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
Reviewer: Prof. Dr. Dr. h.c. Sahin Albayrak
Reviewer: Prof. Dr. habil. Odej Kao

Berlin
February 2016

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: Dial-a-ride problem. firefly algorithm. metaheuristic. integer programming.

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

AMPL	A Mathematical Programming Language
DARP	Dial-a-Ride Problem
FA	Firefly Algorithm
GA	Genetic Algorithms
GLPK	GNU Linear Programming Kit
MD-H-DARP	Multi-Depot Heterogeneous Dial-A-Ride Problem
PDVRP	Pickup and Delivery Vehicle Routing Problem
PSO	Particle Swarm Optimization
VRPTW	Vehicle Routing Problem with Time Windows

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 urban 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 pickup 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 operation costs, here seen as the total time of 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

Cordeau and Laporte (2007, p. 29) states the problem very briefly:

> "The Dial-a-Ride Problem (DARP) consists of designing vehicle routes and schedules for n users who specify pickup and delivery requests between origins and destinations. The aim is to plan a set of m minimum cost vehicle routes capable of accommodating as many users as possible, under a set of constraints."

The author summarizes very well, however, a closer look is to be taken. The instances of the problem specify the following properties, that are taken as input by the solver. Firstly, there is an **amount of homogeneous vehicles**, that is to say, no distinction should be made concerning types of buses. Secondly, it is assumed a **single depot**, which the vehicles start from and which they arrive to. Next, the cars have the attributes of **capacity**, in other words, how many passengers it can carry at a time, and its **maximal duration time** that the car cannot exceed. Then, the passengers have a **maximum allowed travel time** to guarantee their comfort. Finally, there are n **requests**, which have these features:

- Pickup location;
- Destination location;
- Time window for picking up (a time interval in which the vehicle is expected to be at the site);
- Time window for delivering;
- Time needed for boarding or alighting;
- Quantity of passengers;

Additionally, it is possible for a car to move from every location to any other location, the time needed to travel between any pair is considered to be the linear distance between these two points.

To sum up, these are the parameters that form the constraints of the problem and must be considered. In regard to the goal, there are two concepts that can be elucidated, namely feasible and optimal solution. The former is defined thusly:

Definition 1 (Feasible solution)**.** *Determination of the routes of the vehicles, such that, every request is picked up and then delivered by one and only one vehicle, which fulfill every condition of the requests and of itself.*

In order to make it clearer, let a route be defined as:

Definition 2 (Route)**.** *The order of the locations through which a vehicle passes. It starts and ends at the depot and it is possible to be executed without breaking the conditions of time windows of the respective requests considering the boarding, alighting and travel times.*

Every route has then attached a duration time, which can be calculated based on its requests. Summing up the duration of every route in a solution, we get its total duration time. Therewith is an optimal solution defined:

Definition 3 (Optimal solution)**.** *A feasible solution whose duration time is less than or equal to any other feasible solution's duration time.*

In conclusion, given an instance of the Dial-a-ride problem, it is sought an optimal solution.

1.2 Approach

In this work two approaches shall be taken, in order to solve the optimization problem. The exact approach consists of mathematically modeling the problem as an **integer linear program** and then transcribing it into a **mathematical programming language** aiming executing it with a generic linear program solver. The near-optimal approach shall then analyze and develop a program that implements the **firefly metaheuristic** to efficiently seek near-optimal solutions.

Finally, the goal of the research is to compare the performance of the two different models, both through the execution time and the solution quality as well as through the possibility of dealing with large inputs. Besides the evaluation through the the mentioned comparison, a second evaluation is conducted by assessing the firefly metaheuristic implementation when solving the instances used in other works of the literature.

1.3 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 swarm 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 a 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 a business model based on the reduction of transaction costs, 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.

2 LITERATURE REVIEW

According to Cordeau and Laporte (2007, p. 30), the Dial-a-ride Problem (DARP) is very similar to other problems studied in the scientific society, namely the *Pickup and Delivery Vehicle Routing Problem* (PDVRP) and the *Vehicle Routing Problem with Time Windows* (VRPTW), problems that have application in logistics. The authors point out that, 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 as the constraints of the optimization problem.

Cordeau and Laporte (2007) realized a survey on the subject, they show different variations in the formulation and in the approach. Firstly, the DARP occurs in a **static** version in which the requests are known beforehand, this one we would like to treat here, alternatively, there is a **dynamic** version, dealt by Berbeglia, Cordeau, and Laporte (2010), in which requests can be entered during the operation of the transportation service.

