



Example Midterm Questions

Midterm will cover:

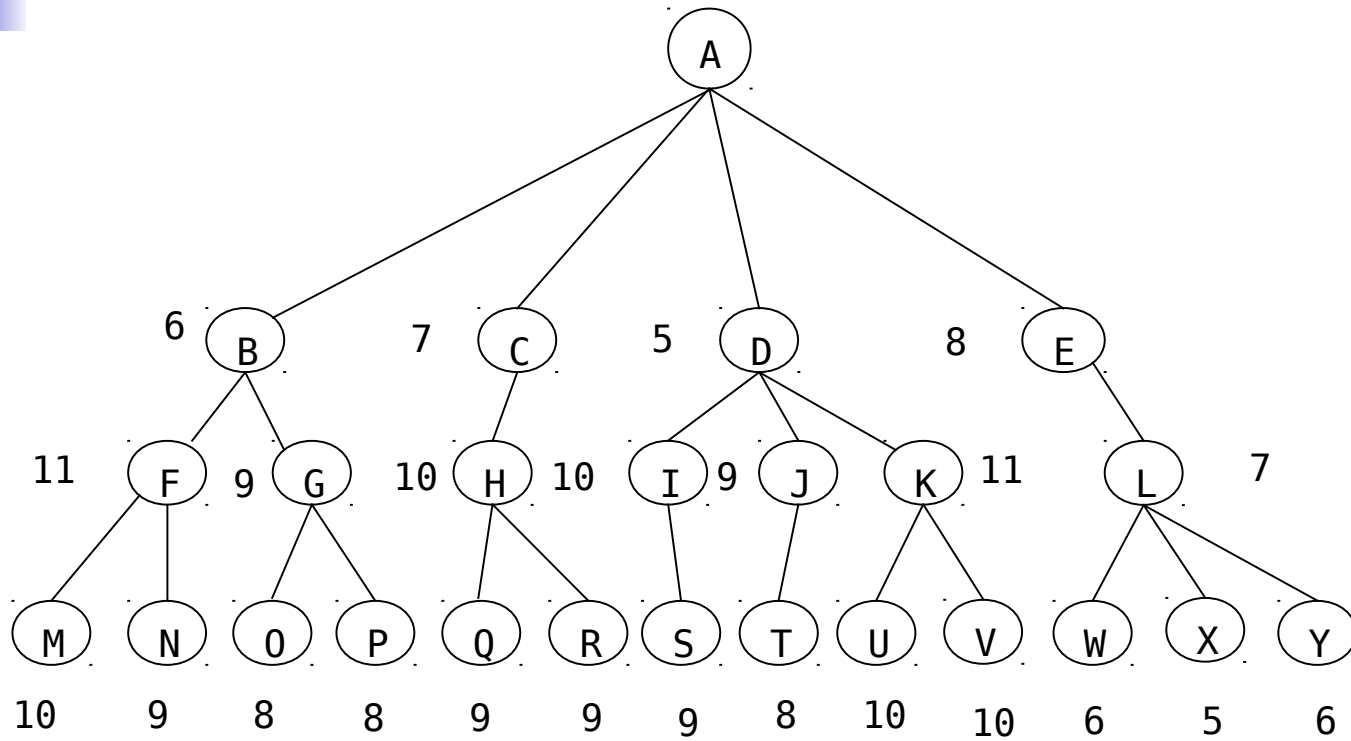
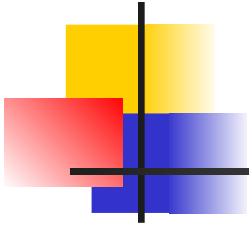
- Search, heuristics and game playing
- Tabu Search
- Simulated Annealing
- GA

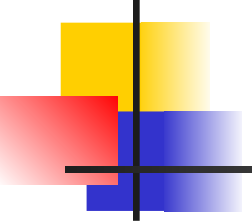


1st Example

1) Search

- a. [6 marks] Discuss the difference between Hill climbing, Backtracking, Beam search, Breadth first and Best first algorithms, with respect to the scope of selecting between alternatives and the scope of recovery. (a diagram may be sufficient).
- b. Consider the following tree as part of a search space





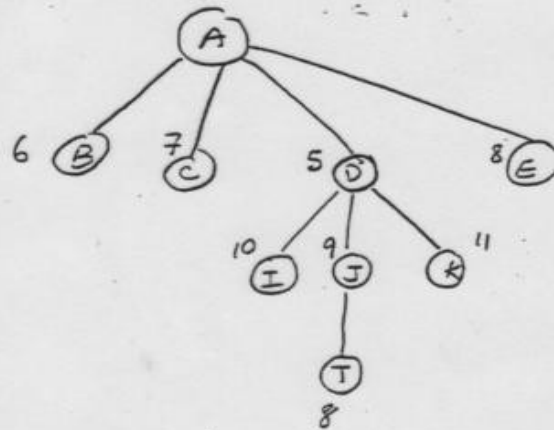
The number indicated beside each node represents the value of an estimated distance to the goal node.

Draw the search tree and write the order of expansion of the nodes when applying:

- i. [6 marks] Hill climbing search.
- ii. [6 marks] Beam search with width of 2.
- iii. [6 marks] Best first search.

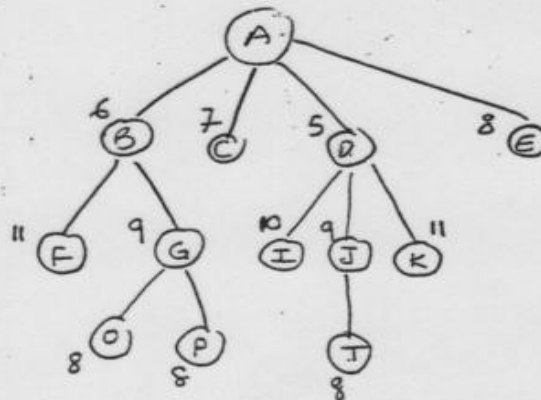
(R)

