DSA 5113: Advanced Analytics and Metaheuristics

Homework #4
Instructor: Charles Nicholson

Homework team:
Saeid Hosseinipoor

April 2017

1 Strategies for the problem

The initial solution was formed based on a random process. A random state with seed of 5113 was assumed and then a vector of zeros and ones was made up. The solution space is a 100-dimensional space.

To generate a neighbor, a neighborhood structure should be specified beforehand. Often the following types of neighborhoods are considered:

transpose neighborhood in which two jobs occupying adjacent positions in the sequence are interchanged:

(1, 2, 3, 4, 5, 6, 7) → (1, 3, 2, 4, 5, 6, 7);

swap neighborhood in which two arbitrary jobs are interchanged:

(1, 2, 3, 4, 5, 6, 7) → (1, 6, 3, 4, 5, 2, 7);

insert neighborhood in which one job is removed from its current position and inserted elsewhere:

(1, 2, 3, 4, 5, 6, 7) → (1, 6, 3, 4, 5, 2, 7)

also, 1-flip neighborhood, 2) 2-flip neighborhood, and 3) 3-flip neighborhood can be used.

In swap neighborhood, the size of the neighborhood is equal to n*(n-1)/2 so that n is the number of the items; in insert neighborhood, the size of the neighborhood is equal to n-1 and in transpose neighborhood the size of the neighborhood is equal to n*(n-1)/2. Also, for the 1-flip, 2-flip and 3-flip neighborhood structures there can be 5000 solutions.

We also think that following neighborhood would not work well:

1- Bisimilarity
2- Geographical neighborhood are very large-scale neighborhood structures and seem to not be very effective for solving NP hard problems such as the Traveling Salesman or the Knapsack problem.

The total weight in the knapsack cannot exceed the total capacity weight of the knapsack. There can not be more than 100 elements picked for this instance of the problem.

2 Local Search with Best Improvement

The given program was edited based on the definitions and given algorithm. The following function is the result of this modification:

```
localSearch (value, weight,
        myPRNG = Random(5113),
        method='Best improvement',
        restart=1,
        show=False)
```

parameters could be tuned for each application. For this question default values were used. Value and weights are given vectors of evaluation function. myPRNG is the random state, method is the algorithm method, restart is number of restart with new random state, and show is a flag to show intermediate calculations on screen.

The function returns five following values: solutionsChecked is number of examined solutions, f_best is the valuation value for the best solution, x_best is the best solution, f_tracked, x_tracked are the best solution tracked in random restart trials.

3 Local Search with First Improvement

For this question, localSearch function were used by method='First Improvement'.

4 Local Search with Random Restarts

For this question, same function with a value of restart=100 were implemented. It was also running with both methods.

5 Local Search with Random Walk

Very similar function was developed for random walk model:

```
randomWalk (value, weight,
        myPRNG = Random(5113),
        method='Best improvement',
        prob=0.5,
        maxIteration=100000,
        show=False)
```

prob is the probability for selection of random direction, and maxIteration is the maximum iteration. This function also has results of solutionsChecked, f_best, x_best.

6 Simulated Annealing

The code also modified for simulated annealing algorigthm:

```
simulatedAnnealing (value, weight,
        myPRNG = Random(5113),
        maxIteration=100,
        T_min=1,
        T_max=10000,
        coolingMethod='Proportioanal',
```

```
        alpha=0.95,
        show=False)
```

New parameters here are; T_min and T_max which are minimum and maximum temperatures for process. Cooling methods are the methods proposed in the course notes that coded as:

```
if coolingMethod == 'Proportioanal':
    T = alpha * T
  elif coolingMethod == 'Cauchy':
    T = T0 / (1 + k)
  elif coolingMethod == 'Boltzmann':
    T = T0 / math.log(1 + k)
  elif coolingMethod == 'Very Fast':
    T = T0 * math.exp(-c * k ** Q)
```

The initial temperature is maximum temperature. It should be a large number. There no specific remedy for all the problems. Very large temperatures result high probability of choosing the value and teds to the random walk models. The value of initial temperature is related to $E_1 – E_2$ value. For this application value of 10000 is a good choice.

Other important parameter is cooling factor. Very fast cooling function led us to very fast stopping criterion and may not have a good precision. In other hands a very slow procedure also wastes lot of time and resources without a sensible improvement. Here a simple proportional cooling rate with rating coefficient of 0.95 was considered.

Two items were considered as stopping criteria. First was the minimum temperature and the second one is the maximum iteration.

7 Tabu Search or Variable Neighborhood Search

Two functions were developed for this question. Both functions are similar in form to previous functions:

```
VNS (value, weight,
        myPRNG = Random(5113),
        maxFlip=3,
        show=False)

tabu (value, weight,
        myPRNG = Random(5113),
        maxIteration=1000,
        tenure=10,
        show=False)
```

In order to explore the space we make some solution as tabu. The criterion to make a solution tabu is that if the flipped component of the solution should not be used in n^{th} iteration, when n is the tabu tenure which kept as a constant. However, if a solution is very good, better than the best solution up to now, the criterion would be overridden, aspiration criterion. After tuning, tabu tenure is chosen as 3.

8 Summary

Algorithm	Parameters	Number of Solutions	Objectives
Hill Climbing	Best Improvement	800	3003
Hill Climbing	First Improvement	219	2812
Hill Climbing	Random restart = 100 Best Improvement	82400	3165
Hill Climbing	Random restart = 100 First Improvement	23804	2895
Hill Climbing	Random Walk Probability = 0.5 Best Improvement	100100	3704
Hill Climbing	Random Walk Probability = 0.3 Best Improvement	100100	3723
Simulated Annealing	Proportional cooling T_0 = 10000 Alpha = 0.95	18000	3101
Simulated Annealing	Proportional cooling T_0 = 5000 Alpha = 0.95	16700	2895
Simulated Annealing	Proportional cooling T_0 = 5000 Alpha = 0.99	16700	3215
Simulated Annealing	Cauchy cooling T_0 = 5000	16700	3215
Simulated Annealing	Boltzmann cooling T_0 = 5000	16700	3215
Tabu Search	Iteration = 1000 Tenure = 10	100000	3666
Tabu Search	Iteration = 1000 Tenure = 100	100000	3672
Tabu Search	Iteration = 100	10000	3652

	Tenure = 100		
Tabu Search	Iteration = 10000 Tenure = 5	1000000	3652
VNS	Flip = 3	166750	2670

Tabu search with long-term memory (frequency-based) and random walk algorithm showed the best objective value, around 3700, while maintaining the efficiency. Other algorithm may find close values. However, the algorithms were significantly slower.