Another distinction to be made is between the **homogeneous** and the **heterogeneous** Dial-a-ride Problem. The former proceeds on the assumption that every vehicle is equal. The latter on the other hand distinguish the vehicles either in capacity or in new features, such as space for wheelchair, etc. (Parragh et al. 2010, p. 593). In the same way, the objectives can differ by aiming **cost minimization** or **satisfied demand maximization** (Cordeau and Laporte 2007, p. 30). Urra, Cubillos, and Cabrera-Paniagua (2015) have succeeded on handling the second one. A last differentiation of the problem is done regarding the depots, which can be a **single** one or **multiple** ones, in this case referred to as *Multi-Depot Heterogeneous Dial-A-Ride Problem* (MD-H-DARP) (Braekers, Caris, and Janssens 2014, p. 166).

In their survey, Cordeau and Laporte (2007) identify in the literature three formulations. Two of them are described as a mathematical linear program and the other one as a scheduling problem. The **three-index mathematical formulation**, proposed by Cordeau (2006), uses binary three-index variables $x_{i,j}^k$ to assign routes to each bus and then to minimize costs. In addition, Ropke, Cordeau, and Laporte (2007) come up with a **two-index formulation** which ignores the maximum ride constraints in order to simplify the model.

Exact algorithms have been proposed by several researchers, to highlight are the **branch-and-cut** by Cordeau (2006) and by Ropke, Cordeau, and Laporte (2007). However, to speed up running times many other methods based on **metaheuristics** have been suggested in the literature, among them are the **genetic algorithms** by Jorgensen, Larsen, and Bergvinsdottir (2007), the **simulated annealing** by Zidi et al. 2012, the **granular tabu search** by Kirchler and Wolfler Calvo (2013) and the approach with **large neighborhood search** performed by Parragh and Schmid (2013). Lately, Urra, Cubillos, and Cabrera-Paniagua 2015 published a new method with the so-called **hyperheuristic**.

All in all, none of the studied methods takes a swarm-based metaheuristic approach to solve the problem. Swarm intelligence is a technique applied in the computer science, more precisely in artificial intelligence and operations research, that is based on observations of nature patterns and behaviors, *particle swarm optimization* (PSO), `ant colony` and the *firefly algorithm* (FA) are example of techniques that apply these ideas. In this point of view, these metaheuristics resemble the *genetic algorithms* (GA), since GAs are also nature-based, but they differ in 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 purely on the behavior of swarms (Yang 2012, p. 189-190).

In this work we apply the **firefly metaheuristic**, that has been showing good results in the solution of nonlinear global optimization problems. It was introduced by Yang (2009), who compares it against the PSO by running simulations in a variety of objective functions and concludes affirming that the FA can outperform the PSO and that it is potentially more powerful in solving $\mathcal{NP}$-hard problems. Additionally, Yang (2012) presents a theoretical analysis on swarm intelligence having as study cases the firefly algorithm and particle swarm optimization. Yang and He (2013) introduce the FA by approaching parameter settings, complexity and applications with examples, at the end he draws a conclusion showing a growing application of the method in the scientific community and foreseeing an expansion of the subject and the improvement of the metaheuristic.

The firefly algorithm has also been applied to discrete optimization problems, also known as **integer programming**. Jati and Suyanto (2011) address the classical **Travelling Salesman Problem** with the help of the FA. In the same way, Apostolopoulos and Vlachos (2010) deal with another combinatorial optimization problem, the *Economic Emissions Load Dispatch Problem*. At last, Sayadi, Ramezanian, and Ghaffari-Nasab (2010) and Sayadi, Hafezalkotob, and Naini (2013) also utilize this technique, in order to solve other $\mathcal{NP}$-hard problems.

3 THEORETICAL BASIS

This chapter presents a theoretical framework that configure a background necessary to understand, analyze, design and implement the solution for the addressed problem.

3.1 Linear and Integer Programming

Linear Programming is a technique in mathematics used for **optimization of linear functions**. With it, it is possible to model optimization problems in which a cost (or utility) linear function is to be minimized (or maximized) given a set of constraints which the function's independent variables are subject to. In short, it enables the formulation of optimization problems in a very simple mathematical language.

To formulate a linear program three specifications are required. At first, the so-called decision variables x_i, that are the ones which values are assigned to. Secondly, the objective function, that is a linear function $f(x)$ to be optimized. Lastly, the constraints, that are linear inequalities describing relations between the decision variables and parameters of the problem instance. Therefore, it is expected from a solver, given a linear program and data parameters, to deliver the set of assignments of values to the decision variables that minimize or maximize the objective function respecting the constraints. The most prominent method for solving linear programming is the **simplex method** (Shenoy 2007, p. 5-6).

