TECHNISCHE UNIVERSITÄT BERLIN
FAKULTÄT IV ELEKTROTECHNIK UND INFORMATIK
BACHELORSTUDIENGANG INFORMATIK

FERNANDO BOMBARDELLI DA SILVA

Solving The Dial-a-Ride Problem With The Firefly Metaheuristic

Work presented in partial fulfillment
of the requirements for the degree of
Bachelor in Computer Science

Advisor: Dr.-Ing. Axel Heßler
Coadvisor: Prof. Dr. Dr. h.c. Sahin Albayrak

Berlin
December 2015

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und eigenhändig sowie ohne unerlaubte fremde Hilfe und ausschließlich unter Verwendung der aufgeführten Quellen und Hilfsmittel angefertigt habe.

Berlin, den

....
Unterschrift

ABSTRACT

Este documento é um exemplo de como formatar documentos para o Instituto de Informática da UFRGS usando as classes LaTeX disponibilizadas pelo UTUG. Ao mesmo tempo, pode servir de consulta para comandos mais genéricos. *O texto do resumo não deve conter mais do que 500 palavras.*

Keywords: Formatação eletrônica de documentos. LaTeX. ABNT. UFRGS.

Lösung des Dial-a-Ride-Problems mit der Firefly Metaheuristik

ZUSAMMENFASSUNG

This document is an example on how to prepare documents at II/UFRGS using the LaTeX classes provided by the UTUG. At the same time, it may serve as a guide for general-purpose commands. *The text in the abstract should not contain more than 500 words.*

Schlagwörter: Electronic document preparation. LaTeX. ABNT. UFRGS.

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

FA	Firefly Algorithm
DARP	Dial-a-Ride Problem

CONTENTS

1 INTRODUCTION

In current urban areas, mainly in very populated cities, there is a huge number of mobility problems. These problems have been seen with much interest by the scientific community, which has been strongly contributing to the improvement of transportation and logistic networks, thus promoting advances towards a better mobility.

The here addressed problem is known as the dial-a-ride problem (DARP). Shortly, it consists of a system with a set of requests of pick up and delivery entered by customers and a fleet of vehicles. The goal is planning the route of the vehicles and the assignment of requests to them in a feasible way, since there are constraints from both the requests and the vehicles to which a solution is subject. These can be several conditions, like location and time limit for picking up and delivering or how much time each vehicle can operate. Besides, it is not only searched a feasible solution but an optimal one that minimizes the costs of the operation defined by a function.

Many difficulties are found when trying to solve the described problem, the combinatorial nature of its solution space make it hard to treat for large inputs because of the high complexity, it leads then to a special difficulty in building a scalable application.

1.1 Definition

1.2 Justification

It is expected that the results of this work may bring relevant contributions to the handling of the presented problem, and even of other ones. By having two distinct approaches it is possible to compare results regarding important features of the problem, such as scalability, feasibility and deviation to an optimal solution. Furthermore, the application of the relatively new firefly algorithm to the problem can show how it performs in a such a solution space.

In addition to the contributions to the understanding of the behavior of swarming metaheuristics applied to optimization problems of transportation, the new procedure of solving the problem serves as a prototype and brings a new perspective to commercial applications which seek constantly to treat the problem in a more efficient and scalable way.

With regard to the current cities' mobility, there is a growing demand for an efficient alternative to the classical means of transportation. The implementation of such a system improves the possibility of movement of the population and makes it more efficient, since it allows the decrease of the number of cars that drive through the urban network everyday causing traffic jams in big cities. Moreover, it helps to solve an demanding problem in today's society where there is an increasing number of elderly or handicapped people, who have the right to mobility and need assistance to travel in the town.

Also in an economical view this research contributes to the win of new markets by companies who aim to enter the branch of public transportation since its main goal is minimizing the operation costs. With such a model a company can take great advantages against competitors in order to capture marketplace.

At last, the concerns of the proposed model shows also an ecological relevance by enabling the decrease of the emission of greenhouse gases in the urban area, directly, considering that in this case costs are direct proportional to the consume of fuel, and consequently to the emission of gases, such as CO_2, and indirectly by reducing the circulation of other automobiles.

