ISM206: Metaheuristics

Professor: Kevin Ross

1 Introduction

**Heuristics:** A heuristic method is a procedure that is used to find a good feasible solution that is at least reasonably close to being optimal. A well designed heuristic method can usually provide a solution that is nearly optimal or can conclude that no solution exists. A disadvantage of heuristic methods is that it is usually designed to fit a specific problem type rather than a variety of applications.

**Metaheuristics:** Metaheuristics are general solution methods which combine general structure and strategy guidelines for developing a specific heuristic method to fit a particular kind of problem.

   Key ideas of metaheuristics:
   
   - Use the information being gathered to guide the search towards the global optimum.
   - It is capable to escaping from a local optima.
   - Examine Neighborhood Structure - some solutions are similar to other neighbour, so we need a neighbourhood structure.

   *The objectives of metaheuristics are:*
   
   - Search using structure without getting stuck.
   - Don’t keep trying the same solutions.
   - Combine one or more properties of good solutions when generating new solutions.

   *Advantage of metaheuristics:*
   
   - It tends to move relatively quickly towards very good solutions, so it provides a very efficient way of dealing with large complicated problems.
   - Useful in cases where traditional methods get stuck at local minimas.
   - Common area of application is combinatorial optimization problems.
Figure 1: An example graph used for the travelling salesmen problem.

*Disadvantage of metaheuristics:*
- There is no guarantee that the best solution found will be the optimal solution.

2 Basic Definitions

We will use the travelling salesmen problem (TSP) on the graph as an example problem for the metaheuristics discussed. Travelling Salesman Problem (TSP):

A salesman spends his time visiting n cities (or nodes) cyclically. Starting from the home city, the salesman wishes to determine which route to follow to visit each city exactly once before returning to the home city so as to minimize the total distance of the tour.

The difficulty of the travelling salesman problem increases rapidly as the number of cities increases. For a problem with n cities and a link between every pair of cities, the number of feasible routes to be considered is (n-1)!/2. Due to enormous difficulty in solving the TSP, heuristic methods guided by the metaheuristics, address such problems.

Heuristic methods involve sequence of feasible trial solutions, where each solution is obtained by making certain adjustment in current trial solution.

**Subtour Reversal:** Adjusting a sequence of cities visited in the current solution by selecting a subsequence of the cities and simply reversing the order.

Eg.
Initial trial solution is the following sequence of cities visited: 1-2-3-4-5-6-7-1 with total distance = 69.
While reversing the sequence 3-4, we obtain new trial solution: 1-2-4-3-5-6-7-1 with total distance = 65.

**Neighbors:** We say 2 tours/solutions/cycles are neighbors if we can transform one to the other by a subtour reversal.

**Degree of Neighbor:** The degree of a neighbor A to B equals the minimum number of subtour reversals required to get from A to B.

**Local Minimum:** A local minimum is when no neighbors are better i.e no neighbour’s subtour gives a better solution.

Problems like TSP have many local minima’s. If we look into the gradient search approach to solve TSP, the steps are:

- Pick a cycle
- Take best neighbour
- Repeat until local minima is obtained.

Because of local minima, this may not yield good solution.

## 3 Simulated Annealing

The process of simulated annealing is inspired by the physical process of annealing from chemistry. Annealing involves slowly heating the metal and then slowly cooling the substance by varying the temperatures until it reaches the low energy stable state, resulting in a global reduction in energy, but locally it may result in an increase in energy. These changes in energy follow a Boltzmann distribution.

The key idea in simulated annealing algorithm is to select an appropriate temperature schedule which needs to specify the initial, relatively large value of T and then decrease the value of T. Starting with relatively large values of T, makes probability of acceptance relatively large, which enables the search to proceed in almost all random directions. Gradually decreasing the value of T as the search proceeds, gradually decreases the probability of acceptance, which emphasizes on climbing upwards.

