Tabu Search Foundation: Short Term Memory

Memory and Tabu Classification

Short term and longer term memory: each type of memory is accompanied by its own special strategies

The effect of both types of memory may be viewed as modifying the neighborhood N(x) of current solution x to modified neighborhood N*(x)

Memory and Tabu Classification

In Short term memory

- N*(x) is a subset of N(x)
- Tabu classification identifies elements of N(x) excluded from N*(x), e.g.,N*(x) = N(x)\T, T: tabu list
- TS may allow a solution x to be visited more than once, but the corresponding reduced N*(x) will be different each time around

In longer term memory

- N*(x) is expanded to include solutions not ordinarily found in N(x)
- TS expands N(x) according to the history of the search: Not static
- Making choices that repeatedly visit only a limited subset of x is almost nonexistent

Recency-Based Memory

The most commonly used short term memory keeps track of solutions attributes that have changed during the recent past

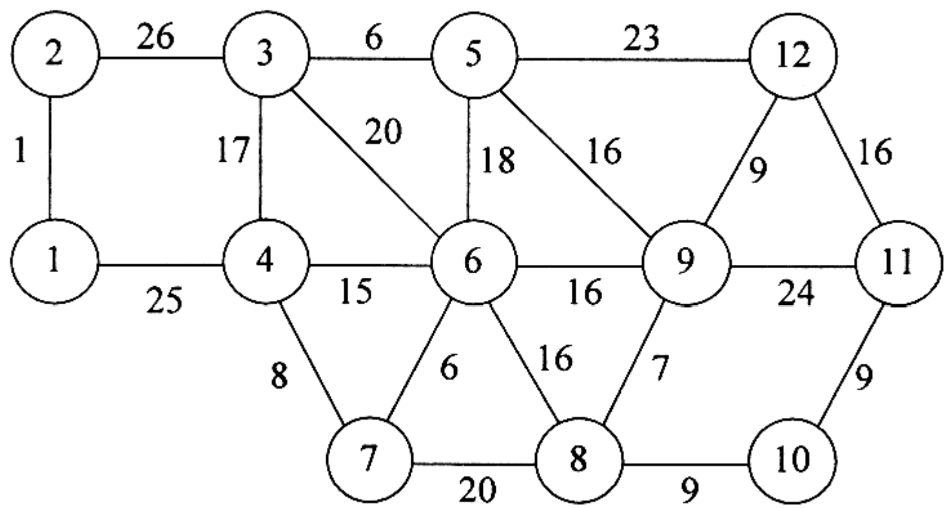
Selected attributes that occur in solutions recently visited are labeled **tabu-active**

This prevents certain recent solutions from belong to N*(x) and hence from being revisited

Recency-Based Memory

Example: *Minimum k-Tree Problem*

Fig. 2.3 Weighted undirected graph.



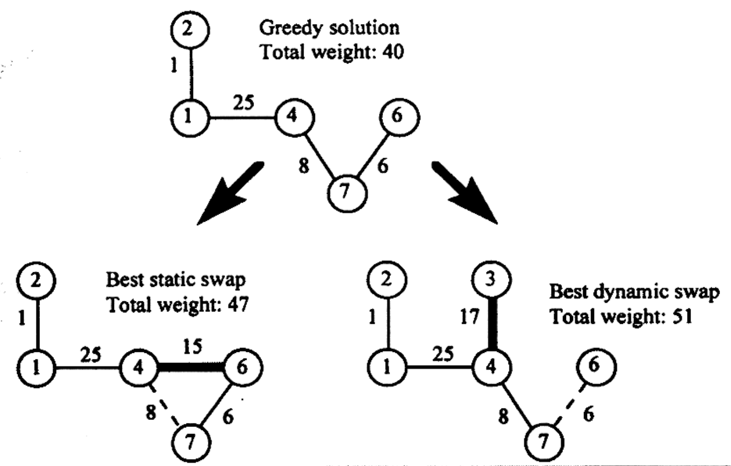
Recency-Based Memory

Move: edge-swapping; static swap, dynamic swap

Table 2.2 Greedy construction.

