***ISE 5113: Advanced Analytics and Metaheuristics***

***Homework #4***

***Instructor: Charles Nicholson***

**Homework Team:**

*Gowtham Taluru*

*Yiwen Gong*

*April 2017*

# Strategies for the Problem (14 Points)

## (a) Initial\_solution (initial\_prop\_of\_items)

* The strategy for determining an initial solution is randomly pick items (generating a random binary number set) from n=100 items. In the question, whether the item *i (i ϵ[1, n])* will be selected is defined as a binary variable.
* Sometimes this leads to an initial solution being infeasible. So, the function initial\_solution (initial\_prop\_of items) is defined through which initial proportion of items in initial solution can be changed to avoid infeasible initial solution.
* So initial solution (0.1) will have 10 random items selected and initial solution(0.3) will have 30 random items selected

## (b)

* 1st neighborhood structure: 1-flip neighborhood.
* 2nd neighborhood structure: 2-flip neighborhood
* 3rd neighborhood structure: 3-flip neighborhood

## (c)

* The size of the 1-flip neighborhood: 100
* The size of the 2-flip neighborhood: = 4950
* The size of the 3-flip neighborhood: = 161700

## (d)

**The bad examples of neighborhood would be:**

**Swap neighborhood:** Since problem is encoded using binary variables, swapping (1,1) and swapping (0,0) will result in similar neighbors to original solution and it may lead algorithms like hill climbing with best improvement end prematurely.

**Insert neighborhood:** Insert neighbourhood is generated by inserting 1 or 0 and shifting all the binary varialbes to right. In knapsack problems where items does not have any relation with their preceding or following items, neighbourhood will have minimum relation with current solution. This kind of neighbourhood may not take the solution in improving direction.

## (e)

**2 strategies for handling infeasibility:**

* Infeasible solutions can be handled by making the value of that solution zero. So, the recursive algorithm will go to the nearby feasible solution. But this technique will get struck if whole neighborhood has infeasible solutions
* Infeasible solutions can be penalized based upon their degree of infeasibility. Instead of making the value of all infeasible solutions zero, the total value of the solution can be reduced in proportion to the weight of that solution exceeded from max weight. This helps the algorithm to search more near the boundaries.

# Local Search with Best Improvement (12 points)

**Initial Solution:** Initial solution is defined by the function **Random\_initial\_solution (initial\_prop\_of\_items):** This function will return an initial solution with predefined number of randomly selected objects. This function is defined to vary number of items selected in the initial solution and also to avoid infeasible initial solutions.

**Handling Infeasibility:** As discussed in 1 (e) infeasible solutions (whose weight is greater that maximum weight is penalized by based on their weight greater than maximum weight.

Value of infeasible solution = Solution weight – Maximum weight

More the weight of infeasible solution more negative will be its value and this will force the solution towards feasible space.

Table 1. Solution parameters for 1 flip Neighborhood with various number of items in initial solution

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Initial proportion of items** | **No. of solutions checked** | **No. of items selected** | **Total weight** | **Total Value** |
| **0 (0)** | **4000** | **39** | **495.3** | **3465** |
| 0.1 (10) | 2800 | 41 | 495 | 3281 |
| 0.2 (20) | 1900 | 41 | 495 | 3029 |
| 0.3 (30) | 2100 | 42 | 496 | 3326 |
| 0.4 (40) | 1400 | 41 | 498 | 3053 |

The solution using various kinds of neighborhoods with random initial solution with initial prop of items 0.5 are shown in table below

Table 2. Solution parameters for 1 different Neighborhoods for hill climbing with best improvement



# Local Search with First Improvement (5 points)

To obtain the first improved solution, neighborhood was searched only until we found the solution better than the initial solution. The result of the first improvement algorithm with different neighborhood structure is summarized in the table below and the best objective value among all structures is highlighted in green:

Table 3. results of Hill climbing with first improvement with different neighborhoods (initial\_prop 0.5)



# Local Search with Random Restarts (8 points)

**def hill\_climbing\_with\_random\_restarts(initial\_prop\_of\_items, restarts)**

**restarts** is the number of random restarts made. The highest value recorded till recent iteration will be recorded in **x\_super\_best** variable and reported in the end.