### 3.1 Acceptance probabilities

The probability of transiting from current state $X_n$ to a current trial solution $X'_n$ is specified by an acceptance probability function $P(e, e', T)$ which depends on the energies $e = E(X_n)$ and $e' = E(X'_n)$ of the two states and a global varying parameter called the *Temperature*. Essential points for designing $P$
• Must be nonzero when $e' > e$, meaning the system might move to the new state $X'_n$ even it is worse than the current one. This feature prevents the method from being stuck in local minimum.

• When $T$ goes to zero, $P(e, e', T)$ must tend to be zero when $e' > e$ and to a positive value if $e' < e$.

Originally simulated annealing is described as setting $P(e, e', T) = 1$ whenever $e' < e$, which means the algorithm moves downhill when there is a chance, irrespective of $T$.

3.2 The Simulated Annealing Algorithm

$X_n$ is the solution after $n$ iterations.
$X_n'$ is the current trial solution.
$T$ is the parameter that measures the tendency to accept the current solution as next trial solution.

The 'move selection rule' is then based on selecting which neighbour will be the next trial solution.

- Select $X_n'$ neighbor of $X_n$
- If neighbour is 'better' i.e. $f(X_n')$ is less than $f(X_n)$, then accept with probability $p_n$, $X_{n+1} = X_n'$.
- If neighbour is 'worse' i.e. $f(X_n')$ is greater than $f(X_n)$, then accept with probability $p_n$, $X_{n+1} = X_n'$.
- or reject with probability $(1 - p_n)$, $X_{n+1} = X_n$.
- we could have $p_n = e^{-\frac{f(X_n') - f(X_n)}{T}}$.
- We let Temperature ($T$) go to zero over time.

To use this method on the TSP we let a subtour reversal define a neighbor and the random selection of neighbors corresponds to randomly selecting the first and last city to reverse.
1. Initial trial solution.
2. Neighbourhood structures.
3. Random selection of immediate neighbour.
4. Temperature schedule: A sample temperature schedules is (where changes are made every $a$ iterations).

\[
T_1 = 0.2f(x) \\
T_2 = 0.5T_1 \\
T_3 = 0.5T_2
\]
The current solution to be the next trial solutions can be an improvement or can be worst than the current solution, therefore we should use the algorithm to accept or reject the solution.

### 3.3 Travelling Salesman Example

Initial trial solution: 1-2-3-4-5-6-7-1 Distance = 69

Iteration 1: Reverse (3-4) 1-2-4-3-5-6-7-1 Distance = 65
Accept as new solution.

Iteration 2: Reverse (3-5-6) 1-2-4-6-5-3-7-1 Distance = 64
Accept as new solution.

Iteration 3: Reverse (3-7) 1-2-4-6-5-7-3-1 Distance = 66
Since the new distance is "worse," accept as new solution with some probability

Continue for some fixed number of iterations, or until temperature function falls below a given threshold.

### 4 Tabu Search

The key process is finding the local optima and then continue the search by allowing non improving moves (or illegal moves, called as tabu moves) to the best solutions in the neighbourhood of the local optima.

One of the features of tabu search is to avoid bad solutions which have already been explored i.e use of memory to guide the search by using tabu list to record recent searches. Essentially, Tabu search makes some moves illegal by maintaining a list of 'tabu' moves.

For example, if A is a neighbor of B in the TSP then B is a neighbor of A. But if you have already chosen B over A, there might not be any reason to search A again.

### 5 Some Useful Definitions

- **Intensify:** To intensify the search is to search the local area( portion of feasible region) more thoroughly.
- **Diversify:** To diversify the search is to force the search away from the current solution( to unexplored areas of feasible region).
• **Length of Tabu List:** The length of the list signifies the balance between intensify/diversify.

### 5.1 The Tabu Search Algorithm

- **Initialize**
- **Iteration:**
  - Compare all possible moves
  - Take best (even if it is worse than the current solution)
  - Update List
  - Stop after a fixed time or CPU usage, or there are no feasible moves.

The optimal solution is the best solution so far.

