MAE 552 – Heuristic Optimization

Lecture 23

March 18, 2002

Topic: Tabu Search

http://unisci.com/stories/20021/0315023.htm

Tabu Search

- The Tabu search begins by marching to a local minima.To avoid retracing the steps used, the method records recent moves in one or more Tabu lists.
- The role of the memory can change as the algorithm proceeds.
  - At initialization the goal is make a coarse examination of the solution space, known as 'diversification’.
  - As candidate locations are identified the search is more focused to produce local optimal solutions in a process of 'intensification'.

Tabu Search

- In many cases the differences between the various implementations of the Tabu method have to do with the size, variability, and adaptability of the Tabu memory to a particular problem domain.
- The Tabu search has traditionally been used on combinatorial optimization problems.
- The technique is straightforwardly applied to continuous functions by choosing a discrete encoding of the problem.
- Many of the applications in the literature involve integer programming problems, scheduling, routing, traveling salesman and related problems.

Tabu Search –Basic Ingredients

- Many solution approaches are characterized by identifying a **neighborhood** of a given solution which contains other so-called transformed solutions that can be reached in a single iteration.
- A transition from a feasible solution to a transformed feasible solution is referred to as a move.
- A starting point for Tabu search is to note that such a move may be described by a set of one or more attributes (or elements).
- These attributes (properly chosen) can become the foundation for creating an attribute based memory.

Tabu Search

- Following a steepest descent / mildest ascent approach, a move may either result in a best possible improvement or a least possible deterioration of the objective function value.
- Without additional control, however, such a process can cause a locally optimal solution to be re-visited immediately after moving to a neighbor, or in a future stage of the search process, respectively.
- To prevent the search from endlessly cycling between the same solutions, a tabu list is created which operates like a short term memory.

Tabu Search –

- Attributes of all explored moves are stored in a list named a **running list** representing the trajectory of solutions encountered.
- Then, related to a sublist of the running list a so-called **tabu list** may be introduced.
- The **tabu list** implicitly keeps track of moves (or more precisely, salient features of these moves) by recording attributes complementary to those of the running list.

Tabu Search –

- These attributes will be forbidden from being embodied in moves selected in at least one subsequent iteration because their inclusion might lead back to a previously visited solution.
- Thus, the tabu list restricts the search to a subset of admissible moves (consisting of admissible attributes or combinations of attributes).
- The goal is to permit "good" moves in each iteration without re-visiting solutions already encountered.

Tabu Search – Pseudo-Code

```
Given a feasible solution x* with objective function value z*:
 Let x := x* with z(x) = z*.
Iteration:
 while  stopping criterion is not fulfilled  do
          begin
          (1) select best admissible move that transforms  x into x' with objective function value
              z(x') and add its attributes to the running list
          (2) perform tabu list management: compute moves  (or attributes) to be set tabu, i.e.,
              update the tabu list
          (3) perform exchanges:
         x = x',  z(x) = z(x');
         if  z(x) < z*  then
                  z* = z(x), x* = x
         endif
endwhile
 Result:  x* is the best of all determined solutions, with objective function value z*.
```

Tabu Search – Example 1: SAT Problem

•Suppose we are solving an SAT problem with n=8 variables.

•For a given logical formula F we are looking for the a truth assignment for all 8 variables such that F is TRUE.

The initial truth assignment for x=(x1…….x8)

$$x=(0,1,1,1,0,0,0,1)$$

•The evaluation function is to maximize a weighted sum of the number of satisfied clauses.

$$F(x_{initial})=27$$

Tabu Search – Example 1: SAT Problem

- Step 1: Examine the neighborhood of $x_{initial}$

The neighborhood consists of all the solutions made by flipping a single bit of x.

$x_{initial}$=(0,1,1,1,0,0,0,1)

N_1= (1,1,1,1,0,0,0,1)	N_5= (0,1,1,1,1,0,0,1)
N_2= (0,0,1,1,0,0,0,1)	N_6= (0,1,1,1,0,1,0,1)
N_3= (0,1,0,1,0,0,0,1)	N_7= (0,1,1,1,0,0,1,1)
N_4= (0,1,1,0,0,0,0,1)	N_8= (0,1,1,1,0,0,0,0)

Evaluate all of N(x) and choose the best solution. At this stage it is very similar to the hill-climbing procedure.

Tabu Search – Example 1: SAT Problem

- Suppose that N_3 provides the best solution, F=31 so this is the new best solution.
- Now we introduce the idea of memory.
- Step 2: Create a memory structure for bookkeeping.