1.3 Approach

2 LITERATURE REVIEW

The Dial-a-ride Problem (DARP) is very similar to other problems studied in the scientific society, namely, it can be cited the Pickup and Delivery Vehicle Routing Problem (PDVRP) and the Vehicle Routing Problem with Time Windows (VRPTW), problems with application in logistics. According to Cordeau (2007, (**??**)), what basically differs the DARP from these other ones is the human perspective, by the fact that people are transported. It often appears presenting two goals, minimizing operation costs subject to the constraints and maximizing the availability and quality of the service. The quality criteria include frequently aspects like route duration, customer waiting and ride time, maximum vehicle ride time, among others and are usually treated in the constraints of the optimization.

Cordeau (2007, (**??**)) realized a survey on the subject, he presents three mathematical models that occur in the literature, two as formulation of a mathematical optimization and one as a scheduling problem. Therefore there are several other approaches made by other authors.

Although there are works handling the problem in an exact way, that tries to find an optimal solution, for instance, with the branch-and-cut algorithm, most of them focus on applying a determined heuristic in order to find a near-optimal solution in the search space, for example, tabu search, genetic algorithms or simulated annealing.

What has not been tried is using swarm-based metaheuristics to solve the problem, such as particle swarming optimization, ant colony or the firefly algorithm. Swarm intelligence is a technique applied in the computer science, more precisely in artificial intelligence and operations research, that is based on nature patterns or behaviors. In this point of view, these metaheuristics resemble the genetic algorithms, since these are also nature-based, but they differ by the fact that, genetic algorithms have mutation and crossover operators and are based on the theory of evolution of the species, whereas swarm intelligence techniques are based on the observation of the behavior of swarms.

In this work we will study and apply the firefly algorithm (FA), that has been showing good results in the solution of nonlinear global optimization problems. Yang (2012, (**??**)) presents a theoretical analysis on swarm intelligence having as study cases the firefly algorithm and particle swarm optimization (PSO). Yang (2013, (**??**)) introduce the FA approaching parameter settings, complexity and applications in a didactic way with examples, at the end he draws a conclusion showing a growing application of the method in scientific articles and foreseeing an expansion in the subject and the improvement of such metaheuristic. Additionally, Yang (2009, (**??**)) compares the FA against the PSO running simulations in a variety of objective function and concludes affirming the superiority of the FA over the PSO and that it is potentially more powerful in solving NP-Hard problems.

3 THEORETICAL BASIS

3.1 Graph Theory

- Cycle - Full tree

3.2 Computational Complexity

3.2.1 Travelling Salesman Problem

3.3 Mathematical Optimization

3.4 Metaheuristics

3.5 The Firefly Metaheuristic

4 ARGUMENTATION

4.1 Integer Linear Programming Formulation

4.2 Firefly Metaheuristic Modeling

This section aims to explain the design of the Firefly Metaheuristic applyed to the hier treated problem. This approach consists of viewing the solutions for the DARP as vectors in a multidimensional space. These solutions are represented by the fireflies, which can movement themselves in the search space by changing the components of the vector. There is a function with takes a vector as input and gives a real number that indicate how bright the firefly is, that is to say, how good the solution is.

The challenge is to fit the problem in this model, so that it can be computed in a effective and efficient way. This includes defining the vector of a solution, the brightness function, distance and movements in the space, the parameters, the randomness and the evolution of the optimization process.

4.2.1 Vector Representation of the Solution

In this model any solution can be represented through a natural number vector. This vector is an element of the search space. In order to understand its construction it is possible to split it into two parts. The first has always one component which describe the assignment of requests to buses. The second part has as many components as there are buses and each one represents a cycle in the requests graph, it means, the route that the bus executes in the solution. So a solution $S \in \mathbb{N} \times \underbrace{\mathbb{N}}_{k-times}$ for k buses.

4.2.1.1 Modeling the Assignment of Requests to Buses

The delegation of n requests to k buses can be represented by an n-tuple, in which each element varies from 0 to $k - 1$. Therefore, the number of possible combinations is k^n. The arrangement of these tuples can be structured in a full tree with a height of n, where every non-leaf node has k children. So the leafs can be enumerated from 0 to k^n, which is the total number of combinations. Thus every walk on the tree yields a possible assignment request-bus and every possibility can be yielded by a unique walk. Hence this enumeration is used in the representation of the solution.

