# Tabu Search

## — Basic Principle and Algorithm


**Zebo Peng**

Embedded Systems Laboratory (ESLAB)
Linköping University

embedded systems lab linköpings universitet

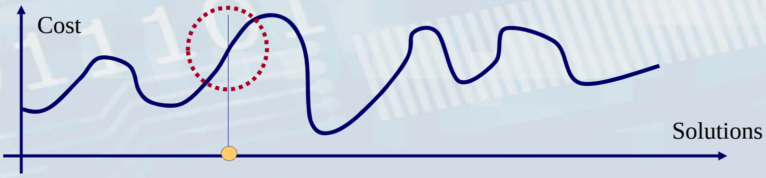
INSTITUTE OF TECHNOLOGY
LINKÖPINGS UNIVERSITET


## Outline

- **Introduction and basic principles**
- **The algorithm**
- **Intensification and diversification**
- **Improvement techniques**

## Introduction

- Tabu search (TS) is a neighborhood search method which employs "intelligent" search and flexible memory technique to avoid being trapped at local optimum.




  - To de-emphasize the use of random selection, in order to speed up the search process.
  - Moves are selected intelligently:
    - best admissible moves are selected;
    - both downhill moves and uphill moves are allowed.
  - Use tabus to restrict the search space and avoid cyclic behavior (trapped at local optimum).
  - The classification of tabus is based on the search history.



## Histrory

- A very simple memory mechanism is described in Glover (1977) to implement the oscillating assignment heuristic for Integer Programming.
- Glover (1986) introduces tabu search as a "meta-heuristic" superimposed on another heuristic.
- Glover (1989) provide a full description of the method.
- Many applications of TB have been reported in the literature since then.

# Main Features

- TS emulates the human problem solving process.
- It takes advantage of search history.
  - The historical record is usually maintained for the characteristics of the moves applied, rather than the solutions visited.
  - Recent moved are classified as tabus to restrict the search space.
- TS is a variable neighborhood method.
- Tabu restrictions are not inviolable under all circumstances.
- Several types of memories are used, both short term and long term, in order to improve the exploration quality.

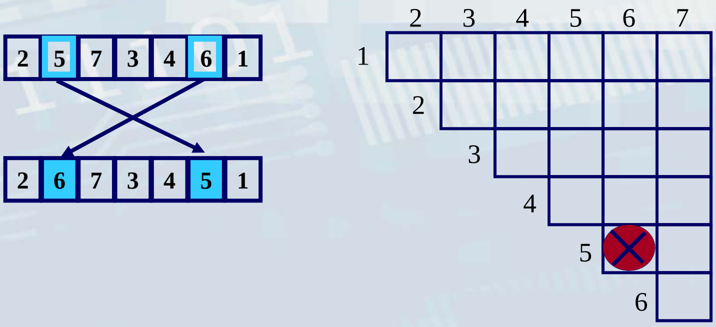


# An Illustrative Example

- A set of tests is to be applied to a given system in order to check its correctness.
- The different orders of applying the tests have different costs (time, power consumption, etc.)
- We want to find the best schedule of the tests.
- A feasible solution can be simply represented by a permutation of the given set of tests.
- A neighborhood move can be defined by swapping two tests.
- The best move will be selected in each step.
- To avoid repeating or reversing swaps done recently, we classify as tabu all most recent swaps.

## Tabu Data Structure

Assume that we have a 7-test problem:

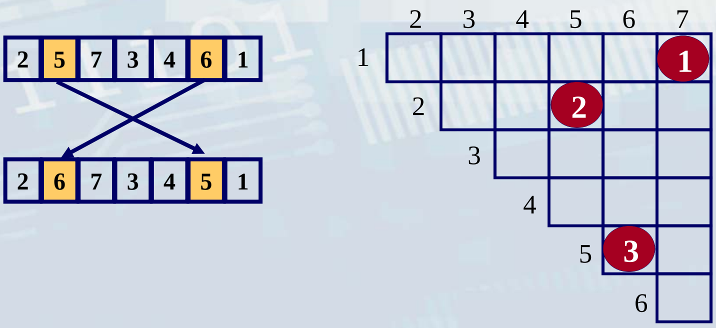



21 moves are possible in each iteration in this example



## Validity of the Tabus

- A paired test tabu will only be valid for three iterations (tabu tenure = 3):




- When a tabu move would result in a solution better than any visited so far, its tabu classification will be overridden (an aspiration criterion).

## A TS Process

**Current solution**

| 2 | 5 | 7 | 3 | 4 | 6 | 1 |
|---|---|---|---|---|---|---|

Cost = 100

| 2 | 4 | 7 | 3 | 5 | 6 | 1 |
|---|---|---|---|---|---|---|

Cost = 94

**Tabu structure**

| | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|
| 1 | | | | | | |
| 2 | | | | | | |
| 3 | | | | | | |
| 4 | | | | | | |
| 5 | | | | | | |
| 6 | | | | | | |

| | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|
| 1 | | | | | | |
| 2 | | | | | | |
| 3 | | | | | | |
| 4 | | | | 3 | | |
| 5 | | | | | | |
| 6 | | | | | | |

