A Course on Meta-Heuristic Search Methods for Combinatorial Optimization Problems

Santosh Kumar Mandal, *Ph.D research fellow*

AutOrI LAB,
DIA, Roma Tre
Email: mandal@dia.uniroma3.it

January 20, 2014

Outline

Tabu search
working process

Input: $s^{(o)}$ - the initial solution;
Output: s^* - the best found solution;
Initialize the Tabu List T;
set: Aspiration criteria;
set: $s = s^{(o)}$ and $s^* = s$;
Repeat
Generate solutions in the neighborhood of s;
Select the best possible solution $s' \notin T$ or satisfying the aspiration criteria;
set $s = s'$;
Insert the solution s (or its attribute) into the tabu list T;
if $f(s) < f(s^*)$
set $s^* = s$;
end if;
Update the tabu list T;
Until (stopping condition is satisfied);

Table: Pseudocode of Tabu Search

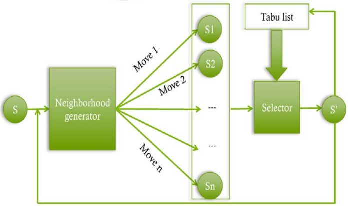


Figure: Tabu search

Tabu search

components

Aspiration criteria:

- if the tabu solution is better than the best found solution.
- if the tabu solution possesses a particular attribute.

Tabu search
components

Tabu list:

- A short term memory
- Stores visited solutions or moves/solutions attributes
- Prevents cycling
- The length of the list, called tabu tenure, controls diversification.

Tabu search

components

Types of Tabu list:

- Static [tabu tenure: 3-10].
- Dynamic: The size changes during the search in a given interval (Robust Tabu Search Algorithm)
- Adaptive: The size is increased or decreased according the search information (e.g, Reactive Tabu search increases the tabu list if cycling occurs.)

Tabu search
tabu list

An illustration on the travelling salesman problem (tabu tenure = 3):

Starting solution: Value = 234

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

Tabu list:

	2	3	4	5	6	7
1	0	0	0	0	0	0
	2	0	0	0	0	0
		3	0	0	0	0
			4	0	0	0
				5	0	0
					6	0

Figure: Iteration 0

Tabu search
tabu list

Current solution: Value = 234

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

Candidate list:

Exchange	Value
5.4	-34
7.4	-4
3.6	-2
2.3	0
4.1	4

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

After move: Value = 200

Tabu list:

	2	3	4	5	6	7
1	0	0	0	0	0	0
	2	0	0	0	0	0
		3	0	0	0	0
			4	3	0	0
				5	0	0
					6	0

Figure: Iteration 1

Tabu search

tabu list

Current solution: Value = 200

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

Candidate list:

Exchange	Value
3.1	-2
2.3	-1
3.6	1
7.1	2
6.1	4

← Choose move (3,1)

Tabu list:

	2	3	4	5	6	7
1	0	0	0	0	0	0
	2	0	0	0	0	0
		3	0	0	0	0
			4	3	0	0
				5	0	0
					6	0

Figure: Iteration 2

Tabu search
tabu list

Current solution: Value = 200

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

Candidate list:

Exchange	Value
3.1	-2
2.3	-1
3.6	1
7.1	2
6.1	4

← Choose move (3,1)

Update tabu list

Tabu list:

	2	3	4	5	6	7
1	0	3	0	0	0	0
2		0	0	0	0	0
3			0	0	0	0
4				2	0	0
5					0	0
6						0

Figure: Iteration 2

Tabu search
tabu list

Current solution: Value = 198

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

Candidate list:

Exchange	Value
1.3	2
2.4	4
7.6	6
4.5	7
5.3	9

Tabu!

Choose move (2,4)

NB: Worsening move!

Tabu list:

	2	3	4	5	6	7
1	0	3	0	0	0	0
2		0	0	0	0	0
3			0	0	0	0
4				2	0	0
5					0	0
6						0

Figure: Iteration 3

Tabu search
tabu list

Current solution: Value = 198

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

Candidate list:

Exchange	Value
1.3	2
2.4	4
7.6	6
4.5	7
5.3	9

Tabu! (1.3)

Choose move (2,4)

NB: Worsening move!

Tabu list:

	2	3	4	5	6	7
1	0	2	0	0	0	0
	2	0	3	0	0	0
		3	0	0	0	0
			4	1	0	0
				5	0	0
					6	0

Update tabu list

Figure: Iteration 3

Tabu search
tabu list

Current solution: Value = 202

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

Candidate list:

Exchange	Value
4.5	-6
5.3	-2
7.1	0
1.3	3
2.6	6

Tabu!

Choose move (4,5)

Aspiration!

Tabu list:

	2	3	4	5	6	7
1	0	2	0	0	0	0
	2	0	3	0	0	0
		3	0	0	0	0
			4	1	0	0
				5	0	0
					6	0

Figure: Iteration 4

Tabu search
tabu list

Observations:

- In the example 3 out of 21 moves are prohibited.
- More restrictive tabu effect can be achieved by
    - Using stronger tabu-restrictions
    - Using OR instead of AND for the 2 cities in a move

Tabu search
strategies

Intensification:

- Use a medium term *recency memory*, which will memorize for each specified component the number of successive iterations the component is present in the visited solutions.
- Start the intensification process in a given period or after a certain number of iterations without improvement.
- Start the search with the best solution obtained, introducing the most visited component(s).

