Tabu Search

Glover and Laguna, Tabu search in Pardalos and Resende (eds.), *Handbook of Applied Optimization*, Oxford Academic Press, 2002

Glover and Laguna, Chapter 3 in Reeves, *Modern Heuristic Techniques for Combinatorial Problems*, Wiley, 1993

Background of TS

- TS has its roots in methods that cross boundaries of feasibility and local optimality
- Examples of such methods include use of surrogate constraints and cutting plane approaches
- TS was first proposed by Glover (1986) and was also developed by Hansen (1986)

Basic notions of TS

- The word tabu (or taboo) comes from Tongan, a language of Polynesia, where it indicates things that cannot be touched because they are sacred
- Now it also means “a prohibition imposed by social custom”
- In TS, tabu status of forbidden elements shift according to time and circumstance, based on an evolving memory

Basic notions of TS (cont.)

- Tabu status can be overruled for a preferrable alternative
- Hence TS uses adaptive (flexible) memory
- TS also uses responsive exploration, i.e. **exploitation** of good solutions and **exploration** of new promising regions

Comparison of TS with others

- Traditional descent methos do not allow non-improving moves, TS does
- SA and GA rely on semi-random processes that use sampling, TS is mostly deterministic
- SA and GA do not have explicit memory, TS does
- B&B has rigid memory designs (fixed branching strategy), TS has adaptive memory
- Can a bad strategic choice yield more information than a good random choice? TS claims yes

TS applications

Table 1.1. Illustrative tabu search applications.

Scheduling
- Flow-Time Cell Manufacturing
- Heterogeneous Processor Scheduling
- Workforce Planning
- Classroom Scheduling
- Machine Scheduling
- Flow Shop Scheduling
- Job Shop Scheduling
- Sequencing and Batching

Design
- Computer-Aided Design
- Fault Tolerant Networks
- Transport Network Design
- Architectural Space Planning
- Diagram Coherency
- Fixed Charge Network Design
- Irregular Cutting Problems

Location and Allocation
- Multicommodity Location/Allocation
- Quadratic Assignment
- Quadratic Semi-Assignment
- Multilevel Generalized Assignment
- Lay-Out Planning
- Off-Shore Oil Exploration

Logic and Artificial Intelligence
- Maximum Satisfiability
- Probabilistic Logic
- Clustering
- Pattern Recognition/Classification
- Data Integrity
- Neural Network |Training and Design

Technology
- Seismic Inversion
- Electrical Power Distribution
- Engineering Structural Design
- Minimum Volume Ellipsoids
- Space Station Construction
- Circuit Cell Placement

Telecommunications
- Call Routing
- Bandwidth Packing
- Hub Facility Location
- Path Assignment
- Network Design for Services
- Customer Discount Planning
- Failure Immune Architecture
- Synchronous Optical Networks

Production, Inventory and Investment
- Flexible Manufacturing
- Just-in-Time Production
- Capacitated MRP
- Part Selection
- Multi-item Inventory Planning
- Volume Discount Acquisition
- Fixed Mix Investment

Routing
- Vehicle Routing
- Capacitated Routing
- Time Window Routing
- Multi-Mode Routing
- Mixed Fleet Routing
- Traveling Salesman
- Traveling Purchaser

Graph Optimization
- Graph Partitioning
- Graph Coloring
- Clique Partitioning
- Maximum Clique Problems
- Maximum Planner Graphs
- P-Median Problems

General Combinational Optimization
- Zero-One Programming
- Fixed Charge Optimization
- Nonconvex Nonlinear Programming
- All-or-None Networks
- Bilevel Programming
- General Mixed Integer Optimization

Four dimensions of TS memory

- Recency based (short term) memory
- Frequency based (long term) memory
- Quality: ability to differenciate the merit of solutions
  - Use memory to identify elements common to good solutions
  - Reinforce (discourage) actions that lead to good (bad) solutions
- Influence: impact of the choices made in search on both quality and structure of solutions

Use of memory in TS

- Use of memory leads to learning
- Memory in TS is explicit and attributive
- Solution attributes (or elements or components) that change in moving from one solution to another are recorded for guiding the search
- Attributes can be nodes or arcs repositioned in a graph or job indeces in scheduling
- In addition to recency or frequency of solution attributes, elite solutions and their attractive neighbors are explicitely recorded

Intensification and diversification

- Intensification: a form of exploitation
  - Based on modifying choice rules to encourage good move combinations and solution attributes
  - May lead to return to attractive regions
  - Examines neighbors of prerecorded elite solutions
- Diversification: a form of exploration
  - Examines unvisited regions, generates different solutions

Problem definition

- Suppose we have the (conceptual) problem

  min or max $f(x)$

  subject to $x \in X$

  where X is the set of constraints
- Both $f(x)$ and the constraints can be nonlinear
- *Neither has to be explicit mathematical formulations*

Neighborhood search in TS

- Let $N(x^{now}) \subset X$ be the neighborhood of x^{now}
- Start by moving from x^{now} to $x^{next} \in N(x^{now})$
- Repeat this a number of times until a termination condition is satisfied
- Choose x^{next} in a systematic manner rather than randomly
- May use a candidate list (as in candidate subgraph for TSP) to narrow down search in $N(x^{now})$

Dynamic neighborhood

- Use recency based (short term) memory to obtain $N^*(x) \subset N(x)$ by eliminating (making tabu) the recently visited solutions in order to avoid cycling
- Use frequency based (long term) memory and elite solutions to obtain $N^*(x) \supset N(x)$ in order to expand the neighborhood and examine unvisited regions

Recency based memory

- Recency based memory records solution attributes (or elements or components) that have changed "recently"
- Selected attributes in recently visited solutions become tabu-active during their tabu tenures
- Solutions containing these attributes are classified as tabu
- Tabu solutions are excluded from $N^*(x)$ and not revisited during the tabu tenure (a certain period of time = certain number of moves)

Aspiration criteria

- Improved-best or best solution criterion: If a tabu solution encountered at the current iteration is better than the best solution found so far, then its tabu status is overridden
- Other aspiration criteria are possible, e.g. setting the tabu tenure shorter for better solutions

Example 1: Ordering of modules

- Problem definition: Find the ordering of modules (filters) that maximizes the overall insulating property of the composite material
- Representation of a solution for 7 modules:

2	5	7	3	4	6	1

- Neighborhood structure: swapping modules

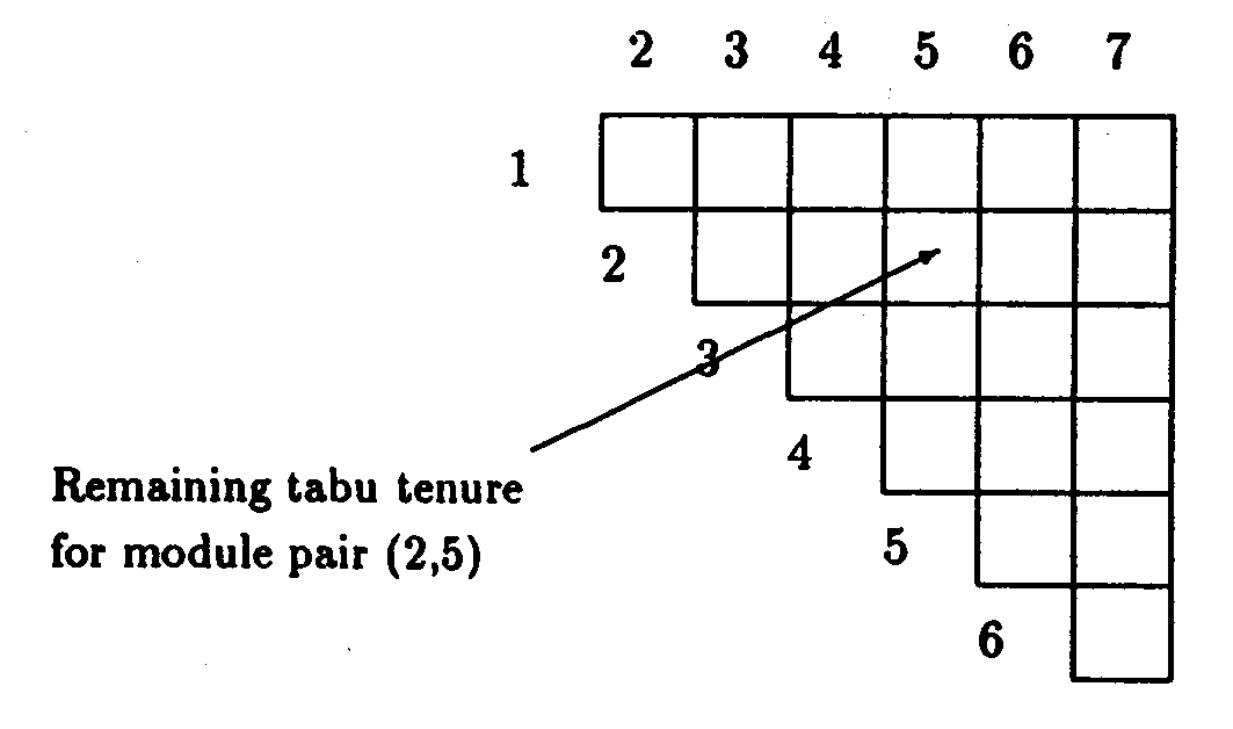
2	6	7	3	4	5	1

- A solution has 21 neighbors (7 choose 2)

Example 1: Recency based memory and tabu classification

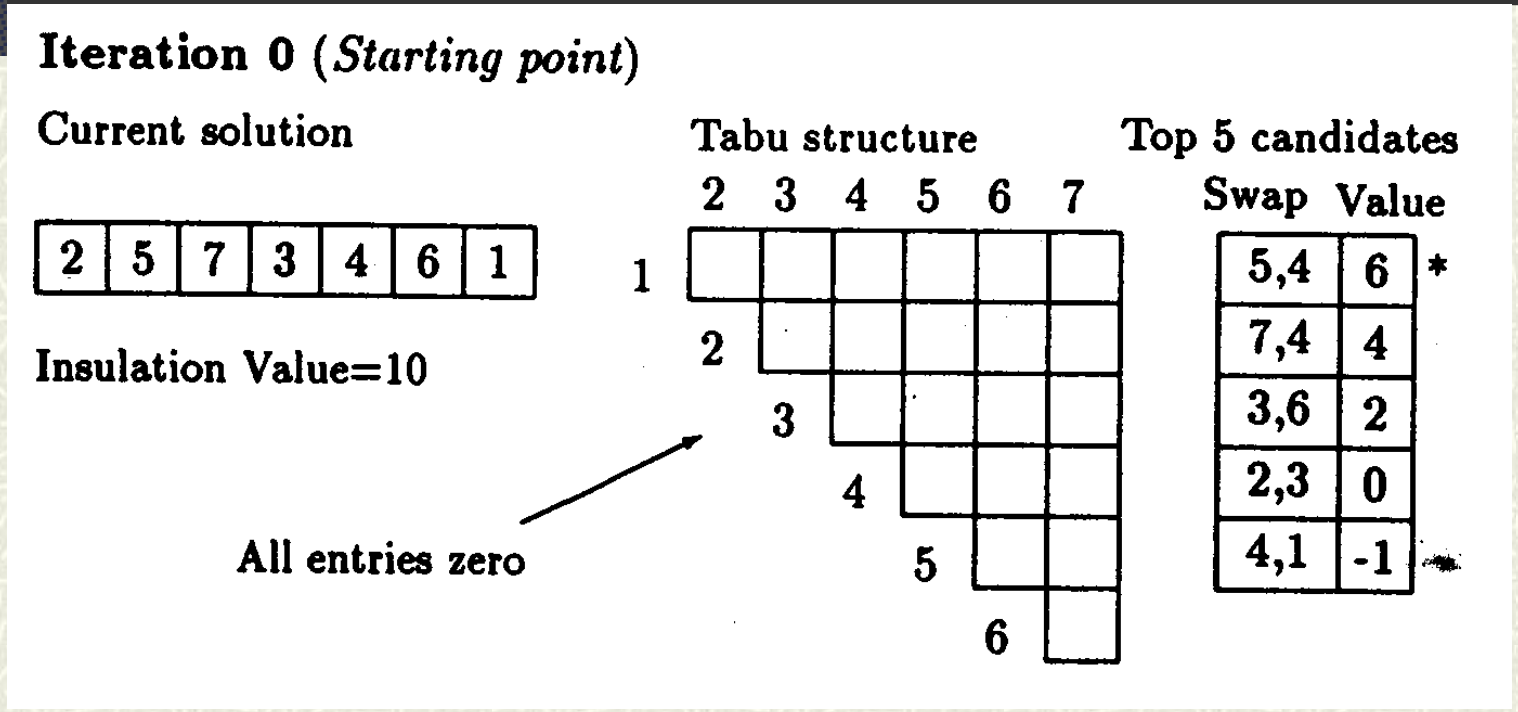
- Tabu attributes are selected as most recently made swaps
- Tabu tenure is set as 3 iterations
- Hence, solutions involving 3 most recent swaps will be classified as tabu
- Aspiration criterion is chosen as best solution