(i) Hill Climbing Depth first, expanding the best successor



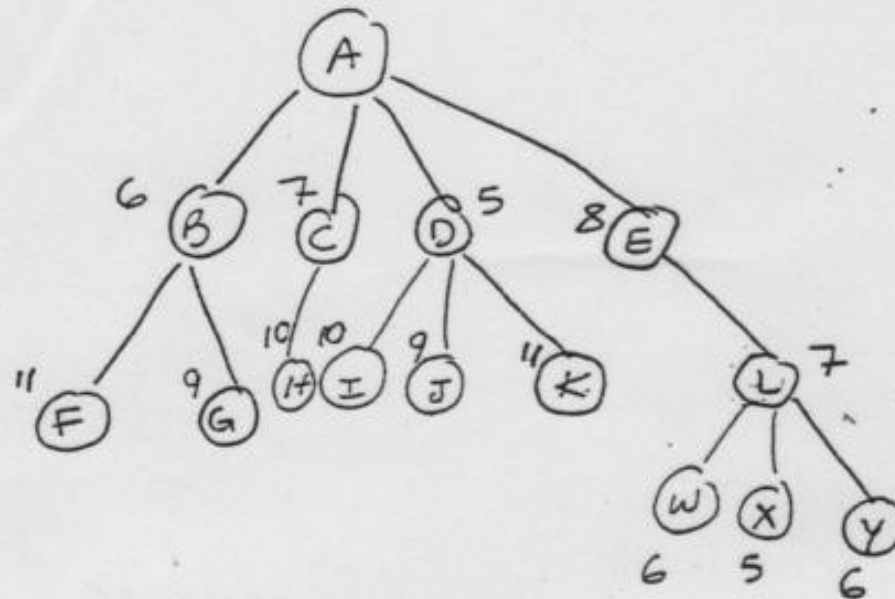
A D J

(ii) Beam Search : Breadth first with ~~list~~ 2 each level

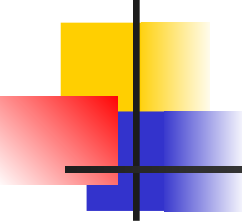


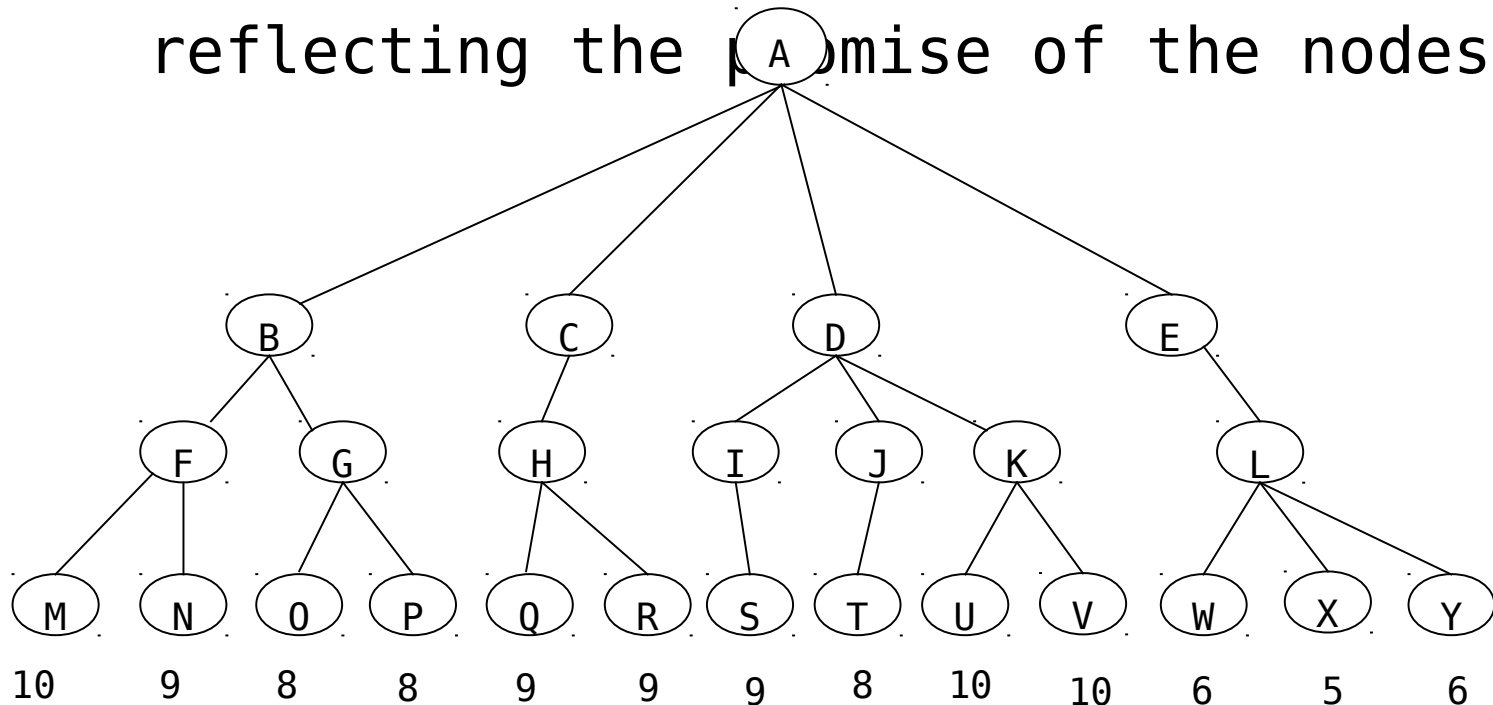
A B D G J

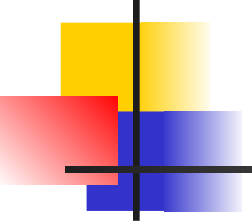
(iii) Best first

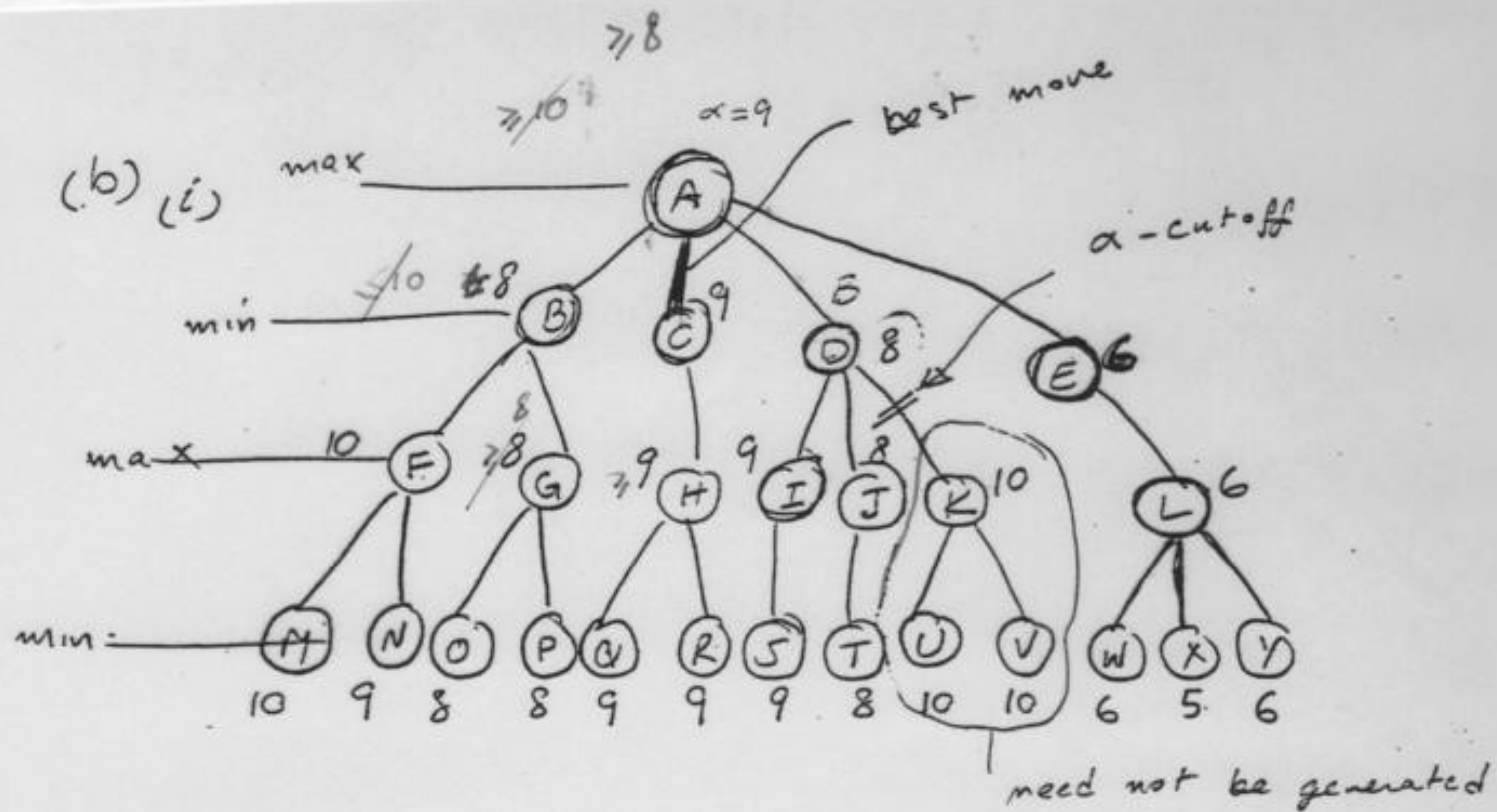


A D B C E L

- 
- b. Consider the following hypothetical game tree (the values at the leafs are values of the evaluation function reflecting the promise of the nodes):



- 
-
- i. Using minimax technique obtain the value at the root and show which move player A should make.
 - ii. Apply Alpha-Beta pruning from left to right and show which parts of the tree need not be generated.



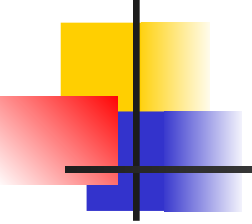
(ii) β value at D of 8 < α value at A of 9, α -cut off.

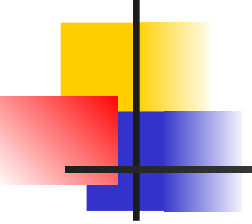


3rd Example

1) Search

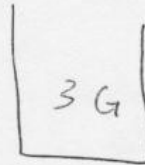
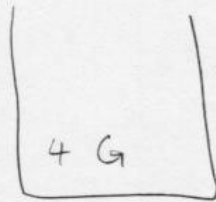
Consider the water jug problem : there are two jugs, a 4-gallon one and a 3-gallon one. Neither has any measuring markers on it and they are initially empty. There is a pump that can be used to fill the jugs with water. The goal is to have exactly 2 gallons of water in the 4-gallon jug and 3-gallon jug empty. A state space description of this problem can be a set of ordered pairs of integers (X,Y) , such that $X=0,1,2,3, \text{ or } 4$ and $Y=0,1,2, \text{ or } 3$; X and Y represent the number of gallons of water in the 4-gallon jug, and the 3-gallon jug respectively.

- 
-
- i. Write the start state and goal state using the given description.
 - ii. Develop a set of operators which can be used to change one state to another.
 - iii. By numbering the nodes in order of their expansion and not to generate the same node from more than one parent, construct the search tree to obtain the goal state, using the following search methods :

- 
-
1. Breadth First search
 2. Depth First Search with depth limit of 7.
 3. Best First search using an evaluation function for node n , $f(n)=g(n)+h(n)$, where $g(n)$ = the level of the node and $h(n)$ is the distance to the goal (use your breadth first search tree to get the values).