### 5.2 Travelling Salesman Example

**Initial trial solution:** 1-2-3-4-5-6-7-1  
**Distance = 69 Tabu list : Null**

**Iteration 1:** Reverse 3-4  
**Deleted links:** 2-3 and 4-5  
**Added links:** 2-4 and 3-5  
**Tabu List:** (2-4), (3-5)  
**New trials Solution:** 1-2-4-3-5-6-7-1  
**Distance = 65**

**Iteration 2:** Reverse: 3-5-6  
**Delete links:** 4-3 and 6-7  
**Add links:** 4-6 and 3-7  
**Tabu List:** 2-4, 3-5, 4-6, 3-7  
**New Solution:** 1-2-4-6-5-3-7-1  
**Distance = 64**

Keep running for more iterations until algorithm terminates at a point where we obtain the best trial solution as the final solution.

While running Tabu search it is necessary to keep track of the Tabu list and the best solution so far. So, delete few links from the tabu list after some number of iterations and while doing this, delete the oldest links of tabu list.

**Advantage:**  
This method is easy to integrate with other methods.

**Disadvantage:**  
Tabu search approach is to climb the hill in the steepest direction and stop at top and then climb downwards to search for another hill to climb. The drawback
is that a lot of iterations are spent climbing each hill rather than searching for tallest hill.

6 Genetic Algorithms

Genetic Algorithms are a class of optimization algorithms based on “survival of the fittest” and combining solutions (parents and children). Here feasible solutions to particular problems correspond to members of species. Fitness of each member is measured by the objective function. The basic idea is that each possible solution is a member of entire population of trial solutions, and any given population is keeping track of multiple solutions.

Parents in a genetic algorithm are selected at random from the available population, and the new trial solutions (children) are created from the parents. When these children are added to the population they occasionally have mutations which add more variety to the population.

*Important property:* When going through a genetic algorithm a good solution is more likely to survive and hence more likely to reproduce.

6.1 A framework for a Genetic Algorithm

1. Choose the initial population of individuals
2. Evaluate the fitness of each individual in that population
3. Repeat on this generation until termination: (time limit, sufficient fitness achieved, etc.)
   (a) Select the best-fit individuals for reproduction
   (b) Breed new individuals through crossover and mutation operations to give birth to offspring
   (c) Evaluate the individual fitness of new individuals
   (d) Replace least-fit population with new individuals

• Initialize the population of solution
• Randomly Select Parents to combine
• Generate Child based on parents
  – keep the child (add to population) if good or adds improvements or
  – reject if infeasible or bad
– mutate randomly

- Possibly throw out all solutions if necessary.

Many decisions affect the effectiveness of genetic algorithm for any particular problem:

- Population Size
- Selection Rule
- Combination rule
- Mutation Rule
- Stopping Rule

A genetic algorithm is often good for solving hard optimization problems which can easily be represented in binary.

6.2 An example solution for the TSP using a genetic algorithm

Some characteristics of the TSP when represented to be used with a genetic algorithm: Need a rule for joining parents (or 2 subtours). Parents are tours and the current city is the home city of a path.

6.2.1 Algorithm

- Identify all links from current in either parent that are not already in the tour.
- Randomly select one of the available.
- Check for mutation.
- The next city is one these
- Use this link to complete the tour.

6.2.2 Example

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

Generate a child:
1-2-4-5-6-7-3-1

The full example is in the text.
Parents are selected using a fitness function, if we were given the choice amongst the following:

1. \( f(x_1) = 69 \)
2. \( f(x_2) = 65 \)
3. \( f(x_3) = 79 \)
4. \( f(x_4) = 86 \)

We would choose numbers 1 and 2 as they are the lowest distance. Alternatively for some problems we would choose to select those which have the highest fitness.

Have mutations to randomly allow other links as this help genetic algorithms to explore a new/better feasible solution. Essence is that parents generate children (new trial solutions) who share some features of both parents. Since the fittest members of the population are more likely to become parents than others, a genetic algorithm tends to improving populations of trial solutions as it proceeds.

7 References

- Text Book
- Class Notes
- Last years scribe notes for Metaheuristics