Example 1: Initialization of recency based memory



An upper triangle is enough

Example 1: Iteration 0



Top 5 candidates constitute the candidate list

"Value" is the gain of swap

Example 1: Iteration 1

Iteration 1

Current solution

2	4	7	3	5	6	1

Insulation Value=16

Tabu structure

	2	3	4	5	6	7
1						
2						
3						
4			3			
5						
6						

Top 5 candidates

Swap	Value	
3,1	2	*
2,3	1	
3,6	-1	
7,1	-2	
6,1	-4	

Move (4,5) has now a tabu tenure of 3 iterations

Example 1: Iteration 2

Iteration 2

Current solution

2	4	7	1	5	6	3

Insulation Value=18

Tabu structure

	2	3	4	5	6	7
1		3				
2						
3						
4			2			
5						
6						

Top 5 candidates

Swap	Value	
1,3	-2	T
2,4	-4	*
7,6	-6	
4,5	-7	T
5,3	-9	

Moves (1,3) and (4,5) have respective tabu tenures 3 and 2

No move with a positive gain, hence best (non-tabu) move will be non-improving

Example 1: Iteration 3

Iteration 3

Current solution

4	2	7	1	5	6	3

Insulation Value=14

Tabu structure

	2	3	4	5	6	7
1		2				
2			3			
3						
4				1		
5						
6						

Top 5 candidates

Swap	Value	
4,5	6	T*
5,3	2	
7,1	0	
1,3	-3	T
2,6	-6	

Move (4,5) has a tabu tenure of 1 iteration

But this move results in the best solution so far

Hence its tabu status is overridden

Example 1: Iteration 4

Iteration 4

Current solution

5	2	7	1	4	6	3

Insulation Value=20

Tabu structure

	2	3	4	5	6	7
1		1				
2			2			
3						
4				3		
5						
6						

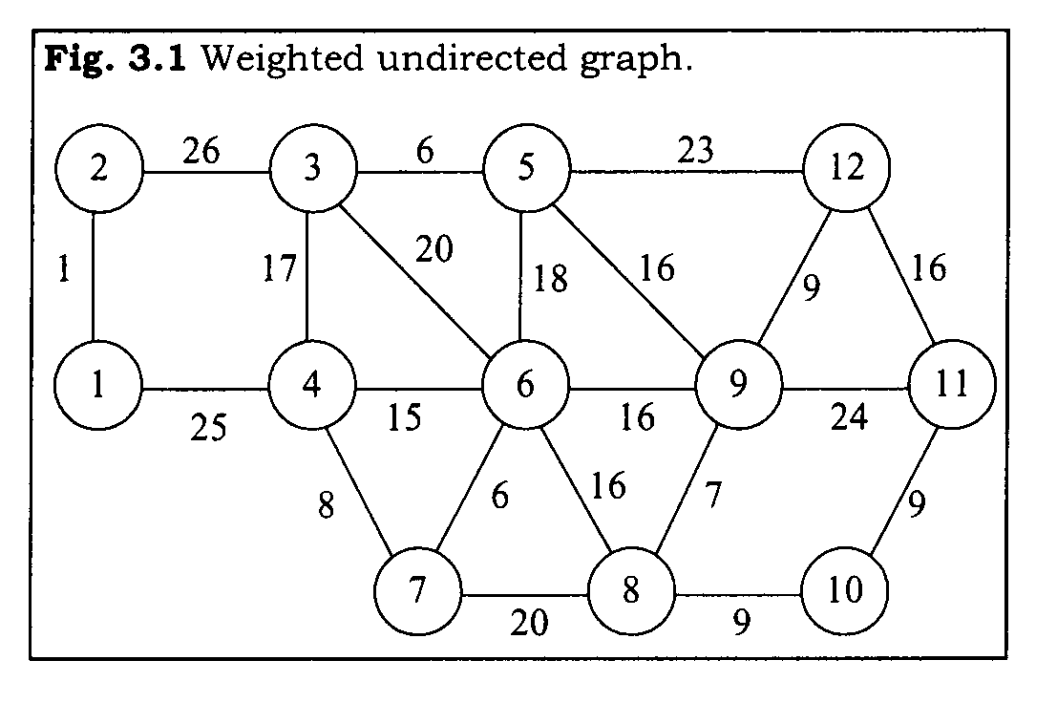
Top 5 candidates

Swap	Value	
7,1	0	*
4,3	-3	
6,3	-5	
5,4	-6	T
2,6	-8	

Best move is (1,7)

Example 2: Minimum *k*-tree

- Problem definition: Find the tree consisting of k-edges in a graph such that the sum of the edge weights is minimum



Example 2: Greedy construction of initial k-tree (k=4)