Solution to the Water Jug Problem

①



Description

(X, Y)

$X = 0, 1, 2, 3, 4$

$Y = 0, 1, 2, 3$

Initial state $(0, 0)$

Goal state $(2, 0)$

operators

① Fill 4G

② Fill 3G

③ empty 4G

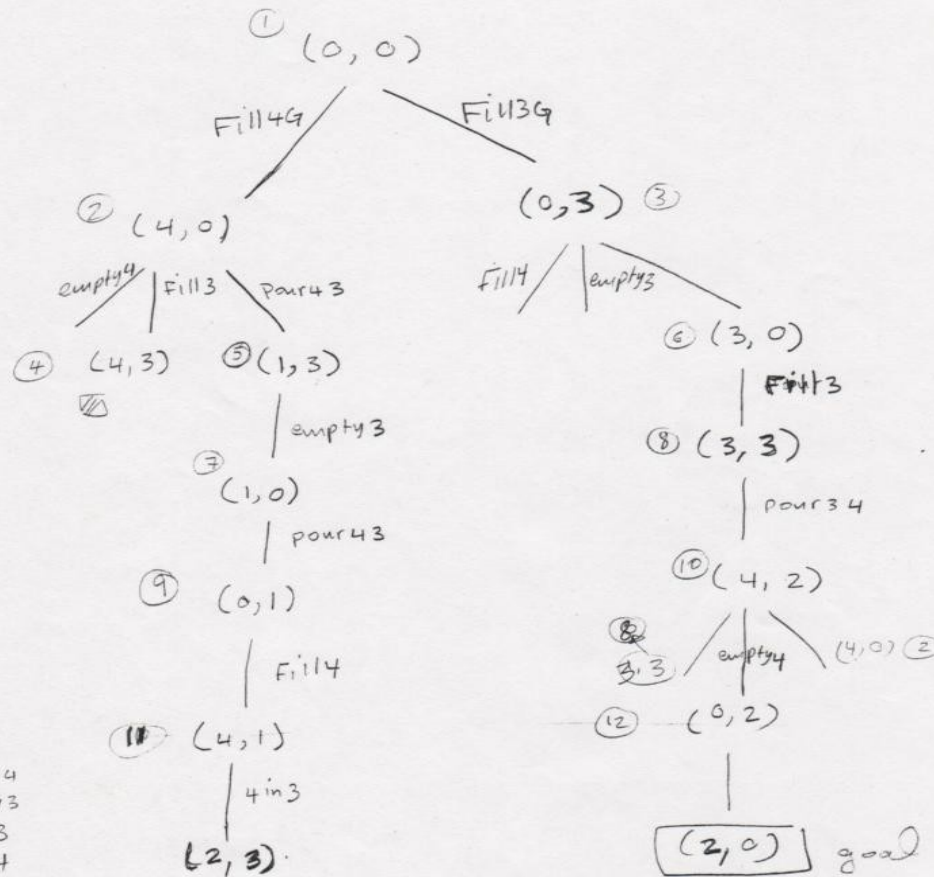
④ empty 3G

⑤ pour 4 into 3

⑥ pour 3 into 4

Breadth First

③

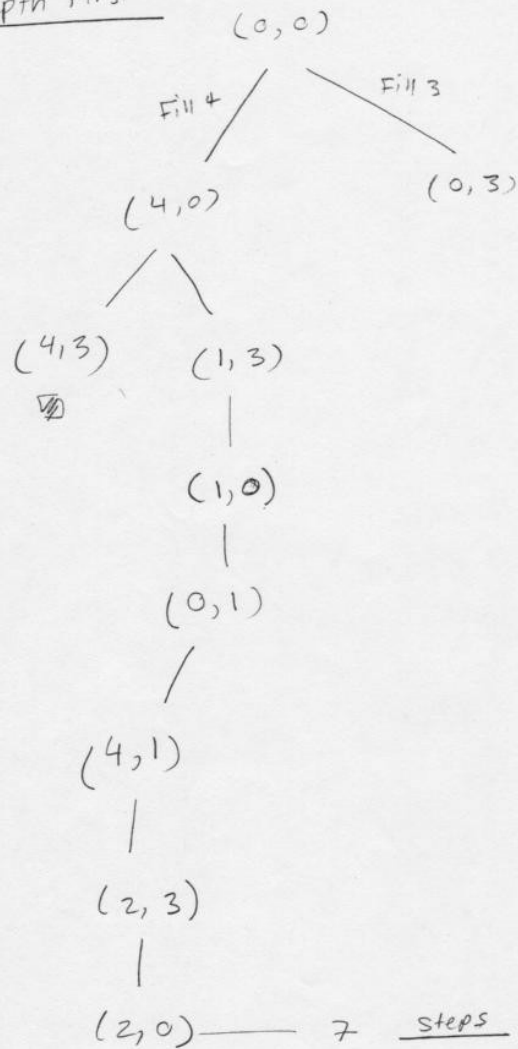


13 Generated

6 steps

Depth First

(4)

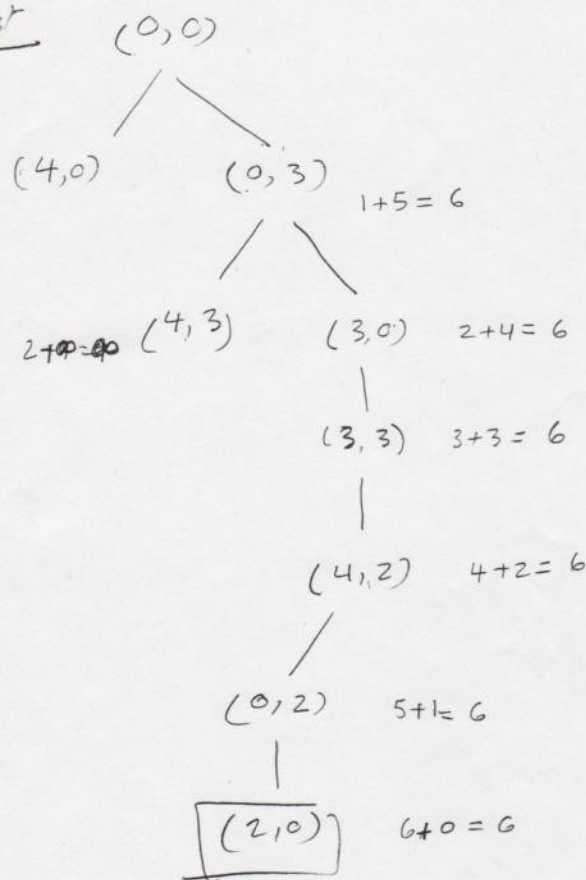


10 generated

Best first

(5)

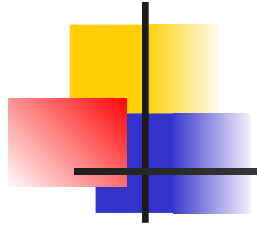
$f = \frac{f}{1+6} = 7$



8 Generated

6 steps





Tabu search



Tabu Search Strategy

- 3 main strategies [7]:
 - **Forbidding strategy**: control what enters the tabu list
 - **Freeing strategy**: control what exits the tabu list and when
 - **Short-term strategy**: manage interplay between the forbidding strategy and freeing strategy to select trial solutions

Parameters of Tabu Search [5]



- Local search procedure
- Neighborhood structure
- Aspiration conditions
- Form of tabu moves
- Addition of a tabu move
- Maximum size of tabu list
- Stopping rule



Basic Ingredients of Tabu Search

- A chief way to exploit memory in tabu search is to classify a subset of the moves in a neighborhood as forbidden (or **tabu**) [1].
- A **neighborhood** is constructed to identify adjacent solutions that can be reached from current solution [8].
- The classification depends on the history of the search, and particularly on the recency or frequency that certain move or solution components, called **attributes**, have participated in generating past solutions [1].
- A **tabu list** records forbidden moves, which are referred to as **tabu moves** [5].
- Tabu restrictions are subject to an important exception. When a tabu move has a sufficiently attractive evaluation where it would result in a solution better than any visited so far, then its tabu classification may be overridden. A condition that allows such an override to occur is called an **aspiration criterion**[1].



Basic Tabu Search Algorithm

[4]

- Step 1: Choose an initial solution i in S . Set $i^* = i$ and $k=0$.
- Step 2: Set $k=k+1$ and generate a subset V^* of solution in $N(i,k)$ such that neither one of the Tabu conditions is violated or at least one of the aspiration conditions holds.
- Step 3: Choose a best j in V^* and set $i=j$.
- Step 4: If $f(i) < f(i^*)$ then set $i^* = i$.
- Step 5: Update Tabu and aspiration conditions.
- Step 6: If a stopping condition is met then stop. Else go to Step 2.

