Route optimization for a fleet of vehicles with temporal constraints

Algorithmics Methods and Mathematicals Models

Master in Research and Innovation - UPC

Juan Francisco Martínez Vera
juan.francisco.martinez@est.fib.upc.edu

January 11, 2017

Abstract

Optimization problems can appear in almost all situations on life. There are specially important those ones that appears on the industry because if we can achieve an optimal solution, we will improve the efficiency of the industrial process and then get lower manufacturing cost. This improvement of the costs have an effect on the competitiveness of the companies and on the final quality of their products and there are always good news for the customers. The challeging part is that those kind of problem have a really high computational complexity therefore there are several methods to face with them. By one hand we have the, let's say, always-optimality methods that obtain the optimal solution but without taking into account the huge amount of time that it could imply. e.g. ILP[1]. By the other hand we have those methods that concerns about the execution time and are looking for a tradeof between the exeution time and the quality of the solution[2].

In this pages will be shown this two approaches in order to get the optimal routes for a fleet of vehicles taking into account temporal constraints.

[1]Integer Linear Programming
[2]The distance to the optimal

Contents

Chapter 1

Introduction

Optimization problems can appear in almost all situations on life. There are specially important those ones that appears on the industry because if we can achieve an optimal solution, we will improve the efficiency of the industrial process and then get lower manufacturing cost. This improvement of the costs have an effect on the competitiveness of the companies and on the final quality of their products. The challeging part is that those kind of problem have a really high computational complexity therefore there are several methods to face with them. By one hand we have the, let's say, always-optimality methods that obtain the optimal solution but without taking into account the huge amount of time that it could imply. e.g. ILP[1]. By the other hand we have those methods that concerns about the execution time and are looking for a tradeof between the exeution time and the quality of the solution[2].

In this project has been developed an Integer Linear Programming model in order to optimize the routes for a fleet of vehicles with temporal constraints derived from the distance between locations and from the tasks that has to be done for every location. Also have been developed different meta-heuristic strategies in order to get results more efficiently in terms of exeuction time even if there are not the optimal but close enough. Finnaly comparissons in term of execution time and solution quality have been done.

1.1 Document structure

The structure of this document is the following: The problem definition is done in chapter 2. Here it is explained the problem that is faced, which constraints is needed to take into account and what we want to optimize. At chapter 3 is explained the mathematical model that has been developed, i.e. the decission variables and also the constraints with a mathematical nomenclature. After this chapter, at chapter 4 is explained how it has been used the heuristics approach in order to face with the problem, two meta-heuristics has been used: GRASP[3] and BRKGA[4]. After perform several executions for those approaches, a comparisson in terms of time and quality of the result is done at chapter 5. Finnally a discussion about the results obtained is done at chapter 6 and the conclusions of the project are explained at chapter 7

[1] Integer Linear Programming
[2] The distance to the optimal
[3] Greedy Adaptative Search Procedure
[4] Biased Random-Key Genetic Algorithm

Chapter 2

Problem definition

2.1 Description

We have been asked to help one logistic company in the design of the daily routes of its fleet of vehicles. The workload for every day consist in some tasks that have to be done at a different locations, therefore, the problem is to identify the optimal routes for the minimum number of vehicles in order to perform all the work in time, i.e. starting at 8 a.m. and finalizing before 8 p.m.

For that purpose we are given a set of locations and a start location l_s where all the vehicles will start. For our point of view, the locations are not needed at all but we need the distance in terms of time between all of them. Because that we will be provided by the $dist_{l_1,l_2}$ input data.

As is already mentioned, in every location (but not in l_s) there is a task that has to be done, then, also the information about how many time is needed for every task is provided as input data. We are talking about $task_l$.

Finally one more constraint is imposed by this logistic company. Every task has to be done but can not be started at any hour of the day. There is a temporal windows for every task that describes when an specific task can be started. This windows consists on a lower boundary min_l and on a upper boundary max_l. Then the starting time can not be before min_l and after max_l.

Considering that the company has an unlimited number of vehicles, the goal of the project is to find the minimum number of vehicles needed to visit all locations and perform all tasks, and define their routes. Given two solutions with the same number of vehicles it is preferred the one in which the latest vehicle arrives at its final destination sooner.

