MAE 552 – Heuristic Optimization

Lecture 24

March 20, 2002

Topic: Tabu Search

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 –Modifications

- 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 – Modifications

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.

- 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.

Tabu Search –

- 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.

- 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.

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

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, 33, 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 might want to 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.
- **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

- Goal is to minimize the distance for a complete tour

- 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)

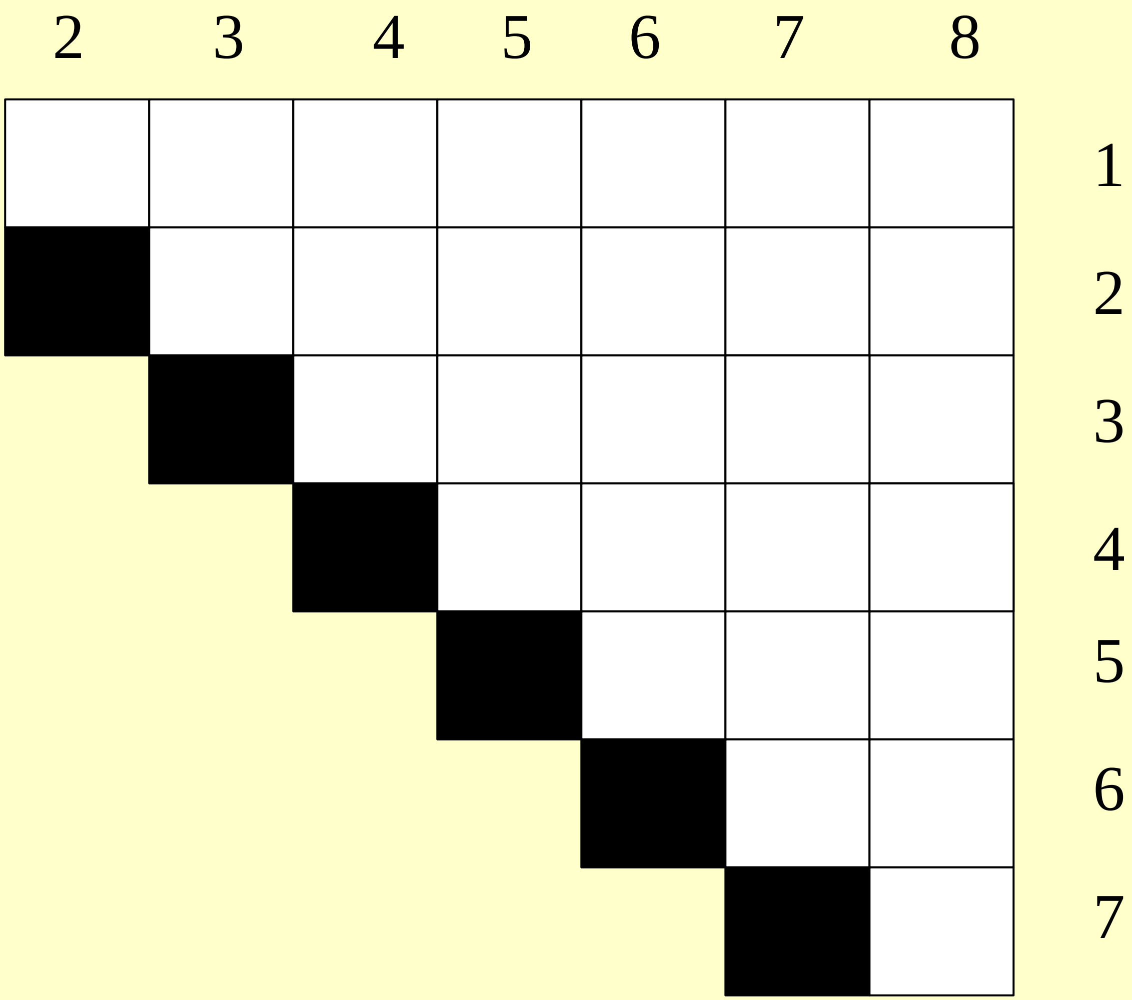
- Let us consider moves that swap any two cities :