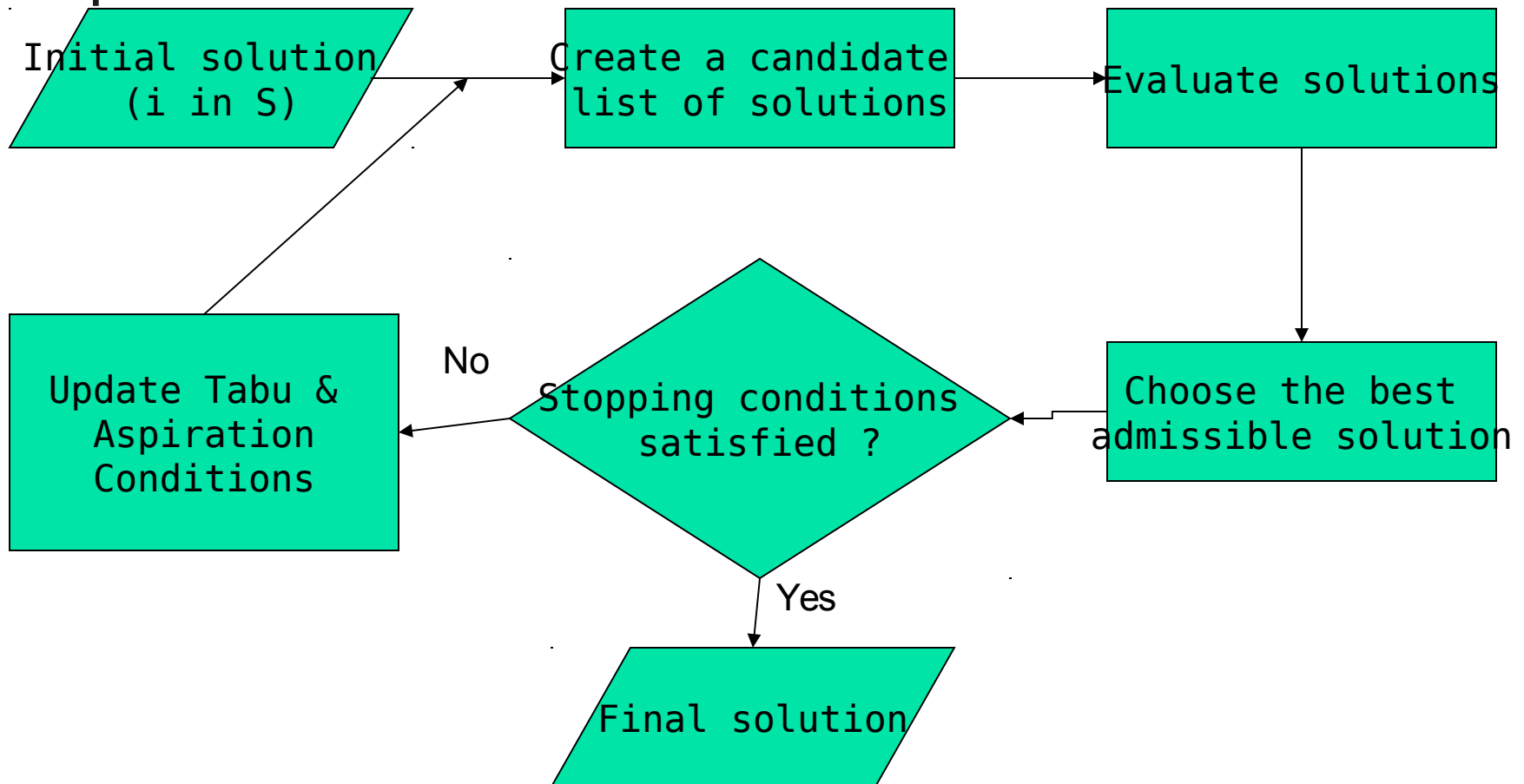


Tabu Search Stopping Conditions

Some immediate stopping conditions could be the following [4]:

1. $N(i, K+1) = 0$. (no feasible solution in the neighborhood of solution i)
2. K is larger than the maximum number of iterations allowed.
3. The number of iterations since the last improvement of i^* is larger than a specified number.
4. Evidence can be given that an optimum solution has been obtained.

Flowchart of a Standard Tabu Search Algorithm [7]

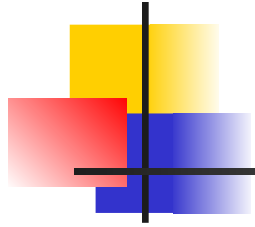




Four dimensions of TS memory

- Recency based (short term) memory
- Frequency based (long term) memory
- Quality: ability to differentiate 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

Example ^[5]

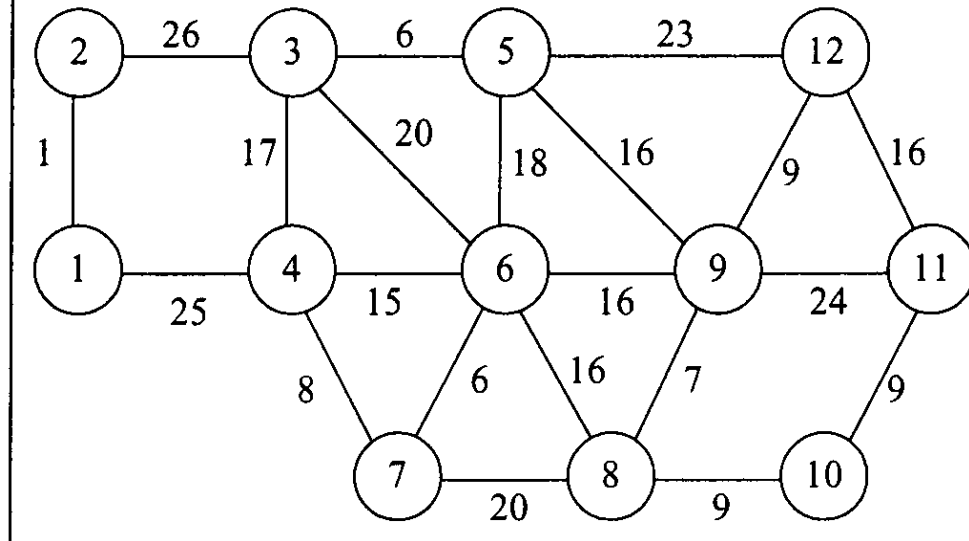



Example 2: Minimum k -tree

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

we

Fig. 3.1 Weighted undirected graph.





Example 2: Greedy construction of

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:

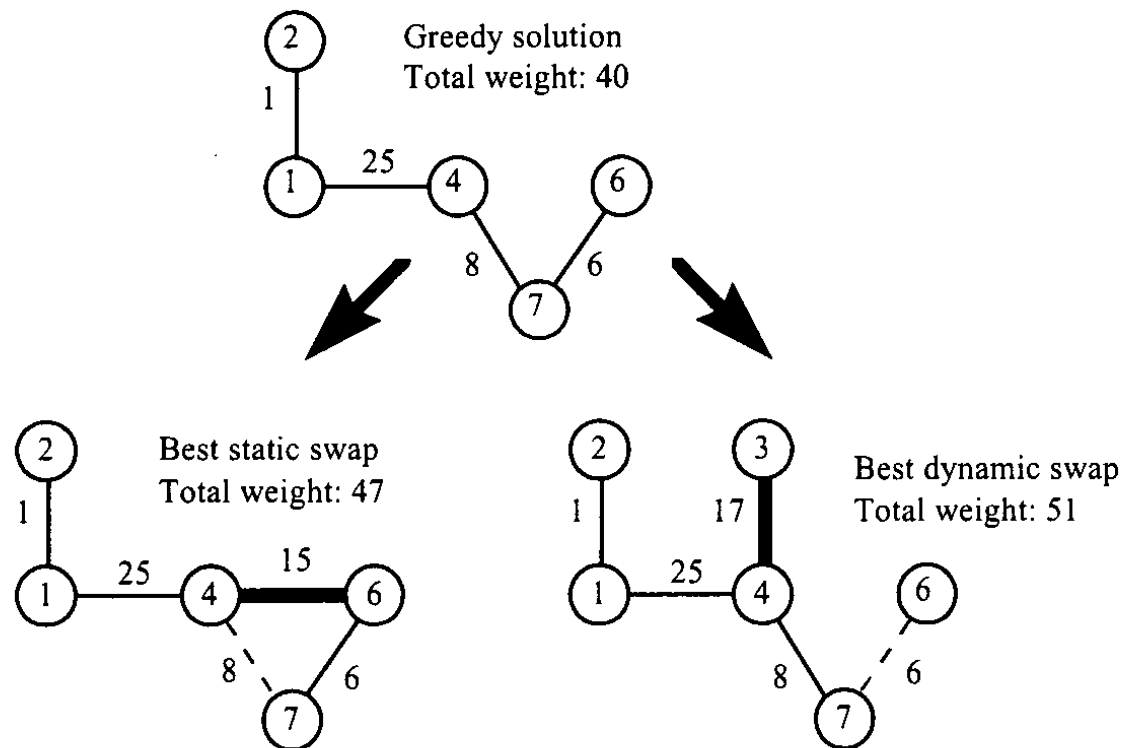
Neighborhood structure

- 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

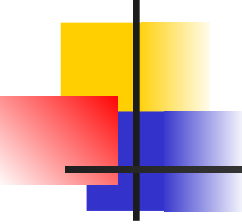
Example 2:

Neighborhood structure

Fig. 3.2 Swap move types.