Step	Candidates	Selection	Total Weight
1	(1,2)	(1,2)	1
2	(1,4), (2,3)	(1,4)	26
3	(2,3), (3,4), (4,6), (4,7)	(4,7)	34
4	(2,3), (3,4), (4,6), (6,7), (7,8)	(6,7)	40

Fig. 2.4 Swap move types.



Recency-Based Memory

Choosing Tabu Classifications

Tabu classifications do not have to be symmetric

Tabu structure can be designed to treat added and dropped elements differently

In static swap of Fig 2.4,

Classify both added and dropped edges tabu-active for the same number of iterations - Symmetric tabu classification

Implement a tabu structure that keeps a recently dropped edge tabu-active for a longer period of time than a recently added edge (There are many more edges outside the tree than in the tree) - Asymmetric tabu classification

Recency-Based Memory

Illustrative Tabu Classifications for the Min K-Tree Problem

Added edges are tabu-active for one iteration, dropped edges are tabu-active for two iterations

Improved-best aspiration criterion is applied

Recency-Based Memory

Fig. 2.3 Weighted undirected graph.

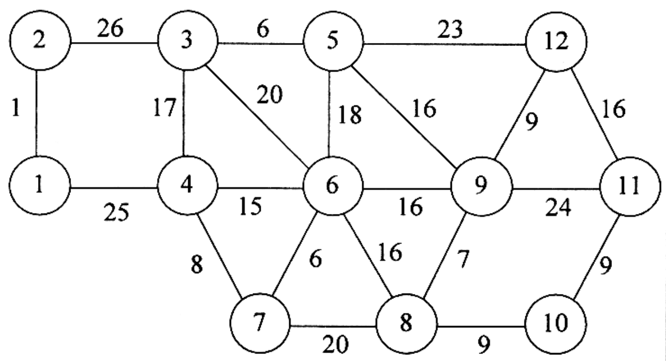
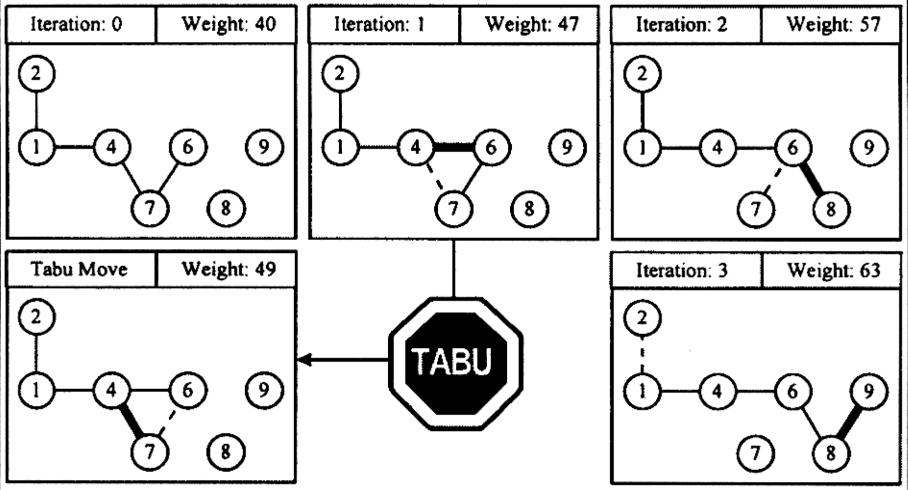


Table 2.3 TS iterations.

Iteration	Tabu-active net tenure		Add	Drop	Weight
	1	2			
1			(4,6)	(4,7)	47
2	(4,6)	(4,7)	(6,8)	(6,7)	57
3	(6,8), (4,7)	(6,7)	(8,9)	(1,2)	63

Fig. 2.5 Effects of attributive short term memory.



A First Level Tabu Search A

Fig. 2.3 Weighted undirected graph.