As a result, a linear program with n variables and m constraints for the minimization of a function can be written as follows.

$$
\begin{aligned}
\textbf{minimize} \quad & \sum_{i=1}^{n} c_i x_i & \\
\textbf{subject to} \quad & \sum_{i=1}^{n} a_{ki} x_i \leq b_m & \forall k = 1, 2, ..., m \\
& x_i \geq 0 & \forall i = 1, 2, ..., n
\end{aligned}
$$

However, there are problems where real values are not accepted as assignment to variables, because they require integer values. These problem are to be treated differently by adding an additional integer-value constraint to the domain of the variables ($x_i \in \mathbb{Z}$). A technique derived from linear programming addresses the issue, namely **integer pro-**

gramming. Although there are other more specific concepts for problems that are similar to integer programming, such as mixed integer programming and zero-one programming, they all can be handled by quite the same principles (Shenoy 2007, p. 175). As an implication, the simplex method is not able to solve this type of optimization problem neither is, generally, numerical approximation a good solution. Thus, there are several algorithm in the literature, for instance the **branch-and-bound** and the **cutting planes** are widespread and one of the most common.

The integer-value constraint implies a vast difference regarding the complexity of the integer problem when compared to the linear one. It is then discussed, whether these problems can be solved in polynomial time, referred to as *certificate of optimality*. Wolsey and Nemhauser (2014) point out that there are algorithms capable of solving linear programming in polynomial time, whereas no such algorithm is known to integer programming. Moreover, many known $\mathcal{NP}$-complete problems, such as the TSP and the knapsack problem, can be reduced to an instance of integer programming. It turns out that integer programming is not only $\mathcal{NP}$-hard but $\mathcal{NP}$-complete as well (Schrijver 1998, p. 21).

3.2 Metaheuristics

Osman and Kelly (2012, p. 1) state in their work: "Meta-heuristics are the most recent development in approximate search methods for solving complex optimization problems.". As he follows, a metaheuristic guides the development of algorithms using artificial intelligence, biological, physical, mathematical and natural phenomena as source of inspiration. Firstly should be understood what heuristics are. According to Carvalho, Savransky, and Wei (2004, p. 13), there is no common definition for heuristics. They suggest that: "Heuristics can be rules, strategies, principles or methods for increasing the effectiveness of a problem resolution [...]. [They] neither provide direct and definite answers, nor guaranty a solution for a problem.". On the whole, heuristics are a powerful weapon to attack difficult combinatorial optimization problems.

On metaheuristics the concept is concise, Osman and Kelly (2012, p. 3) explain in the following way.

> "A Meta-heuristic is an interactive generation process which guides a subordinate heuristic by combining intelligently different concepts for exploring and exploiting the search space using learning strategies to structure information in order to find efficiently near-optimal solutions."

Therefore, metaheuristics do not necessarily deliver an optimal solution, but the benefit in terms of performance overcome the possibility of not obtaining a proved optimal solution. Even though, the late advances in the field have contributed strongly in order to get very good in quality results (ibid.).

As a metaheuristc gives quasi instructions to write an algorithm, it is common to almost every technique four basic elements which should be modeled. First, a **solution representation** is to be determined, this should be able to describe any possible solution and to be evaluated by a function which determines its cost or benefit. Secondly, the algorithm should **initialize solutions**, it means, there is a procedure to create the so-called initial solutions. Thirdly, **new solutions** should be generated from existing ones, in some metaheuristics this step is called *movement*, *genetic operator*, among other terms. Finally, a **stop criterion** is established in order to determine the termination of the method.

The metaheuristics can be classified in two types, the ones based on **local search** (or neighborhood search) and the others based on **nature observations**. The former explores the search space in search of an optimal solution by generating new solutions that lie near the current analyzed one, it then requires a move mechanism besides a *selection* and a *acceptance* criteria, which decide whether a new generated solution is approved in order to progress with the iterations. Whereas simpler methods, such as *variable neighbohood search*, have the drawback of tending to a local optimum, others, like *simulated annealing*, *tabu search* and *GRASP*, aim to workaround the issue. In contrast, the nature-inspired metaheuristcs are generally based on the behavior of populations of living beings and how the its members interact. They are characterized by having a set of solutions, instead of only one, with which local searches are performed in different areas of the search space (Osman and Kelly 2012, p. 7). Examples of this kind of method are the *genetic algorithms*, *particle swarm optimization*, *ant colony* and the *firefly algorithm*.

3.3 The Firefly Algorithm

The **firefly algorithm** (FA), or firefly metaheuristic, was brought forth by Yang (2009). It is a **nature-inspired metaheuristic** based on observations of fireflies populations that resembles other techniques, such as the *particle swarm optimization* and the *bacterial foraging algorithm*. The following explanations relies on the work of (ibid.).

A firefly is a flying insect that emits flashes of light from its tail. This luminous signal serves then as a mean of communication between these animals, that is used for sexual attraction and selection. So, the main conception in the FA is related to this light emission. Let the **position of a firefly** in the space be a unique representation of a solution to a optimization problem, thus this position can be represented through a mathematical vector. Now, for reason of abstraction, let the **intensity of the light** emitted by the firefly be proportional to the function to be optimized and assume that the **attractiveness between the fireflies** is stronger when the light intensity is greater. Consequently, one can construct a model that takes these factor into account in order to **simulate the movements** of the several fireflies. As a result, due to the tendency to a convergence of them to a set of optimal positions (solutions), it is possible to use the mechanism for optimization.