Static swap is selected



Example 2: Tabu classification

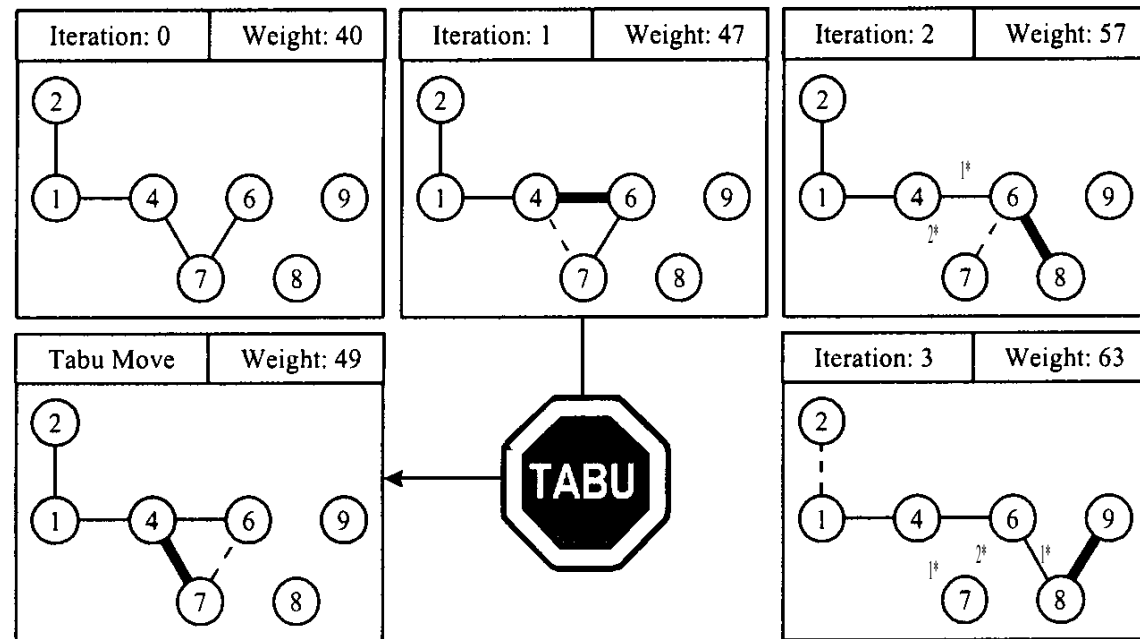
- Tabu attributes are edges swapped
- Out-edge is $\{4,7\}$, in-edge is $\{4,6\}$ in Figure 3.2
- Tabu tenure for in-edges (bounded by k) is set as 1
- Tabu tenure for out-edges is chosen as 2 (since there are more outside edges)
- A swap move is classified as tabu if either the out-edge or the in-edge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

Example 2: Tabu

classification may prevent

ions

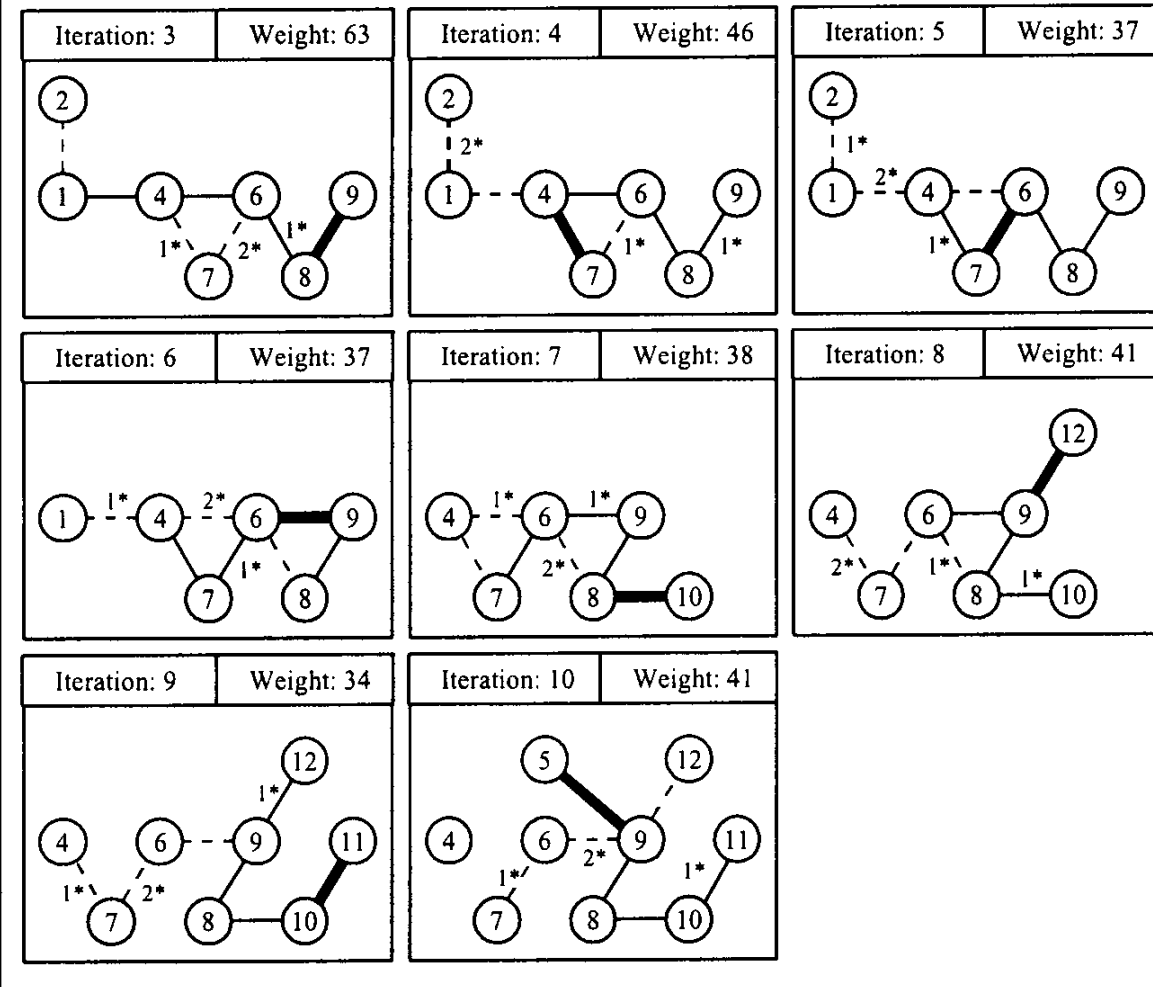
Fig. 3.3 Effects of attributive short term memory.



1] ge {6,7}
 with objective function value 49 cannot be
 visited since {4,7} is tabu-active, and we end up
 with an inferior solution

Example 2: Additional

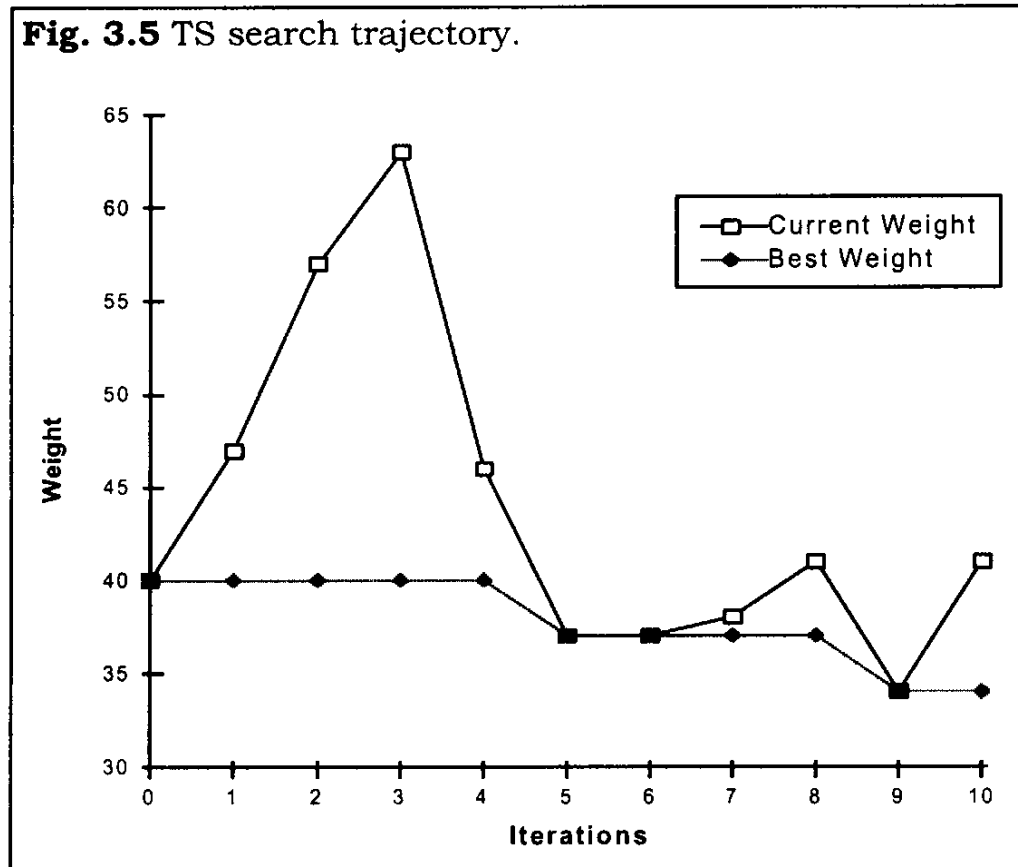
Fig. 3.4 Graphical representation of TS iterations.



34 is the
optimal
solution

Example 2: TS search trajectory

Fig. 3.5 TS search trajectory.



Note the local optima in iterations 5 and 6



Extensions

- Long term memory
- We'll see how this information is used in a minute
- How often have solution elements been used during the search so far?
- Which moves lead to local optima?