(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

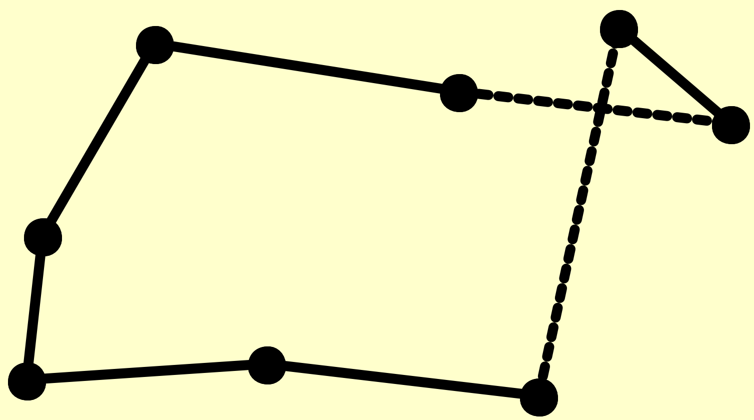
•The neighborhood of this tour was selected to be a swap operation of two cities on the tour.

•This is not the best choice for Tabu Search.

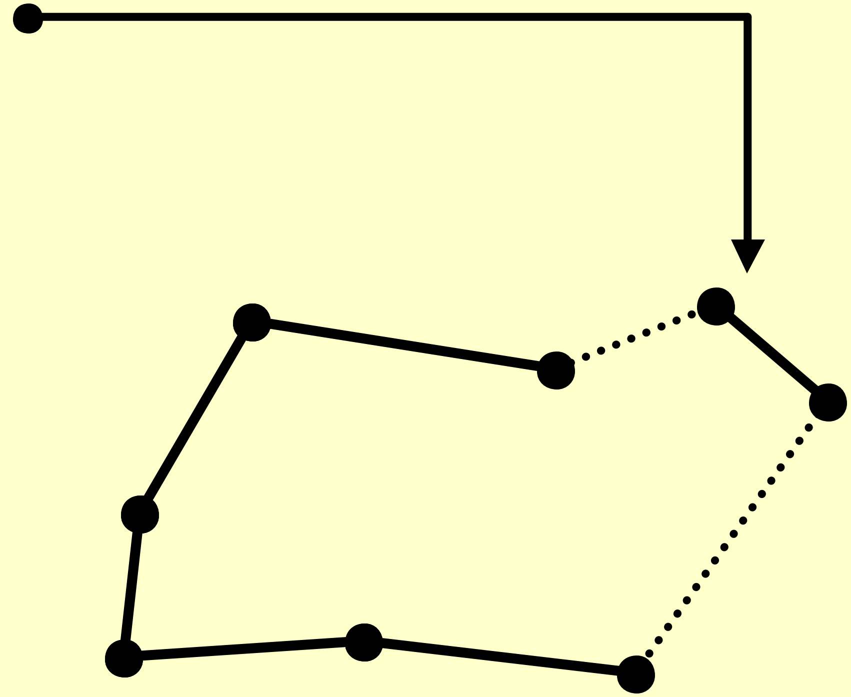
•Many researchers have selected larger neighborhoods which work better.

•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.).

Tabu Search –Length of Tabu List

- Ways to Select the Length of the Tabu List
  1. The length of the tabu-list is randomly selected from a range [r_{lower} r_{upper}]after every n iterations.

  2. The length of the Tabu list is a function of the size of the problem e.g. L=sqrt(n)

  3. The length of the Tabu list changes based on intensification diversification needs.
     - Shorten Tabu list to intensify
     - Lengthen Tabu list to diversify