For the abstraction of the natural system three principles are to be considered:

1. The fireflies are attracted to each other regardless of their sex;
2. The attractiveness is proportional to the brightness so that a less brighter firefly moves in the direction of the brighter one. In the same way, the brightness is relative to the observer, thus, the farther away two fireflies are from each other, the weaker the attractiveness between them is. If there is no firefly brighter than a particular one, it moves randomly;
3. The brightness is determined by the landscape of the objective function.

Hence, the firefly algorithm can be described as follows.

Algorithm 1 Firefly Algorithm

```
function FIREFLY ALGORITHM
    Define a objective function f(x), x = (x_1, ..., x_d)^T
    Generate initial population of fireflies x_i(i = 1, 2, ..., n)
    Let the intensity I_i be proportional to f(x_i)
    while t < MaxGeneration do
        for i = 1 : n do
            for j = 1 : i do
                if I_j · Attractiveness > I_i then
                    Move firefly i towards j
                end if
            end for
        end for
        if Firefly i did not move then
            Move i randomly
        end if
        Rank the fireflies to determine the current best
    end while
    return Best solution
end function
```

In the algorithm it is assumed that the brightness of a firefly in a position $\mathbf{x}$ is determined by the objective function applied to the vector $\mathbf{x}$ so that $I(\mathbf{x}) \propto f(\mathbf{x})$. However, the brightness perception depends on the distance r between the vectors. So, an attractiveness function $\beta(r)$ is to be defined. Given a medium absorption coefficient γ and the brightness β_0 in the source , this function varies according to the inverse square law and can be written as

$$\beta(r) = \beta_0 e^{-\gamma r^2},$$

which is, after a series expansion analysis, equivalent to

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

Concerning the distance, it can be, as commonly, defined as the euclidean distance $r_{ij} = \sqrt{(x_{i1} - x_{j1})^2 + (x_{i2} - x_{j2})^2 + \cdots + (x_{id} - x_{jd})^2}$ although another distance definition may be used depending on the type of the problem (Yang 2012, p. 193). At last, the movement is calculated by a sum of vectors which characterize that a firefly will move a larger distance towards the other one, as the brightness is stronger and the current distance is shorter. Hence, one can describe it as

$$\mathbf{x}^{t+1} = \mathbf{x}^t + \beta_0 e^{-\gamma r_{ij}^2} + \alpha_t \epsilon^{\mathbf{t}},$$

where the third term models a stochastic variation and t is the iteration number. In the formula α^t represents the radius in which the random variation is allowed and $\epsilon^{\mathbf{t}}$ is a vector of numbers sampled from a random distribution. This distribution can be uniform, Gaussian or also another one depending on the desired effect.

Yang (2009, p. 177) carries out a statistical comparison between the FA, PSO and GA with a set of functions. The firefly algorithm outperformed both in every presented case. The author states also that “FA is potentially more powerful in solving $\mathcal{NP}$-hard problems”, which is the case of the DARP. Insights about parameter settings, complexity, recent applications and efficiency analysis can be found in Yang and He (2013).

4 ARGUMENTATION

The objective of this chapter is to present in details the design and implementation of the two approaches taken in order to solve the DARP. Firstly, the integer programming formulation is explained and discussed. Then, the model for the firefly metaheuristic is introduced showing the main procedures and decisions.

4.1 Integer Linear Programming Model

As the DARP is an optimization problem, one can define it as an mathematical integer program. On the one hand, This is a powerful way to solve this kind of problem, since it is relatively easy to formulate and quite flexible in adding new rules and conditions. On the other hand, the solving by a generic solver or by an exact algorithm is usually cost intensive and limited by the strictness of the linear program formulation.

4.1.1 Mathematical Formulation

In this work it is considered the **three-index formulation** proposed by Cordeau (2006, p. 574-575). For this purpose, consider n the number of requests of a problem instance. The DARP can then be defined through a directed graph $G(V, A)$, where V is the set of vertices representing stop locations. This set can be partitioned in three subsets, (1) the initial and final depot, that contains the vertices 0 and $2n + 1$; (2) the pickup points comprehending the vertices $P = \{1, \ldots, n\}$; (3) the deliver points including the vertices $D = \{n + 1, \ldots, 2n\}$. This way, one define a request as a ordered pair $(i, n + i)$. Additionally, A is the set of edges that interconnect the locations to represent the travel time y_{ij} and the travel costs c^k_{ij} for a vehicle k. With that, G becomes a complete digraph, that is to say, there is exact two edges connecting every pair of distinct vertices, one edge in each direction. For reason of simplification, one can assume an undirected graph, where the costs of driving from a stop i to another j are the same as from j to i, for the same reason, one can set the costs $c^k_{ij} = t_{ij}$.