M=

- One element of an array for each design variable

Tabu Search – Example 1: SAT Problem

- The design specifies how long an element should remain in memory
- For this problem we decide that a move should remain ‘Tabu’ for 5 iterations. Then the memory after one iterations would be:

$M_1=$

0	0	5	0	0	0	0	0

- This implies that bit 3 is unavailable for flipping for 5 iterations
- After each iteration the elements in the memory are decreased by 1
- During the second iteration bit 7 is flipped

$M_2=$

0	0	4	0	0	0	5	0

Tabu Search – Example 1: SAT Problem

- Let us say that after 3 additional iterations of selecting the best neighbor - which is not necessarily better than the current point- the memory looks like:

M_5=

3	0	1	5	0	4	2	0

- F=33
- Bits 2,5, and 8 are available to be flipped any time.
- Bit 1 is not available for the next 3 iterations
- Bit 3 is not available for the next iteration
- Bit 4 is not available for the next 5
- Bit 6 is not available for the next 4
- Bit 7 is not available for the next 2.

Tabu Search – Example 1: SAT Problem

- Exercise: Based on the current Tabu list what were the last moves made and what does the current solution look like?

- Known initial solution: $x_{initial}$=(0,1,1,1,0,0,0,1)

M_5=

3	0	1	5	0	4	2	0

$\mathbf{x_1}$

$\mathbf{x_2}$

$\mathbf{x_3}$

$\mathbf{x_4}$

$\mathbf{x_5}$

Tabu Search – Example 1: SAT Problem

- Since our current memory looks like

M_5=

3	0	1	5	0	4	2	0

We can choose from which remaining solutions?

- What is M_6???????
- M_6=

Tabu Search – Modifications

- What happens if we come upon a very good solution and pass it by because it is Tabu?
- Perhaps we should incorporate more flexibility into the search.
- Maybe one of the Tabu neighbors, x_6 for instance provides an excellent evaluation score, much better than any of the solutions previously visited.
- In order to make the search more flexible, Tabu search evaluates the ‘whole’ neighborhood, and under normal circumstances selects a non-tabu move.
- But if circumstances are not normal i.e. one of the tabu solutions is outstanding, then take the tabu point as the solution.

Tabu Search –

- Overriding the Tabu classification occurs when the *‘aspiration criteria’* is met.
- There are other possibilities for increasing the flexibility of the Tabu Search.

1. Use a probabilistic strategy for selecting from the candidate solutions. Better solutions have a higher probability of being chosen.
2. The memory horizon could change during the search process.
3. The memory could be connected to the size of the problem (e.g. remembering the last $n^{1/2}$ moves) where n is the number of design variables in the problem.

Tabu Search –

4. Incorporate a ‘long-term’ memory in addition to the short term memory that we have already introduced.

- The memory that we are using can be called a *recency-based* memory because it records some actions of the last few iterations.
- We might introduce a *frequency-based* memory that operation on a much longer horizon. For example a vector H might be introduced as a long term memory structure.

Tabu Search – Example 1: SAT Problem cont.

- The vector H is initialized to zero and at each stage of the search the entry

  $$H(i)=j$$

  is interpreted as 'during the last *h* iterations of the algorithm the *i-th* bit was flipped *j* times.'
- Usually the value of *h* is set quite high in comparison to the length of the short-term memory.
- For example after 100 iterations with h =50 the long term memory *H* might have the following values displayed. H:

5	7	11	3	9	8	1	6

Tabu Search –

- H shows the distribution of moves during the last 50 iterations.How can we use this information?
- This could be used to *diversify* the search.
- For example H provides information as to which flips have been underrepresented or not represented at all, and we can diversify the search by exploring these possibilities.
- The use of long term memory is usually reserved for special cases.
- For example we could encounter a situation where are non-tabu solutions lead to worse solutions. To make a meaningful decision, the contents of the long term memory can be considered.

Tabu Search –

- The most common way to incorporate long term memory into the Tabu search is to make moves that have occurred frequently less attractive. Thus a penalty is added based on the frequency that a move has occurred.

$$F(x) = \text{Eval}(x)+P(\text{Frequency of Move})$$

Tabu Search –Long Term Memory

- To illustrate the use of the long term memory assume that the value of the current solution **x** for the SAT problem is 35. All non-tabu flips, say of bits 2,3, and 7 provide values of 30, 3, and 31.
- None of the tabu moves provides a value greater than 37 (the highest value so far), so we cannot apply the aspiration criteria.
- In this case we look to the long term memory to help us decide which move to take.
- A penalty is subtracted from F(x) based on the frequency information in the long term memory.
- Penalty=0.7 * H(i) is a possible penalty function.

