overview

1.1 Pseudo-code

```
Initilze_Population(50);
While(iter < MAX_NUM_OF_GEN && time_spent < MAX_TIME){

 Mtp=tournament(malloc_Create_mating_pool());  //parents are selected from here, tournament select 4 people from population randomly ,choose the best one ,put it into the mating pool, iteration++,in all create a mating pool of 50

Solution sons[50];
while(count<50){
parents=random_select_two(MTP);
sons[count]=crossover(parents);   //50% 50% chance choose from mom or dad, a new DNA
mutate(sons[count]);
repair(sons[count]);
local_search_first_descent(sons[count]);
if(objective same with current population or sons){
 continue; //back to the start and regenerate a son
}
count++;
}
Replace(pop[50],sons[50]);  //replace all population with children
Update_solution();//loop through new population and find the best solution, update it
 Free();
}
```

In case the pseudo code is not clear here is the word description, first initialize population and then went into the loop, in the loop first use tournament to select more fitting individuals in random 4 people and do this 50 times to create a mating pool. And then choose two random parent from mating pool to produce offspring(UX),mutate them, check if you have to use feasibility repair, local search to get optimal data, do this 50 times. If the same solution has already been generated, regenerate it. Then replace worst population with best children. After the loop, search through all the population and update the best result.

2 Parameter tuning

2.1 Final Parameter

The final adoption is that I use the crossover rate as 0.5(use rand_int(0,1)so set as 1), and the child has the same probability to take a gene from its father or mother. This would result in a fair chance for every gene to exchange position, and this increase more chance than the original 0.8.There is only one children after production. The mutation rate is 0.001 as indicating there should be 1 gene in every 1000 genes should mutate.
Tournament size is 4 according to the paper and the population size is 50 not too large nor too small, max number of generation is 100000.After this the loop will end Actually this is a pretty random tuning, because the process uses so many random events, there is no guarantee that changing one parameter will surely improve the result. But if you put the max_time to a higher number the better chance you would get a better result.
I have tried to change the mutation rate , compare 0.001 to a dynamic ratio of 0.001*n in short period of time， the result was pretty random too. So either way I would prefer a fixed 0.001 in case of bugs and a probability larger than 1.

3 Result Compare

3.1 Gap%

Make sure set the time to 300 , so that you can get a better result
Here we take three problem 1,5,8 as example

Mknapcb1 (30 results) <=1% in 25th , all the rest are 0%,no new best result
0% 0% 0% 0% 0% 0% 0% 0% 0% 0% 0% 0% 0% 0% 0% 0% 0% 0% 0% 0% 0% 0% 0% 0% 0% 0% 0.019% 0% 0% 0% 0% 0% 0%

Mknapcb5(30 results)it is the same with best, all are a little less than the best result, no new best result
0.2% 0.1% 0.2% 0.2% 0.2% 0.3% 0.1% 0.3% 0.1% 0% 0.1% 0.1% 0.1% 0.1% 0.1% 0.1% 0.1% 0.1% 0.1% 0.1% 0.2% 0.1% 0.2% 0.1% 0.2% 0% 0.1% 0.1% 0.1% 0.1% 0.1%

Mknapcb8(30 results) 4th and 10th result exceed 1% others are less then 1%
0.5% 0.5% 0.6% 1.2% 0.4% 0.5% 0.7% 0.7% 0.9% 1.4% 0.2% 0.4% 0.3% 0.3% 0.3% 0.5% 0.2% 0.2% 0.5% 0.2% 0.1% 0.1% 0.2% 0.3% 0.4% 0.1% 0.1% 0.1% 0.1% 0.1% 0.1%

Maybe if the time set to more than 300, every solution would be 0%, and maybe new best result would be found, from the previous result we can see that most result were far from best solution 0-1300 compare to the best solution the gap do exist but it is quite small. Longer time iteration should be chosen to testify the assumption.
And indeed after I test with 600 sec, the result was improved, some better results were found , but it still preserve hardship to reach new best result.

By assumption the rest of the file result should not vary very greatly from the upper three examples. Here are some results from other problem files.

MKnapcb2(30 results) Found new best at 14th 20th 22th
0.08% 0.006% 0.1% 0.03% 0.01% 0.03% 0.02% 0.03% 0.02% 0.03% 0.005% 0.0006% 0.2% -0.01% 0.02% 0.0005% 0.01% 0.05% 0.02% -0.004% 0.008% -0.002% 0.01% 0.01% 0% 0% 0.1% 0% 0.02% 0%

MKnapcb3(30 results) All under 1%
0.01% 0% 0.02% 0.01% 0.02% 0.01% 0.02% 0.01% 0.01% 0.02% 0.01% 0.02% 0.01% 0.01% 0.02% 0.01% 0% 0.02% 0.03% 0.05% 0.01% 0.04% 0.01% 0.05% 0.02% 0.01% 0.04% 0.06% 0.04% 0.01%

MKnapcb4(30 results)All under 1%
0.03% 0.2% 0% 0.03% 0% 0% 0.3% 0.1% 0% 0% 0.016% 0% 0.1% 0.4% 0.003% 0.1% 0.3% 0.02% 0.05% 0% 0% 0.02% 0.1% 0.2% 0% 0% 0.15% 0.1% 0% 0%

4 Reflection

4.1 Strength

1. There is no need for a greedy solution, when searching, it covers more solution than traditional heuristic algorithm
2. Can evaluate several solutions at the same time , reduce the chance to get stuck in the local optimum
3. Reduced randomness, according to survival of the fitness, individual with higher fitness got the chance to reproduce and higher the chance to find the optimal solution

4.2 Weakness

1. It is possible Genetic algorithms are prone to premature convergence(meet local optimal)
2. The efficiency of genetic algorithm is usually lower than other traditional optimization methods
3. Need a repair operator, add more burden to computation complexity

This may vary from individual code and modification