Moreover, every vertex has associated a load q_i that tells the quantity of passenger boarding or alighting in the point, these values are then positive or negative, respectively, thus, $q_i \geq 1, (\forall i \in P)$ and $q_i = -q_{i-n}, (\forall i \in D)$ and $q_i = 0, (\forall i \in \{0, 2n + 1\})$.

Likewise, the vertices indicate also a duration di which is a positive number that informs the time necessary to realize the service at that stop. Furthermore, a time window $[e_i, l_i]$ is associated to every vertex and represents, respectively, the earliest and the latest time at which the service may start at the location.

Regarding the vehicle attributes, let K be the quantity of buses at disposal, so, it is defined a vehicle capacity Q_k and a maximal route duration T_k, but, as we are considering the homogeneous version of the DARP, the indexation by $k \in K$ does not play an important role. At the last, the maximal route duration in respect of the passenger is given by the constant L.

Having considered all these parameters, it is possible to define the set of decision variables which identify a solution to the addressed problem. So, for each edge $(i, j) \in A$ and vehicle $k \in K$ there is a variable x_{ij}^k which is equal to *one* if the vehicle passes through this edge and equal to *zero* otherwise. Further, let u_i^k be the time at which the car k starts the boarding at the location i, w_i^k the load of k as leaving that point, and r_i^k the travel time of the users of the request identified by i. The linear program is then written as follows.

$$
\begin{aligned}
\textbf{minimize} \quad & \sum_{k\in K}\sum_{i\in V}\sum_{j\in V} c_{ij}^k x_{ij}^k && && (1)\\
\textbf{subject to} \quad & \sum_{k\in K}\sum_{j\in V} x_{ij}^k = 1 && \forall i \in P && (2)\\
& \sum_{i\in V} x_{0i}^k = \sum_{i\in V} x_{i,2n+1}^k = 1 && \forall k \in K && (3)\\
& \sum_{j\in V} x_{ij}^k - \sum_{j\in V} x_{n+i,j}^k = 0 && \forall i \in P, k \in K && (4)\\
& \sum_{j\in V} x_{ji}^k - \sum_{j\in V} x_{ij}^k = 0 && \forall i \in P \cup D, k \in K && (5)\\
& u_j^k \geq (u_i^k + d_i + t_{ij}) x_{ij}^k && \forall i, j \in V, k \in K && (6)\\
& w_j^k \geq (w_i^k + q_j) x_{ij}^k && \forall i, j \in V, k \in K && (7)\\
& r_i^k \geq u_{n+i}^k - (u_i^k + d_i) && \forall i \in P, k \in K && (8)\\
& u_{2n+1}^k - u_0^k \leq T_k && \forall k \in K && (9)\\
& e_i \leq u_i^k \leq l_i && \forall i \in V, k \in K && (10)\\
& t_{i,n+i} \leq r_i^k \leq L && \forall i \in P, k \in K && (11)\\
& max\{0, q_i\} \leq w_i^k \leq min\{Q_k, Q_k + q_i\} && \forall i \in V, k \in K && (12)\\
& x_{ij}^k \in \mathbb{B} && \forall i, j \in V, k \in K && (13)
\end{aligned}
$$

In the program above, the objective function (1) minimizes the total operation costs which is calculated by the sum of the costs of every arc (i, j) where a vehicle travels through, since in this case on would have $x_{ij}^k = 1$. The equation (2) ensure that every request is taken by one and only one bus. Equation (3) model the single depot rule, which says that every vehicle starts and ends at the garage, identified by the vertices 0 and $2n + 1$. Equation (4) gives consistency to the assignments of vehicles to requests, since it constraints the solutions such that the bus k that picks up a request i is the same that delivers it to the final destination $n+i$. Analogously, (5) guarantee route consistency, that means that the bus that arrives at a determined site is the same to departure from this.

In the next constraints are defined the complementary decision variables. These are represented in the conditions (6), (7) and (8), which indirectly determine values to the time in which a vehicle k starts serving i (u_i^k), the load of k when leaving vertex i (w_i^k) and the ride time of the passengers served by i (r_i^k), respectively. Constraint (9) assures that a vehicle does not ride longer that it is allowed, the idea is that by subtracting the initial time of the route from the time when the vehicle terminates it, yields the the total ride time of a bus and then the condition can be easily expressed. Additionally, the equation (10) guarantees the restriction of time windows from the requests. In the same way, equation (11) guarantees that the ride time r_i^k of the passengers of a request i does not exceed the limit L.