Tabu Search – Long Term Memory

- The new scores for the three possible solutions are:

H=

5	7	11	3	9	8	1	6

Solution 1 (bit 2) = 30 -0.7*7=25.1

Solution 2 (bit 3) = 33-0.7*11 =25.3

Solution 3 (bit 7) = 31-0.7*1 =30.3

•The 3rd solution is selected

Tabu Search –Other Ways of Diversifying the Search

- Diversifying the search by penalizing the high frequency moves is only one possibility.
- **Other possibilities if we have to select a Tabu move:**
  - Select the oldest.
  - Select the move that previously resulted in the greatest improvement.
  - Select the move that had the greatest influence on the solution – resulted in the greatest change in F(x)

Tabu Search –TSP Example

- Consider a TSP with eight cities:

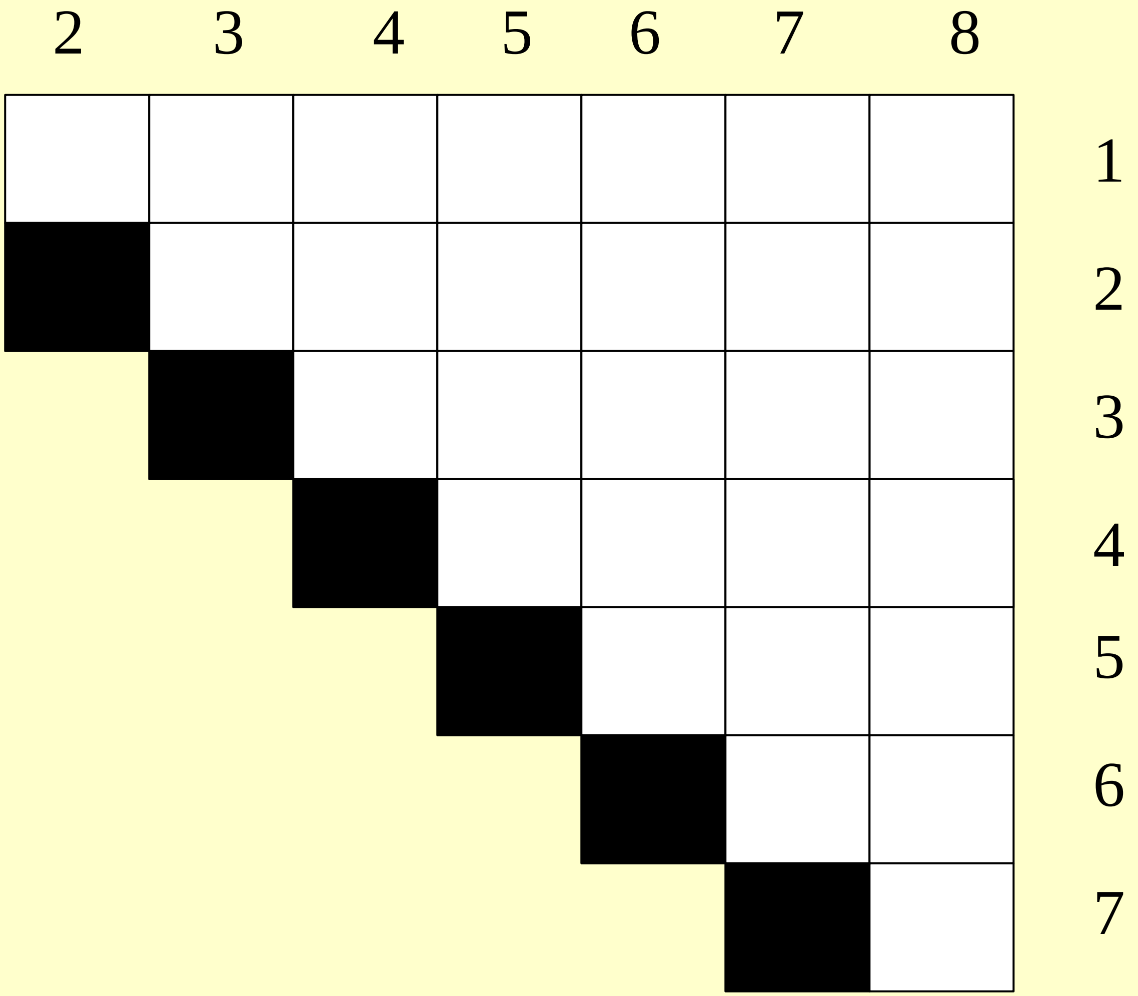
- Recall that a solution can be represented by a vector indicating the order the cities are visited
- Example: (2, 4, 7, 5, 1, 8, 3, 6)
- Let us consider moves that swap any two cities :

(2, 4, 7, 5, 1, 8, 3, 6) →(4, 2, 7, 5, 1, 8, 3, 6)---swap cities 1 and 2

- Each solution has 28 neighbors that can be swapped.

