Evolutionary Computation

Assignment 2: TTP
Exercise 1

Zhuoying Li(a1675725),
Jingwen Wei(a1671836),
Puzhi Yao(a1205593),
XueYang Wang(a1690260)

May 7, 2017

1 Introduction

In Exercise 1, we are required to design an algorithm to solve a problem that any item can be picked up in any city without overloading the capacity of the knapsack. Due to the item sets and TSP tour is unchanged, the target problem can be considered as simple 0/1 Knapsack Problem(KP). The two criteria are:

- Outperform the two referenced algorithms (RLS and (1+1)EA),
- 10-minute limitation for each optimization.

In the rest of this paper we will introduce the implemented operators for Exercise 1 and corresponding results.

2 Operators

In this section we will introduce the operators we implemented for finding an optimal solution for Exercise 1. At the end of this section, we will define the algorithm designed to solve this problem.

2.1 Packing Plan to Item & Item to Packing Plan

$PackingPlan2Item$ and $Item2PackingPlan$ are designed to convert the index of packing plan into the index of items and vice versa. Although the sizes of these two components are the same, we found that their indexes are not matched. Therefore, we need these two operators to link the indexes of packing plan and items.

2.2 Penalty function

According to TTP, picking an item will increase the total traveling time due to the weight of the item and the decreasing velocity, and the total amount of rent will increase. In this case, if we ignore the negative effect of picking an item, it may lead us away from the optimal solution because of the increasing travel cost. Therefore, we introduce a penalty function into our algorithm, which is to calculate a corresponding profit (NOT modifying the value in *instance*) of an item when we are deciding the packing plan. Comparison results between using Penalty function and not are shown as in Table 1.

Here are the steps to calculate a corresponding profit:

1. Calculate the distance of the rest tour according to the location of an item,
2. Calculate the velocity if taking an item,
3. Calculate the increase travel time based on the distance of the rest tour and the changed velocity,
4. Calculate the increase travel cost according to the rent and the increase travel time,
5. Calculate the corresponding profit which is equal to the item profit minus the increase travel cost

Instances	OB-ComplexHeuristic-1	OB-ComplexHeuristic-1-without-penalty	OB-Ref-EA	OB-ReferenceValue
a280-n279-bounded-strongly-corr-01.ttp	15049.09418	14397.8746	15785.30014	16099
a280-n1395-uncorr-similar-weights-05.ttp	104363.0173	104365.7309	104365.7309	104366
a280-n2790-uncorr-10.ttp	411714.7896	411714.7896	411714.7896	411715
fnl4461-n4460-bounded-strongly-corr-01.ttp	231330.6795	206745.3473	230204.5192	235856
fnl4461-n22300-uncorr-similar-weights-05.ttp	1478485.646	1471832.317	1478911.333	1478962
fnl4461-n44600-uncorr-10.ttp	6260475.53	5815369.802	5898476.872	6256200
pla33810-n33809-bounded-strongly-corr-01.ttp	1694414.697	1226807.085	1577971.038	1639008
pla33810-n169045-uncorr-similar-weights-05.ttp	1.32E+07	-7927897.44	-5.89E+06	1
pla33810-n338090-uncorr-10.ttp	4.03E+07	2.58E+07	-4.89E+07	5.81E+05

Table 1: Penalty Function Test Results

2.3 Random Local Search

The Random Local Search operator is adopted from the in the given codes. We apply this referenced algorithm as part of our purposed algorithm. The rationale of using it will be discussed in the next section.

2.4 Complex Heuristic 1

To solve the problem in Exercise 1, we designed Complex Heuristic 1 which integrates the above operators. The main requirement of Exercise 1 is to outperform the two referenced algorithms (RLS and (1+1)EA). Our Complex Heuristic 1 is shown as in Algorithm-1.

Algorithm 1: ComplexHeuristic-1

```
compute penalty score for each of the items I_m ∈ M
sort the items of M in descending order based on the penalty score
set current packing plan P = ∅ and W' = 0
set half-time = Max-Time-Limit * 0.5
while size of P < M and loop-Time < half-time do
    if W'+ weight of I_m <= instance's capacity and penalty value of I_m > 0 then
        W'+ = weight of I_m add I_m to packing plan P
    end
end
using P as new packing plan and perform Bit-flip Random Local Search until half-time is up
return results;
```

Before designing our algorithm, we first tested the two referenced algorithms with the given instances. We found that two algorithms can give acceptable solutions in small instances but not in large instances due to the time limit. Considering the theory of local search, it takes much more steps to approach an optimal solution in large instances. To solve this problem, the main idea of our algorithm is to provide a better "starting point" for the local search method to accelerate the process of "approaching". In this case, we employ a greedy heuristic to get a better "starting point". By using the greedy heuristic we can get a better packing plan as an initial solution for

local search.