Tabu search

strategies

Diversification:

- Restart
- Continuous
- Strategic oscillation

Tabu search
diversification

Restart diversification:

- Use a long term *frequency memory*, which will memorize for each specified component the number of times the component is present in all visited solutions.
- Start the diversification process periodically or after a certain number of iterations without improvement.
- Start the search with the best solution obtained, introducing the least visited component(s).

Tabu search
strategies

Continuous diversification:

≫ This is achieved by penalizing worsening moves.

$$f(x) := f(x) + \delta_{penalty} \tag{1.1}$$

For VRP (Taillard (1993)):

$$\delta_{penalty} = \gamma \times \sqrt{mn} \times fr_u \tag{1.2}$$

fr_u: frequency of moving vertex u in the past.

Tabu search
diversification

Strategic oscillation:

- Proceed beyond the feasible boundary for a set depth.
- Turn around to enforce feasibility.

For CVRP:

$$f(x) = f(x) + \alpha \times |Q(x)| \quad (1.3)$$

$Q(x)$: total violation in the loading capacity.

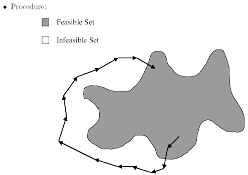


Figure: strategic oscillation

Genetic algorithm
working process

Quick overview:

- Developed by Holland (1975).
- A population based meta-heuristic.
- Based on Darwinian's principle of *competition*.
- A very successful algorithm, but not too fast.

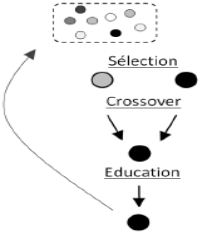


Figure: GA illustration

Genetic algorithm

Components:

- Solution representation
- Population initialization
- Fitness function
- Parent selection mechanism
- Crossover and mutation operators
- Survivor selection
- Parameter settings

Genetic algorithm
components

Solution representation:

≫ In the Genetic algorithm, the encoded solution is referred as chromosome while the decision variables within a solution (chromosome) are genes. The possible values of variables (genes) are the alleles and the position of an element (gene) within a chromosome is named locus.

0	1	1	0	0	0	1	0	1	0

Table: Binary chromosome

10	8	7	1	2	5	3	9	4	6

Table: Permutation chromosome

0.23	0.10	1.0	0.89	0.95	0.64	1.0	0.45	0.76	0.34

Table: Real-valued chromosome

Genetic algorithm
components

Population initialization:

- Uniformly at random
- Use any heuristic

Facts:

- Lower population size: poor solution quality
- Higher population size: good solution quality, but more computational time
- Find a satisfactory balance point.

Genetic algorithm
components

Parent selection methods:

- Two widely used:
  - Tournament selection.
  - Roulette-wheel selection.
- Others:
  - Rank-based selection.
  - Sigma scaling
  - Boltzmann selection

Genetic algorithm
selection methods

Roulette wheel procedure:

step 1: Consider a roulette wheel and assign each solution i some portion of the wheel. The area of the wheel allocated to an individual solution i is equal to:

$$\frac{fitness(i)}{\sum_i fitness(i)} \times 100\ \% \qquad (2.1)$$

step 2: Rotate the wheel and select the solution corresponding to the selection point.

step 3: Inset a copy of the selected solution in the mating pool, and repeat the process until it is full (population size times).

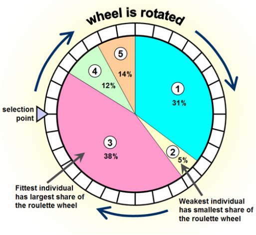


Figure: Roulette wheel illustration

Genetic algorithm

selection methods

Tournament selection:

step 1: Draw *t* solutions from the population and select the fittest one with some probability.

step 2: Inset a copy of the selected solution in the mating pool, and put solutions back into the population.

step 3: Repeat the process until the mating pool is full.

Genetic algorithm
selection methods

Tournament selection:

- *Deterministic tournament*: Always select the best one.
- *Binary tournament*: Only two players are involved.

Facts:

- Better in maintaining selection pressure than the Roulette wheel procedure.
- Tournament size should be set appropriately.

Genetic algorithm

components

Crossover/Recombination:

$\gg$ It is the process of generating new solutions, called off-springs, by mixing genes of two or more parent solutions from the mating pool. The operation is executed with some probability.

$\gg$ The crossover probability should be set high.