Therefore, the process of transforming the component of the solution vector into a tuple describing the delegation of the requests to buses can be described by a bijective function described by a simple algorithm:

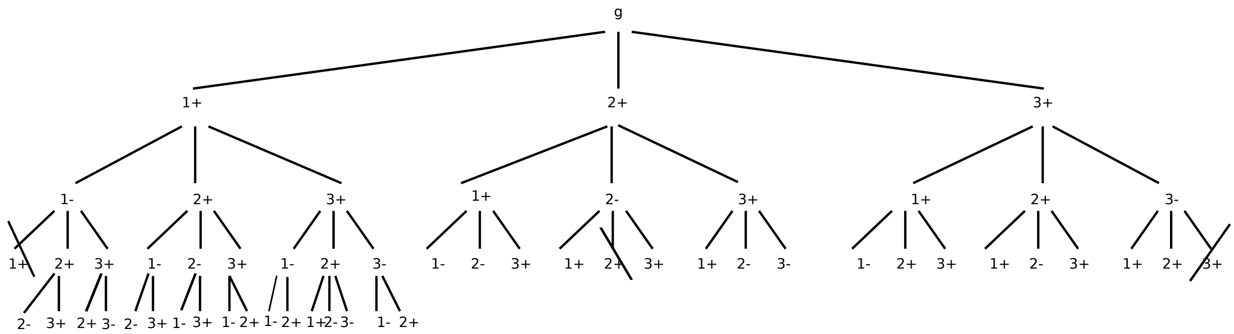
function TRANSFORM COMPONENT INTO ASSIGNMENT REQUEST-BUS(x)
  $a \leftarrow x$
  for $i = 1 : n$ **do**
    $T_i \leftarrow \lfloor \frac{a}{k^{n-i}} \rfloor$
    $a \leftarrow a \bmod k^{n-i}$
  end for
  return T
end function

4.2.1.2 Modeling the Routes of the Buses

A similar analysis, as in the modeling of the previous section, can be done to the definition of the routes of the buses with the difference that in this case there is a permutation of the boarding and alighting nodes. Moreover, more constraints are applied to the numerical representation of a route, namely, that an alighting node of a given request cannot occur before the boarding of the same.

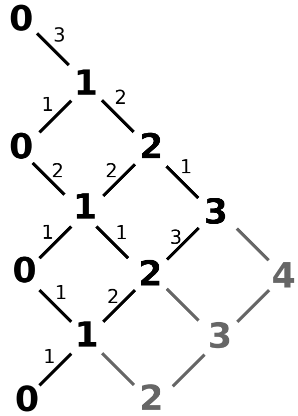
The following figure illustrates the route tree of a bus which was assigned three requests. **EXPLAIN!!**

Figure 4.1: Route possibilities represented through a tree



Determining how the tree structure at each level is and how many leafs altogether there are depends on what the load of the bus in a given vertex is, it means, it depends on how many requests are being carried on and how many there are to pick up. The figure 4.2 helps to explain the structure of the tree. In the illustration each knot represents the load of the bus in a depth of the tree, how further down it is greater is its respective depth. The edges under a knot tells, how many children with the following load are generated by each vertex with a given load.

Figure 4.2: Graph describing the structure of the routes tree



Nevertheless, in order to build a function that allows us to have a mapping from each walk to a natural number, thus enumerating all the possible routes of a bus, is necessary to know, given a depth in the tree how many leafs are under each vertex of this depth. Once this information is available, the transformation function can be computed in a very efficient way.

From the graph illustrated in the figure 4.2 it is possible to extract two matrices to represent the structure by denoting the numbers on the edges that leave a knot on the right and on the left in the matrix Q_{in} and Q_{out} respectively. Each column of the matrix Q_{in} has the quantity of children with a larger load to be created by each vertex with a given load, the matrix Q_{out} is analogously constructed, with the difference that it carries the quantity of children with a lower load in the next stage. In addition, each row corresponds then to a load of the bus, beginning from 0 in the first row to n in the last row, be n the number of requests. Note that, for the generalization it is assumed that a bus can at a moment be carrying all the requests to him assigned. Further constraints should be checked in a future procedure, in order to decide whether a route is feasible according to the input. Knowing that there are n requests implies that there are $2n$ columns, since every request must be picked up and delivered. The mentioned matrices for the figure 4.2 are as follows represented.

$$Q_{in} = \begin{bmatrix} 3 & 0 & 2 & 0 & 1 & 0 \\ 0 & 2 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, Q_{out} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 2 & 0 & 2 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 \end{bmatrix}$$

An iterative algorithm with these two matrices as input yields a new matrix, which shows, for each depth of the above shown routes tree, the number of vertices for each bus load.

function GENERATE MATRIX OF KNOTS PER LOAD AND DEPTH(Q^{in}, Q^{out})