Lastly, the consistency of the vehicle load is determined by the constraint (12). Note that, in the case which the vehicle picks passenger up, i.e. $q_i > 0$, this condition ensures a minimum load of q_i and a maximum of the load limit Q_k, the integrity of the rule is then fully covered by the equation (7). In the case which passengers drop off, i.e. $q_i < 0$, the constraint guarantees, besides a positive load, a superior value according to the load limit and the number of alighting passengers. Finally, the line (13) define the binary domain restriction over the variable x_{ij}^k.

4.1.2 Linearization

This proposed formulation is nevertheless not completely linear, since the the constraints (6) and (7) contain a multiplication of variables $u_i^k \cdot x_{ij}^k$ and $w_i^k \cdot x_{ij}^k$, respectively. Thus, they should pass through a linearization process which adds the constants M_{ij}^k and

W_{ij}^k. As a result, these two equations are replaced by the following ones.

$$u_j^k \geq u_i^k + d_i + t_{ij} - M_{ij}^k(1 - x_{ij}^k) \qquad \forall i, j \in V, k \in K \tag{14}$$

$$w_j^k \geq w_i^k + q_j - W_{ij}^k(1 - x_{ij}^k) \qquad \forall i, j \in V, k \in K \tag{15}$$

In order that these new constraints are valid, the constants M and W are subject to the next presented conditions.

$$M_{ij}^k \geq max\{0, l_i + d_i + t_{ij} - e_j\} \qquad \forall i, j \in V, k \in K \tag{16}$$

$$W_{ij}^k \geq min\{Q_k, Q_k + q_i\} \qquad \forall i, j \in V, k \in K \tag{17}$$

This procedure closely resembles *Miller-Tucker-Zemlin* subtour elimination constraints for the TSP (Cordeau 2006, p. 575). Yet, the current formulation is not able to run in a generic linear programming solver. Then, in order to workaround the issue, it is enough to assign M_{ij}^k and W_{ij}^k a sufficient large positive number such that the condition (16) and (17) hold always true for the problem instances (Häll et al. 2009, p. 44).

4.1.3 Implementation

As part of the approach of this research, we seek exact solutions for a set of instances available in the literature. The first step in order to achieve that is transcribing the above detailed mathematical model into a programming language. The chosen language was AMPL[1] which permits a quasi direct translation of the integer programming description to AMPL's notation (Fourer, Gay, and Kernighan 1990, p. 520). Additionally, in a lower level lies the GLPK[2] (GNU Linear Programming Kit) solver whose role is to interpret the written program, admit the input, validate the linear model, execute an optimization algorithm and deliver an optimal solution by outputting the assignment of value to decision variables and calculating the objective function. Details of the experiments are showed in the corresponding chapter.

1. See `http://ampl.com/`
2. See `https://www.gnu.org/software/glpk/`

4.2 Firefly Metaheuristic Model

This section aims to explain the design of the Firefly Metaheuristic applied to the here 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 which 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.

Algorithm 2 Transformation Vector-Solution

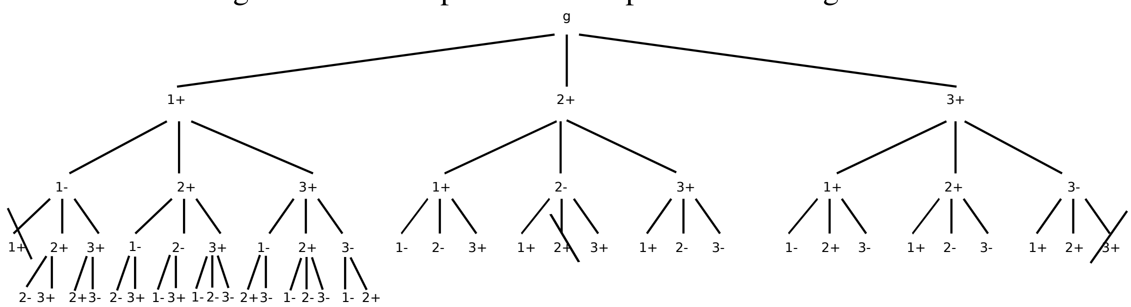
function TRANSFORM COMPONENT INTO ASSIGNMENT REQUEST-BUS(x)
  $a \leftarrow x$
  for $i = 1 : n$ **do**
    $T_i \leftarrow \lfloor \frac{a}{k^{n-i}} \rfloor$
    $a \leftarrow a \bmod k^{n-i}$
  end for
  return T
end function

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!!**

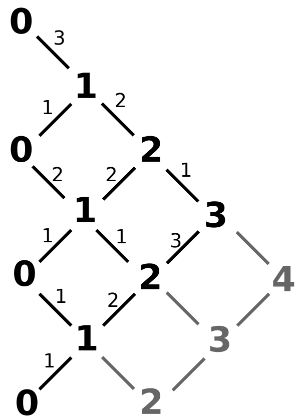
Figure 4.1: Route possibilities represented through a tree



Source: Own authorship

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



Source: Own authorship

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.