2.2 Formal definition

In this section is defined how the problem will be managed, also the input variables, the constraints and the objective function. This is a formal definition, therefore is implementation independant. These definitions have been used both for ILP and for meta-heuristics.

2.2.1 A relaxed TSP with temporal constraints

This problem has been formally defined as a relaxed TSP[1] with temporal constraints. It is relaxed because allows more than one cycle, i.e. more than one travel salesman (vehicle), and it is with temporal constraints because in addition to the usual weighted edges (routes) in the original TSP, we have temporal windows and tasks times for every node (location). The key point is that all the cycles in the graph, i.e. all routes, **must imply the** l_s.

This point of view allows to manage this problem without taking into account vehicles. The number of vehicles will be, in fact, the number of cycles in the solution.

As in TSP we can represent the towns and routes as a directed weighted complete graph, then, we can describe a route of every vehicle by mean of a set of edges that describes a cycle. Remember that the cycle must implie the l_s node.

[1] Travel Salesman Problem

The definition of the graph is defined by the below expression. Is important to mention that the cardinal of the A set is like that because the distance of the travel from location a to location b could be different than the travel from b to a because the roads or whatever else.

$$G = (V, A) \quad where \quad \text{V: set of nodes, A: set of directed edges}$$

$$|V| = n, \text{being n = number of locations} \quad |A| = n^2 - n$$

Now, let's define the set the whole possible cycles in the graph as C where every cycle is defined as a subset of edges that fulfill all the conditions to be a cycle. Then, the set of all possible vehicles $VH \subset C$ are all the cycles that imples the l_s as is described in the next expression:

$$vh \in VH \Leftrightarrow \exists a \in vh \text{ s.t. } a_{source} = l_s \vee a_{destination} = l_s$$

We can describe a set of potential solutions for the problem $PS \subset VH$ as all of subsets of VH that cover all the locations without repeat any one of them. Those are potential solutions because for the moment we have not taked into account the temporal constraints. Finally, the set of feasible solutions will be $S \subseteq PS$. And the challenging point is to get the optimal solution from this set.

2.2.2 Input data

According the definition, this variables are provided and describes a single instance of this problem. The following list describe all the input data that is provided. In order to improve the understanding, the input data have been split into two subsets. By one hand all data related to the graph stuff, and by the other hand the input data related to temporal restrictions.

Graph input data

This input data is used in order to build up the graph that represents the problem and has been described several lines above. On next lines this input data is showed and explained.

- **Number of locations** that the vehicles have visit. From here we can define the V set that is the nodes set of the graph.

$$n \in \mathbb{N}$$

- **Starting location** where all vehicles are placed at the beginning of times.

$$l_s \in L$$

- **Distances** in minutes from one locations to the other. This data form the A set because it is the weights for the edges for our completed directed and weighted graph. Note that when $a = b$, then $dist_{l_a,l_b} = 0$

$$dist_{l_a,l_b} \in A \quad \forall l_a, l_b \in V$$

  Could be said that this is a temporal data but as this data is also needed to build up the graph, and is present on the original TSP as well, is prefered to be here.

Time constraints input data

This input data is used in order to build up the time constraints needed for the problem definition. On next lines this input data is shoed and explained.

- **Lower boundary** for the temporal windows for every location. It indicates from when a task can be executed. If one vehicles arrives before this time, it must wait. The temporal unit are the minutes. This minimum windows can be at most 720 that is the journey time in minutes $(8h * 60min/h = 720min)$

$$min_l \in [0, 720] \quad \forall l \in L$$

- **Upper boundary** for the temporal windows for every location. It indicates until when a task can be executed. Take into account that does not have sense that $max_l < min_l$

$$max_l \in [min_l, 720] \quad \forall l \in L$$

- **Time** that a task spends for every location.

$$task_l \in [0, 720], \forall l \in L$$

2.2.3 Constratints definition

In this section are explained the fundamental constraints that are implementation independant. As well as with input data on the previous section, in order to improve the understanding, the constraints are splitted into two subsets, the graph constraint and the time constraints.