Table 3.1 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

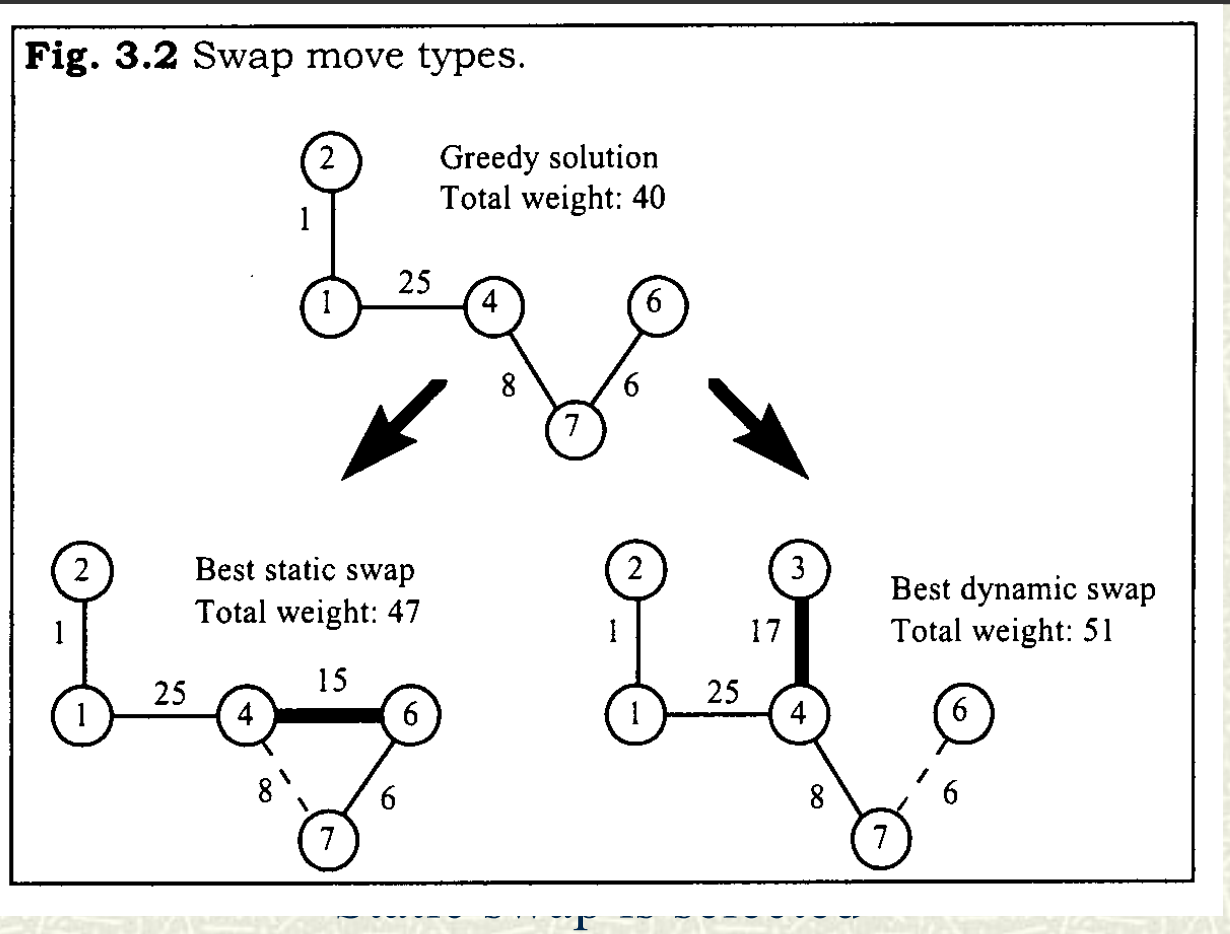
Start with the edge having the minimum weight

Choose remaining k-1 edges successively to minimize the increase in total weight in each step

Example 2: Neighborhood structure

- Basic move is edge swapping
- Consider all feasible swaps that result in a tree and that are not tabu (in general, infeasible moves may be allowed occasionally)
- Choose the move yielding the minimum total weight
- A static swap keeps current nodes, a dynamic one allows change of nodes

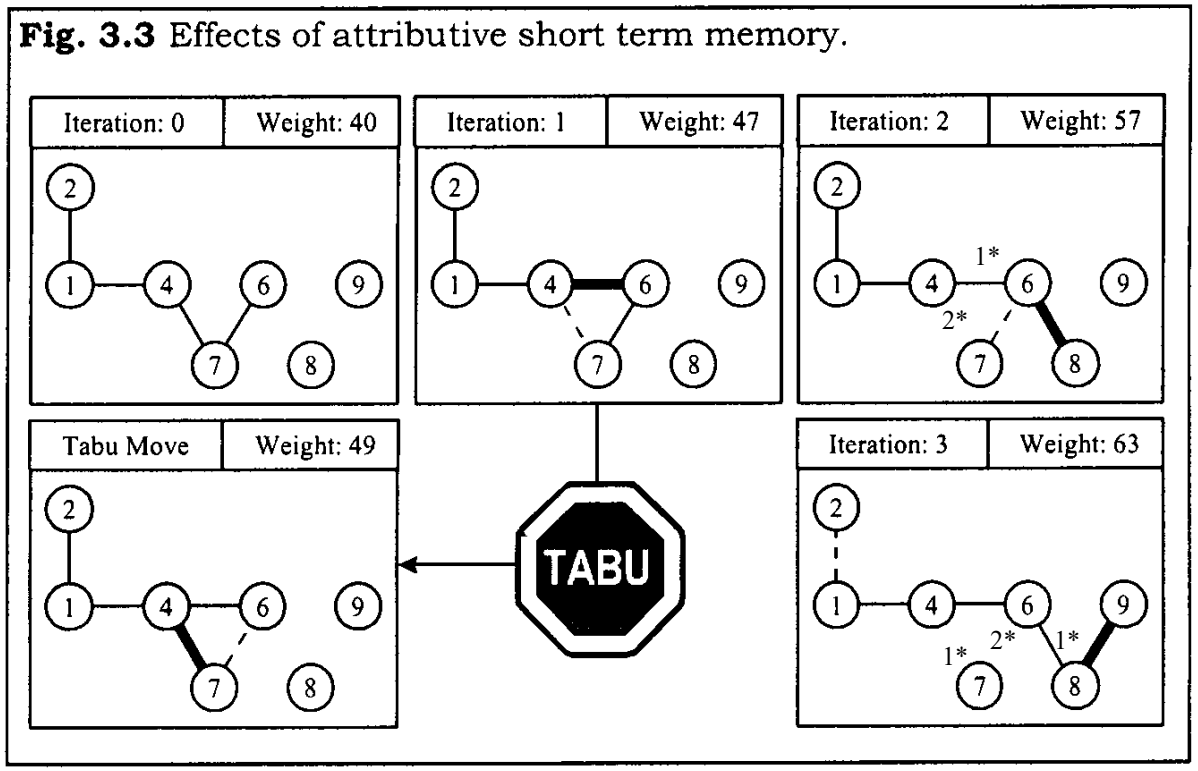
Example 2: Neighborhood structure (cont.)