Algorithm 3 Matrix Generation

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.

Algorithm 4 Transformation Vector-Solution

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.

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

4.2.3 Attractiveness

As proposed by Yang (2009, p. 173) 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 Yang (2010, p. 80), "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, Yang and He (2013, p. 37-38) presents an 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.

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

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 Yang and He (2013, p. 37-38). 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

It is to note in a regular problem instance that the distribution of requests to bus has a greater weight it the cost function. Empirical observations suggest that this distribution is roughly the most determinant factor in the calculation of the costs. For instance, the grouping requests that are geographically near tends to show better results regardless of the route. Yet, the route of each bus plays an important role in finding an optimal result.

Moreover, as the contribution to the cost function by the vector components that describe the routes is strongly dependent on the distribution of requests to each bus, it makes no sense trying to optimize these components if this assignment varies strongly in the cost minimization procedure.

On the basis of these observations, an optimization process with two phases is advised. In this schema, there is a first stage, where only the first vector component is optimized by moving the fireflies only in this dimension of the search space. In a second stage, the other components are optimized by anchoring the first component and allowing the movement of the fireflies in these other dimension.

[PICTURE SHOWING PROGRESS OF THE ITERATIONS]

4.2.10 Implementation

One of the main difficulties in implementing the algorithm is the magnitude that the numbers may have. To workaround the problem the programming language Python[3] was 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[4] and Scipy[5] were used for that purpose. For drawing the graphs the was adopted library Matplotlib[6], which has an 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)

3. See `https://www.python.org/`
4. See `http://www.numpy.org/`
5. See `http://www.scipy.org/`
6. See `http://matplotlib.org/`

5 EVALUATION

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

6 CONCLUSION

- Further Research

BIBLIOGRAPHY

Apostolopoulos, Theofanis, and Aristidis Vlachos. 2010. "Application of the Firefly Algorithm for Solving the Economic Emissions Load Dispatch Problem". *International Journal of Combinatorics, International Journal of Combinatorics* 2011, 2011 (). ISSN: 1687-9163, 1687-9163.

Berbeglia, Gerardo, Jean-François Cordeau, and Gilbert Laporte. 2010. "Dynamic pickup and delivery problems". *European Journal of Operational Research* 202, number 1 (): 8–15. ISSN: 0377-2217.

Braekers, Kris, An Caris, and Gerrit K. Janssens. 2014. "Exact and meta-heuristic approach for a general heterogeneous dial-a-ride problem with multiple depots". *Transportation Research Part B: Methodological* 67 (): 166–186. ISSN: 0191-2615.

Carvalho, Marco Aurelio de, Semyon D. Savransky, and Tz-Chin Wei. 2004. *121 Heuristics for Solving Problems.* Lulu.com. ISBN: 978-1-4116-1689-9.

Cordeau, Jean-François. 2006. "A Branch-and-Cut Algorithm for the Dial-a-Ride Problem". *Operations Research* 54, number 3 (): 573–586. ISSN: 0030-364X.

Cordeau, Jean-François, and Gilbert Laporte. 2007. "The dial-a-ride problem: models and algorithms". *Annals of Operations Research* 153, number 1 (): 29–46. ISSN: 0254-5330, 1572-9338.

Fourer, Robert, David M. Gay, and Brian W. Kernighan. 1990. "A Modeling Language for Mathematical Programming". *Management Science* 36, number 5 (): 519–554. ISSN: 0025-1909.

———. 2003. *AMPL: A Modeling Language for Mathematical Programming.* 2nd. Pacific Grove, CA: Cengage Learning. ISBN: 978-0-534-38809-6.

Gribkovskaia, Irina, Øyvind Halskau sr., Gilbert Laporte, and Martin Vlček. 2007. "General solutions to the single vehicle routing problem with pickups and deliveries". *European Journal of Operational Research* 180, number 2 (): 568–584. ISSN: 0377-2217.

Häll, Carl H., Henrik Andersson, Jan T. Lundgren, and Peter Värbrand. 2009. "The Integrated Dial-a-Ride Problem". *Public Transport* 1 (1): 39–54. ISSN: 1866-749X, 1613-7159.

Jati, Gilang Kusuma, and Suyanto. 2011. "Evolutionary Discrete Firefly Algorithm for Travelling Salesman Problem". In *Adaptive and Intelligent Systems,* edited by Abdelhamid Bouchachia, 393–403. Lecture Notes in Computer Science 6943. Springer Berlin Heidelberg. ISBN: 978-3-642-23856-7 978-3-642-23857-4.

Jorgensen, Rene Munk, Jesper Larsen, and Kristin Berg Bergvinsdottir. 2007. "Solving the Dial-a-Ride problem using genetic algorithms". *Journal of the Operational Research Society* 58 (10). ISSN: 0160-5682.