Tabu Search –Length of Tabu List

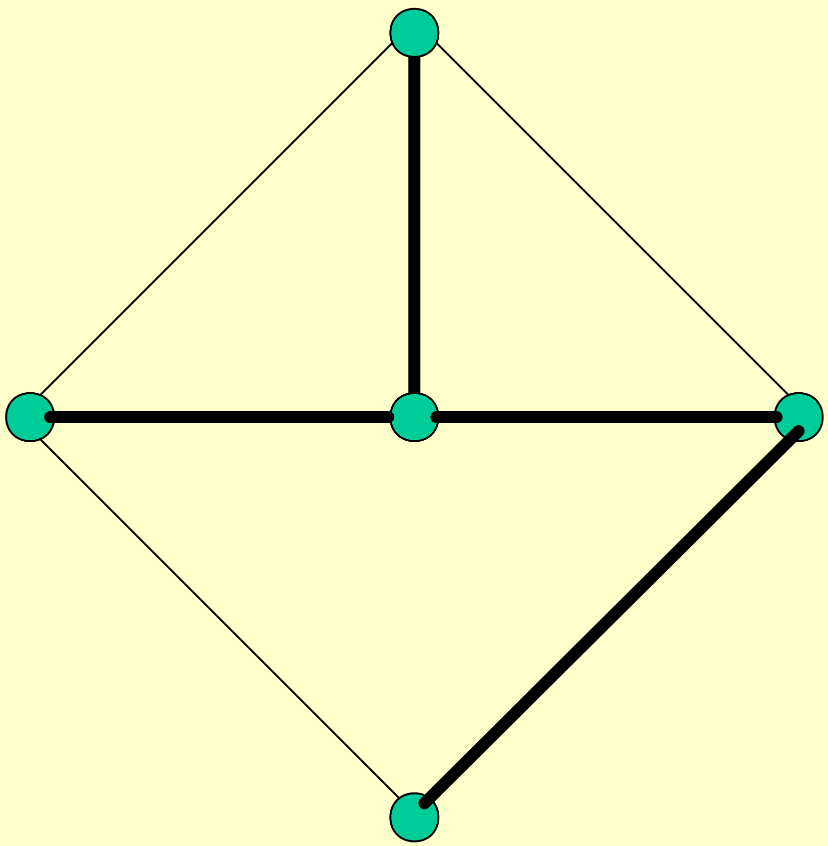
Reference: “A tutuorial on Tabu Search”, by Hertz

A Simple Illustration of Tabu Search

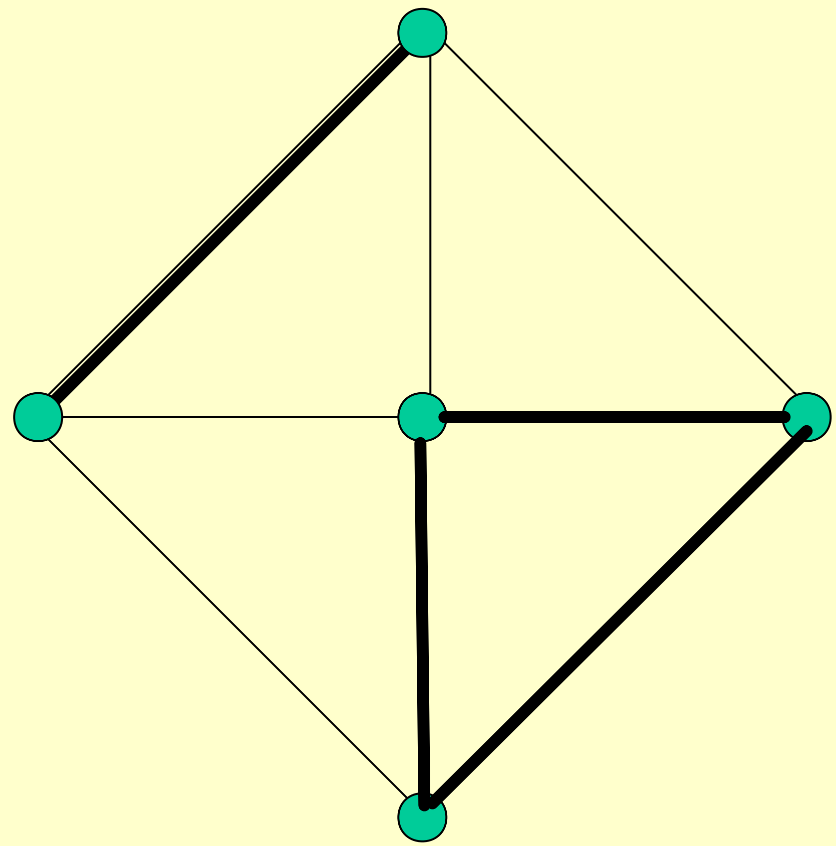
- A Simple Version of the short term memory component of the Tabu Search is illustrated in this example.
- The problem is known as a minimum spanning tree problem
- The minimum spanning tree (MST) of a graph defines the cheapest subset of edges that keeps the graph in one connected component.
- Telephone companies are particularly interested in minimum spanning trees, because the minimum spanning tree of a set of sites defines the wiring scheme that connects the sites using as little wire as possible.