Example 2: Tabu classification

- Tabu attributes are edges swapped
- Out-edge is {4,7}, in-edge is {4,6} in Figure 3.2
- Tabu tenure for in-edges (bounded by *k*) is set as 1
- Tabu tenure for out-edges is chosen as 2 (since there are more outside edges)
- A swap move is classified as tabu if either the out-edge or the in-edge involved is tabu-active (an alternative rule could have been "if both are tabu-active")

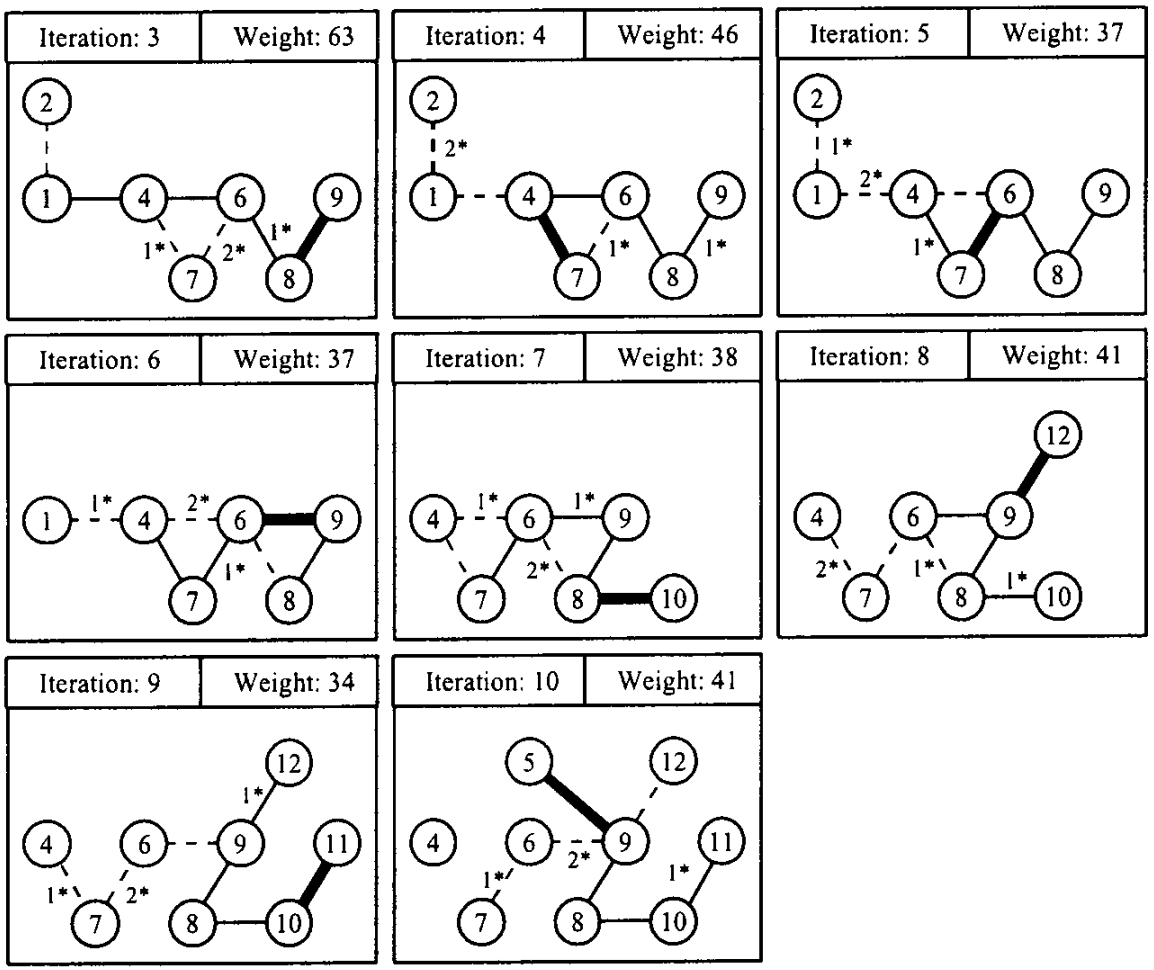
Example 2: Tabu classification may prevent visiting unexamined solutions



In iterat function value 49 cannot be visited since {4,7} is tabu-active, and we end up with an inferior solution

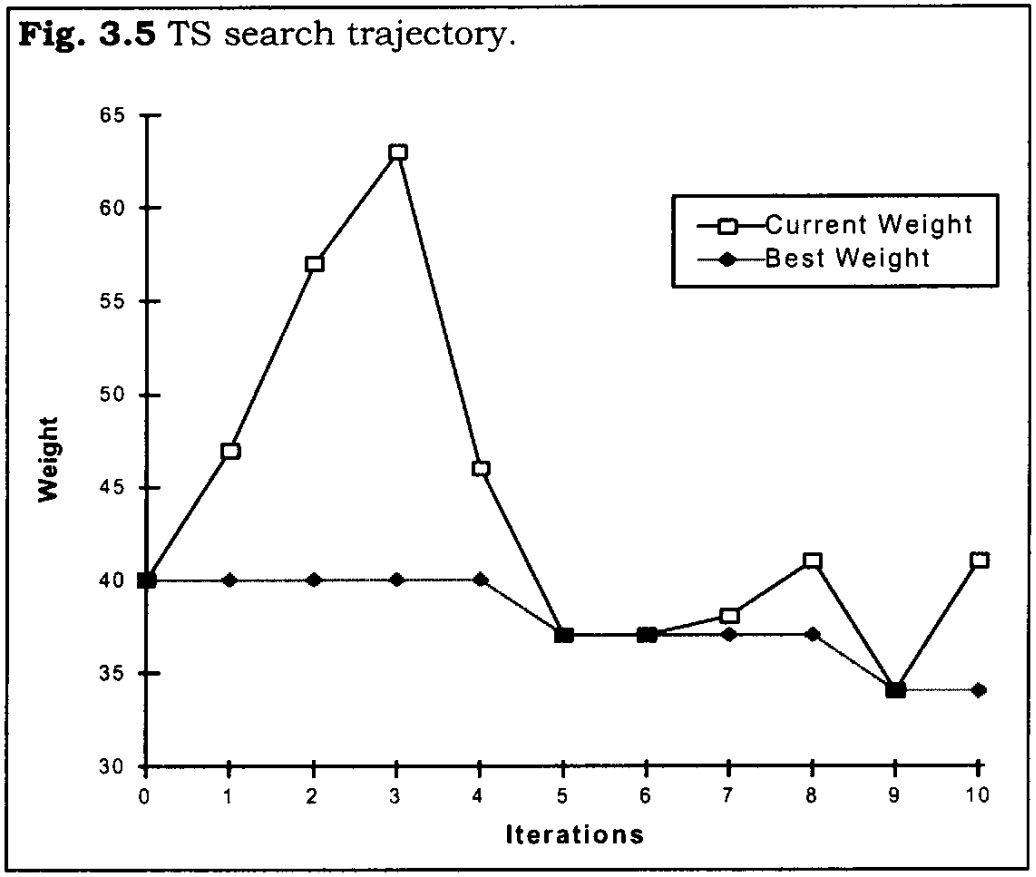
Example 2: Additional iterations

Fig. 3.4 Graphical representation of TS iterations.