Kirchler, Dominik, and Roberto Wolfler Calvo. 2013. "A Granular Tabu Search algorithm for the Dial-a-Ride Problem". *Transportation Research Part B: Methodological* 56 (): 120–135. ISSN: 0191-2615.

Nanry, William P, and J Wesley Barnes. 2000. "Solving the pickup and delivery problem with time windows using reactive tabu search". *Transportation Research Part B: Methodological* 34, number 2 (): 107–121. ISSN: 0191-2615.

Oki, Eiji. 2012. *Linear Programming and Algorithms for Communication Networks: A Practical Guide to Network Design, Control, and Management.* CRC Press. ISBN: 978-1-4665-5263-0.

Osman, Ibrahim H., and James P. Kelly. 2012. *Meta-Heuristics: Theory and Applications.* Springer Science & Business Media. ISBN: 978-1-4613-1361-8.

Parragh, Sophie N., Jean-François Cordeau, Karl F. Doerner, and Richard F. Hartl. 2010. "Models and algorithms for the heterogeneous dial-a-ride problem with driver-related constraints". *OR Spectrum* 34, number 3 (): 593–633. ISSN: 0171-6468, 1436-6304.

Parragh, Sophie N., and Verena Schmid. 2013. "Hybrid column generation and large neighborhood search for the dial-a-ride problem". *Computers & Operations Research* 40, number 1 (): 490–497. ISSN: 0305-0548.

Ropke, Stefan, Jean-François Cordeau, and Gilbert Laporte. 2007. "Models and branch-and-cut algorithms for pickup and delivery problems with time windows". *Networks* 49, number 4 (): 258–272. ISSN: 1097-0037.

Sayadi, Mohammad Kazem, Ashkan Hafezalkotob, and Seyed Gholamreza Jalali Naini. 2013. "Firefly-inspired algorithm for discrete optimization problems: An application to manufacturing cell formation". *Journal of Manufacturing Systems* 32, number 1 (): 78–84. ISSN: 0278-6125.

Sayadi, Mohammad Kazem, Reza Ramezanian, and Nader Ghaffari-Nasab. 2010. "A discrete firefly meta-heuristic with local search for makespan minimization in permutation flow shop scheduling problems". *International Journal of Industrial Engineering Computations* 1, number 1 (): 1–10. ISSN: 19232926, 19232934.

Schrijver, Alexander. 1998. *Theory of Linear and Integer Programming.* John Wiley & Sons. ISBN: 978-0-471-98232-6.

Shenoy, G. V. 2007. *Linear Programming: Methods and Applications.* New Age International. ISBN: 978-81-224-1034-1.

Urra, Enrique, Claudio Cubillos, and Daniel Cabrera-Paniagua. 2015. "A Hyperheuristic for the Dial-a-Ride Problem with Time Windows". *Mathematical Problems in Engineering, Mathematical Problems in Engineering* 2015, 2015 (). ISSN: 1024-123X, 1024-123X.

Wassan, Niaz A., and Gábor Nagy. 2014. "Vehicle routing problem with deliveries and pickups: Modelling issues and meta-heuristics solution approaches". *International Journal of Transportation* 2 (1): 95–110.

Wassan, Niaz A., A. Hameed Wassan, and Gábor Nagy. 2007. "A reactive tabu search algorithm for the vehicle routing problem with simultaneous pickups and deliveries". *Journal of Combinatorial Optimization* 15, number 4 (): 368–386. ISSN: 1382-6905, 1573-2886.

Wolsey, Laurence A., and George L. Nemhauser. 2014. *Integer and Combinatorial Optimization.* John Wiley & Sons. ISBN: 978-1-118-62686-3.

Yang, Xin-She. 2009. "Firefly Algorithms for Multimodal Optimization". *Stochastic Algorithms: Foundations and Applications* 5792:169–178.

———. 2010. "Firefly algorithm, stochastic test functions and design optimisation". *International Journal of Bio-Inspired Computation* 2 (2): 78–84.

Yang, Xin-She. 2012. “Efficiency Analysis of Swarm Intelligence and Randomization Techniques”. *Journal of Computational and Theoretical Nanoscience* 9 (2): 189–198.

Yang, Xin-She, and Xingshi He. 2013. “Firefly Algorithm: Recent Advances and Applications”. *International Journal of Swarm Intelligence* 1 (1): 36–50. ISSN: 2049-4041, 2049-405X.

Zidi, Issam, Khaled Mesghouni, Kamel Zidi, and Khaled Ghedira. 2012. “A multi-objective simulated annealing for the multi-criteria dial a ride problem”. *Engineering Applications of Artificial Intelligence* 25, number 6 (): 1121–1131. ISSN: 0952-1976.