A Simple Illustration of Tabu Search

Legal Spanning Tree

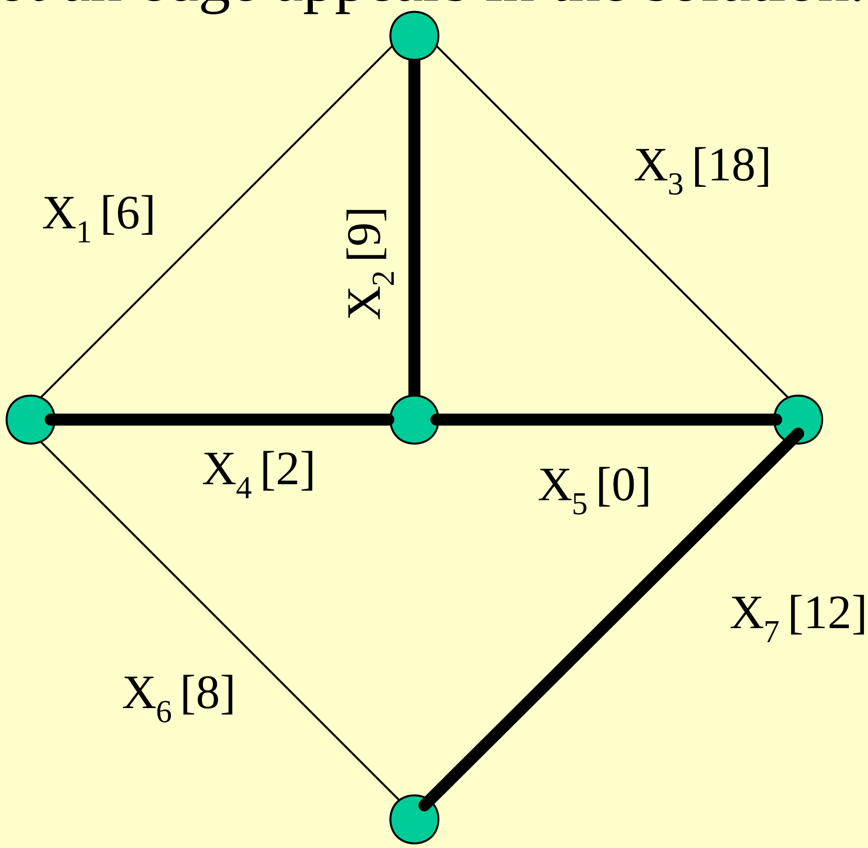


Illegal Spanning Tree



A Simple Illustration of Tabu Search

• A solution can be represented in terms of a vector indicating whether or not an edge appears in the solution.



This solution is (0,1,0,1,1,0,1) and F=23

A Simple Illustration of Tabu Search

- Additionally there are constraints imposed on this problem.
- Constraint 1: At most only one of edges 1, 2, or 6 can be used at the same time.

$$x_1+x_2+x_6 \leq 1$$

- Constraint 2: Edge 1 can be in the tree only if edge 3 is also in the tree

$$x_1 \leq x_3$$

- To permit the evaluation of the infeasible trees a penalty of 50 is added for each unit violation of a constraint. The a unit violation is when the left side of the constraint exceeds the right side by 1.