Let n be the the number of requests and $\odot$ the element-wise multiplication

$$Q_{2n+1} \leftarrow \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

for $i = 2n : 1$ **do**

$$Q_i \leftarrow Q_i^{in} \odot \begin{bmatrix} Q_{i+1,1..n-1} \\ 0 \end{bmatrix} + Q_i^{out} \odot \begin{bmatrix} 0 \\ Q_{i+1,2..n} \end{bmatrix}$$

end for

return Q

end function

The result of applying the function for the shown matrices Q_{in} and Q_{out} is shown below. This provide additionally a new information about the size of the domain which the component of the vector find feasible values, namely in $Q_{1,1}$, since it tells the total number of leafs in the tree. It is useful for constraints check and for a fast creation of the initial generation of fireflies.

$$Q = \begin{bmatrix} 90 & 0 & 6 & 0 & 1 & 0 & 1 \\ 0 & 30 & 0 & 3 & 0 & 1 & 0 \\ 0 & 0 & 12 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 6 & 0 & 0 & 0 \end{bmatrix}$$

With help of these three matrices and given a correspondent vector component value, the path of a bus can be computed by a transformation function. Below is shown a function that takes these arguments and delivers a path performed by a bus, where its elements do not represent the requests themselves, but the knots of the walk in the tree, which is relative to each bus. A mapping to the absolute requests can be trivially done, once the requests assigned to each bus are known.

function TRANSFORM COMPONENT TO A CYCLE IN THE GRAPH(Q^{in}, Q^{out}, Q, x)

Let n be the the number of requests of the bus

$Path \leftarrow []$

$row \leftarrow 0$

$pointer \leftarrow 0$

for $col = 1 : 2n$ **do**

$$ChildrenSizes = \begin{bmatrix} \underbrace{Q_{row-1,col+1}}_{Q^{out}_{row,col} times} \\ \underbrace{Q_{row+1,col+1}}_{Q^{in}_{row,col} times} \end{bmatrix}$$

for $child = 1 : Q^{in}_{row,col} + Q^{out}_{row,col}$ **do**

if $x - pointer < ChildrenSizes_{child}$ **then**

if $child < Q^{out}_{row,col}$ **then**

$row \leftarrow row - 1$

else

$row \leftarrow row + 1$

end if

else

$pointer \leftarrow pointer + ChildrenSizes_{child}$

end if

end for

$Path_{col} \leftarrow child$

end for

return $Path$

end function

4.2.2 Distance

For reasons of efficiency of the implementation and numerical error the Manhattan distance can be used to approximate the distance between two vectors in the search space. The Manhattan distance can be described as follows:

$$d(\mathbf{p}, \mathbf{q}) = \| \mathbf{p} - \mathbf{q} \| = \sum_{i=1}^{n} | \mathbf{p}_i - \mathbf{q}_i |$$

4.2.3 Attractiveness

As proposed by (**??**) the attractiveness is calculated with the following quotient, which varies with the squared distance between two vectors.

$$\beta(r) = \frac{\beta_0}{1 + \gamma r^2}$$

However, without loss of quality in the method, the attractiveness can be set to vary directly with the distance as the search space is too large and concerns with numerical errors play a important role. So the function is rewritten as follows.

$$\beta(r) = \frac{\beta_0}{1 + \gamma r}$$

4.2.4 Randomization Term

The random term of the metaheuristic movement equation is by default $\alpha \cdot \epsilon$, where $\epsilon \sim N(0, 1)$. Though, as stated by (**??**), *"it is a good idea to replace α by αS_k where the scaling parameters $S_k(k = 1, ..., d)$ in the d dimensions should be determined by the actual scales of the problem of interest"*. As the size of each dimension are easily obtainable, they are used in this model. Moreover, (**??**) presents a iterative change of the alpha parameter, by turning it variable to the optimization evolution t. Thus, he introduces a new variable delta and that parameter is updated by the equation

$$\alpha_t = \alpha_{t-1} \cdot \delta, (0 < \delta < 1),$$

where α_0 is the initial scaling factor. The author gives also an advice about setting delta: *"δ is essentially a cooling factor. For most applications, we can use $\delta = 0.95$ to 0.97"*.

At last, it is wanted a integer number for the stochastic term, as the search space is discrete. In order to achieve that, rounding is performed, proper concerns about numerical errors should be taken in the implementation, so that they are reduced at most.

