ALGORITHMS FOR MULTI-TRIP VEHICLE ROUTING AND DEVICE TO DEVICE COMMUNICATIONS

LEILA KARIMI
Bachelor of Science, University of Shiraz, 2009
Master of Science, Shahid Chamran University of Ahvaz, 2011

A thesis submitted
in partial fulfilment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

in

THEORETICAL AND COMPUTATIONAL SCIENCE

Department of Mathematics and Computer Science
University of Lethbridge
LETHBRIDGE, ALBERTA, CANADA

ALGORITHMS FOR MULTI-TRIP VEHICLE ROUTING AND DEVICE TO DEVICE COMMUNICATIONS

LEILA KARIMI

Date of Defence: September 26, 2022

Dr. D. Gaur Thesis Supervisor	Professor	Ph.D.
Dr. R. Benkoczi Thesis Examination Committee Member	Professor	Ph.D.
Dr. A. Zovoilis Thesis Examination Committee Member	Assistant Professor	Ph.D.
Dr. R. Yalamova Internal External Examiner Dhillon School of Business	Professor	Ph.D.
Dr. O. Alp External Examiner University of Calgary Calgary, AB	Associate Professor	Ph.D.
Dr. W. Osborn Chair, Thesis Examination Committee	Associate Professor	Ph.D.

Dedication

To my beloved family.

Abstract

Motivated by the transportation needs of modern-day retailers, we consider a variant of the vehicle routing problem with time windows in which each truck has a variable capacity. In our model, each vehicle can bring one or more wagons. The clients are visited within specified time windows, and the vehicles can also make multiple trips. We give a mathematical formulation for the problem and a branch and price algorithm is used to solve the model. In each iteration of branch and price, column generation is applied. Based on the different capacities, different subproblems are created to find the best column. We extend Solomon's instances to evaluate our approach. We report on the computational results using CPLEX. Ours is the first such study to the best of our knowledge. For the second part of the thesis, we study the sharing of spectrum in the device-to-device (D2D) communications in an underlay cellular network. Our model maximizes the total sum-rate such that i) each D2D link is assigned at most one sub-channel, ii) the total interference is at most the required maximum. Our model can also minimize the interference subject to i) the total sum-rate being bounded by some required amount. We give a branch-n-cut algorithm for solving both models. We give a Lagrangian relaxation which is solved optimally and combinatorially for the minimization objective. We give an iterative rounding algorithm that achieves at least a quarter of the optimal sum rate and no more than the required maximum of the total interference when the objective is to maximize sum-rate. Detailed experiments are performed on synthetic as well as network simulator data. Our experiments establish the effectiveness of the branch-n-cut and the iterative rounding approach for channel assignment.

This thesis is a study on the use of branch and cut, branch and price, and iterative rounding for solving two real world optimization problems.

Acknowledgments

I would like to express my special thanks of gratitude to my supervisor, Dr. Daya Gaur for his continues support, motivation, guidance and immense knowledge. It was a great privilege and honor to work and study under his guidance.

Besides my supervisor, I would like to thank the rest of my thesis committee Dr. Robert Benkoczi and Dr. Athanasios Zovoilis for their comments and valuable suggestions throughout my research. I would like to thank Dr. John Shariff, Dr. Howard Cheng and all members of Mathematics and Computer Science department . I am grateful to SGS, Dr. Daya Gaur and Dr. Robert Benkoczi for the financial assistantship.

I am very appreciative of my family's support throughout my life, especially my mother who motivates me to pursue my goals, my sister Maryam Karimi, and also Farzin Keykavoos.

I am extremely thankful to Dr. Salimur Choudhury, Vijay Adoni, Nawrin Chowdhury and Peash Saha for their valuable discussion and suggestion throughout my research.

Last, but not least, I would like to thank my friends who make life easier for me these four years including Ajay Raj, Nawrin Chowdhury, Vijay Adoni, Oluwaseun Lijoka, Abha Goel, Jannatul Maowa.

Contents

List of Tables

List of Figures

Glossary

μ_0 Dual variable associated with number of vehicle. 61

μ_1 Dual variable associated with the number of wagons. 61

0 Depot. 11

A Set of edges between nodes. 11

B Bipartite graph. 85

C Set of customers. 11

C_r Total distance of route r. 45

D Set of D2D pairs. 81

G A complete directed graph. 11

$I_{(c,d)}$ interference for pair (c,d) . 82

J^* Optimal solution of mixed integer problem. 70

J_R Optimal solution of LP-relaxation of mixed integer problem (MILP). 71

L Number of resources. 26

L_i Label on node i. 36, 64

M A big number. 20

N Set of nodes. 11

P Space solution of LP-relaxation of mixed integer problem (MILP). 71

Q Capacity of identical vehicles. 11

S^* Optimal solution of LP-relaxation of D2D problem . 94

T_i^l Quantity of resource l. 30

T_r Departure time of the route r. 61

V Set of vehicles. 11

X Feasible solutions of mixed integer problem. 70

X_{pi} A path. 30

X_{rs} Is one if the route (r,s) is used and zero otherwise. 61

Ω Set of all feasible tours. 49

$\mathbb{R}_{+}^{n}$ Space of n dimensional vectors of positive real numbers . 71