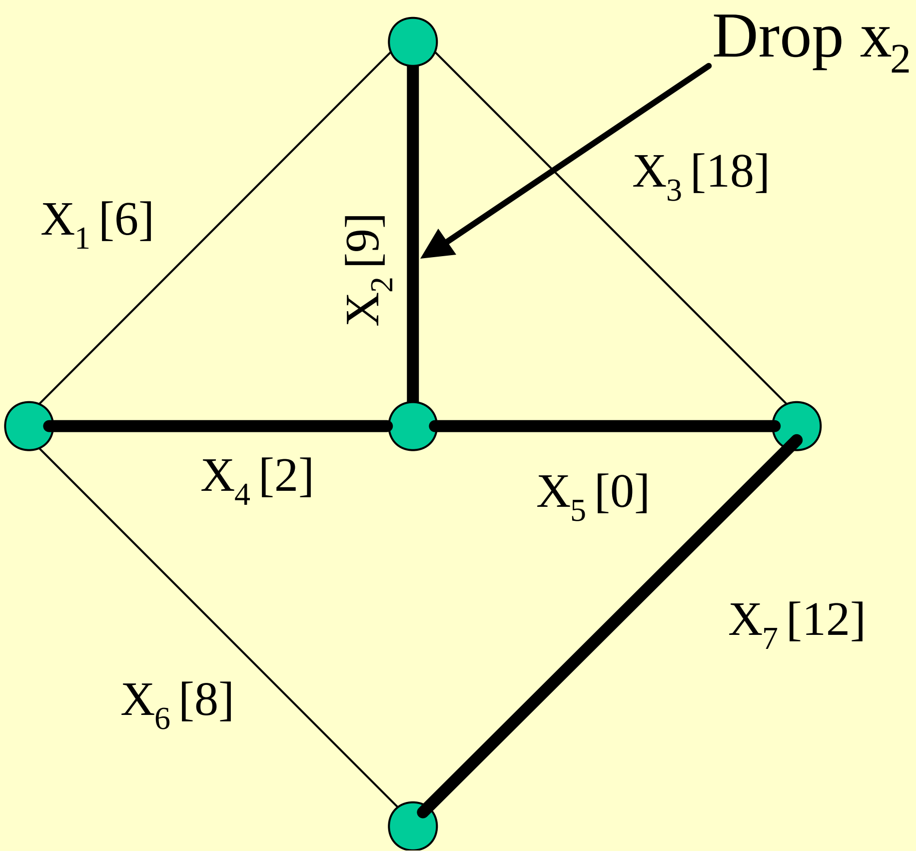
A Simple Illustration of Tabu Search

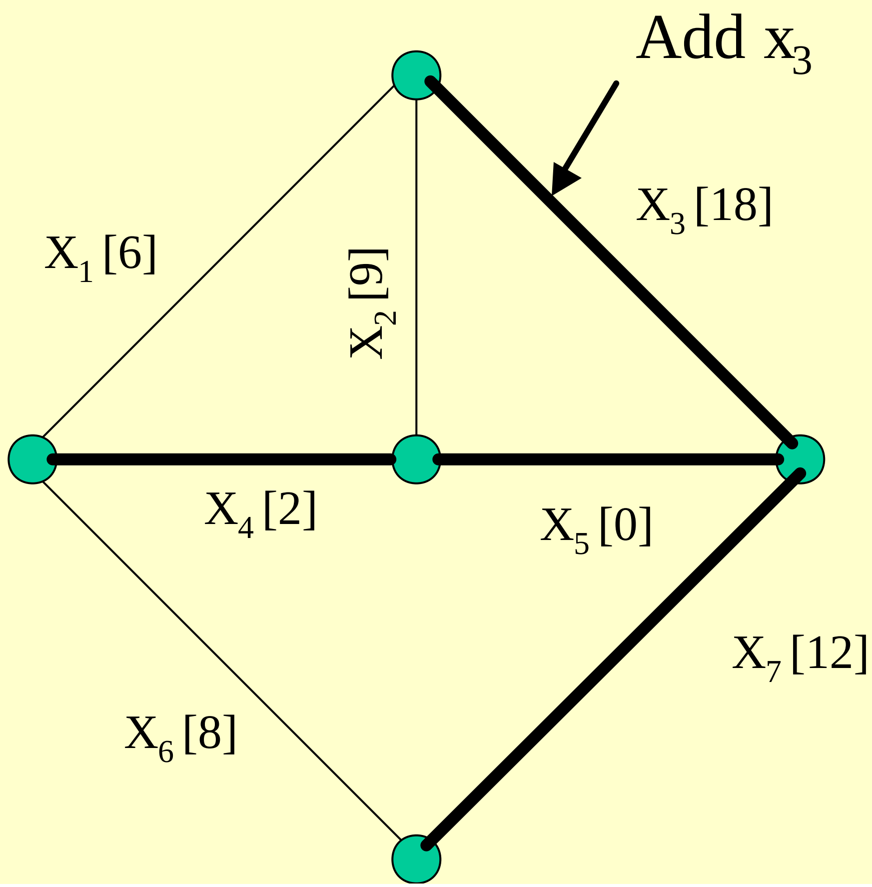
- To define a Tabu restriction, we have decided to use the *added* edge to be the move attribute assigned Tabu status.
- This forbids a future move from dropping the edge as long as it remains Tabu.
- The length of the tabu list for this example is 2.
- A move remains Tabu for two iterations and then is dropped from the list
- The aspiration criteria that we have selected is that a tabu restriction can be overridden if the resulting tree is better that any yet produced so far.