**Top 5 candidates**

| Swap | Value |
|---|---|
| 5, 4 | -6 |
| 7, 4 | -4 |
| 3, 6 | -2 |
| 2, 3 | 0 |
| 4, 1 | +1 |

| Swap | Value |
|---|---|
| 3, 1 | -2 |
| 2, 3 | -1 |
| 3, 6 | +1 |
| 7, 1 | +2 |
| 6, 1 | +4 |



## A TS Process

**Current solution**

| 2 | 4 | 7 | 3 | 5 | 6 | 1 |
|---|---|---|---|---|---|---|

Cost = 94

| 2 | 4 | 7 | 1 | 5 | 6 | 3 |
|---|---|---|---|---|---|---|

Cost = 92

**Tabu structure**

| | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|
| 1 | | | | | | |
| 2 | | | | | | |
| 3 | | | | | | |
| 4 | | | | 3 | | |
| 5 | | | | | | |
| 6 | | | | | | |

| | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|
| 1 | | 3 | | | | |
| 2 | | | | | | |
| 3 | | | | | | |
| 4 | | | | 2 | | |
| 5 | | | | | | |
| 6 | | | | | | |

**Top 5 candidates**

| Swap | Value |
|---|---|
| 3, 1 | -2 |
| 2, 3 | -1 |
| 3, 6 | +1 |
| 7, 1 | +2 |
| 6, 1 | +4 |

| Swap | Value | |
|---|---|---|
| 1, 3 | +2 | X |
| 2, 4 | +4 | |
| 7, 6 | +6 | |
| 4, 5 | +7 | X |
| 5, 3 | +9 | |

# A TS Process

**Current solution**

| | | | | | | |
|---|---|---|---|---|---|---|
| 2 | 4 | 7 | 1 | 5 | 6 | 3 |

Cost = 92

**Uphill moves are allowed**

| | | | | | | |
|---|---|---|---|---|---|---|
| 4 | 2 | 7 | 1 | 5 | 6 | 3 |

Cost = 96

**Aspiration criterion applies**

**Tabu structure**

| | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|
| 1 | | 3 | | | | |
| 2 | | | | | | |
| 3 | | | | | | |
| 4 | | | | 2 | | |
| 5 | | | | | | |
| 6 | | | | | | |

| | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|
| 1 | | 2 | | | | |
| 2 | | | 3 | | | |
| 3 | | | | | | |
| 4 | | | | 1 | | |
| 5 | | | | | | |
| 6 | | | | | | |

**Top 5 candidates**

| Swap | Value |
|---|---|
| 1, 3 | +2 |
| 2, 4 | +4 |
| 7, 6 | +6 |
| 4, 5 | +7 |
| 5, 3 | +9 |

| Swap | Value |
|---|---|
| 4, 5 | -6 |
| 5, 3 | -2 |
| 7, 1 | 0 |
| 1, 3 | +3 |
| 2, 6 | +6 |



# A TS Process

**Current solution**

| | | | | | | |
|---|---|---|---|---|---|---|
| 4 | 2 | 7 | 1 | 5 | 6 | 3 |

Cost = 96

| | | | | | | |
|---|---|---|---|---|---|---|
| 5 | 2 | 7 | 1 | 4 | 6 | 3 |

Cost = 90

**Best so far!**

**Tabu structure**

| | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|
| 1 | | 2 | | | | |
| 2 | | | 3 | | | |
| 3 | | | | | | |
| 4 | | | | 1 | | |
| 5 | | | | | | |
| 6 | | | | | | |

| | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|
| 1 | | 1 | | | | |
| 2 | | | 2 | | | |
| 3 | | | | | | |
| 4 | | | | 3 | | |
| 5 | | | | | | |
| 6 | | | | | | |

**Top 5 candidates**

| Swap | Value |
|---|---|
| 4, 5 | -6 |
| 5, 3 | -2 |
| 7, 1 | 0 |
| 1, 3 | +3 |
| 2, 6 | +6 |

| Swap | Value |
|---|---|
| 7, 1 | 0 |
| 4, 3 | +3 |
| 6, 3 | +5 |
| 5, 4 | +6 |
| 2, 6 | +8 |

## Tabu Memories

- The paired test tabu makes use of recency-based memory (short-term memory).
- It should be complemented by frequency-based memory (long-term memory), which record the frequencies of moves.
- The long-term memory is used to diversify the search into new regions.
- Diversification should be restricted to operate only on particular occasions.
  - For example, when no admissible improving moves exist.



## Tabu Memory Structure

**Iteration 26**

**Current solution**

| 1 | 3 | 6 | 2 | 7 | 5 | 4 |
|---|---|---|---|---|---|---|

Cost = 66

**(Recency-based)**

| | 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-based)**

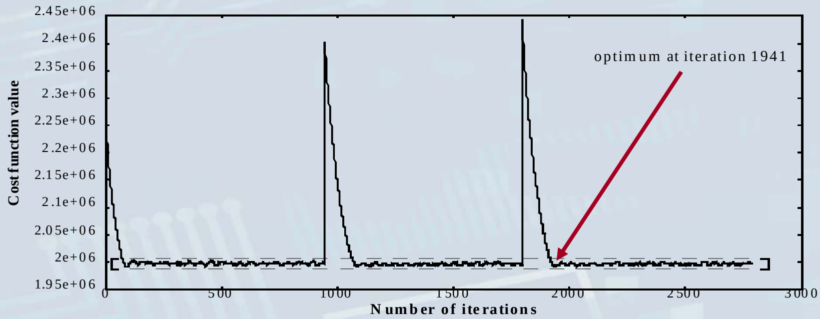
**Top 5 candidates**

| Swap | Value | Penalized Value |
|---|---|---|
| 1,4 | -3 | -2 |
| 2,4 | +1 | +6 |
| 3,7 | +3 | +3 |
| 1,6 | +5 | +5 |
| 6,5 | +4 | +6 |