$\mathbb{Z}_{+}^{p}$ Space of p dimensional vectors of positive integer numbers. 71

$\overline{C}_r$ Reduced cost of route w. 46

$\overline{J}$ Upper bound of the optimal solution of mixed integer problem (MILP). 72

$\overline{d}_w$ Reduced cost of a tour w. 50

$\bar{t}_0^r$ Latest departure. 55

$\bar{t}_{n+1}^r$ Latest arrival. 55

$\delta(v)$ Set of edges incident on v . 88

π optimal solution of dual problem. 46

π_i Dual variable associated with each customer. 46

$\underline{t}_0^r$ Earliest departure . 55

$\underline{t}_{n+1}^r$ Earliest arrival . 55

a_{ir} Is one if customer i is in route r, zero otherwise. 45

c_{ij} Cost of edge. 26

d_w Total distance of tour w. 49

d_{ij} Distance between any two customers. 20

d_{ij}^l Consumption of resource l on an edge. 26

d_i Demand of customer. 11

$n+1$ Depot. 11

$s_{(c,d)}$ Sum rate for pair (c,d) . 82

s_{iv} Number of visited nodes on path X_{pi}. 33

t_{ij} Travelling time between any two nodes. 11

$x_{(c,d)}$ Is one if a pair (d) uses radio resources from a cellular user (c), zero otherwise. 82

x_{ijk} Is one if vehicle k, visits customer j immediately after customer i, zero otherwise. 12

y_r Is one if route r is chosen, zero otherwise . 45

Chapter 1

Introduction

The Vehicle Routing Problem (VRP) initially emerged, when Dantzig and Ramser formulated and resolved the problem of supplying fuel to service stations about the end of the fifties of the last century [7].

The VRP definition states that n customers with discrete quantities of goods must be served by m vehicles initially located at a depot. A VRP is to determine the optimal routes taken by a group of vehicles while serving a group of users. The objective is to minimize the overall transportation cost. The solution to the classical VRP problem is a set of routes that all begin and end in the depot while satisfying the constraint that all customers must be visited only once. The transportation cost can be improved by reducing the total travelled distance [8].

The majority of the real-world problems are often much more complex than the classical VRP. Therefore, in practice, the classical VRP problem is augmented by constraints, such as vehicle capacity or time interval in which each customer has to be served, revealing the Capacitated Vehicle Routing Problem ($CVRP$) and the Vehicle Routing Problem with Time Windows ($VRPTW$), respectively. In the last fifty years, many real-world problems have required extended formulation that resulted in the multiple depot VRP, periodic VRP, split delivery VRP, stochastic VRP, VRP with backhauls, VRP with pickup and delivery and many others [8].

VRP is an *NP* hard combinatorial optimization problem that can be exactly solved only for small instances of the problem. Although the heuristic approach does not guarantee optimality, it yields the best results in practice. In the last twenty years the meta-heuristics approach has emerged as the most promising direction of research for the *VRP* family of problems [8].

1.1 Multi-trip *VRP* with time windows (*TW*)

The Multi-Trip Vehicle Routing Problem with Time Windows ($MTVRPTW$) is a type of the classical Vehicle Routing Problem with Time Windows ($VRPTW$) with more than one trip for a vehicle during a workday. A trip is a timed route in a context such that more than one route can be assigned to a vehicle. The multi-trip feature is needed when the vehicle fleet size is limited. In this case, a benefit is a reduced number of drivers and vehicles besides, in practice, industries can't provide an unlimited number of vehicles to serve all customers and they tend to prefer a limited number of vehicles to do more than one trip. Despite its apparent practical relevance, this variant of the classical VRP has not been the subject of a large number of studies.

1.2 Multi-trip *VRP* with a variable number of wagons and *TW*

Multi-trip vehicle routing problem with a variable number of wagons and time windows defines a variant of the classical vehicle problem in which the capacity of vehicles can be determined given the total demand of the route when a vehicle is prepared to leave the depot. In this situation, one, two, or three wagons can be attached to make a vehicle ready to service the customers. The number of wagons and vehicles is limited and the configuration of the vehicle will stay the same during all trips of the vehicle. This new methodology is suitable to decrease time and cost by reducing the number of vehicles, drivers, and fuel consumption which is specifically more important in distributing goods over large distances like two different cities or from large cities to rural areas.

1.3 Device to Device communications

D2D communication enables two devices to communicate without needing a base station. By using direct links, nearby devices can interact with one another. This allows for power savings in the network due to the proximity of D2D users. It enhances energy efficiency and decreases delays and offloads traffic from the base station [9].

The number of smart handheld devices has increased exponentially in recent years. So, applications and services needed by devices nearby are growing in popularity. These applications include communication between emergency vehicles and location-based services. D2D communication can be a crucial enabling technology in case of an emergency like a natural disaster where the traditional network may experience problems, then using D2D; a wireless network can be established [9].