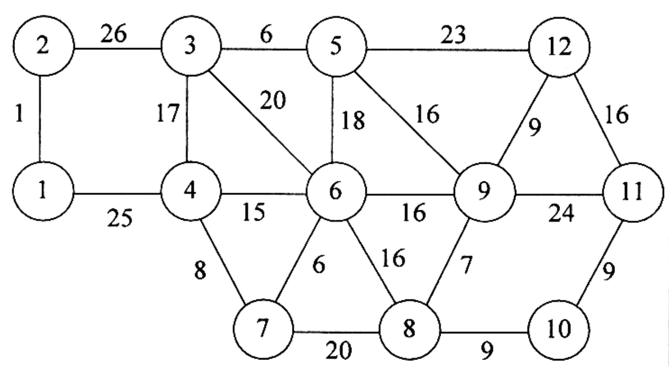


Table 2.4 Iterations of a first level TS procedure.						
Iteration	Tabu-active net tenure		Add	Drop	Move Value	Weight
	1	2				
3	(6,8), (4,7)	(6,7)	(8,9)	(1,2)	8	63
4	(6,7), (8,9)	(1,2)	(4,7)	(1,4)	-17	46
5	(1,2), (4,7)	(1,4)	(6,7)	(4,6)	-9	37*
6	(1,4), (6,7)	(4,6)	(6,9)	(6,8)	0	37
7	(4,6), (6,9)	(6,8)	(8,10)	(4,7)	1	38
8	(6,8), (8,10)	(4,7)	(9,12)	(6,7)	3	41
9	(4,7), (9,12)	(6,7)	(10,11)	(6,9)	-7	34*
10	(6,7), (10,11)	(6,9)	(5,9)	(9,12)	7	41

A First Level Tabu Search A

Fig. 2.3 Weighted undirected graph.