P.V. = Value + Frequency_count

# Diversifications

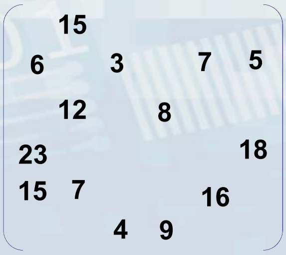
- Diversifications are used to drive the search towards a new area.
  - Moves that are not applied or rarely applied are tried.
- They can also be done in a more dramatical way by:
  - Applying a large number of rarely applied moves, or
  - Generating random solutions.






# Diversification for the TSP Problem

- Keep track of the edges visited so far by maintaining a two dimensional array of occurrence of each edges.




- After each move, the entries corresponding to all the edges in the new tour are incremented.
- After a specified number of iterations, a new starting tour is generated based on the edges that have been visited the least.

# Outline

- **Introduction and basic principles**
- **The algorithm**
- **Intensification and diversification**
- **Improvement techniques**



# The Basic TS Algorithm

Step 1 (Initialization)

(A) Select a starting solution $\boldsymbol{x}^{now} \in \mathbf{X}$.

(B) $\boldsymbol{x}^{best} = \boldsymbol{x}^{now}$, *best_cost* = $\boldsymbol{c}(\boldsymbol{x}^{best})$.

(C) Set the history record *H* empty.

Step 2 (Choice and termination)

Determine *Candidate_N*($\boldsymbol{x}^{now}$) as a subset of $N(H, \boldsymbol{x}^{now})$.

Select $\boldsymbol{x}^{next}$ from *Candidate_N*($\boldsymbol{x}^{now}$) to minimize $c(H, \boldsymbol{x})$.

Terminate by a chosen iteration cut-off rule.

Step 3 (Update)

Re-set $\boldsymbol{x}^{now} = \boldsymbol{x}^{next}$.

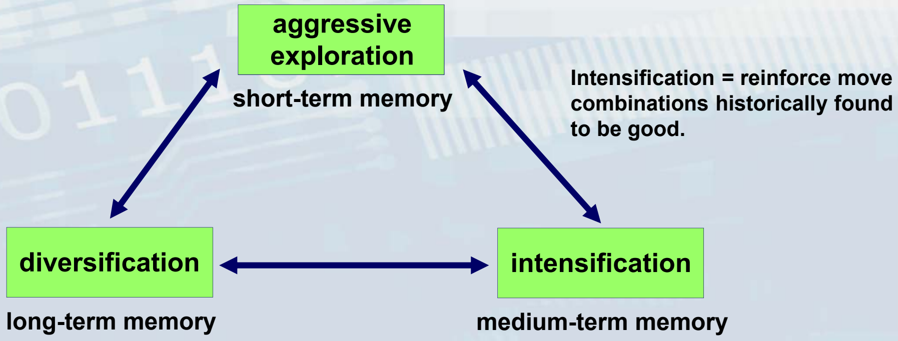
If $c(\boldsymbol{x}^{now}) <$ *best_cost*, perform Step 1(B).

Update the history record *H*.

Return to Step 2.

## Problems to Solve

- How to define and use the history record?




- How to determine *Candidate_N*($x^{now}$)?
- How to evaluate $c(H, x)$?



## Neighborhood Definition

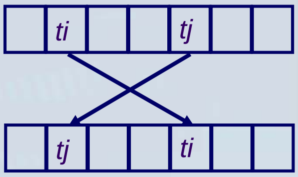
- The moves that define the neighborhood of the current solution must be carefully designed:
  - A very large number of move options must be examined and compared in each iteration, since all moves must in principle be analyzed.
  - The impact of a move to the cost function (move value) must be able to be quickly evaluated:
    - Incremental changes are preferred.
- In some variation of the TS implementation, a sampling technique can be used to select a subset of the moves to evaluate.
  - Usually based on random selection.
  - The sampling size should be still relatively large.

## Short-Term Memory = Tabus

- In general, a tabu is specified by some attributes of the moves.
- When a move having an attribute $e$ is performed, a record is maintained for the reverse attribute $\overline{e}$, in order to restrict a move having some subset of the reverse attributes.

  => To preventing reversals or repetitions!
- A tabu can be defined with different restriction levels, for examples:

  (R1) $t_i$ is involved in a swap.
  (R2) $t_j$ is involved in a swap.
  (R3) either (R1) or (R2) occurs.
  (R4) both (R1) and (R2) occur (assumed).
  (R5) $t_i$ is scheduled to an earlier position.
  (R6) $t_j$ is scheduled to an later position.
  (R7) both (R5) and (R6) occur (reversed move?).