34 is the optimal solution

Example 2: TS search trajectory



Note the local optima in iterations 3 and 6

Diversification or restart in TS

- We may need to diversify or restart TS when, for example, no admissible improving moves exist or rate of finding new best solutions drops
- Instead of choosing the restarting point randomly, TS employs diversification strategies based on:
  - Frequency based memory: records frequently used attributes or visited solutions
  - Critical event memory: an aggregate summary of critical events (local optima or elite solutons) during the search

Example 1: Diversification using frequency based memory

Iteration 26

Current solution

1	3	6	2	7	5	4

Insulation Value=12

Tabu structure (Recency)

	1	2	3	4	5	6	7
1				3			
2							
3	3					2	
4	1	5					1
5		4		4			
6			1		2		
7	2			3			

(Frequency)

Top 5 candidates

Swap	Value	Penalized Value	
1,4	3	3	T
2,4	-1	-6	
3,7	-3	-3	*
1,6	-5	-5	
6,5	-4	-6	

Diversify when no admissible improving moves exist

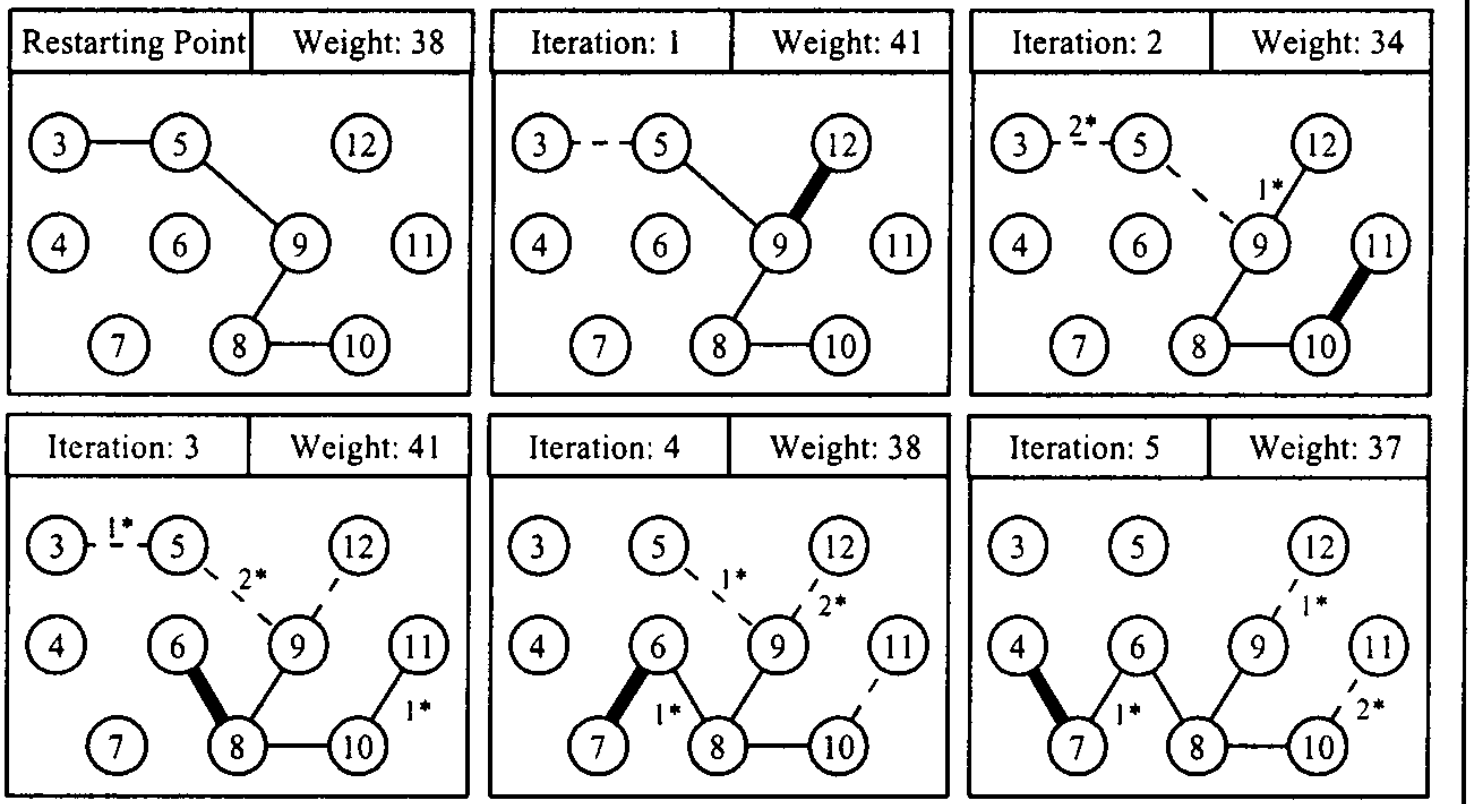
Penalize non-improving moves by assigning larger penalty to more frequent swaps, choose (3,7) using penalized value

Example 2: Diversification using critical event memory

- Summary of the starting solution and local optima (elite solutions) found in previous start(s)
- Assuming we choose to restart after iteration 7, these are the starting solution (40), and solutions found in iterations 5 (37) and 6 (37)
- In finding the restarting solution, penalize the use of edges in these solutions for diversification
- May use frequency based memory (frequency of appearance of these edges) for weighing the edges

Example 2: Restart iterations

Fig. 3.6 Graphical representation of TS iterations after restarting.