1.4 Literature review

Multi trip vehicle routing problem (*MTVRP*) is an important type of vehicle routing problem in the real world that is studied less compared to other versions of vehicle routing problem, specifically, for exact methods. In 1990, Fleischmann [10] proposed a modification of the savings heuristic and used a Bin Packing Problem heuristic to assign the routes to vehicles with multiple uses of vehicles. Taillard, Laporte, and Gendreau [11] (1996) presented a tabu search algorithm with three phases to solve the problem. Brando and Mercer [12] (1997) proposed another tabu search algorithm with a variable neighborhood to find a solution with the least cost. The algorithm is a three-phase algorithm that creates an initial solution by a heuristic and then uses tabu search (reinsertion and exchange of customers) to improve the solution and finally restore feasibility. Brandão and Mercer (1998) [13] also presented a more complex type of problem when mixed fleets and maximum overtime constraints are allowed. Salhi [14] (1998) proposed the many-to-many location-routing problem. In (2004), Campbell and Savelsbergh [15] described insertion heuristics that can be used effectively when time windows constraints are added to the problem as well. Petch

and Salhi [16] (2004) developed a multi-phase constructive heuristic for the $MTVRP$ which in phase one generates a VRP solution using a savings approach and phase 3 generates a VRP solution by route population approach. Phase 2 is a $VRPM$ construction and improvement stage in which an $MTVRP$ solution is constructed using bin-packing with the minimization of the overtime as the objective. In 2007, Salhi and Petch [17] improved their previous method to a hybrid Genetic Algorithm with the same objective. Olivera and Viera [18] presented an adaptive memory approach to minimize total routing cost. Cattaruzza et al. [19] (2014) used a hybrid genetic algorithm with a new local search operator that is a combination of standard VRP moves and swaps between trips to minimize total travelling time. Naveed et al. [20] (2017) proposed a two level variable Neighborhood Search to generate an $MT-VRPB$ initial solution to minimize the total cost. More heuristic approaches are in [21] (2018), [22] (2019), [23] (2020), and [24] (2020), in which a hybrid genetic algorithm, a simulated annealing, and a hybrid particle swarm optimization algorithm, and a hybrid genetic algorithm are used respectively.

There are a limited number of papers on the exact methods for $MTVRP$. We describe some of them. Desrosiers and Solomon (1992) [25] were the first to use column generation in a Dantzig-Wolfe decomposition framework. Halse (1992) [26] implemented Lagrangean decomposition. After that, Kohl and Madsen (1997) [27] extended Lagrangean relaxation. These approaches were further developed using Dantzing-Wolfe decomposition including cutting planes or parallel platforms in Kohl, Desrosiers, Madsen, Solomon, and Soumis (1999) [28]; Larsen (1999) [29]; Cook and Rich (1999) [30]. A hybrid algorithm, a combination of Lagrangean relaxation and Dantzig-Wolfe decomposition was presented by Kallehauge (2000) [31]. Chabrier, Danna and Le Pape (2002) [32]; Feillet, Dejax, Gendreau and Gueguen (2004) [33]; Rousseau, Gendreau and Pesant (2004) [34]; Larsen (2004) [29]; Chabrier (2005) [35]; Irnich and Villeneuve (2005) [36]; Danna and Le Pape (2005) [37] presented algorithms based on the subproblem methods. Hernandez et al.

[38] and Nabila Azi et al. [1] suggested the branch and price algorithm with two phases, in the first phase, all paths are generated and in the second phase, the problem is solved by column generation. Macedo et al. [39] proposed an approach using a pseudo-polynomial model. More recent works are [40] which presented a branch price cut for multi-trip vehicle routing problem; [41] has two integer programs for the open vehicle routing problem and uses column generation to solve them; [42] gave column generation embedded in branch and price algorithm to solve multi trip vehicle routing problem with time windows using dynamic programming to generate all non-dominated paths by label correcting algorithm which are used in the sub-problem; [43] presented a branch and price algorithm to solve the multi trip vehicle routing problem with time windows and driver work hours.

A heterogeneous fleet vehicle routing problem ($HVRP$) is a variant of VRP. The vehicles are of different capacities located at the depot, and a fixed cost is associated with each vehicle. Many heuristics have been proposed for $HVRP$, like [44], which takes into account the three-dimensional (3D) loading constraints that must be satisfied given the vehicle's capacity. They used a local search and simulated annealing to find a solution. Gendreau et al. [45] proposed a tabu search for heterogeneous fleet vehicle routing problem. Berghida and Boukra [46] developed a quantum-inspired algorithm for a VRP with heterogeneous fleet mixed backhauls and time windows. Penna et al. [47] solved the problem using a hybrid heuristic that mixed a local search with a variable neighbourhood search. An updated version of the Green Vehicle Routing Problem with time windows is presented in [48]. To optimize the routing of a mixed vehicle fleet made up of electric and conventional vehicles, they suggest an iterative local search algorithm. [49] is a survey on heterogeneous vehicle routing problem and its variants. Additionally, the paper compares the metaheuristic algorithms that have been suggested. We can find only a small number of exact algorithms for $HVRP$. A column generation approach for the heterogeneous fleet vehicle routing problem was proposed in [50]. Pessoa et al. [51] studied a branch-cut-and-price algorithm for