## Tabu Status

- A tabu restriction is typically activated only under certain condition:
  – Recency-based restriction: its attributes occurred within a limited number of iterations prior to the present iteration;
  – Frequency-based restriction: occurred with a certain frequency over a longer span of iterations.
- The condition of being tabu-active or tabu-inactive is called the *tabu status* of an attribute.
- The tabu restrictions and tenure should be carefully selected, in order to
  – achieve cycle prevention, and
  – induce vigor into the search.
- It is preferable to select an attribute whose tabu status less rigidly restricts the choice of moves.

# Tabu Tenure Decision

- The tabu tenure, *t*, must be carefully selected:
  - For highly restrictive tabus, *t* should be smaller than for lesser restrictive tabus.
    - Ex. highly restrictive tabu: $t_i$ is involved in a swap.
    - Ex. lesser restrictive tabu: $t_i$ and $t_j$ are swapped back.
  - It should be long enough to prevent cycling, but short enough to avoid driving the search away from the global optimum.
- *t* can be determined using static rules or dynamic rules:
  - Static rule chooses a value for *t* that remains fixed:
    - $t$ = constant (typically between 7 and 20).
    - $t = f(n)$, where $n$ is the problem size, typically $\in [0.5\, n^{1/2}, 2\, n^{1/2}]$.
  - Dynamic rule changes the value of *t* during the search process.
- Experimentation must be carried out to choose the best tenure!



# Dynamic Tenure

The tabu tenure *t* will be changed dynamically:

- Simple dynamic: vary between two fixed bounds:
  - Randomly, or
  - Systematically:
    - Longer at the beginning.
    - Shorter at the end.
- Attribute-dependent dynamic: the bounds are determined based on the properties of the tabu attributes.
  - Larger for more "attractive" attributes:
    Ex. Good quality moves will have a larger *t*.
  - A weaker restriction should also have a larger *t*.
    Ex. TSP:
    - Tabus: the arcs recently added and the arcs recently dropped.
    - Tabu preventing arcs from being dropped should have a shorter tenure.
    - Tabu preventing arc from being added can have a much longer tenure.

# Aspiration Criteria (AC)

- Used to determine when tabu restrictions should be overridden.
  - To encourage selection of high-quality solution.
- They contribute significantly to the quality of the TS technique.
- They are often selected based on the influence of a move, which measures the degrees of change in solution structure or feasibility.
  - To drive the search to break away from local optimum.
- There are mainly two kinds of aspirations:
  - Move aspirations which revoke a move’s tabu classification.
  - Attribute aspirations which revoke an attribute’s tabu-active status.



# Aspiration Criterion Examples

- Aspiration by Default:
  - If all available moves are classified as tabu, and are not made admissible by some other AC, then the "least tabu" move is selected.
  - This is always implemented, e.g., by selecting the tabu with the shortest time to become inactive.
- Aspiration by Objective:
  - Global optimum: $c(x^{trial}) < best\_cost$.
  - Regional optimum:
    - Subdivide the search space into regions $R \in \mathbf{R}$;
    - Let $best\_cost(R)$ denote the minimum $c(x)$ for $x$ found in $R$;
    - If $c(x^{trial}) < best\_cost(R)$, we have a move aspiration.

# Aspiration Criterion Examples

- Aspiration by Search Direction:
  - Let *direction*(*e*) = IMPR, if the most recent move containing *e_bar* was an improving move; otherwise *direction*(*e*) = NON_IMPR.
  - An attribute aspiration for *e* is satisfied, if *direction*(*e*) = IMPR and $\boldsymbol{c}(x^{trial}) < \boldsymbol{c}(x^{now})$.

    That is, even though you reverse an earlier improving move, it is ok if it improves the cost.



# Aspiration Refinement

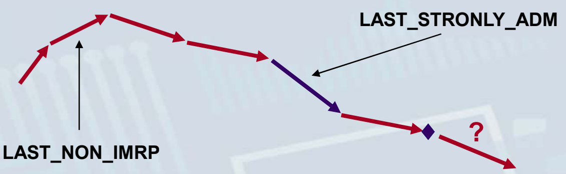
A more refine scheme can be used to handle aspiration:

- When an aspiration criterion is satisfied for an attribute that otherwise is tabu-active, it is called a pending tabu attribute.
- A move that has a pending tabu attribute is called a pending tabu move.
- A pending tabu move can be treated in two ways:
  - It will be a candidate for selection if no improving moves exist except those that are tabu.
  - Additionally, it must be an improving move to qualify for selection.
- A move is strongly admissible if:
  - It is admissible and does not rely on aspiration criteria to qualify for admissibility; or
  - It qualifies for admissibility based on the Global Aspiration by Objective (A tabu move which produces the best solution so far).

