UNIVERSITY OF PADUA

Department of Mathematics

Computer Science Master Degree

Methods and Models for Combinatorial Optimization project

Beccaro Alex

Professors

De Giovanni Luigi

Di Summa Marco

06/02/2018

Contents

1   Abstract

The project is composed by two parts:

- Implementing an Integer Linear Programming model with the Cplex API

- Implementing a meta-heuristic solution method

For the second part my choice of method is a genetic algorithm, details on implementation and decisions in section 3. Both solution methods will be applied to the following problem:

> A company produces boards with holes used to build electric frames. Boards are positioned over a machines and a drill moves over the board, stops at the desired positions and makes the holes. Once a board is drilled, a new board is positioned and the process is iterated many times. Given the position of the holes on the board, the company asks us to determine the hole sequence that minimizes the total drilling time, taking into account that the time needed for making an hole is the same and constant for all the holes.

The two different approaches are then tested using some benchmark tests from literature and compared to spot differences in results (optimality, execution time, reliability, ...)

2   Integer Linear Programming Model

2.1   Network Flow Model Representation

The problem can be represented on a complete weighted graph $G = (N, A)$ where N is the set of nodes (holes on the board) and A is the set of the arcs $(i, j), \forall i, j \in N$ (trajectory of the drill moving from hole i to hole j).To each arc will be associated a weight c_{ij} that represents the time needed to move from starting hole to destination.

Using this representation the problem is equivalent to determine the minimum weight hamiltonian cycle on G, so it is like the very popular Travelling Salesman Problem (TSP).

To solve the TSP problem we can formulate it as a network flow model on G. Given a starting node $0 \in N$, let $|N|$ be the amount of its output flow, the problem can be solved by finding the path for which:

- Each node receives 1 unit of flow

- Each node is visited once

- The sum of costs in selected arcs is minimum

2.2   Variables and Constraints

The problem can then be represented like this:

SETS:

N = the graph nodes (holes)
A = arcs in the form $(i, j), \forall i, j \in N$ (trajectories between holes)

PARAMETERS:

c_{ij} = time taken by the drill to move from i to j, $\forall (i, j) \in A$ 0 = starting hole, $0 \in N$

DECISION VARIABLES:

x_{ij} = amount of the flow shipped from i to j, $\forall (i,j) \in A$ $y_{ij} = 1$ if arc (i,j) ships some flow, 0 otherwise, $\forall (i,j) \in A$

OBJECTIVE FUNCTION:

$$min \sum_{i,j|(i,j) \in A} c_{ij} \cdot y_{ij}$$

CONSTRAINTS:

$$\sum_{j|(0,j) \in A} x_{0j} = |N|$$

$$\sum_{i|(i,k) \in A} x_{ik} - \sum_{j|(k,j) \in A} x_{kj} = 1 \qquad \forall k \in N \setminus \{0\}$$

$$\sum_{j|(i,j) \in A} y_{ij} = 1 \qquad \forall i \in N$$

$$\sum_{i|(i,j) \in A} y_{ij} = 1 \qquad \forall j \in N$$

$$x_{ij} \leq |N| \cdot y_{ij} \qquad \forall (i,j) \in A$$

$$x_{ij} \in \mathbb{Z}_+ \qquad \forall (i,j) \in A$$

$$y_{ij} \in \{0,1\} \qquad \forall (i,j) \in A$$

4

3   Genetic Algorithm

Genetic algorithms are a population based heuristic method that simulates nature's evolutionary system to evolve a starting population of random solutions to obtain very good (possibly optimal) solutions at the end of the process. To do this a random starting population is generated, then each solution is evaluated with a fitness function. After that some solutions, usually the best ones, are chosen to create offsprings that will replace worst solutions. This way the algorithm performs a sort of natural selection keeping the best solutions and improving them over generations.

3.1   Encoding

To represent a TSP solution the path representation has been used, it consists in a sequence of N nodes, where N is the total number of nodes, sorted from the starting one to the last visited on encoded path.