HVRP. Another branch-cut-and-price algorithm for multi depot *HVRP* is in [52] in which different cuts and pricing strategies are proposed and implemented. Sundar and Rathinam [53] presented a branch and cut for multi-depot heterogeneous vehicle routing problem that customers are divided into two distinct subsets: those that must be visited by particular vehicles and those that can be reached by any vehicle. Bettinelli et al. [54] provided a branch-cut-and-price algorithm for multi-depot heterogeneous-fleet pickup and delivery problems with soft time windows. Later, Pessoa et al. [55] improved the branch-cut-and-price algorithm for heterogeneous fleet vehicle routing problems. *HVRP* assumes that each can make only one trip during a work day, and it has not been studied in the context of multiple trips. All these exact methods used a set covering formulation. In the case of cut or column generation, the vehicle capacity is affected given a new column which is a solution to the sub-problem. Next, we comment on how the problem in this thesis is different from *HVRP*. In our problem, there are a limited number of trailers and wagons. The only budget is an existing number of wagons, so there is no extra cost for a trailer when wagons are attached to it. These wagons are attached to trailers based on the routes. This gives vehicles three different capacities. No extra cost is incurred to have a vehicle with a different capacity. In this work, a trailer has the possibility of having various capacities. To the best of our knowledge, a mathematical model for our problem has not been studied in the past. Past heuristic and exact methods on related problems use vehicle capacities when constructing routes, not in the mathematical model. The problem in this thesis allows the vehicle to change configuration (different capacities) daily. Our work is a new variant for *MTVRP*.

In addition to the multi-trip VRP, we present the vehicle routing problem with dynamic capacity in this thesis. We are not aware of any work on exact methods for the multi trip vehicle routing problem with time windows and with the flexibility of having different wagons attached to service customers as in this thesis.

The second part of the thesis is on device-to-device communication and approaches

such as iterative rounding and branch and cut to solve the problem. Device to device (D2D) communication occurs when neighboring cellular devices communicate directly with each other typically over a shared spectrum. D2D communication improves the transmission rate, frequency reuse, and reduces the hop count. But can lead to interference with a cellular user, and even stronger interference with another D2D pair in a neighboring cell. Therefore new approaches are needed for radio resource allocation [56]. The approaches should increase the system sum rate without too much deterioration in the signal, and the total interference should be limited.

With the emergence of 4G and 5G networks, resource allocation for D2D communication underlaying cellular networks has been studied extensively. These studies deal with various aspects such as minimization of interference, maximization of sum-rate, fair allocation of resources, restricted allocation of resources, improving run-time, uniform graph, exact or heuristic algorithms, etc.

One of the early papers in the area is [57] which proposed a mechanism for session setup and management for D2D communications in LTE-A networks. Islam et al. [58] proposed a minimum knapsack-based resource allocation for D2D communication (MIKIRA) underlaying cellular networks. This algorithm which takes $(O(n^2 log(n))$ time, is a knapsack-based approach aimed at maintaining a target sum rate while minimizing interference. This approach wasn't fair as the algorithm stopped assigning resources when the sum rate was met. Islam et al. [59] addressed the fairness issue with a two-phase auction-based fair resource allocation algorithm (TAFIRA). The algorithm starts with a minimum interference solution and tries to gain a better sum rate via an auction. Hassan et al. [6] showed that MIKIRA doesn't provide a feasible solution in most cases and that TAFIRA has an unbounded integrality gap. They gave a two-phase algorithm, a maximum weight matching is found in phase one using the Hungarian method, and a local search algorithm improves this matching in phase two. Saha et al. [5] proposed a two-phase polynomial-time algorithm when interferences are uniform when the objective is to minimize the interferences while

satisfying a target sum rate. Phase one is similar to the first phase in [6] and phase two improves on phase one by iteratively finding special triples.

Hussain et al. [60] studied one-to-many and many-to-many sharing of resources. Zhang et al. [61] formulated the interference relationships among different D2D and cellular communication links into a novel interference graph with newly defined attributes and proposed the InGRA algorithm that can effectively lead to a near-optimal solution with low computational complexity. They showed that D2D communication increases the total throughput. Janis et al. [62] gave a scheme to monitor interference between cellular users and D2D pairs, and use it to minimize interference. Their simulations demonstrate substantial gains in sum-rate. Xu et al. [63] developed a second price auction for the allocation of spectrum resources. The D2D pairs bid in sequence. Their method shows an improved sum rate and fairness in simulations. A reverse iterative combinatorial auction in which spectrum resources are auctioned off as good was proposed by [64]. They prove that auction is cheat-proof and converges. Simulations show that the method yields a good sum-rate. An admissibility-based approach was developed by [65]. The scheme has three phases. Admissible D2D pairs are identified based on the distance from the base station in the first phase. This ensures that sum-rate requirement is not violated. In the second stage, power is allocated to each D2D pair and cellular user. The third stage identifies the resource allocation by solving a weighted matching problem.

In the next chapter, we describe the methodology and present the work that we build on in this thesis.