# Aspiration Criterion Examples (Cont'd)

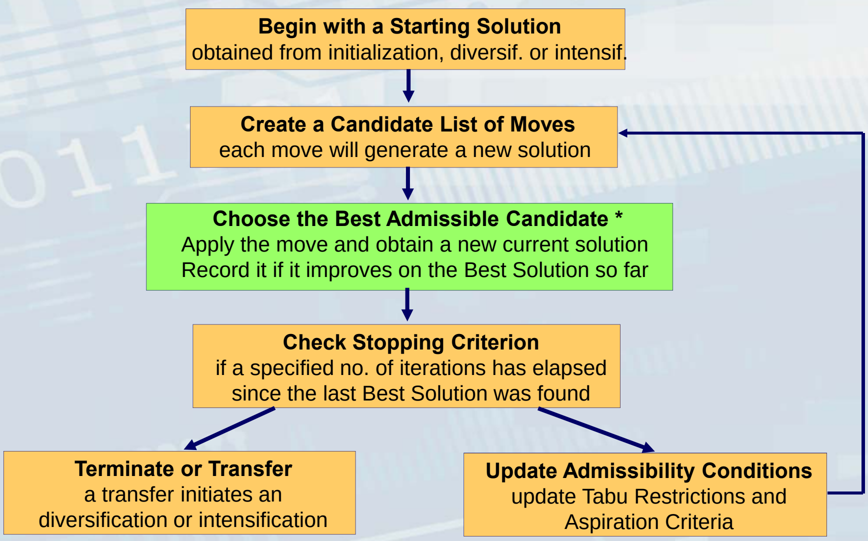
- Aspiration by Strong Admissibility:
  - Let LAST_NON_IMPR = the most recent iteration that a non-improving (uphill) move was made, and
    LAST_STRONLY_ADM = the most recent iteration that a strongly admissible move was made.
  - If LAST_NON_IMRP < LAST_STRONLY_AMD, every improving tabu move will be aspired, and be classified as a pending tabu move.
  - The above condition implies:
    - A strongly admissible improving move has been made since the last non-improving move; and
    - The search is currently generating an improving sequence.

    => the search is moving towards an optimum.

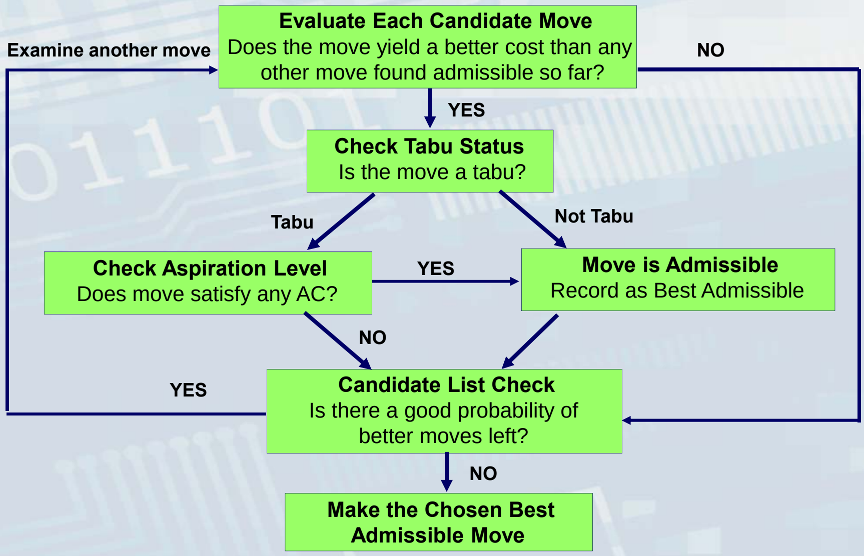





# TS Short-Term Memory Component



# Selecting the Best Admissible Move






# Some Conclusions

- One basic element of the tabu search technique is based on the interplay between tabu restrictions and aspiration criteria.
- Tabu restrictions are used to prevent cyclic behavior.
- The basic idea of aspiration criteria is to allow improving moves to escape a tabu classification under certain conditions.
  - These moves are usually treated as inferior improving moves.
- While tabus are defined by properties of the moves, aspiration criteria are usually based on the outcome of the moves.
- Another interaction which plays an important role in TS is that between diversification and intensification.

# Outline

- Introduction and basic principles
- The algorithm
- Intensification and diversification
- Improvement techniques



# Frequency-based Memory

- Frequency-based memory is used to complement recency-based memory, in order to broaden the foundation for selecting preferred moves.
- A frequency measure is given by the ratio of the number of occurrences of a particular event to a given denominator.

Ex.

$$\frac{\text{number\_of\_swapping}(n_i, n_j)}{\text{Total\_number\_of\_swaps}}$$ **Move frenquency**

$$\frac{\text{number\_of\_occurrences\_of\_edge}(i,\ j)}{\text{Total\_number\_of\_iterations}}$$ **Solution frenquency**

$$\frac{\text{number\_of\_occurrences\_of\_edge}(i,\ j)\text{\_in\_elite\_solutions}}{\text{Total\_number\_of\_elite\_solutions}}$$

**Selected solution frenquency**

## Utilization of the Frequncy Information

- Restrictions in terms of evaluation penalties:
  - The more frequently applied moves are discouraged.
- Incentives in terms of evaluation enhancement:
  - The good solutions or moves should be tried more often.
- They support graduated tabu status.
  - The tabu status of a move or attribute is not longer simply black or white.



## Residence Frequency

Let ***S*** denote a subsequence of the solution sequence generated to the current point.

***S***(*solution_attribute*) = {*x* ∈ ***S*** : *x* contains *solution_attribute*}

Ex. ***S***(<$c_i$, $c_j$> ∈ *solution_path*) denotes all solutions where the edge <$c_i$, $c_j$> resides in the solution of TSP.