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 solutions) during the search

Example 1:

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

Penalized
Swap Value Value

1,4	3	3	T
2,4	-1	-6	
3,7	-3	-3	*
1,6	-5	-5	
6,5	-4	-6	

; exist

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



Example 2:

Diversification using

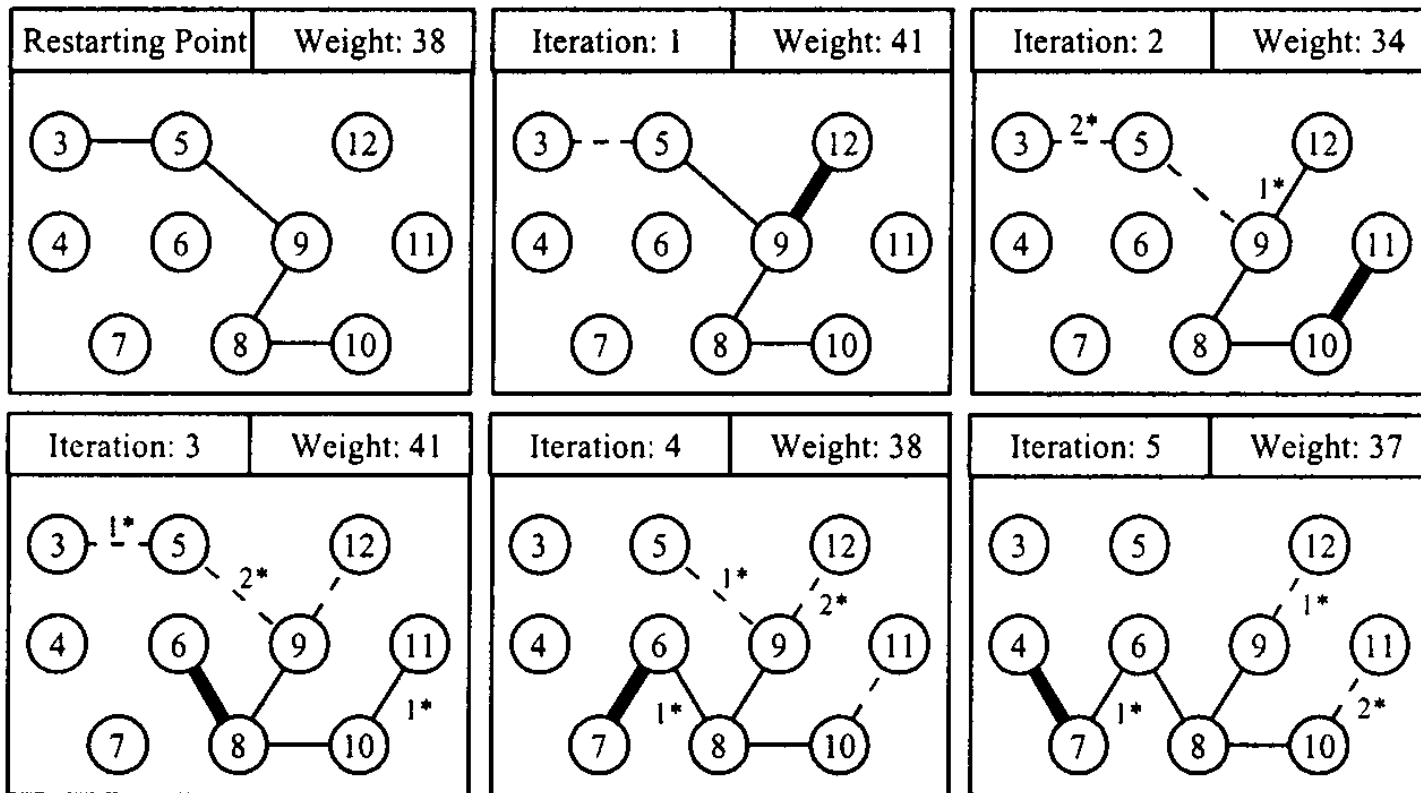
critical event memory

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

Fig. 3.6 Graphical representation of TS iterations after restarting.





A TS Algorithm

1. Find an initial solution $x_0 \in X$,
 $x^{now} = x^{best} = x_0$ set
 , 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 in 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



Diversification

- When search not going well; e.g., haven't found an improving neighbor in some iterations
- New best solutions become rare
- Restart: avoid beginning near known local optima (stored in LTM!)
- Modify objective function for a fixed number of iterations
 - Penalize frequently performed moves or moving to oft visited solutions (LTM!)
 - Change absolute constraints to large penalties
 - Note: both of these could be used throughout the search; the 2nd one was in our example



Intensification

- Intensify search in promising regions
- When to intensify vs. diversify: interesting; further topic we will not cover
- Revisit one of the best solutions so far (LTM!) and shrink Tabu list for small number of iterations
- Temporarily change the objective function to reward solutions with “good” components based on performance in search so far



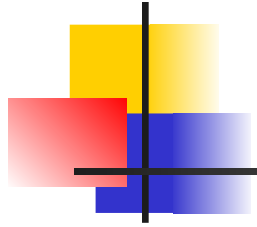
Pros and Cons

■ Pros:

- Allows non-improving solution to be accepted in order to escape from a local optimum (true too for GA, SimAnn, AntColOpt)
- The use of Tabu list (helps avoid cycles and move reversal)
- Can be applied to both discrete and continuous solution spaces
- For larger and more difficult problems (scheduling, quadratic assignment and vehicle routing), tabu search obtains solutions that rival and often surpass the best solutions previously found by other approaches [1].

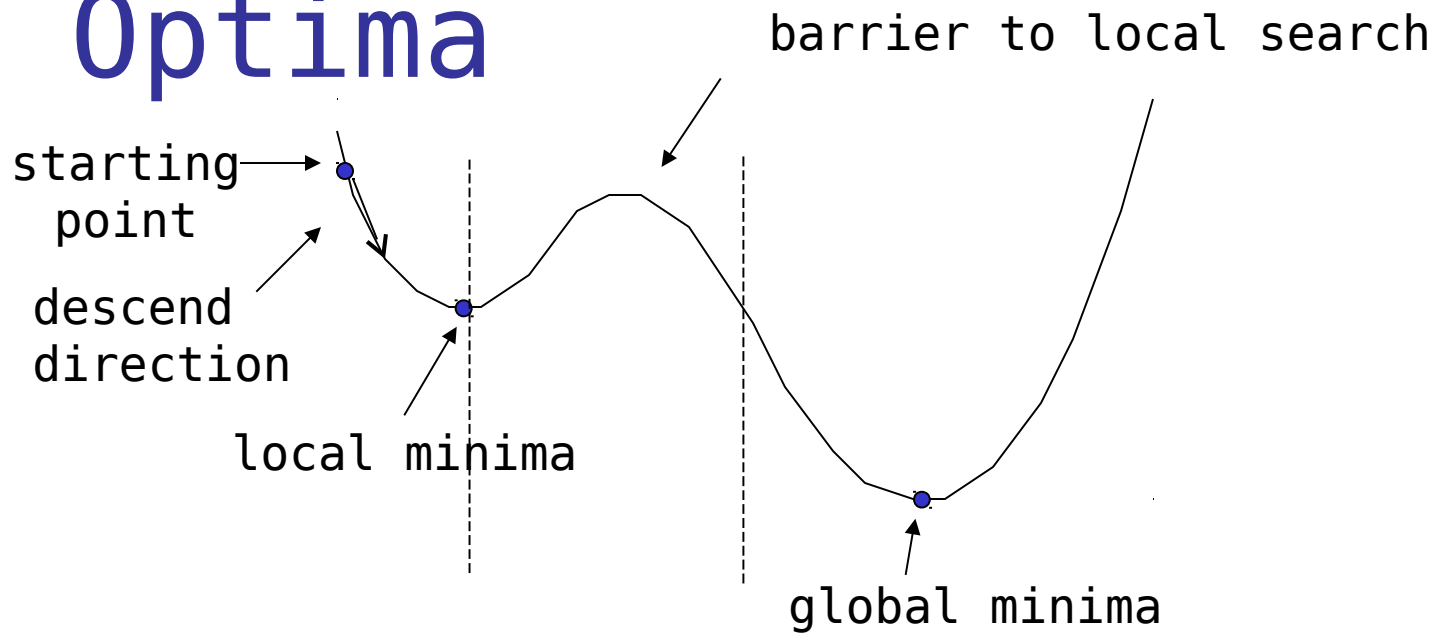
■ Cons:

- Too many parameters to be determined
- Number of iterations could be very large
- Global optimum may not be found, depends on parameter settings (also true of GA, SimAnn, AntColOpt)



Simulated annealing

Difficulty in Searching Global Optima

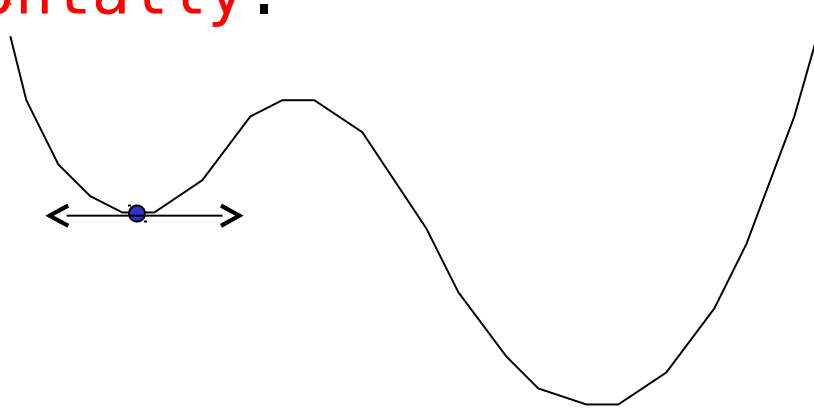




Possible Solutions

Solution 1:

Randomly select another starting point whenever trapped in a local optima – **shake the picture horizontally.**





Possible Solutions

Solution 2:

In addition to solution 1, we allow search to occasionally go on an ascending direction – **shake the picture vertically.**





Intuition of Simulated Annealing

Origin:

The annealing process of heated solids.