For the Greedy heuristic, we employ a value-per-unit strategy, which is the ratio of profit and weight, because the weight of an item is also an important factor when calculating the object value. Due to the negative effect of taking an item we discussed above, we use the corresponding profits resulted from the $penaltyfunction$ for the value-per-unit strategy.

Once an initial solution is ready, RLS can be applied to further improve this solution. According to the performance of RLS and (1+1)EA, RLS has a great benefit on computation time while the solution quality is very close to (1+1)EA. Considering the computation work load in Exercise 2 and 3, we finally choose RLS for our local search strategy.

3 Results

Instances	OB-ComplexHeuristic-1	OB-Ref-EA	OB-Ref-RLS	OB-Reference	Time-ComplexHeuristic-1	Time-Ref-EA	Time-Ref-RLS
a280-n279-bounded-strongly-corr-01.ttp	15049.09418	15785.30014	14453.45846	16099	427	974	163
a280-n1395-uncorr-similar-weights-05.ttp	104363.0173	104365.7309	104363.0173	104366	445	3738	461
a280-n2790-uncorr-10.ttp	411714.7896	411714.7896	411714.7896	411715	1363	9680	1325
fnl4461-n4460-bounded-strongly-corr-01.ttp	231330.6795	230204.5192	198180.2353	235856	5957	95829	4928
fnl4461-n22300-uncorr-similar-weights-05.ttp	1478485.646	1478911.333	1478571.219	1478962	59664	543706	73252
fnl4461-n44600-uncorr-10.ttp	6260475.53	5898476.872	6259319.462	6256200	327900	599816	353148
pla33810-n33809-bounded-strongly-corr-01.ttp	1694414.697	1577971.038	1617308.752	1639008	265374	599810	292302
pla33810-n169045-uncorr-similar-weights-05.ttp	1.32E+07	-5.89E+06	4061207.553	1	303618	599844	599829
pla33810-n338090-uncorr-10.ttp	4.03E+07	-4.89E+07	-3.09E+07	5.81E+05	304966	599861	599855

Table 2: Test Results

Our test results are shown as in Table 2. The parameters used for these results are 10000 Duration without Improvement and 600000 ms maxRuntime. In order to intuitively illustrate the solution quality comparison among the three algorithms and the referenced objective values, we apply a normalized approach: Pick the highest objective value of each instance as the standard value (100%), and the others are divided by this standard into percentages to show their qualities. For those whose objective values are negative, we set them as 0. As can be seen from Figure 1, our purposed algorithm, Complex Heuristic 1, outperforms the referenced RLS and (1+1)EA, especially in large-scale instances. This algorithm even has better objective values in large-scale instances compare to the referenced value from the problem website.

In order to further investigate the performance of Complex Heuristic 1, we also conduct a test using parameters of 600000 ms maxRuntime and 900000000 Duration without Improvement. This test is aimed investigate whether the two referenced algorithms can beat our Complex Heuristic 1 if giving enough computation time. The results of this test is shown as in Figure 2. We can see that with max runtime of 10 minutes both of the two referenced algorithms cannot reach an acceptable result.

Figure 1: 10000 Duration Without Improvement Termination Result

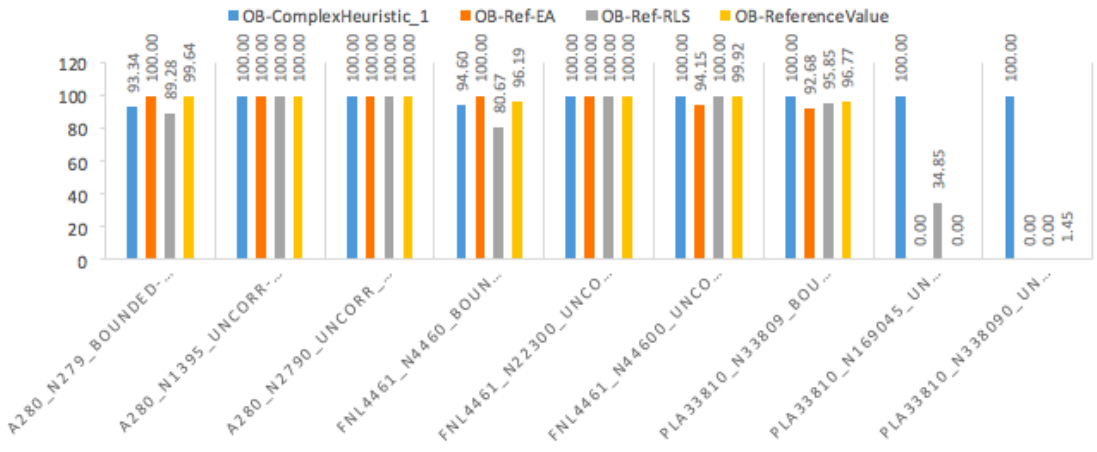

Figure 2: 90000000 Duration Without Improvement Termination Result