Genetic algorithm
crossover

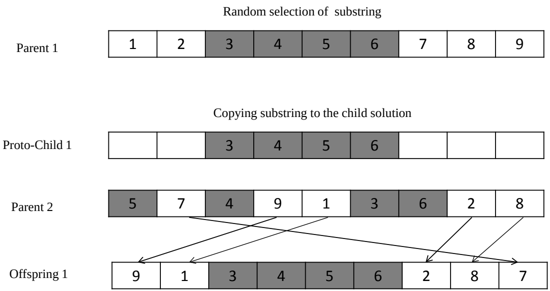


Figure: An illustration of Order crossover [Oliver et al. (1987)]

Genetic algorithm
crossover

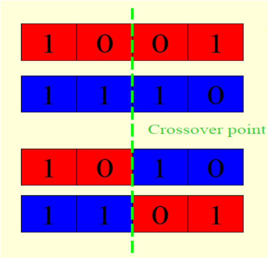


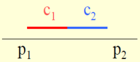
Figure: An illustration of one-point crossover

Genetic algorithm
crossover

β: Spread factor
$p_1 \& p_2$: Parent solutions
$c_1 \& c_2$: Off-springs

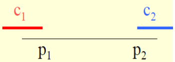
$$\beta = \left| \frac{c_1 - c_2}{p_1 - p_2} \right|$$

- Contracting Crossover $\beta < 1$



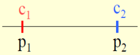
The offspring points are enclosed by the parent points.

- Expanding Crossover $\beta > 1$



The offspring points enclose the parent points.

- Stationary Crossover $\beta = 1$



The offspring points are the same as parent points

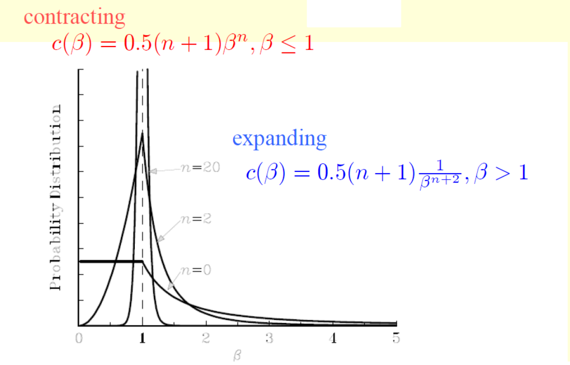
Genetic algorithm

crossover

Simulated binary crossover [Agrawal and Deb (1994)]:

≫ High values of n will create off-springs near the parents; vice versa in the case of low values of n.

$n = 2$ *for mono − objective problems*



Genetic algorithm
Simulated binary crossover

≫ Generates off-springs symmetrically about the parents.

$$c_1 = \frac{p_1 + p_2}{2} - 0.5 \times \beta^* \times (p_2 - p_1) \quad (2.2)$$

$$c_2 = \frac{p_1 + p_2}{2} + 0.5 \times \beta^* \times (p_2 - p_1) \quad (2.3)$$

To calculate β^*:

- Generate a random number μ in [0, 1]
- Get β value that makes area under the curve $= \mu$

Genetic algorithm
components

Mutation:
≫ In this process, the structure of off-spring generated via crossover is further changed slightly. The mutation is also with some probability.

Facts:

- The mutation probability (i.e. probability of mutating each gene) should be set low.

$$P_m = \frac{1}{L} \quad L : \textit{length of string} \tag{2.4}$$

- Probability of mutating an individual becomes

$$P_{string} = 1 - (1 - P_m)^L \tag{2.5}$$

Genetic algorithm
mutation

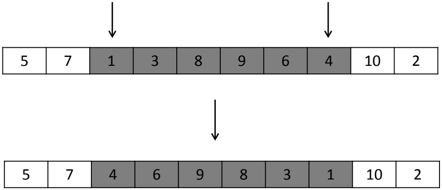


Figure: An illustration of Inversion mutation

Genetic algorithm
mutation

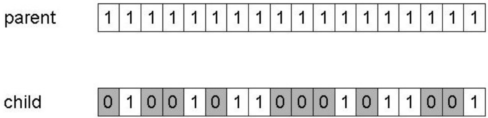


Figure: An illustration of flip mutation

Genetic algorithm

mutation

Polynomial mutation:

$P(\delta)$: Probability distribution function
$\eta_m = 20$ *is generally used*

To calculate δ_i:

- Generate a random number μ in [0, 1]
- Get δ value that makes area under the curve $= \mu$

$$x_i' = x_i + \left(x_i^u - x_i^L\right) \delta_i$$

$$P(\delta) = 0.5(\eta_m + 1)(1 - |\delta|^{\eta m})$$

Genetic algorithm
components

Survivor selection:
≫ Select the top best individuals among parent and offspring solutions for the next generation of search.

Facts:

- This strategy promotes faster convergence.
- Premature convergence may occur.

Agrawal, R. B. and Deb, K. (1994). Simulated binary crossover for continuous search space. Technical report, Indian Institute of Technology, Indian Institute of Technology, Kanpur.

Holland, J. H. (1975). *Adaptation in natural and artificial systems*. Ann Arbor: The University of Michigan Press.

Oliver, I. M., Smith, D. J., and Holland, J. R. C. (1987). A study of permutation crossover operators on the travellng salesman problem. In *Proceedings of 2nd International Conference on Genetic Algorithms and Their Application*, pages 224–230.

Taillard, É. D. (1993). Parallel iterative search methods for vehicle routing problems. *Networks*, 23:661–673.