#***S***(*solution_attribute*) is a residence measure, which gives the number of times the given attribute resides in the solutions.

- Residence frequency is usually used for diversification.
- Residence frequency is collected from all solutions, and therefore more time-consuming to obtain.

## Transition Frequency

$\boldsymbol{S}$(*move_attribute*) = {$x \in \boldsymbol{S}$ : $x$ results from a move containing *move_attribute*}

$\boldsymbol{S}$(*from_attribute*) = {$x \in \boldsymbol{S}$ : $x$ initiates a move containing *from_attribute*}

$\boldsymbol{S}$(*to_attribute*) = {$x \in \boldsymbol{S}$ : $x$ results from a move containing *to_attribute*}

Ex. $\boldsymbol{S}(x_i = p \text{ to } x_i = q)$ denotes all solutions which are generated by a move which changes $x_i$ from $p$ to $q$, by, e.g., rescheduling task $x_i$ from time step $p$ to $q$).

$\#\boldsymbol{S}(x_i = p \text{ to } x_i = q)$, $\#\boldsymbol{S}(\text{from } x_i = p)$, and $\#\boldsymbol{S}(\text{to } x_i = q)$ are transition measures, which identify the number of times $x_i$ changes from and/or to specified values.

- ▹ Transition frequency are easier to obtain, since it concerns some features of the applied moves, which are directly available.



## Intensification and Diversification

- Diversification encourages compositions of attributes significantly different from those encountered previously during the search.

  => Jumping out from local optimum.

- Intensification encourages the incorporation of "good attributes" in the solutions.

  => Converging to local optimum (or the global one).

- Intensification and diversification counterbalance and reinforce each other.

## Intensification and Diversification

- For diversification, ***S*** is usually chosen to be a significant subset of the full solution sequence.
- For intensification, ***S*** is chosen to be a small subset of the elite solutions (high quality local optima):
  - that share a large number of common attributes, and
  - whose members can reach each other by relatively small numbers of moves.



## Basic I/D Techniques

- Selection of penalty/incentive function, *PI*, which is used as penalty in diversification and as incentive in intensification.
- Ex. For diversification of the TSP problem:

  When considering to add two edges, let

  Panalized_Move_Value(Edge($i$, $j$), Edge($p$, $q$)) =

  Move_Value(Edge($i$, $j$), Edge($p$, $q$)) -

  [ Max{$F$(Edge($i$, $j$) $\in x^{next} - x^{now}$),

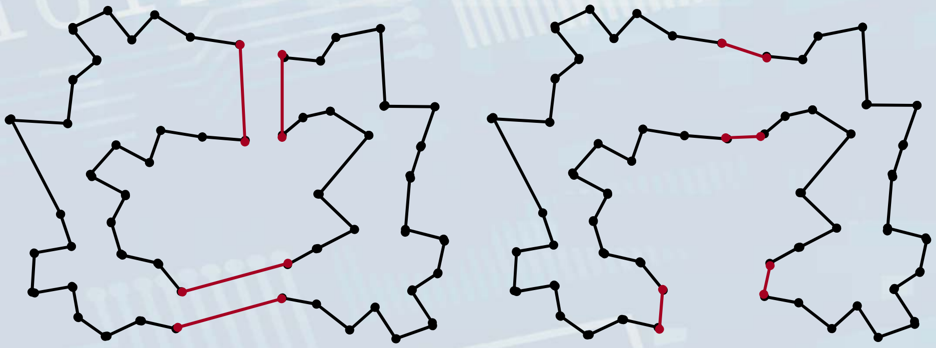
  $F$(Edge($p$, $q$) $\in x^{next} - x^{now}$)} - 20 ]*.

  - where the frequency measure ***F*** is given by the number of occurrences.
  - *20* is a given threshold.

## Basic I/D Techniques (Cont'd)

- Diversification can also be achieved by expanding the neighborhood.

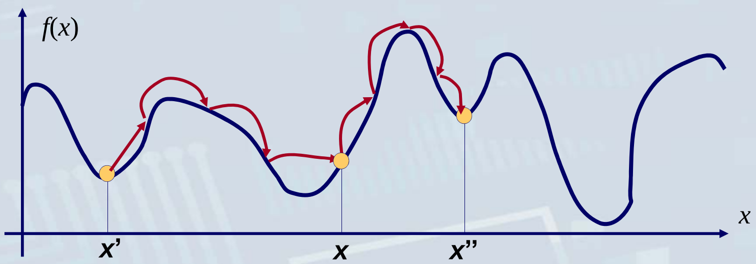
  Ex. TSP: 2-opt basic moves with swapping several edges at a time for diversification.