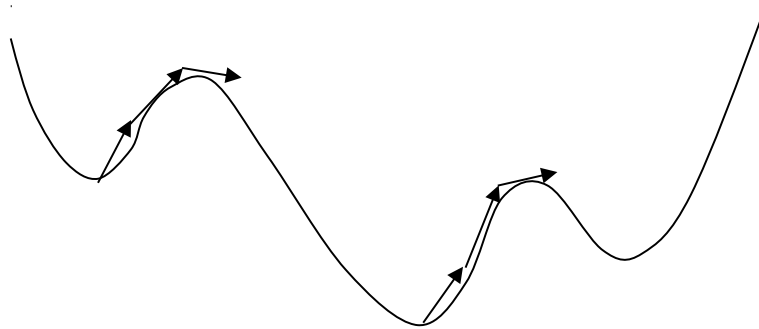
Intuition:

By allowing occasional ascent in the search process, we might be able to escape the trap of local minima.

Consequences of the Occasional Ascents

desired effect

Help escaping the
local optima.



adverse effect

Might pass global optima
after reaching it



Control of Annealing Process

Acceptance of a search step (Metropolis Criterion):

- Assume the performance change in the search direction is Δ .
- Always accept a descending step, i.e. $\Delta \leq 0$
- Accept an ascending step only if it passes a random test, $\exp(-\Delta/T) > \text{random}[0,1)$



Control of Annealing Process

Cooling Schedule:

- T , the *annealing temperature*, is the parameter that control the frequency of acceptance of ascending steps.
- We gradually reduce temperature $T(k)$.
- At each temperature, search is allowed to proceed for a certain number of steps, $L(k)$.
- The choice of parameters $\{T(k), L(k)\}$ is called the *cooling schedule*.



Simulated Annealing Algorithm

- 0) $k = 0$;
- 1) Search ($i \rightarrow j$), performance difference Δ ;
- 2) If $\Delta \leq 0$ then accept, else
 if $\exp(-\Delta/T(k)) > \text{random}[0,1)$ then accept;
- 3) Repeat 1) and 2) for $L(k)$ steps;
- 4) $k = k+1$;
- 5) Repeat 1) – 4) until stopping criterion is met;



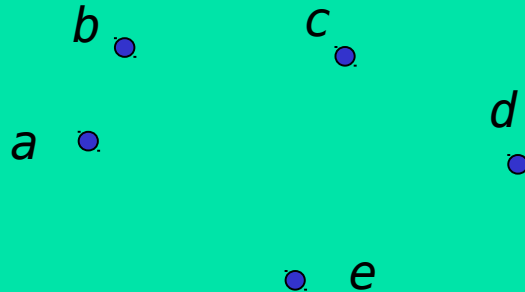
Implementation of Simulated Annealing

- Select a local search scheme
 - Define a neighborhood structure
 - Sample a new design from the neighboring designs of the current design.
 - Accept the new design based on Metropolis Criterion.



An Example

A Traveling Salesman Problem



Visit each city once and only once with minimal overall traveling distance.



An Example

Possible Solutions

1:abcde	2:abced	3:abdce	4:abdec
5:abecd	6:abedc	7:acbde	8:acbed
9:acdbe	10:acebd	11:adbce	12:adcbe

Choose a neighborhood structure

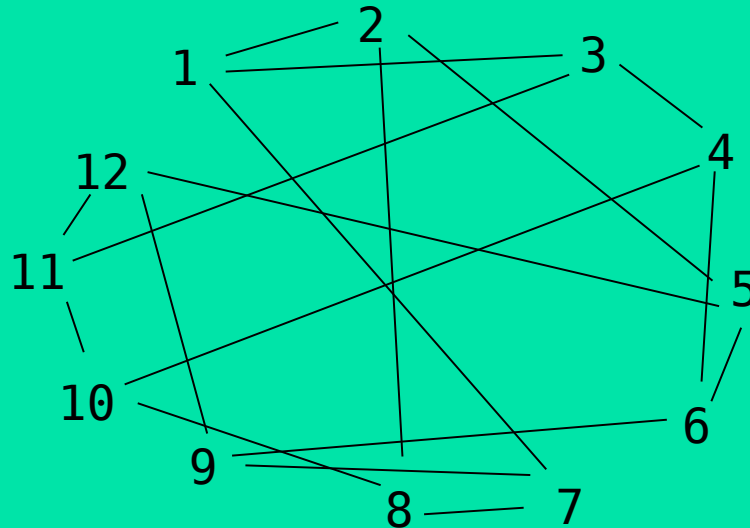
Example 1: An arbitrary sequence

1	—	2	—	3	—	4	—	5	—	6
12	—	11	—	10	—	9	—	8	—	7

An Example

Example 2:

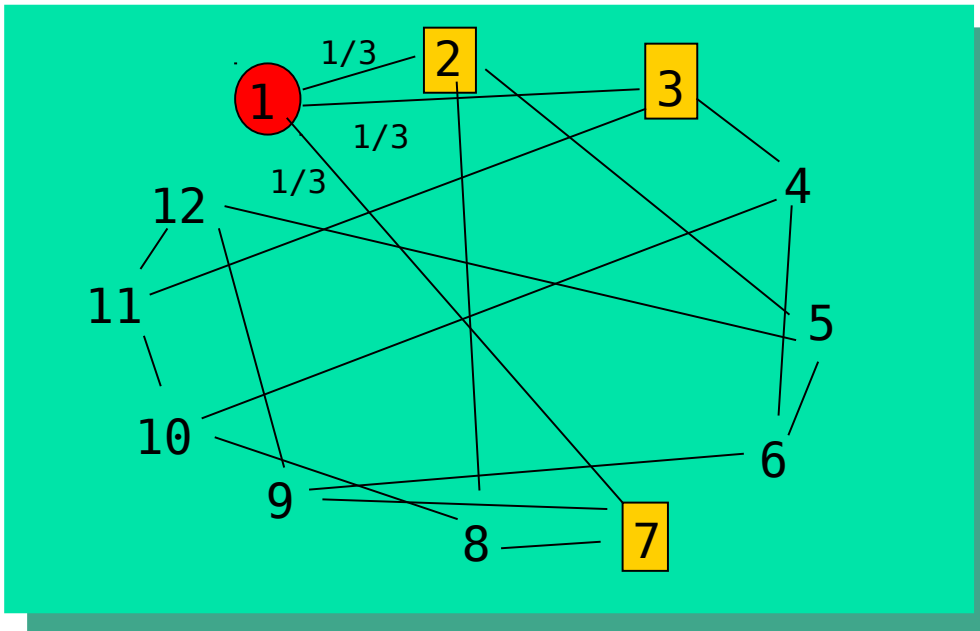
Two designs are neighbors iff two neighboring cities in one design are in the opposite order in another design.



An Example

Sample a neighboring design

Suppose current design is 1 at example 2.
Then we select one of its neighbors, design 2, 3 and 7, with probability $1/3$.





Implementation of Simulated Annealing

- Determine the cooling schedule

For example:

- Set $L = n$, the number of variables in the problem.
- Set $T(0)$ such that $\exp(-\Delta/T(0)) \approx 1$.
- Set $T(k+1) = \alpha \bullet T(k)$, where α is a constant smaller but close to 1.



Implementation of Simulated Annealing

- Understand the result:
 - This is a stochastic algorithm. The outcome may be different at different trials.
 - Convergence to global optima can only be realized in asymptotic sense.
 - Slow convergence: to ensure convergence, the decreasing rate for $T(k)$ should be set at no faster than $O([\log k]^{-1})$



Comparison of TS with others

- Traditional descent methods 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





2) **Tabu Search**

- a. Explain what is meant by the following and illustrate your answers using simple examples:
 - i. Memory structures, their types and functions,
 - ii. Neighborhood structure,
 - iii. Intensification and diversification.



TS Memory Structures, their types and functions

- **Short-term memory, a.k.a Tabu list.** It stores the recent solutions to prevent cycling when the local search is moving away from a local optimum.
- **Long-term memory, a.k.a frequency memory.** It holds the total number of iterations (since the beginning of the search) in which certain components always appear in the current solution or are involved in the selected moves. This allows doing the **Diversification**, which forces or biases components that appear rarely in the current solutions.



Neighborhood Structure

- Set of solutions that could be reached from a given solution by applying a local move, operator or transformation



Intensification and diversification.

- Intensification, which refers to the process of exploiting a small portion of the search space, or penalizing solutions that are far from current solution.
- Diversification, which refers to forcing the search into previously unexplored areas of the search space, or penalizing solutions that are close to current.