A TS Algorithm

1. Find an initial solution $x_0 \in X$, set $x^{now} = x^{best} = x_0$; initialize memory
2. Intensification phase:
   - 2.1 If termination condition (e.g. simple iteration count, no admissible improving move, no change in x^{best} in so many iterations) is satisfied, then go to step 3
   - 2.2 Choose $x^{next} \in N(x^{now})$ such that x^{next} is not tabu or satisfies aspiration criterion
   - 2.3 Move from x^{now} to x^{next}, i.e. set $x^{now} = x^{next}$

A TS algorithm (cont.)

2.4 If x^{now} is better than x^{best}, then set $x^{best} = x^{now}$

2.5 Update recency based memory (tabu classifications), frequency based memory and/or critical event memory (elite solutions), return to step 2.1

3. Diversification phase:

3.1 If termination condition is satisfied, then stop

3.2 Using frequency based memory and/or critical event memory, find a new starting point x^{now}, return to step 2

TS decisions

- Neighborhood structure (basic move, candidate list)
- Recency based memory
- Intensification strategy
  - Tabu attribute(s)
  - Tabu tenure(s)
  - Tabu classification of a solution (as a function of tabu attributes)

TS decisions (cont.)

- Aspiration criterion
- Frequency based memory and/or critical event memory (summary of elite solutions)
- Diversification strategy
- Termination conditions for intensification and diversification phases

Tabu Search (cont.)

Glover and Laguna, Chapter 3 in Reeves, *Modern Heuristic Techniques for Combinatorial Problems*, Wiley, 1993

Section 3.2.5 and on

TS Algorithm

1. Find an initial solution $x_0 \in X$, set $x^{now} = x^{best} = x_0$, initialize memory
2. Intensification phase:
   - 2.1 If termination condition (e.g. simple iteration count, no admissible improving move, no change in x^{best} in so many iterations) is satisfied, then go to step 3
   - 2.2 Choose $x^{next} \in N(x^{now})$ such that x^{next} is not tabu or satisfies aspiration criterion
   - 2.3 Move from x^{now} to x^{next}, i.e. set $x^{now} = x^{next}$

TS algorithm (cont.)

2.4 If x^{now} is better than x^{best}, then set $x^{best} = x^{now}$

2.5 Update recency based memory (tabu classifications), frequency based memory and/or critical event memory (elite solutions), return to step 2.1

3. Diversification phase:

3.1 If termination condition is satisfied, then stop

3.2 Using frequency based memory and/or critical event memory, find a new starting point x^{now}, return to step 2

Move attributes

Illustrative Move Attributes for a Move x^{now} to x^{trial}

(A1)	Change of a selected variable x_j from 0 to 1.
(A2)	Change of a selected variable x_k from 1 to 0.
(A3)	The combined change of (A1) and (A2) taken together.
(A4)	Change of $c(x^{now})$ to $c(x^{trial})$.
(A5)	Change of a function $g(x^{now})$ to $g(x^{trial})$ (where g may represent a function that occurs naturally in the problem formulation or that is created strategically).
(A6)	Change represented by the difference value $g(x^{trial}) - g(x^{now})$.
(A7)	The combined changes of (A5) or (A6) for more than one function g considered simultaneously.

Move attributes (cont.)

- x^{trial} is the potential x^{next} we are considering
- Setting a solution variable x_j to 1/0 may mean adding/deleting a node or an edge to/from a graph, or assigning/removing a facility to/from a location
- A move can be devised based on multiple solution attributes as in (A3)
- Function g may be completely independent of objective function c, e.g. the distance (dissimilarity) between x^{trial} and x^{best}, or x^{trial} and the last local optimum visited

Move attributes (cont.)

- In some cases, move attributes can be classified as from-attributes and to-attributes, i.e.

  from-attribute $\in A(x^{now}) - A(x^{trial})$

  to-attribute $\in A(x^{trial}) - A(x^{now})$

  where A is the set of attributes of a solution
- For example, if $x_{ij}=1$ indicates that facility i is assigned to location j, then in the move setting $x_{ij}=0$ and $x_{ik}=1$, x_{ij} is the from-attribute and x_{ik} is the to-attribute

Tabu restrictions

Illustrative Tabu Restrictions

A move is tabu if:

(R1) x_j changes from 1 to 0 (where x_j previously changed from 0 to 1).

(R2) x_k changes from 0 to 1 (where x_k previously changed from 1 to 0).

(R3) at least one of (R1) and (R2) occur. (This condition is more restrictive than either (R1) or (R2) separately—i.e. it makes more moves tabu.)

(R4) both (R1) and (R2) occur. (This condition is less restrictive than either (R1) or (R2) separately—i.e. it makes fewer moves tabu.)

(R5) both (R1) and (R2) occur, and in addition the reverse of these moves occurred simultaneously on the same iteration in the past. (This condition is less restrictive than (R4).)

(R6) $g(x)$ receives a value v' that it received on a previous iteration (i.e. $v' = g(x')$ for some previously visited solution x').

(R7) $g(x)$ changes from v'' to v', where $g(x)$ changed from v' to v'' on a previous iteration (i.e. $v' = g(x')$ and $v'' = g(x'')$ for some pair of solutions x' and x'' previously visited in sequence.)

Tabu restrictions (cont.)

- Tabu restrictions are used to avoid reversals or repetitions (cycling)
- As long as at least one to-attribute of a current move is different from a from-attribute of a previous move, cycing cannot occur
- Revisiting a solution encountered before may still be possible in the long run
- Tabu tenures may be based on recency or frequency

Recency based tabu restrictions

- Different attributes may have different tabu tenures, e.g. in-edges and out-edges in the *k*-tree problem or TSP
- Tabu tenures can be static, e.g. a constant or $\sqrt{n}$
- Tabu tenures can also be set dynamically within a range, e.g. they may vary between

$$t_{\min} = 0.9\sqrt{n} \text{ and } t_{\max} = 1.1\sqrt{n}$$

Recency based tabu restrictions (cont.)