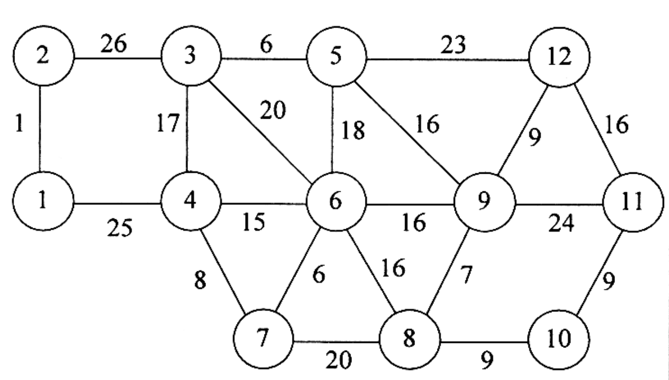
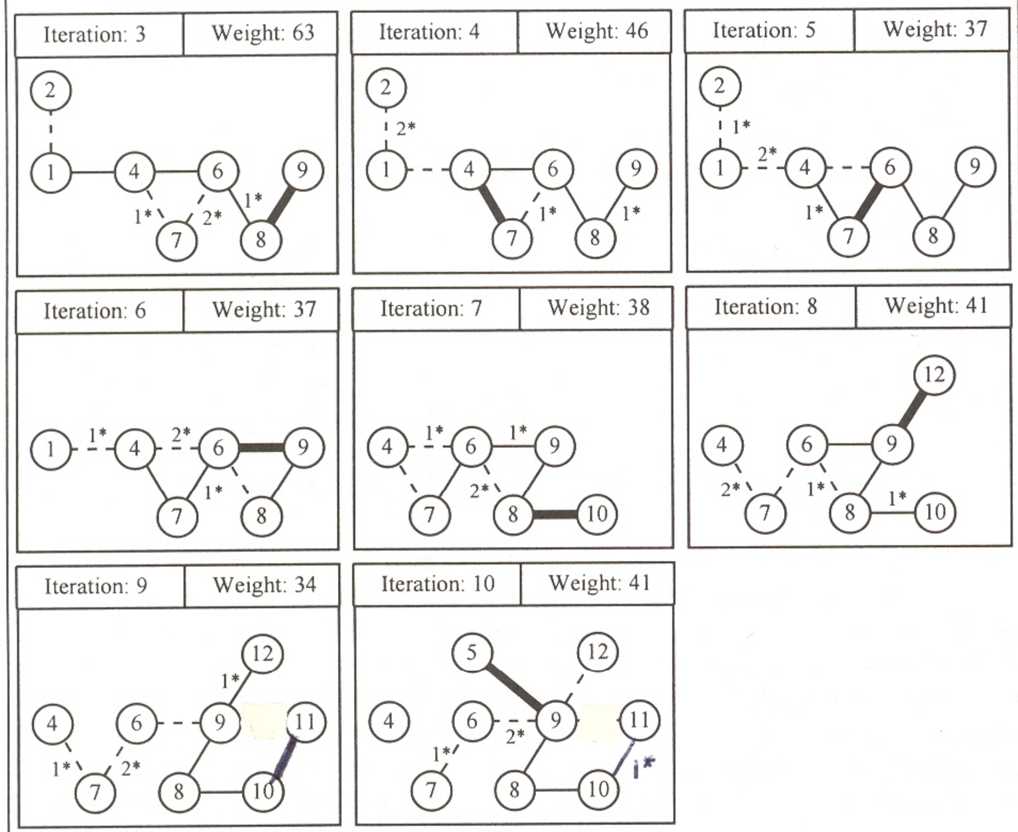
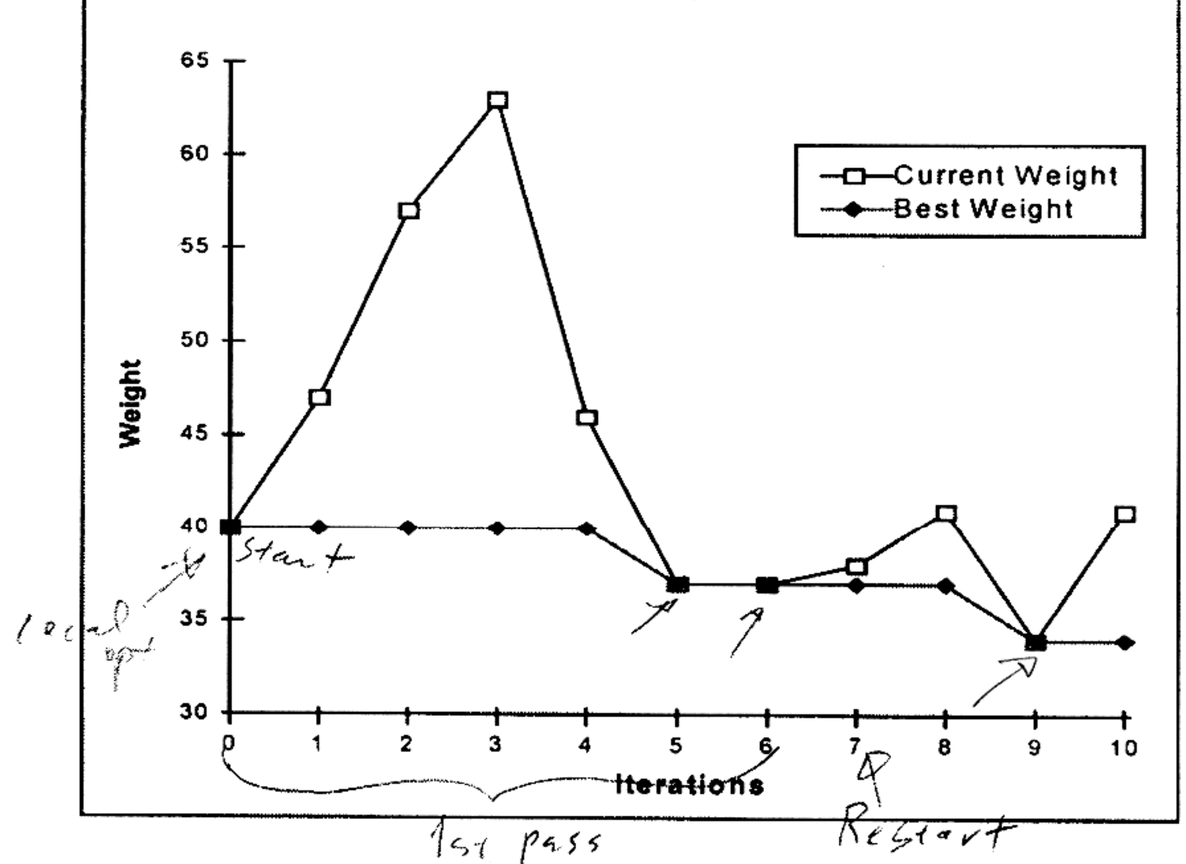