1.5 Original contributions in this thesis

A list of the original contributions in this thesis is as follows:

1. We give the definition and mathematical model for MTVRP-VW-TW in section 2.4 and 2.5.

2. Column generation for MTVRP-VW-TW is presented in section 4.3.

3. MTVRP-VW-TW is solved using the branch and price algorithm in section 5.2.

4. We provide a branch and cut algorithm to solve D2D communication to maximize the total sum-rates in section 6.2.

5. We describe a Lagrangian relaxation for solving D2D in section 6.3.

6. An iterative rounding algorithm for solving D2D and a proof of the performance ratio is given in section 6.4.

7. A branch and price algorithm is implemented to solve MTVRP-VW-TW. We used Solomon's instances to test the algorithm. It is the first time that the model is presented and solved with exact methods. We could find the optimal solution for types $R1$, $R2$, $RC1$, and $RC2$ of Solomon's instances in section 7.1.

8. Detailed experiments for D2D are performed on synthetic as well as network simulator data. We evaluated the performance of the branch and cut algorithm and the iterative rounding algorithm on both simulated network instances, and computationally hard instances in section 7.2.

Chapter 2

Vehicle Routing Problem

About seventy years ago, Dantzig and Ramser gave the mathematical formulation and an algorithm to solve the problem of distributing gasoline to a set of stations [7]. That was the beginning of research on the Vehicle Routing Problem (VRP). Later, more research emerged to meet the diverse demand of clients.

2.1 Vehicle Routing Problem

This section is the study of the distribution of goods between depots and customers, which have some requests and requirements that need to be satisfied. Vehicles have prescribed capacities, and all start working from the depot, visiting customers, and returning to the depot again. Several variants of vehicle routing problems arise from this basic setup in the real world. Below we explain some of them and the methods to solve them.

2.2 Basic Model

2.3 Vehicle Routing Problem and its variants

Vehicle Routing Problem is an extension of the traveling salesman (TSP) problem where a set of tours must be determined to visit all the customers instead of one tour as in TSP. In TSP there is one vehicle with unlimited capacity that must visit all the customers using a tour of minimum cost. An example of TSP and its solution is shown in Fig. 2.1. The costs of traversing the edges are given on each edge. There is zero demand at customers.

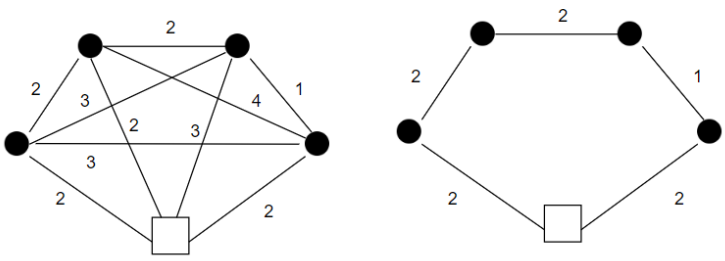


Figure 2.1: Example with one vehicle

If there are demands at the customers, then it may not be possible to serve all customers by one vehicle with fixed capacity, and a fleet of vehicles is required. After explaining the Capacitated Vehicle Routing Problem below, we will provide one example to illustrate the differences between TSP and VRP.

In the classical version of VRP, any tour starts at the depot, services several customers, and returns to the depot again. Each customer is visited exactly once.

2.3.1 Capacitated Vehicle Routing Problem

The most simple and basic type of VRP is the Capacitated Vehicle Routing Problem ($CVRP$). In this case, each vehicle has a specific capacity (Q). The vehicles are all homogeneous and can perform at most one route per day. Each customer has a deterministic demand that must be satisfied exactly once, which means the demand can not be split, and everything is determined in advance; therefore, none of these conditions can change.

The problem is specified as a complete directed graph $G = (N, A)$ where $N = \{0, \ldots, n+1\}$ is a set of nodes, 0 and $n+1$ are the depot and $C = \{1, \ldots, n\}$ is the set of customers, there are edges between all customers as well as depot defined as $A = \{(i, j) : i, j \in N, i \neq j\}$ and a set of vehicles V. It must be noted that there are no arcs ending at vertex 0 or originating from vertex $n+1$. The distance between every two customers is the cost of the edge (t_{ij}), and each customer has a demand (d_i). Each vehicle can perform at most one

route, starting at vertex 0, ending at vertex $n+1$, and visiting each customer at most once. Decision variable x_{ijk} is one if vehicle k, visits customer j immediately after customer i, zero otherwise. The aim is to design a set of routes to minimize the total distance traveled by all vehicles while visiting every customer. One simple mathematical model is presented below:

$$\min \sum_{i\in N}\sum_{j\in N}\sum_{k\in V} t_{ij}x_{ijk} \tag{2.1}$$

s.t.