A Simple Illustration of Tabu Search

•For this example a move will be a standard edge swap that consists of removing an edge and adding an edge to make a new legal tree.

•The solution selected will be the admissible move with the lowest cost including penalty costs.

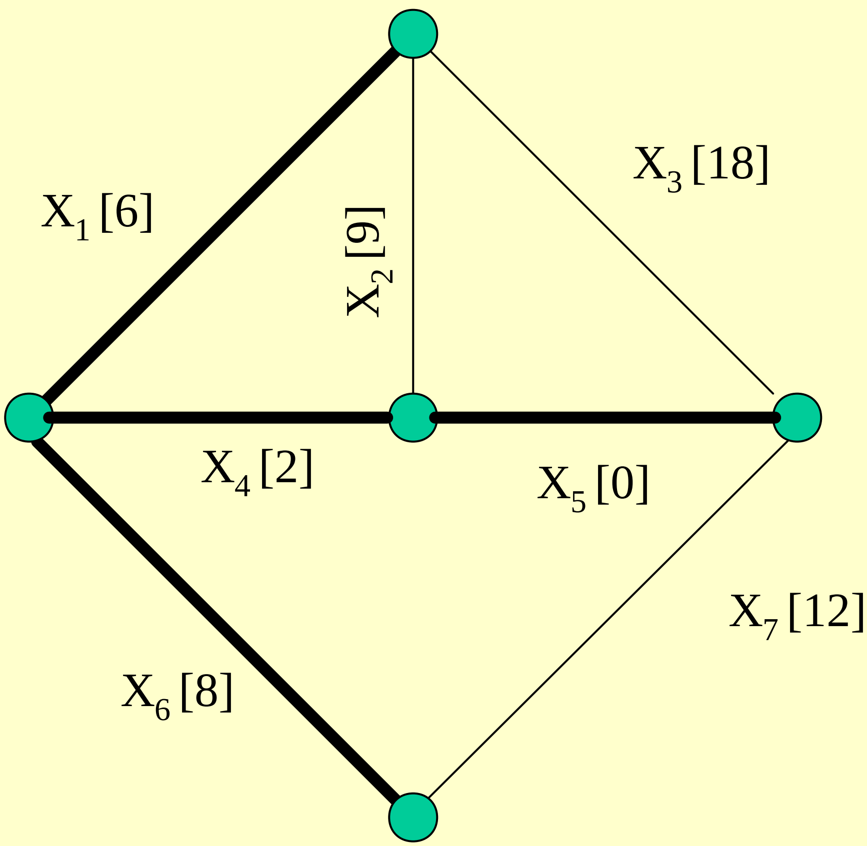




A Simple Illustration of Tabu Search

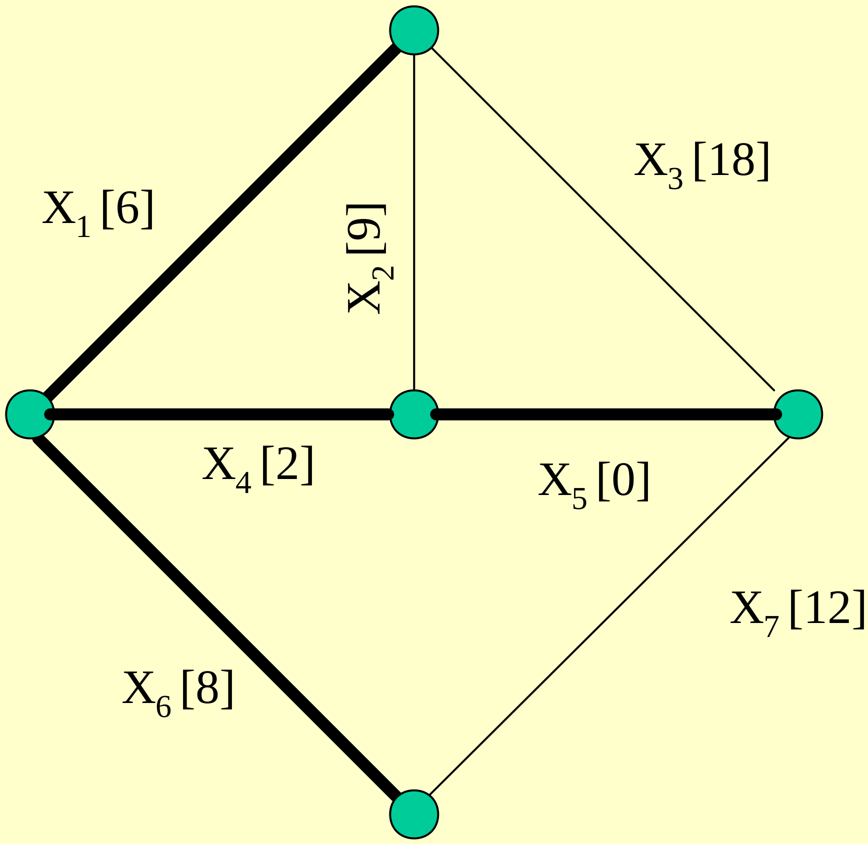
- Initial Solution Cost = 16 + 100 = 116