Fig. 2.6 Graphical representation of TS iterations.



A First Level Tabu Search Approach

Fig. 2.7 TS search trajectory.



A First Level Tabu Search Approach

Critical Event Memory for Restarting Procedures

To generate a new starting solution, a critical event that is clearly relevant is the generation of the previous starting solution

New starting solutions has to differ not only from preceding starting solutions, but also from other solutions generated from previous passes

In Fig. 2.7, four solutions may be qualified as critical: the starting solution and three local TS optima

To execute a restarting procedure, one may penalize the inclusion of the edges in the critical solutions at early steps: Edges in the solution of iteration 0, 5, 6 are penalized for two steps in Table 2.5

A First Level Tabu Search Approach

penalized

Table 2.5 Restarting procedure.			
Step	Candidates	Selection	Total Weight
1	(3,5)	(3, 5)	6
2	(2,3), (3,4), (3,6), (5,6), (5,9), (5,12)	(5, 9)	22
3	(2,3), (3,4), (3,6), (5,6), (5,12), (6,9), (8,9), (9,12)	(8, 9)	29
4	(2,3), (3,4), (3,6), (5,6), (5,12), (6,8), (6,9), (7,8), (8,10), (9,12)	(8, 10)	38

edges connected to 3&5 nodes

3,5&9

3,5,8&9

penalize the edges in subgraph

no penalty

Table 2.6 TS iterations following restarting.						
Iteration	Tabu-active net tenure		Add	Drop	Move Value	Weight
	1	2				
1			(9,12)	(3,5)	3	41
2	(9,12)	(3,5)	(10,11)	(5,9)	-7	34*
3	(3,5), (10,11)	(5,9)	(6,8)	(9,12)	7	41
4	(5,9), (6,8)	(9,12)	(6,7)	(10,11)	-3	38
5	(9,12), (6,7)	(10,11)	(4,7)	(8,10)	-1	37

A First Level Tabu Search A

Fig. 2.3 Weighted undirected graph.