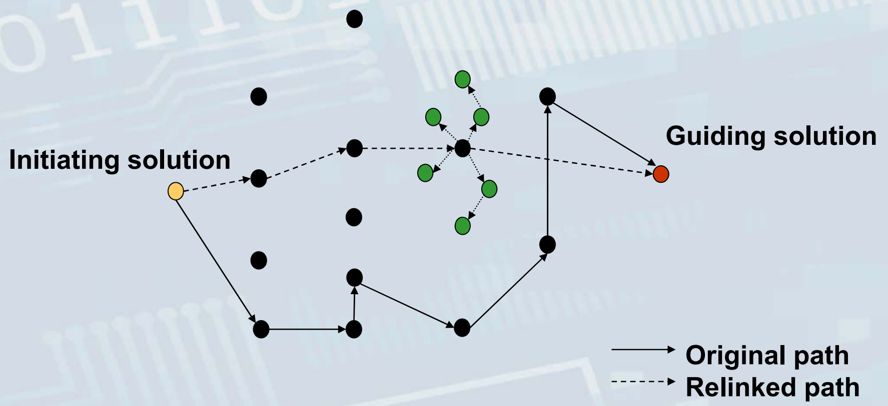
## Path Relinking (PR)

- Basic technique:
  - Select two solution *x*′ and *x*′′ from a collection of elite solutions.
  - A path is generated from *x*′ to *x*′′ by choosing a move that leaves the fewest number of moves remaining to reach *x*′′ at each step.
  - One solution in the path is selected as the starting solution of the next search phase.



# Path Relinking (Cont'd)

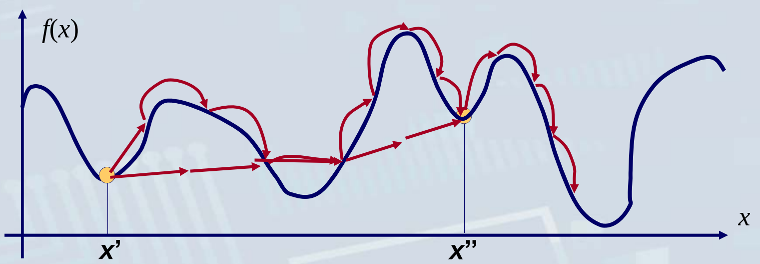
- PR generates new solutions by exploring trajectories that connect elite solutions






# Application of PR

- PR can be used both for intensification and diversification:
  - Choosing $x'$ and $x''$ to share many attributes in common will stimulate intensification.
  - Choosing $x'$ and $x''$ to share very few attributes in common will enforce diversification.
- Extrapolated relinking or tunneling can also be used for diversification.

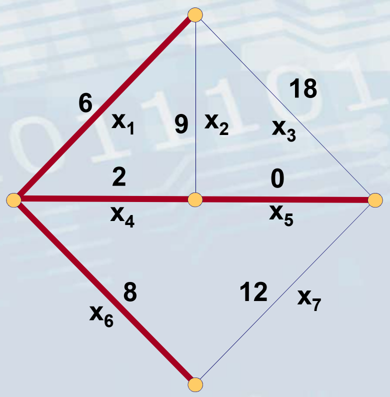


# Outline

- Introduction and basic principles
- The algorithm
- Intensification and diversification
- Improvement techniques



# Minimum Cost Spanning Tree Example




Cost = 16 + 100 (2 Constraint Penalty)

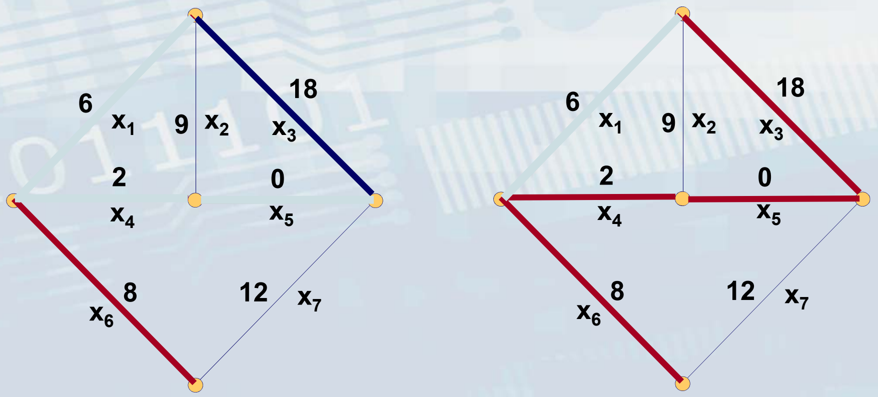
- To find the spanning tree (undirected graph that is connected and acyclic) with minimal total length of its edges that satisfies a set of constraints.

  Ex.

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

  $x_1 \leq x_3$.
- Infeasible solutions are allowed with Violation_penalty = 50.
- A move is defined by adding an edge and dropping another edge to form a new tree.

## Minimum Cost Spanning Tree Example




Cost = 16 + 100 (2 Constraint Penalty)

Cost = 28

Tabu list = $\{x_3\}$
(not to be dropped for 2 iterations)



## Improvements and Variations

- Neighborhood reduction.
- Neighborhood expansion.
- Creation of new attributes during the search process using, e.g., the vocabulary building method:
  - Collect a set of solution ***S***, which is viewed as a text;
  - Analyze the text to extract common patterns, which becomes units of vocabulary.
  - New attributes will be generated based on those units which have a high occurrence frequency.
- Probabilistic tabu search:
  - Moves are selected according to probabilities based on the status and evaluations assigned to these moves.