Tabu Search –TSP Example

The main memory component (short term memory) can be stored in a matrix where the swap of cities i and j is recorded in the i-th row and j-th column



Tabu Search –TSP Example

- We will maintain in the Tabu list the number of remaining iterations that given swap stays on the Tabu list (5 is the Max).
- We will also maintain a long term memory component *H* containing the frequency information for the last *50* swaps.
- After 500 iterations the current solution is:

(7,3,5,6,1,2,4,8) and F(x)=173

- The current best solution encountered in the 500 iterations is 171

Tabu Search – TSP Example

Short Term Memory (*M*) after 500 iterations

2	3	4	5	6	7	8	
0	0	1	0	0	0	0	1
■	0	0	0	5	0	0	2
	■	0	0	0	4	0	3
		■	3	0	0	0	4
			■	0	0	2	5
				■	0	0	6
					■	0	7

Most
Recent
Swap

Tabu Search – TSP Example

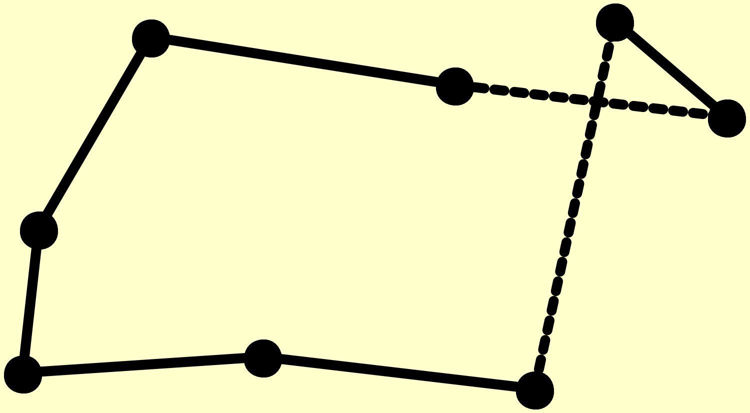
Long Term Memory (*H*) last 50 iterations

2	3	4	5	6	7	8	
0	2	3	3	0	1	1	1
■	2	1	3	1	1	0	2
	■	2	3	3	4	0	3
		■	1	1	2	1	4
			■	4	2	1	5
				■	3	1	6
					■	6	7

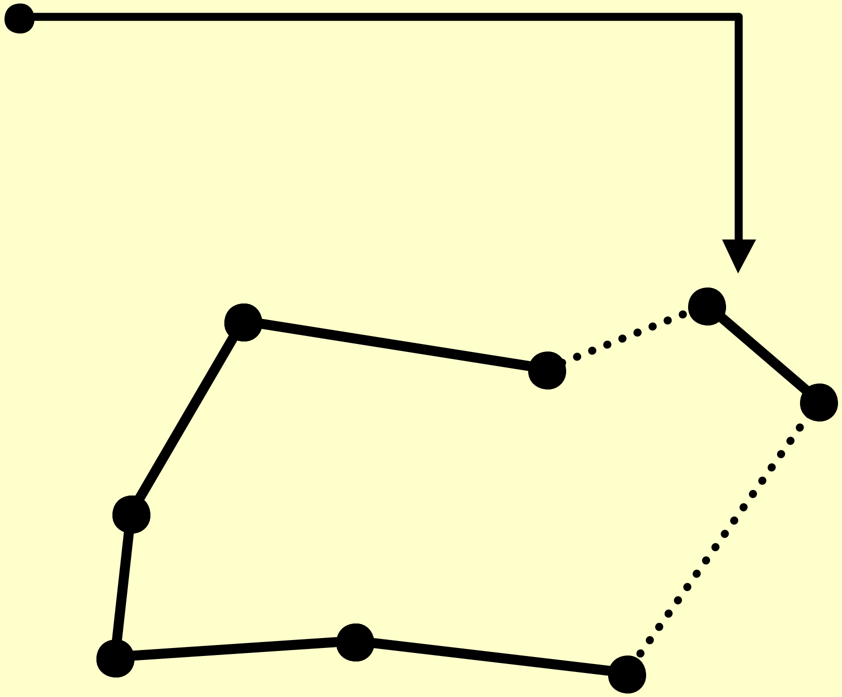
Tabu Search –TSP Example

- The neighborhood of this tour was selected to be a swap operation of two cities on the tour.
- This is not the best choice for Tabu Search.
- Many researchers have selected larger neighborhoods which work better.
- A two interchange move for the TSP is defined by changing 2 non-adjacent edges.

Tabu Search –TSP Example



2-Interchange Move



- For a 2-interchange move a tour is Tabu if both added edges are on the Tabu list.

Tabu Search –Summary

- Tabu Search works by redirecting the search towards unexplored regions of the design space.
- There are a number of parameters whose values are decided by the designer:

1. What characteristics of the solution to store in the Tabu list
2. The aspiration criteria – what criteria will be used to override the Tabu restrictions.
3. How long to keep a move on the Tabu list.
4. Whether to use long-term memory (H) and what to base it on (frequency, search direction, etc.).