Table 4. Results of local search (Hill Climbing) with random restarts (initial\_prop=0.5, restarts= 100)



# Hill Climbing with Random Walk (8 Points)

**hill\_climbing\_with\_random\_walk(initial\_prop\_of\_items, random\_walk\_prob,**

**max\_super\_best\_steps)**

**Neighbourhood used:** One flip neighborhood

**Recursive Algorithm:** Local search with **Best** **improvement**

**Probability of random walk:** The probability of random walk is soft coded using parameter **random\_walk\_prob.** This function can be called with different with different probabilities of random walk

**Memory:** The best solution among the searched solutions is stored in **x\_super\_best** and its evaluation is stored in **f\_super\_best**

**Stopping condition:** The search algorithm will be stopped if **x\_super\_best, f\_super\_best** does not change for **max\_super\_best\_steps**

The solutions found with various probabilities of random walk are as follows (initial prop of items 0.1, max\_super\_steps = 500)

**( \*Bonus)**

Table 5. Results of local search (Hill Climbing) with random walk

|  |  |  |  |
| --- | --- | --- | --- |
| **Random walk Prob** | **Solutions checked** | **Best Value found** | **Weight** |
| 0.05 | 58200 | 3567 | 499.8 |
| 0.1 | 119400 | 3609 | 499.1 |
| 0.15 | 54600 | 3596 | 495 |
| 0.2 | 51300 | 3609 | 499 |
| 0.5 | 50400 | 3597 | 498 |

\*\*\* These values are stochastic and may change from run to run

One of the best solutions is as follows:

**Final number of solutions checked:** 56100

**Best value found:** 3620.18299808

**Weight is:** 499.866997185

**Total number of items selected:** 42

**Best solution:** [0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0]

# Simulated Annealing

**simulated\_Annealing (initial\_prop\_of\_items= 0.1, initial\_temp=8000, iter\_per\_temp= 1000, final\_temp==5)**

**Initial Temperature and Cooling schedule:** Various cooling schedules were tried with different initial temperatures and probability of making bad moves were plotted in excel. The probabilities of making bad moves of different magnitudes (20, 100, 500) with various initial temperatures and Cauchy cooling schedule is shown in figure below. It can be observed that the probability of making bad moves is high initially and reduces to very low when temperature reduces below 5.

**Cooling Schedule:** Cauchys cooling schedule is used for updating temperature:

where *k* is the temperature step.

The probability of making non improving moves is shown in figures provided.

The change in maximum value found with change in number of interactions at each temperature for same initial random solution is shown in table below

**(Bonus\*)** Table 6. Results of simulated annealing ( Initial temp = 8000, intial prop 0.1)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Iterations per temp** | **Max Value found** | **Number of Solution checked** | **Total random steps** | **Total Improvements** |
| 50 | 3605 | 79950 | 3553 | 2591 |
| 100 | 3622 | 159900 | 6079 | 5254 |
| 300 | 3641 | 479700 | 19422 | 15667 |
| 1000 | 3699 | 1599000 | 61609 | 52072 |

**Stopping criteria:** The probability of making bad moves becomes very small as the temperature reaches below 10. The algorithm is set to stop when the temperature becomes less than 5.





The best solution found is

**Final best solution** [0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0]

**Value:** 3699.64387471 **Weight:** 499.826398433

**Total random steps:** 61609 **Total improvements** 52072

**Solutions checked:** 1599000

# Tabu Search or Variable Neighborhood Search (30 Points)

## Tabu Search

Function defined in the code: **taboo\_search (initial\_prop\_of\_items, taboo\_tenure, max\_super\_best\_steps)**

**Initial\_prop\_of\_items** is proportion of items selected in initial solution

**Neighborhood criteria used:** 1-flip neighborhood

**Taboo Criteria:** If any item is selected or deselected in the current solution, changing its status will become taboo (not possible) until **taboo\_tenure** number of steps

**Aspiration Criteria:** If any solution in the neigbourhood of current solution has objective value higher than the objective values observed till then, taboo criteria will be overruled and the current solution will move to that neighbor.

**Stopping Criteria:** If **f\_super\_best** (Highest objective value found) does not change for **max\_super\_best\_steps** the taboo search will stop.

Taboo search algorithm is run with different **taboo\_tenures** and **max\_super\_best\_steps** and the value of highest value found is shown in table below

(Bonus\*) Table 7. Highest value found vs Taboo tenure vs Max\_super\_steps (intial prop 0.1)

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Taboo tenure** | | |
| **Max\_super\_steps** | **10** | **30** | **50** |
| **100** | 3570 | 3621 | 3559 |
| **300** | 3572 | 3621 | 3601 |
| **500** | 3572 | 3621 | 3601 |
| **1000** | 3572 | 3621 | 3601 |
| **2000** | 3572 | 3639 | 3601 |
| **10000** | 3572 | **3659** |  |

Taboo tenure of 30 steps showed good betterment in the solution with increase in max\_super\_steps. The best solution found is

**Final best** [0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0]

**Value:** 3659

**Weight:** 498

**Solutions checked:** 2506600

## Variable Neighborhood Search (Bonus)

The defined neighborhood structures for the Variable Neighborhood Search is 1-flip, 2-flip and swap in here. The reason why the 3-flip is not selected is because its neighborhood structure is huge so that it severely slows down the variable neighborhood search’s speed. In addition, the best value that was obtained from 1-flip, 2-flip as well as 3-flip are fairly good enough with a relatively lower time span. The evaluation is going through all three neighborhoods and find the best solution among 3-best solutions from 3 types of the neighborhood structures.