Current Best Point is Infeasible



A Simple Illustration of Tabu Search

- Initial Solution Cost = 16 + 100 = 116



<u>Search neighborhood</u>

Drop x_1 Add x_2 F=119

Drop x_1 Add x_3 F=28

Drop x_5 Add x_7 F=128

Drop x_5 Add x_3 F=84

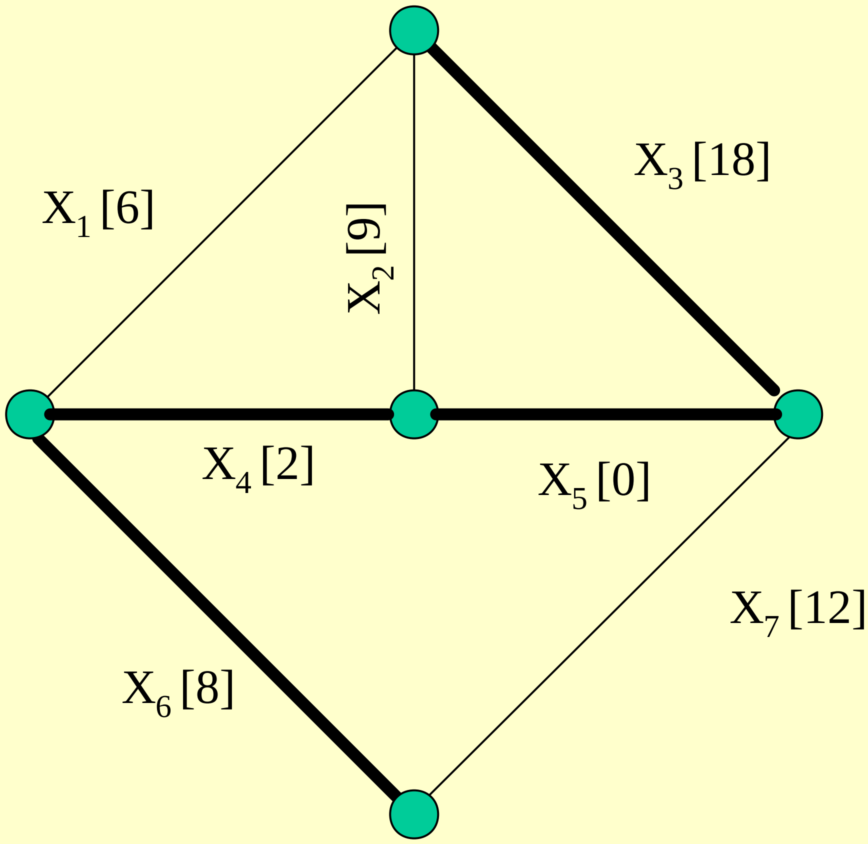
Drop x_6 Add x_7 F=70

<u>Best Choice</u>

Drop x_1 Add x_3 F=28

A Simple Illustration of Tabu Search

•Current Cost 28 Tabu List: x_3 M=[0 0 2 0 0 0 0]



<u>Search neighborhood</u>

Drop x_3 Add x_1 F=116 Tabu

Drop x_3 Add x_2 F=69 Tabu

Drop x_5 Add x_2 F=87