$$\sum_{k\in V}\sum_{i\in N} x_{ijk} = 1 \qquad \forall j \in C \tag{2.2}$$

$$\sum_{i\in C}\sum_{j\in N} d_i x_{ijk} \leq Q \quad \forall k \in V \tag{2.3}$$

$$\sum_{j\in N} x_{0jk} = 1 \quad \forall k \in V \tag{2.4}$$

$$\sum_{i\in N} x_{ihk} - \sum_{j\in N} x_{hjk} = 0 \quad \forall h \in C,\ k \in V \tag{2.5}$$

$$\sum_{i\in N} x_{i,n+1,k} = 1 \quad \forall k \in V \tag{2.6}$$

$$x_{ijk} \in \{0,1\} \qquad \forall i,j \in N,\ k \in V \tag{2.7}$$

Constraints (2.2) indicate that each customer must be visited exactly once, and each trip starts and ends at the depot (2.4), (2.6). The sum of the demand of customers visited by a vehicle must not exceed the vehicle capacity (2.3). Finally, the flow constraints assert that the vehicle must not stop on customers (2.5). Fig. 2.1 is an instance of a *CVRP* with demands on customers. Several vehicles with a capacity of 6 will serve the customers. The optimal solution to the problem is in Fig. 2.3. Four vehicles are needed to serve each customer in one trip as they cannot serve any pair of customers due to the vehicle capacity and customer demands.

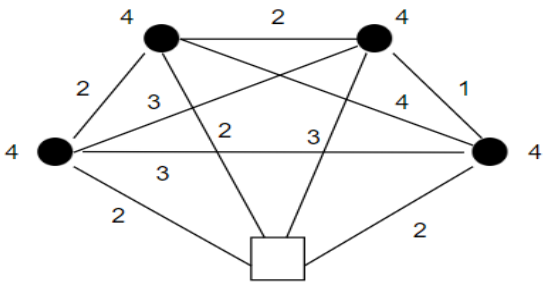


Figure 2.2: Vehicles with $Q = 6$

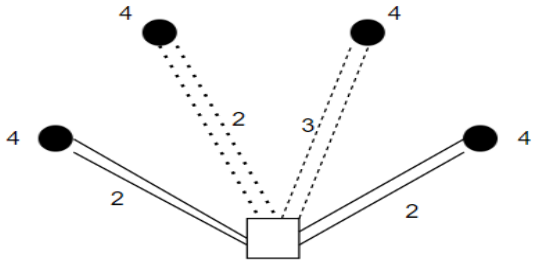


Figure 2.3: 4 tours with cost=18

After getting familiar with *CVRP*, we study more variants of the problem.

2.3.2 Vehicle Routing Problem with Time Windows (*VRPTW*)

Solomon and Desrosiers [66], in 1988, proposed *VRPTW* for the first time. There is an interval of time for each customer that the goods must be delivered and each vehicle has a fixed capacity. This kind of problem usually is of two different types: soft time windows, in that customers, will accept the delivery, even late, hard time windows when the customers don't accept the products that arrive outside of their time window.

2.3.3 Multi-Depot Vehicle Routing Problem (*MDVRP*)

The classic *VRP* has a unique warehouse for all routes and vehicles, while multiple depot VRP that Laporte [67] develops says that vehicles can depart from and return to multi warehouses. Therefore, the depot that a customer will be visited from must be decided in addition to the decisions for *VRP*.

2.3.4 Split Delivery Vehicle Routing Problem (*SDVRP*)

The only assumption in *VRP* that will change for the split delivery *VRP* is that a customer may be visited more than once as the demand may be serviced over multiple visits. An optimal solution to *SDVRP* always uses a minimum number of vehicles.

Fig. 2.4 shows an example *SDVRP*, and its optimal solution.

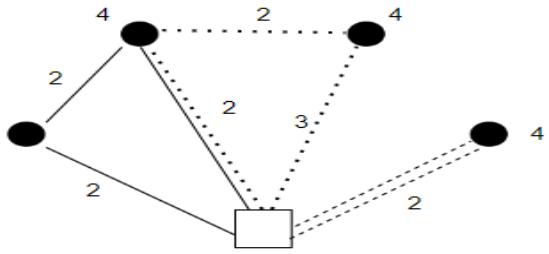


Figure 2.4: 3 Vehicles of capacity 6 with cost=17

One of the customers is visited two times, and each vehicle served two units of four unit demands that the customer requested. Ultimately, All four customers are satisfied. Therefore, three vehicles could serve all customers instead of four as explained in Fig. 2.3 with less cost.

2.3.5 Dynamic Vehicle Routing Problem (*DVRP*) and Vehicle Routing Problem with Static Demand (*VRPSD*)

Dynamic Vehicle Routing Problem is one of the important classes of VRP in which new orders can arrive during the execution of the routes. In the static routing problem, all customers are known a priori, while new customers can request service once the vehicles start the routes in a dynamic VRP [68]. In contrast to the classical definition of the problem, real-world problems have other dimensions like revolution and quality of information [69]. Revolution of information means that the information that the planer has access to can change when the vehicles are executing the tours like a new customer can request to be served. Quality of information relates to the uncertainty of the available data; in this case, all

customers are known. Still, their demands are not certain. The planer has an estimation of the demands. *VRP* with static demand can be planned for and solved statically to determine all routes apriori. So based on these characteristics we have Fig. 2.5 to show the different cases.