The infeasible solutions are always obtained if we kept the original definition for the initial solution. Thus, the initial solution for this question is switched with the weighting approach as explained earlier in question 1 part e for avoiding excessive infeasible issue. The result of VNS is shown below:

Total number of solutions checked: 5037900

**Best value found:** 3883

Weight of knapsack: 499

**Best solution:**

[1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0]

# Summary (5 points)

**8.1 Calculation of global maximum using Excel.**

Since the problem give is very straight forward, global maximum can be calculated using simple calculation using excel. The solution is nothing but items having highest value to weight ratio within weight limit of the knapsack. The methodology can be seen in the table below

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Item No** | **Value** | **Weight** | **Value density** | **Selection** | **Weight Contribution** | **Value Contribution** |
| 17 | 88.0 | 5.1 | **17.2** | 1.0 | 5.1 | 88.0 |
| 61 | 98.2 | 6.4 | **15.3** | 1.0 | 6.4 | 98.2 |
| 95 | 83.4 | 5.9 | **14.2** | 1.0 | 5.9 | 83.4 |
| 85 | 86.0 | 6.1 | **14.1** | 1.0 | 6.1 | 86.0 |
| 54 | 81.8 | 6.4 | **12.7** | 1.0 | 6.4 | 81.8 |
| 58 | 64.5 | 5.2 | **12.3** | 1.0 | 5.2 | 64.5 |
| 67 | 91.9 | 7.6 | **12.0** | 1.0 | 7.6 | 91.9 |

\*This technique is possible only due to simple objective function.

**Global** **Maximum: 3738**

**Weight: 499.4**

Global best [1 1 1 0 0 0 0 1 1 0 1 1 1 1 0 0 1 1 0 0 1 0 1 0 1 1 1 1 0 1 1 1 0 1 0 1 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 1 0 0 1 1 0 0 1 1 1 0 0 0 1 1 1 0 1 0 0 0 1 0 1 0 0 0 1 0 0 0 1 1 1 1 1 0 1 0 0 0 0 0]

**8.2 Conclusions**

Size of the solution space =

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Method** | **Solution checked** | **Max value** | **Percentage of search space evaluated** | **Solution as % of global maximum** |
| Hill climbing with best improvement (3 flip) | 3395700 | 3729 | 0.0000000000000000000268% | 99.8% |
| Hill climbing with first improvement (3 flip) | 475726 | 3680 | 0.0000000000000000000038% | 98.4% |
| Hill climbing with random restarts (3 flip) | 315800100 | 3710 | 0.0000000000000000024912% | 99.3% |
| Hill climbing with random walk | 56100 | 3620 | 0.0000000000000000000004% | 96.8% |
| Simulated Annealing | 1599000 | 3699 | 0.0000000000000000000126% | 99.0% |
| Tabu Search | 2506600 | 3659 | 0.0000000000000000000198% | 97.9% |

* The size of the solution space is so huge that it becomes impossible to check each and every possible solution.
* It can seen from the above table that all the neighborhood based heuristics reached up to 96% of global maximum by searching very tiny part of the solution.
* The results shown in the table has lot of stochasticity in them. Any technique above giving best result in one instance doesn’t mean it will generate best solution every single time. I would recommend Simulated annealing due to its organized way of allowing bad moves and leading the solution to global maximum.
* We tried changing the number of items in the initial solution. In most of the techniques, initial solution having no item selected lead to maximum objective value. This could be due to more flexibility to the algorithm to select the items.
* There is a tradeoff regarding the performance of neighborhood-based the heuristics problem: the speed of the heuristics searching and the size of the neighborhood. Generally, the larger the neighborhood is, the slower the searching speed would be. For example, simulated annealing algorithm has the advantages of high quality, robust initial and easy while the searching speed is relatively longer compare to others. Tabu search’s performance is strongly depends on the initial solution setting while it actually has strong local search ability of global iterative optimization (Zhang, 2011).

# References

Zhang, J. (2011). Comparative Study of Several Intelligent Algorithms for. *Procedia Environmental Sciences* , 163 – 168.