Graph constraints

These constraints are derived from the format description of the problem as a directed weighted graph. All the solutions that exists in set VH (section 2.2.1) fullfil the graph constraints. Just for summing up, those constraints are

- Every location has to be visited exactly one time. It means that all cycles in a solution must cover all nodes and can not share any node.
- All cycles in a solution must involve the starting location.

Time constraints

These constraints have to guarantee that the time restrictions are fullfiled. Those constraints are:

- The arrival time to location l must be less or equal max_l.
- The arrival time to location l must be $max(arrivalTime, min_l)$. It means that if one vehicle arrive before min_l it will wait.
- The total time of a cycle must be less than 720 minutes that is 8 hours.

Note that the arrival time to a location is the sumation of the previous location arrival time plus the task of the previous location plus the time spent in the path (the weight of the edge).

2.2.4 Objective function definition

In order to get the optimal solution we need to clasify every solution by a number that can tell us the quality of every one of them. This number is provided by the **objective function**.

As the description of the problem said, we want to minimize the number of vehicles and when two solutions have the same number of vehicles, we want the solution that end the whole job before. We will need two variables for this purpose, by one hand, the number of vehicles, lets say $nVehicles$ and by the other hand the time when the last vehicle ends its job, lets say $lastArrival$. How to get the correct value for these two variable is implementation dependant and therefore will be explained on the next chapters.

The objective functions is described as:

$$Minimize \quad nVehicles * M + lastArrival$$

In this expression, the M is a big enough number. It is needed in order to priorize the solutions with less value for $nVehicle$. When two solutions have the same number of vehicles, then the important value is the $lastArrival$.

Chapter 3

ILP model

ILP[1] is the first of the two methods that has been used in this project. This method is always concerned on having the optimal solution without taking into account the amount of computational resources needed. This model is developed in CPLEX and the model implemented is described at the next sections.

3.1 Input data

From chapter 2 we already know which is the data that is provided. Now we need them in an specific shape in order to deal with them in CPLEX.

- **Number of locations** n is declared as an integer as well as **starting location**

```
int nLocations=...;
int startLocation=...;
```

- The **distances** are modeled as two-dimensional matrix of integers that have n^2 positions. If you take into account that the values on the diagonal of the matrix will not be taked into account, the total number of distances is $n^2 - n$ as we have described in the previous chapter.

```
range N=1..nLocations;
int distances[n in N, n2 in N]=...;
```

- The remaining data fields are related to the locations, because that they are defined as a vector on n positions. Those data fields are **lower boundary** and **upper boundary** of the temporal windows and the time spent in a **task**.

```
int task[n in N]=...;
int minW[n in N]=...;
int maxW[n in N]=...;
```

3.2 Decission variables

The solution of the problem will be encoded in a certain way in a decission variable. The decission variables are not only used for get the final result but also for enforcing some situation with constraints. Those variables unlike input data ones, are not statics and can change their value in order to fulfill the constraints (section 3.4), as is just said, and in the same time in order to maximize or minimize the objective function (section 3.3). Therefore, even if the important numbers for the solution are two. The number of used vehicles and the time when the last vehicle done the work is needed a sort of other decission variables. In the next list is presented a relation of the defined decission variable for this project.

[1] Integer Linear Programming

- The **tracked** variable. It is a nxn matrix and keep information about whether a path from location n to location $n2$ is taked, i.e. keep information about which edges are on the solution. It is needed in order to control the number of input and output edges for every location and for calculate the time spent in a route.

```
dvar boolean tracked[n in N, n2 in N];
```

- In order to build the temporal constraints that we have at section 2.2.3 we need information about when a vehicle arrives to a location. Because that the **arrivingTime** decission variable has been defined. It is an n-vector that keeps the arriving time for all locations (nodes).

```
dvar int arrivingTime[n in N];
```

- One of the most important decission variables is the numbers of vehicles needed. **nVehicles** keep the number of vehicles in the solution.

```
dvar int nVehicles;
```

- The other really important value for our result is the end time of the last vehicle. **lastDone** have the time in minutes of the last arrival vehicle.

```
dvar int lastDone;
```

3.3 Objective function

3.4 Constraints

blah blah blah

Chapter 4

Meta-heuristics

Chapter 5

Comparissons

Chapter 6

Discussion

Chapter 7

Conclusions