		Information evolution	
		Known	Dynamic
Information quality	Stochastic	Stochastic and static	Stochastic and dynamic
	Deterministic	Deterministic and static	Deterministic and dynamic

Figure 2.5: Vehicle Routing Problem by evolution and quality information

2.3.6 Vehicle Routing Problem with pickups and deliveries (*PDP*)

PDP is required when we have multiple origins and destinations. A customer location can either be for pickup or delivery (or both). Other aspects of the problem, like whether the customer location is the origin and destination of the commodity or if a product is being transshipped, whether pick up and delivery points are paired or unpaired, can arise. Any of these situations can make the problem a little bit different in modeling and solving. For paired locations, the problem definition from [70] is a directed graph $G = (V,A)$ where the set of vertices is V, and A is the set of arcs connecting pairs of vertices. Set $P \subset V$ is the set of pickup locations, whereas $D \subset V$ is the set of delivery locations, and vertex $0 \in V$ is the depot where all routes start and end. Every arc $(i, j) \in A, i, j \in V$ has an associated cost c_{ij}, and time t_{ij} to travel from location i to j.

A customer has a demand and a pair of locations. The vehicle must pick up and deliver from this pair. The vehicles have capacities as usual. One example and its solution from

[71] is in Fig. 2.6.

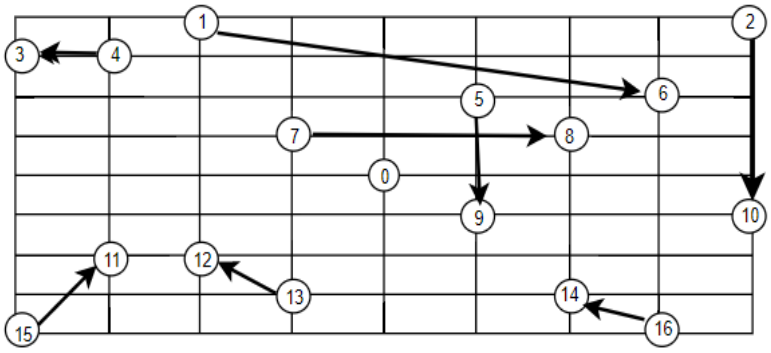


Figure 2.6: Vehicle Routing Problem with Pickup and Deliveries

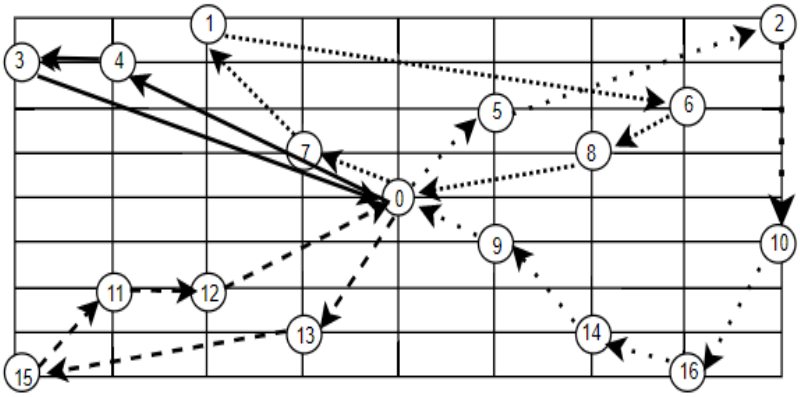


Figure 2.7: Routes for Vehicle Routing Problem with Pickup and Deliveries

2.3.7 Periodic Vehicle Routing Problem (*PVRP*)

In *PVRP*, schedules are created for T-days to service customers and satisfy their demands. Each customer has a certain daily demand and a frequency that specifies how often the customer must be visited during these T-days. The solution is to have T-sets of routes by considering a customer's daily demand, frequency, and the capacity of the vehicles.

2.3.8 Heterogeneous fleet vehicle routing problem (*HVRP*)

HVRP is a variant of the vehicle routing problem in that there is a heterogeneous fleet of vehicles to service customers. In contrast, each vehicle has a different cost, and the number of each type of vehicle is assumed to be unlimited. The problem is to find a set of routes to visit all customers while minimizing the cost [72].

2.3.9 Multi-Trip Vehicle Routing Problem (*MTVRP*)

In *VRP*, each vehicle is able to perform a trip during a workday which is extended to a number of trips in *MTVRP*. Therefore, the vehicle can return to the depot after servicing a number of customers and start a new trip after reloading the goods. The number of trips may be fixed at the beginning of the workday. *MTVRP* is the model studied in this thesis.

2.3.10 Vehicle Routing Problem with Variable Number of Wagons

This thesis introduces this model. This is the first study that uses a variable number of wagons to build different capacities for a vehicle. In this way, the capacity is adjustable, and the vehicle can even have a new capacity the day after. Three different capacities are used to configure the vehicles. It is supposed that a number of wagons exist, and the wagons can be hitched to the vehicle for the workday. In this study, one, two, or three wagons are attached, and the mathematical formulation in section 2.5 is written based on this assumption.

2.3.11 Multi-trip Vehicle Routing Problem with a Variable Number of Wagons and Time Windows

Our study is on Multi-trip vehicle routing problem with a variable number of wagons and time windows. Vehicle routing problem with a variable number of wagons is illustrated next. The model we are interested in is a combination of all these variants listed below. At the beginning of the workday, the vehicle is configured. During different trips, the capacity of the vehicles remains the same, either one wagon, two wagons, or three for the day. Each vehicle can make a few trips during the workday to service clients during a day. The clients

must be visited in time windows. A mathematical model will be described in section 2.5. Fig. 2.8 shows an instance of the problem.

