1 Introduction

Metaheuristics are general solution methods which combine clues given by locality with a bigger picture. Key ideas of metaheuristics:

- Examine Neighborhood Structure
- Useful in cases where traditional methods get stuck at local minimas.
- Some problems are too large and complicated to be solved by traditional methods, but metaheuristics may make some progress.

2 Basic Definitions

We will use the travelling salesmen problem (TSP) on the graph in figure 1 as an example problem for the metaheuristics discussed. A definition of the travelling salesmen problem taken from http://www.pcug.org.au/dakin/tsp.htm:

A salesman spends his time visiting n cities (or nodes) cyclically. In one tour he visits each city just once, and finishes up where he started. In what order should he visit them to minimize the distance travelled?

**Subtour Reversal:** A reverse order of some subsequence in a graph. Eg. 1-2-3-4-5-6-7-1 becomes 1-2-4-3-5-6-7-1.

**Neighbors:** We say 2 "tours" 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 number of subtour reversals required to get from A to B. It must be always be possible to get from A to B.

![Figure 1: An example graph used for the travelling salesmen problem.](image-url)
Local Minimum: A local minimum is when no neighbors are better.
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.

3 Simulated Annealing
The process of simulated annealing is inspired by the physical process of annealing from chemistry. Annealing involves slowly heating and cooling materials to strengthen them. When cooling, annealing results in a global over time reduction in energy, but locally there may result in an increase in energy. These changes in energy follow a Boltzmann distribution.

3.1 The Simulated Annealing Algorithm

\[ X_n \] is the solution after \( n \) iterations.
- Select \( X_n \) neighbor of \( X_n \)
- If \( f(X_n) \) is an improvement then \( X_{n+1} = X_n \).
- If not then still accept with probability \( p_n \).
  - else \( X_{n+1} = x_n \)
- For example we could have \( p_n = e^{\frac{f(X_n) - f(x_n)}{T_n}} \)
- 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.

A sample temperature schedules is (where changes are made every \( a \) iterations):

\[ T_1 = 0.2f(x) \]
\[ T_2 = 0.5T_1 \]
\[ T_3 = 0.5T_2 \]

4 Tabu Search
Tabu search is an approach which seeks to avoid bad solutions which have already been explored. 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. Essentially, Tabu search makes some moves illegal by maintaining a list of ‘tabu’ moves.

5 Some Useful Definitions
- **Intensify:** To intensify the search is to search the local area more thoroughly.
- **Diversify:** To diversify the search is to force the search away from the current solution.
- **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

Start with 1-2-3-4-5-6-1

Iteration 1: Reverse 3-4
Deleted: 2-3 and 4-5
Added: 2-4 and 3-5
Tabu List: (2-4), (3-5)
New Solution: 1-2-4-3-5-6-7-1

Tabu = what would take you back to the previous solution
Iteration 2:
Reverse: 3-5-6
Delete: 4-3 and 6-7
Add: 4-6 and 3-7
Tabu List: 2-4, 3-5, 4-6, 3-7
New Solution: 1-2-4-6-3-5-7-1

While running Tabu search it is necessary to keep track of the Tabu list and the best solution so far. A modification to Tabu search which is often used is to have the tabu list only hold moves for a fixed number of iterations.

6 Genetic Algorithms

Genetic Algorithms are a class of optimization algorithms based on “survival of the fittest”. The basic idea is that each possible solution is a member of a population, and any given population is keeping track of multiple solutions. When going through a genetic algorithm a good solution is more likely to survive and hence more likely to reproduce.

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.

6.1 A framework for a Genetic Algorithm

- Initialize
- Randomly Select Parents
- Generate Child
keep (add to population) or
− mutate and reject if infeasible

• Possibly throw out all solutions if necessary.

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

• Population Size
• Selection Rule
• Feature combination to produce children
• Mutation
• 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:

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.