For example the path $0 \rightarrow 3 \rightarrow 1 \rightarrow 4 \rightarrow 2 \rightarrow 0$ is encoded as:

$$[0, 3, 1, 4, 2]$$

3.2   Fitness

As fitness concerns I took a different approach from usual genetic algorithms and considered the sum of costs in path as the fitness and I will try to minimize it. This choice was taken because other considered fitness functions had drawbacks, mainly because most of them were in the form $upperBound - costs$ but upper bounds are not so strict and then a proportional choice based on fitness would yield a very similar probability for each individual (because $maxFitness - minFitness$ is much lower than $upperBound - anyFitness$).

By setting the fitness function equal to the sum of path costs it it directly proportional to objective function resulting in a more correct reward based on fitness.

3.3   Select

The select operator is a little tricky because of the fact that solutions with lower fitness should have higher probabilities, so both Montecarlo and Linear ranking methods can't be used. N-Tournament is a valid alternative, but it was discarded because if a subset of k solutions is chosen, the $k-1$ worst solutions are never used for crossover and this reduces (slightly) diversification.

The final choice is a probability calculated based on position in an array of solutions sorted by fitness. The selected solution is the one with index

$$index = \lfloor N * r^2 \rfloor$$

where N is the total number of nodes and r is a random number $\in [0, 1)$

This way the solutions with lower cost are more likely to be chosen than the others, but each solution might be selected.

3.4   Crossover

The crossover operator used in the project is the Order Crossover (OX), that works like this:

```
Parent 1: 8 4 7 3 6 2 5 1 9 0
Parent 2: 0 1 2 3 4 5 6 7 8 9

Child 1:  0 4 7 3 6 2 5 1 8 9
```

Figure 1: Example of Order Crossover (OX)

1. Select a random subsequence of consecutive alleles from parent 1. (underlined)

2. Drop the subsequence down to Child and mark out these alleles in Parent 2.

3. Starting on the right side of the subsequence, grab alleles from parent
   2 and insert them in Child at the right edge of the subsequence. Since
   8 is in that position in Parent 2, it is inserted into Child first at the
   right edge of the subsequence.

   Notice that alleles 1, 2 and 3 are skipped because they are marked out
   and 4 is inserted into the 2nd spot in Child.

3.5   Mutation

Only one type of mutation is used and it is the 2-opt, an algorithm that
takes a substring of the path representation and reverse it. The idea is to
take a route that crosses over itself and reorder it so that it does not. This
is good for the main purpose of mutations (prevent genetic drift) as well as
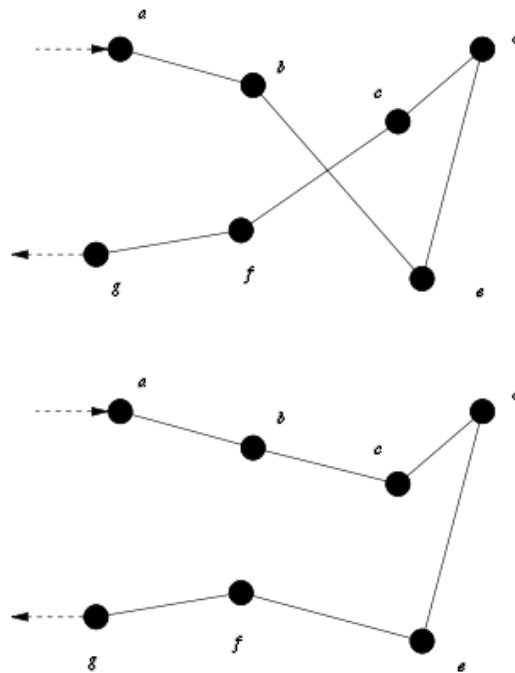for improving solutions.

Figure 2: Example of 2-opt mutation

To control the mutation there is a parameter MUTATION_CHANCE that
sets the probability that this mutation happens.

3.6   Generational Replacement

When a population advance to next generation we need to replace the old bad solutions with newly created offsprings. This is done by elitist, so only the best solutions are kept and all the others are replaced by new solutions. The percentage of survivors (and replaced solutions) is set by the parameter SURVIVAL_RATIO.

4   Tests

TODO

5   Comparison

TODO

6   Conclusions

TODO

A   Bibliography

- Project pt. 1: http://www.math.unipd.it/~luigi/courses/metmodoc/z01.eserc.lab.01.en.pdf

- Project pt. 2: http://www.math.unipd.it/~luigi/courses/metmodoc/z02.eserc.lab.01.en.pdf