## Neighborhood Selection Techniques

To avoid evaluating moves from the entire neighborhood:

- Candidate list strategies:
  - Monte Carlo methods:
    - Sample the neighborhood space at random;
    - Repeat the process if the outcome is unacceptable.
  - Systematic selection:
    - Decompose the neighborhood into critical subsets;
    - Select one subset of evaluation at each iteration;
    - Ensure that subsets not examined on one iteration will be evaluated in the subsequent iterations.
  - Master list:
    - Use a master list to keep the best moves.
    - The master list is used for move selections for several iterations.
    - A threshold of acceptability triggers the creation of a new master list.
- This has led to implicit diversification and facilitated parallel implementation.



## Compound Neighborhoods

- Use compound moves, where a sequence of simpler moves is treated as a single complex move, to expand the size of neighborhoods.
- Ejection chain strategy:
  - An element is assigned to a new state, with the outcome of ejecting some other element from its current state.
  - The ejected element is then assigned to a new state, in turn ejecting another element, and so forth.

  Ex. Job sequencing problem:
  - Move a job to a new position occupied by another job, rejecting it from its position.
  - The second job is then moved to a new position to eject another job.
  - This will continue and end by inserting the last job between two other jobs.

=> Very useful for scheduling, routing and partitioning problems.

# Strategic Oscillation

- Perform a type of moves until hitting a boundary, represented, e.g., by feasibility, that normally would be a point where the method stops.
- The neighborhood definition is then extended, or the evaluation criteria for selecting moves is modified, to permit the boundary to be crossed.
- It proceeds for a specified depth beyond the boundary.
- It then turns around and proceeds in the opposite direction by performing another type of moves.

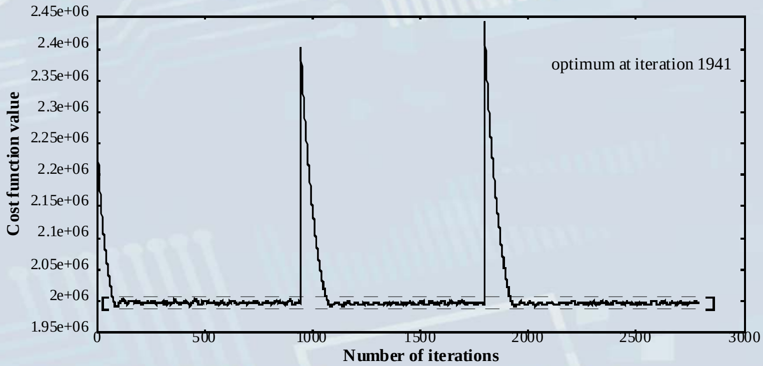
  Ex. Find a spanning tree:
  - Add edges to a growing tree until it is spanning.
  - Continue to add edges to cross the boundary.
  - A different graph structure results when the current solution no longer constitutes a tree, and hence a different neighborhood is required.
  - Edges are then removed in order to proceed in the opposite direction.
- This technique allows infeasible solutions in the search path.



# Stopping Conditions

TS does not converge naturally.

- A fixed number of iterations has elapsed in total.
- A fixed number of iterations has elapsed since the last best solution was found.
- A given amount of CPU time has been used.



# Applications of TS

- Scheduling is one of the most fruitful areas.
  - TS outperforms classical heuristics in most cases.
  - For special classes of problems, optimal solutions are always found, given sufficient CPU time.
- Transportation problems:
  - TSP.
  - Vehicle routing.
- Layout and circuit design.
- Path assignment and bandwidth packing.
- Graph problems.
- Probabilistic logic and expert systems.
- etc.

- TS is extremely useful when the feasibility condition is very strong and the randomly generated neighborhood solutions are usually unfeasible ones.



# TS vs. SA

- Neighborhood space exploration:
  - TS emphasizes complete neighborhood evaluation to identify moves of high quality.
  - SA samples the neighborhood solutions randomly.

- Move evaluation:
  - TS evaluates the relative attractiveness of moves in relation not only to objective function change, but also to factors of influence.
  - SA evaluates moves only in terms of their objective function change.

# TS vs. SA (Cont'd)

- Search guidance:
  - TS uses multiple thresholds, reflected in the tabu tenures and aspiration criteria, which varies also non-monotonically.
  - SA is based on a single threshold implicit in the temperature parameter that only changes monotonically.
- Use of memory:
  - SA is memoryless.
  - TS makes heavily and intelligently use of both short-term and long-term memory.



# Main Weakness of TS

- No theory has yet been formulated to support TS and its convergence behavior.
- Good understanding of the problem structure is required. Domain specific knowledge is needed for selection for tabus and aspiration criteria.
- Experiment might have to be done using different tabu classification schemes and aspiration criteria.
- It needs considerable memory resources.
- Efficient data structure must be used for tabu list manipulation.
- TS is still a relatively young technique and has many unexplored issues.

## Summary I

- TS is a meta-heuristic which can be superimposed on other procedures to prevent them from being trapped at local optimum.
- Three main elements:
  - Flexible memory;
  - Tabu restrictions and aspiration criteria;
  - Intensification and diversification.
- TS has a natural rationale: it emulates intelligent uses of memory in the human problem solving process.



## Summary II

- Parallel implementation of TS is relatively straightforward and efficient.
  - Parallel evaluations of moves.
  - Independent searches using different initial solutions or/and different parameters.
- When properly implemented, TS often outperforms the best previously known techniques.