4.2.5 Movement in the Discrete Space

The movement of a firefly i towards j is determined by

$$\mathbf{x_i^{t+1}} = \mathbf{x_i^t} + \beta(d(\mathbf{x_i^t}, \mathbf{x_j^t})) \cdot (\mathbf{x_j^t} - \mathbf{x_i^t}) + RandomTerm(t)$$

As the fireflies move in a discrete vector space, the terms of the sum should be also integer number. Since β is less or equal than one and its image is the set of the real numbers, the equation can be modified, by turning the multiplication into a division by the inverse function and by rounding the result of the function β, or rather, implementing it, so that its image is the set of the natural numbers.

$$\mathbf{x_i^{t+1}} = \mathbf{x_i^t} + \lfloor \frac{(\mathbf{x_j^t} - \mathbf{x_i^t})}{\beta^{-1}(d(\mathbf{x_i^t}, \mathbf{x_j^t}))} \rfloor + RandomTerm(t)$$

AND WHEN IT IS OUT OF DOMAIN? WHAT TO MAKE?

4.2.6 Intensity Function

The intensity function models the brightness of a firefly and should be directly proportional to the utility function of the problem to be maximized. However, the problem's goal is to minimize the operation costs, and in the Firefly Algorithm it is not enough utilizing the negative costs function instead, since the intensity function is by definition non-negative.

Nevertheless, the costs function has an lowest possible value, that can be estimated. Here it is proposed to sum the v most distance vertices in the requests graph, so that v is twice the number of requests, since every request has two correspondent vertices, one for getting in and one for getting out. Once this value is calculated, the costs of a given solution can be subtracted from it, in order to obtain an utility function. The process is equivalent to translating the function.

Let m be the mentioned superior limit for the costs, f a function that tells in a binary matrix which edges are visited by a solution vector and C the adjacency matrix of the requests graph with the costs between any one of them, then the intensity can be defined like in the following formula.

$$I(\mathbf{x}) = \begin{cases} 0 & \text{if } \mathbf{x} \text{ does not meet the constraints} \\ m - \displaystyle\sum_{i=1}^{2n+1} \sum_{j=1}^{2n+1} f(\mathbf{x})_{i,j} \cdot C_{i,j} & \text{else} \end{cases}$$

4.2.7 Initial Solution

A initial set of feasible solutions is calculated by simply generating random natural numbers in an interval to each component of the vector. Firstly, the first component is randomized, its range is $[0, k^n)$, where there are n requests and k buses. Secondly, the other components are randomized, their domain intervals are determined based on the distribution of requests to buses resulted from the first component randomization. So the first generation of fireflies can be efficiently created. [ARE THEY FEASIBLE WHEN THE BUSES HAVE LIMIT OF PASSENGERS? OR TIME CONSTRAINTS? NO!]

4.2.8 Parameters

The choice of the parameters is basically based on the work of (**??**). For that the scale L of the problem is taken into consideration, it represents the size of the search space and is calculated by multiplying the sizes of the intervals, in which the intensity function is defined, of each dimension. The parameter gamma is then set $\gamma = 1/\sqrt{L}$. In order to have an broad exploration of the space it is set $\alpha_0 = 0.1$, the parameter is reduced along the optimization process. Lastly, it is set $\beta_0 = 1$, $\delta = 0.95$ and the number of fireflies to 40.

4.2.9 Two-Phase Optimization

4.2.10 Implementation

One of the main difficulties in implementing the algorithm is with the large that the numbers may have. To workaround the problem the programming language Python[1] was

[1]<https://www.python.org/>

chosen, since it has a native implementation of unlimited integers. Besides, the proposed model has a considerable quantity of matrix operations, therefore, the libraries NumPy[2] and Scipy[3] were used for that purpose. For drawing the graphs the was adopted library Matplotlib[4], which has a easy-to-use interface for the programmer.

Care by the numerical operations and data type had to be specially taken to ensure no capacity overflows and a controlled numerical error at a lowest level.

To obtain a better performance and the most recent implemented features it is required that the implementation program runs on the newest possible tool versions. Namely, it was developed on the versions 3.3 of Python, 1.10 of the NumPy, 0.16 of the SciPy and 1.3 of the Matplotlib.

- Input (Manual,help)

[2]<http://www.numpy.org/>
[3]<http://www.scipy.org/>
[4]<http://matplotlib.org/>

5 EVALUATION

- Method for evaluation - Experiments Setup - Dataset - Results - Comparison

6 CONCLUSION

- Further Research