- Tabu tenures can be dynamic depending on both type and quality of an attribute (keep a good attribute longer)
- Experience shows that dynamic rules are more robust than static rules
- Attribute-dependent dynamic rules have proved effective for difficult problems (scheduling and routing)

Aspiration criteria

- Aspiration criteria are based on influence, which measures the degree of change in solution structure, quality, or feasibility
- Influence is often associated with move distance
- High-influence moves must be given priority in the absence of improving moves (indication of getting stuck in local optima)

Types of aspiration criteria

- There are two types of aspirations
  - Move aspiration: revokes solution's tabu classification
  - Attribute aspiration: revokes attribute's tabu-active status
- Aspiration by default: If all available moves are tabu, then select the least tabu one
- Aspiration by objective: Revoke tabu classification of x^{trial}, if x^{trial} is better than x^{best}

Types of aspiration criteria (cont.)

- Aspiration by search direction: Revoke tabu-active status of an attribute, if direction indicated by the attribute is improving and x^{trial} is better than x^{now}
- Aspiration by influence: Revoke tabu-active status of an attribute, if it is associated with a low-influence move, and a high-influence move has been performed since it has been tabu-active (hence cycling is unlikely)

Mesures for frequency based memory

In general, a ratio where the numerator is the number of occurences of a particular event

Denominators for Frequency Measures

(D1)	The total number of occurrences of all events represented by the numerators (such as the total number of associated iterations).
(D2)	The sum of the numerators.
(D3)	The maximum numerator value.
(D4)	The average numerator value.

Mesures for frequency based memory (cont.)

The numerator can be the number of times

- a solution has appeared
- a move attribute has appeared
- a from-attribute has appeared
- a to-attribute has appeared

over the solutions visited so far

Types of frequency

- Residence frequency: Number of times a particular value of an attribute resides in the solution; high frequency in good solutions may indicate a highly attractive attribute, or vice versa
- Transition frequency: Number of times an attribute changes from a particular value and/or to a particular value; high frequency may indicate the attribute is in and out of solution for fine tuning

Use of frequency based memory

- Attributes with higher frequency may also become tabu-active just as those having greater recency
- However, frequency based memory is typically used
  - To define penalty and incentives in evaluating moves (as in Example 1)
  - For long term diversification

Intensification vs Diversification

- Intensification encourages incorporation of good attributes
  - Short term: incorporate attributes receiving highest evaluations
  - Long term: incorporate attributes from a small subset of elite solutions easily reachable from each other
- Diversification seeks different solutions
  - Consider, for example, using all local optima distributed across the entire search space

TS algorithm revisited

- Note that intensification and diversification are in fact not entirely separated in TS
- In step 2 (intensification phase), some short term diversification may take place, e.g. penalize frequent moves as in Example 1
- In step 3 (diversification phase), some long term intensification may take place, e.g. assign incentives (instead of penalties) to elite solutions to find a restarting point in Example 2

TS algorithm revisited (cont.)

	Short term	Long term
Intensification	Choosing as x^{next} the best improving move from $N(x^{now})$	See broader aspects of intensification and diversification
Diversification	Tabu restrictions, penalizing frequent moves	

Broader aspects of intensification and diversification

- TS hypotesis: systematic diversification is better than random search
- We are interested in not only a diversified collection of solutions but also a diversified sequence of solutions
- To create a diversified sequence, we can use a metric that maximizes the distance between solutions in terms of from- and to-attributes (solution structure)

Broader aspects of intensification and diversification (cont.)

$Z(k)=\{z(1),z(2),...,z(k)\}$ is a diversified sequence of solutions, relative to distance metric d by requiring each subsequence $Z(h)$ of $Z(k)$, $h<k$, and each associated point $z=z(h+1)$ to satisfy the following hierarchy of conditions:

(A) z maximizes the minimum distance $d(z,z(i))$ for $i \leq h$

(B) subject to (A), z maximizes the minimum distance $d(z,z(i))$ for $1<i \leq h$, then for $2<i \leq h$,..., etc. (in strict priority order in case of ties)

(C) subject to (A) and (B), z maximizes the distance $d(z,z(i))$ for $i=h$, then for $i=h-1$,..., and finally for $i=1$.

Mechanisms of intensification and diversification

- Penalty and incentive functions
- Reinforcement by restriction
- Path relinking
- Variations of path relinking: tunneling and extrapolated relinking
- Solutions evaluated but not visited
- Creating new attributes
- Strategic oscillation

Penalty and incentive functions

- Assign penalties to frequent moves for diversification
- Assign incentives to elite solutions for intensification
- Interval specific penalties and incentives, e.g.
  - Classify move evaluations into improving and nonimproving intervals, assign penalties (incentives) to those in nonimproving (improving) interval
  - Classify move evaluations into high and low influence intervals, reduce or cancel incentives accordingly

Reinforcement by restriction

- If a few attribute values have a high frequency, then explicitely restrict moves to take on those attribute values, i.e. fix certain attributes
- Restricts solution space and simplifies some problems
- Restricting some attributes (intensification) allows better exploration of remaining attributes (diversification) within limited computation time

Path relinking

- Given two selected elite solutions x' and x'', generate a path (a sequence of moves) to reach x'' from x'
- At each step, choose a move that leaves the fewest number of moves remaining to reach x''
- If x' and x'' share more (fewer) attributes, then path relinking serves intensification (diversification)

Variations of path relinking

- Tunneling: Proceed from both x' and x'' (occasionally allow infeasible moves)
- This basically corresponds to exchanging components of two solutions
- Extrapolated relinking: Do not stop when you reach x'' but go beyod it for further diversification

Solutions evaluated but not visited

- Count the number of times an attribute is considered in x^{trial}, even though x^{trial} is not actually visited
- Give incentive to an attribute that has a high count above, but that has a low frequency over actually visited solutions, to be incorporated into future moves
- Here, recency and frequency interact, serving both intensification and diversification simultaneously

Creating new attributes

- Create new attributes by implicitly combining or subdividing existing attributes
- Vocabulary building: Discover attribute combinations (vectors) shared by various solutions, e.g. a subpath in TSP or a partial sequence in scheduling (similar to building sentences from words)
- If necessary fill in the blanks in these vectors to obtain unvisited feasible solutions

Strategic oscillation

- Move until hitting a boundary, e.g. infeasibility
- Instead of stopping there, extend neighborhood definition or modify evaluation criteria for selecting moves, to permit crossing of the boundary
- Proceed for a specified depth beyond the boundary, then turn around and cross the boundary in reverse direction
- This way, explore the boundary of feasibility better, where the global optimum is likely to be found
- SO may also take place at a region searched intensely