Drop x_5 Add x_7 F=40

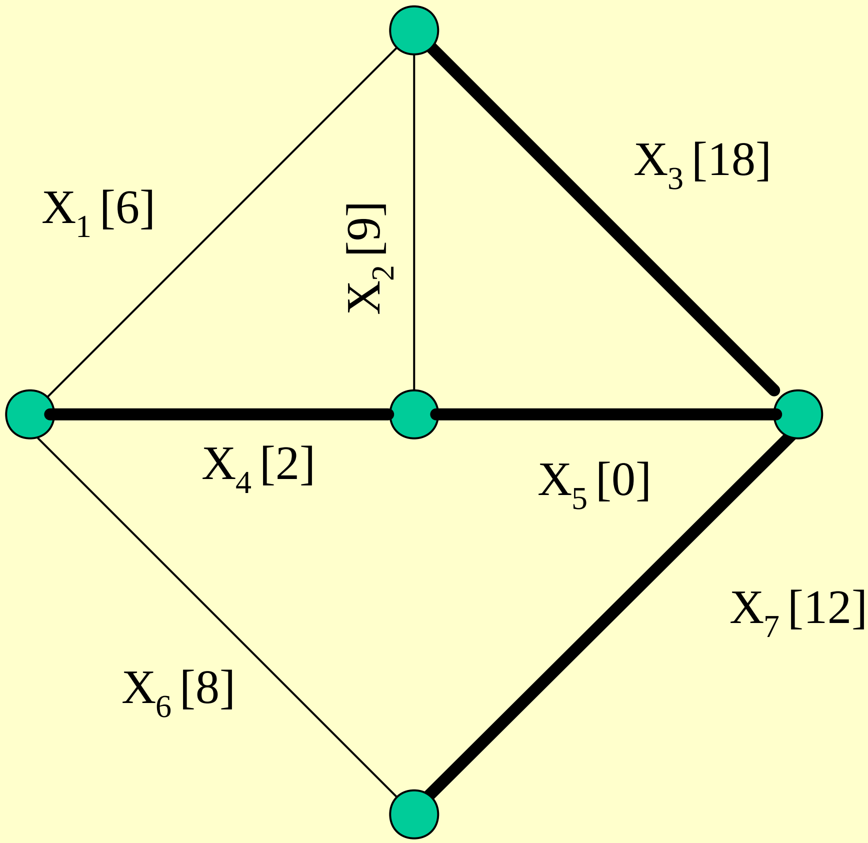
Drop x_6 Add x_7 F=32

<u>Best Choice</u>

Drop x_6 Add x_7 F=32

A Simple Illustration of Tabu Search

•Current Cost 32 Tabu List: x_3 M=[0 0 2 0 0 0 2]



<u>Search neighborhood</u>

Drop x_3 Add x_2 F=23 Tabu

Drop x_3 Add x_1 F=70 Tabu

Drop x_4 Add x_6 F=38

Drop x_4 Add x_1 F=36

Drop x_5 Add x_6 F=40

<u>Best Choice</u>

Drop x_3 Add x_2 F=32

Aspiration Criteria Overrides Tabu Status

A Simple Illustration of Tabu Search

- Final Cost 23 Tabu List: x_3 M=[0 2 0 0 0 0 0]