2) Simulated Annealing (SA)

a. [10 marks]

- i. What is meant by the *cooling schedule* and what should be observed in selecting such schedule.
- ii. Explain the advantage and disadvantage of SA.
- iii. Mention some of the approaches that have been proposed to improve the computational cost of the basic SA method.



What is meant by the *cooling schedule* and what should be observed in selecting such schedule?

- The cooling schedule is the change of temperature across the search process (initial temperature, the temperature decrement rule, the number of iterations per temperature, and the final temperature). What should be observed? The initial temperature should be high enough to allow a move to possibly any state in the search space. The temperature should decrease slowly with enough number of iterations at every temperature for the system to be stable at that temperature.



Explain the advantage and disadvantage of SA.

- Advantages: Ease of use. Providing good solutions for a wide range of problems.
- Disadvantages: A lot of computer time for many runs. A lot of tunable parameters



Approaches to make SA adaptive

- Using lookup table to do the exponential calculation.
- Using non-exponential formulas to calculate acceptance probability.
- Using update functions for calculating the cost.



3rd example

2) **Tabu Search**

- a. **[10 marks]** Explain what is meant by the following and illustrate your answers using simple examples:
 - i. Short term memory and how it is used
 - ii. Intensification and diversification.
 - iii. Explain approaches that have been proposed to make TS adaptive.
- b. **[25 marks]** 7 Queen Problem



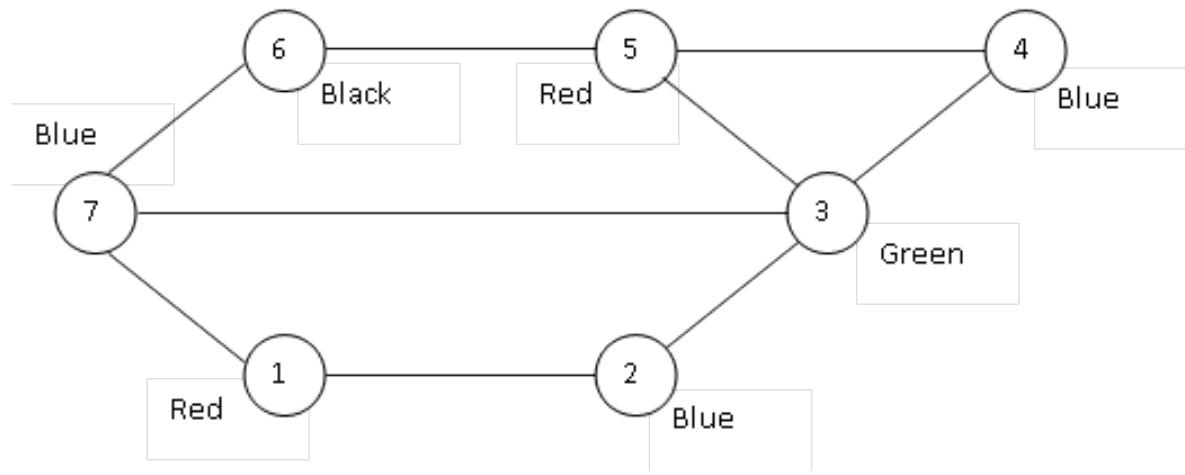
Approaches that have been proposed to make TS adaptive

- One of the adaptive approaches incorporated into TS is to allow the length of the short term memory (tabu list) to vary dynamically,
- **Method 1:** A range for the tabu list length is computed in advance according to the problem size, A new list length is randomly selected from this range every predetermined number of iterations.
- **Method 2:** The tabu list length is restricted between L_{min} and L_{max} , If the current solution is better than the best-so-far, the tabu list length is set to 1, If in an improving phase (the solution has improved over the last iteration) the tabu list length is decreased by 1, If not in an improving phase (the solution has deteriorated over the last iteration) the tabu list length is increased by 1, The values of L_{min} and L_{max} are randomly changed every A iterations.



3. **Simulated Annealing**

- a. **[10 marks]** Explain approaches that have been proposed to make SA adaptive.
- b. The graph node coloring is concerned with allocating a “color” to each node such that adjacent nodes are not given the same color. The objective is to find an allocation, which uses the smallest possible number of colors. The problem can be solved using SA. Consider the following graph and starting with a solution where nodes are colored as shown using 4 colors (**Red, Blue, Green, Black**).



- [5 marks]** Develop a representation for the possible solutions,
- [5 marks]** Define an operation on the representation to generate neighboring solutions,
- [5 marks]** Suggest a cost function (objective function) for this problem.
- [10 marks]** Perform few steps of SA till you reach a better solution.



Genetic Algorithms (GA)

a. [10 marks]

- I. Explain fitness based parent selection methods,
- II. Explain crossover operators suitable for real-valued problems,
- III. Explain mutation operators suitable for permutation type problem



i. Parent Selection

Fitness-Proportionate Selection (FPS)

Probability of parent selection

$$p_i = \frac{f_i}{\sum f_j}$$

- Expected number of copies of an individual i

$$E(n_i) = f_i / \bar{f}$$

(f_i is the fitness and \bar{f} is the average fitness



FPS

- Roulette wheel algorithm:
 - Given a probability distribution, spin a 1-armed wheel n times to make n selections
 - No guarantees on actual value of n_i
- Baker's Stochastic Universal Sampling algorithm:
 - n evenly spaced arms on wheel and spin once
 - Guarantees $\text{floor}(E(n_i)) \leq n_i \leq \text{ceil}(E(n_i))$



ii. Real-valued GAs

- Two types of crossover operators:
 - Discrete, use any of the crossover operators identified before for binary representations,
 - Intermediate (arithmetic):
 - Single arithmetic,
 - Simple arithmetic,
 - Whole arithmetic.



Real-valued GAs

- Single arithmetic crossover:
 - Parents: $\langle x_1, \dots, x_n \rangle$ and $\langle y_1, \dots, y_n \rangle$,
 - Pick random gene (k) at random,
 - Child 1 is:
$$\langle x_1, \dots, x_{k-1}, \alpha x_k + (1 - \alpha) y_k, \dots, x_n \rangle$$
 - Reverse for other child.



Real-valued GAs

- Simple arithmetic crossover:
 - Parents: $\langle x_1, \dots, x_n \rangle$ and $\langle y_1, \dots, y_n \rangle$,
 - Pick random gene (k), after this point mix the values,
 - Child 1 is:

$$\langle x_1, \dots, x_k, \alpha \cdot y_{k+1} + (1 - \alpha) \cdot x_{k+1}, \dots, \alpha \cdot y_n + (1 - \alpha) \cdot x_n \rangle$$

- Reverse for other child.



Real-valued GAs

- Whole arithmetic crossover:
 - Most commonly used,
 - Parents: $\langle x_1, \dots, x_n \rangle$ and $\langle y_1, \dots, y_n \rangle$,
 - Child 1 is:

$$a \cdot \bar{x} + (1 - a) \cdot \bar{y}$$

- Reverse for other child.



iii. Permutations GAs

- Insert mutation:
 - Pick two genes values at random,
 - Move the second to follow the first, shifting the rest along to accommodate,
 - Preserves most of the order and the adjacency information,

1	2	3	4	5	6	7	8	9
---	---	---	---	---	---	---	---	---



1	2	5	3	4	6	7	8	9
---	---	---	---	---	---	---	---	---



Permutations GAs

- Swap mutation:
 - Pick two genes at random and swap their positions,
 - Preserves most of adjacency information (4 links broken), disrupts order more.

1	2	3	4	5	6	7	8	9
---	---	---	---	---	---	---	---	---



1	5	3	4	2	6	7	8	9
---	---	---	---	---	---	---	---	---



Permutations GAs

- Inversion mutation:
 - Pick two genes at random and then invert the substring between them,
 - Preserves most adjacency information (only breaks two links) but disruptive of order information.

1	2	3	4	5	6	7	8	9
---	---	---	---	---	---	---	---	---



1	5	4	3	2	6	7	8	9
---	---	---	---	---	---	---	---	---



Permutations GAs

- Scramble mutation:
 - Pick two gene values at random,
 - Randomly rearrange the genes in those positions (note subset does not have to be contiguous)

1	2	3	4	5	6	7	8	9
---	---	---	---	---	---	---	---	---



1	3	5	4	2	6	7	8	9
---	---	---	---	---	---	---	---	---



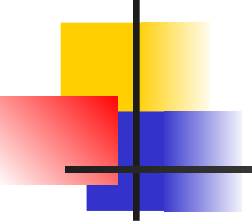
b.

i- A suitable representation will be a vector where the index is the location and the value is the facility assigned to the location

ii- Cost function can be
$$\sum_{i=1}^n \sum_{j=1}^n F_{p_i p_j} D_{ij}$$

iii- Crossover operator can be Order 1 crossover

iv- for mutation we can use swap mutation

- 
-
- v. GA steps
 - Initialize a population of n random assignments.
 - Evaluate all the individual
 - While termination conditions is not satisfied
 - Select parents (according to FPS)
 - Select a random number $r1$. If probability of crossover $> r1$ Apply crossover
 - Select a random number $r2$. If probability of mutation is $> r2$, apply mutation
 - Apply survival criterion (such as elitist) to form the new generation