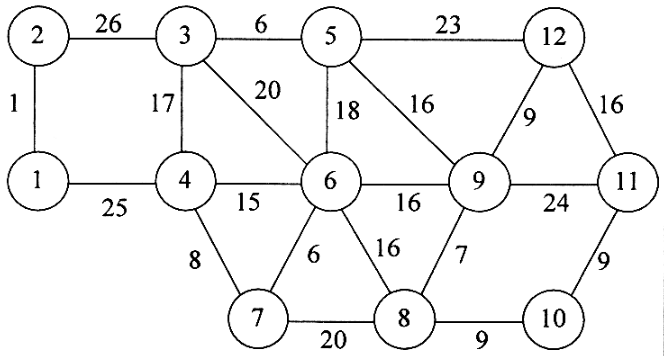
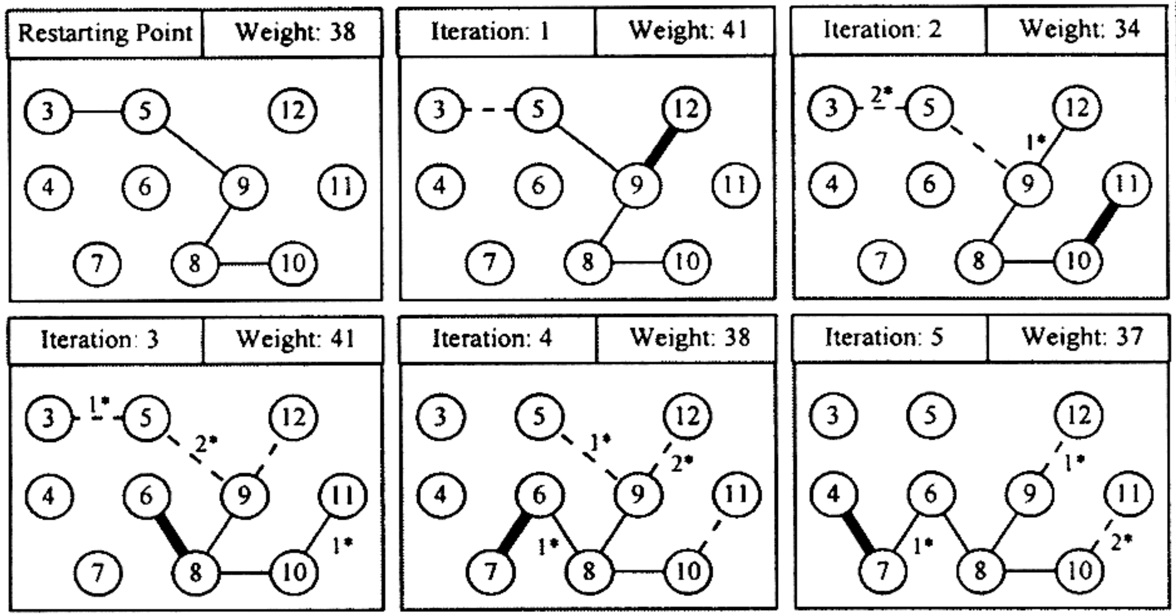


Fig. 2.8 Graphical representation of TS iterations after restarting.



Recency-Based Memory for Add/Drop

Useful notation

TabuStart(Added)=Iter

TabuStart(Dropped)=Iter

TabuEnd(Added)=Iter+TabuDropTenure

TabuEnd(Dropped)=Iter+TabuAddTenure

TestAdd is tabu-active when:

$Iter \leq TabuEnd(TestAdd)$

TestDrop is tabu-active when:

$Iter \leq TabuEnd(TestDrop)$

Tabu Tenure

In general, recency-based memory is managed by creating one or several tabu lists, which record the tabu-active attributes and identify their current status

Tabu tenure can vary for different types of attributes and can also vary over different intervals of time of the search

It is advantageous to record the iteration number that identifies when the tabu-active status of an attribute starts or ends

Effective tabu tenures have been empirically shown to depend on the size of the problem instance

Tabu Tenure

An appropriate tabu tenure depends on the strength of the tabu activation rule employed (more restrictive rules are generally coupled with shorter tenures)

Varying the tabu tenure during the search provides one way to induce a balance between closely examining one region and moving to different parts of the solution space - ***intensification* and *diversification***

Tabu Tenure: Dynamic Tabu Tenure

Random Dynamic Tenure:

The tabu tenure t is randomly selected within a range [t_{min}, t_{max}], usually following a uniform distribution

1st form: The chosen tenure is maintained constant for αt_{max} iterations, and then a new tenure is selected by the same process

2nd form: Apply a new t for every attribute that becomes tabu at a given iteration

Systematic Dynamic Tenure:

The tenure t alternately increases and decreases according to a specified sequence

Aspiration Criteria and Regional Dependencies

Aspiration criteria

- Aspiration by the improved-best
- Aspiration by default
  - If all available moves are classified tabu, a "least tabu" (least penalty) move is selected
- Aspiration by influence

Aspiration Criteria and Regional Dependencies

Influence measures the degree of change induced in solution structure or feasibility

See Fig. 2.9

High influence moves may or may not improve the current solution, though they are less likely to yield an improvement when the current solution is relatively good

But high influence moves are important, especially during intervals of breaking away from local optimality

Aspiration Criteria and Regional Dependencies

Fig. 2.9 Influence level of two moves.