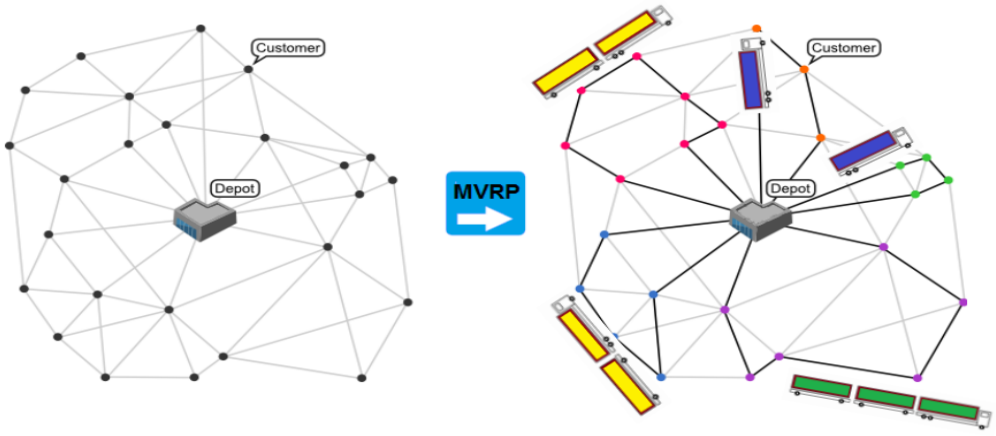


Figure 2.8: Multi-trip Vehicle Routing Problem with a Variable Number of Wagons and Time Windows

2.4 Multi-trip Vehicle Routing Problem with a Variable Number of Wagons and Time Windows

An instance of this problem is defined by a set of customers $C = \{1,2,\cdots,n\}$ and the depot is represented by the vertices 0 and $n+1$. Depot 0 is the start depot, and $n+1$ is the return depot. The set $\{0,1,\ldots,n+1\}$ is denoted N in a complete directed graph $G = (N,A)$, where A is a set of arcs $\{(i,j) : i \neq j, i,j \in N\}$. A traveling time of t_{ij} is associated with each arc $(i,j) \in A$ which we consider as the distance of two vertices i and j, where $i,j \in N$. A fleet of wagons W, with identical capacities q which can be connected as one wagon, two wagons or three wagons to organize a set of vehicles V. Vehicles in V are used to serve the customers. Then, $|V|$ and $|W|$ are the number of vehicles and wagons, respectively. The set of arcs A represents all the connections between customers and the depot. There are no arcs ending at vertex 0 or originating from vertex $n+1$. Any customer $i \in C$ has a demand d_i, a service time s_i, and a time window $[a_i,b_i]$, which means that a vehicle must arrive at the customer before b_i. If it arrives before the time window opens, it has to wait until

a_i to service the customer. The time windows for both depots are assumed to be $[a_0, b_0]$, representing the scheduling horizon. The vehicles may not leave the depot before a_0 and must return at the latest time b_0. A route of a vehicle is a closed path that the vehicle starts from the depot, visits a number of customers based on the left capacity of the vehicle and the demand of a customer in addition to the time windows requested for a customer, and comes back to the depot again. The workday of each vehicle is a sequence of routes where each route starts and ends at the depot and we call it a tour. These routes are denoted by the set R and $|R|$ is a fixed number for all vehicles in our problem.

Each vehicle can have a configuration of attaching one wagon or two wagons or three ones. The configuration of each vehicle must stay the same on all routes. Next, we define some of the mathematical model's decision variables. The decision variable s_{ir}^k denotes the time that the vehicle k starts to service customer i in route r. If the vehicle k does not service customer i in route r, s_{ir}^k has no meaning; consequently, its value is considered irrelevant. Variable x_{ijr}^k is one if vehicle k drives directly from customer i to customer j and zero otherwise in route r. Variable z_n^k is used to determine how many wagons vehicle k needs, where n is in $\{1,2,3\}$. If $z_2^3 = 1$, vehicle 3 has 2 wagons. Therefore, z_1^3 and z_3^3 must be 0, which means vehicle 3 does not have 1 or 3 wagons. Moreover, finally, q^k is the k^{th} vehicle's capacity, depending on the number of wagons attached.

$$x_{ijr}^k = \begin{cases} 1 & \text{if vehicle } k \text{ drives directly from vertex } i \text{ to vertex } j \text{ on route } r \\ 0 & \text{otherwise} \end{cases}$$

$$\mathrm{z}_n^k = \begin{cases} 1 & \text{if the } n \text{ wagons are attached for vehicle } k \\ 0 & \text{otherwise} \end{cases}$$

We assume $a_0 = 0$ and therefore $s_{0r}^k = 0$, for all k and r. The goal is to design a set of routes that minimizes the total distances of all routes such that

- Each customer is serviced exactly once;
- Every route starts at vertex 0 and ends at vertex $n+1$;
- The time windows of the customers are satisfied;
- The total demand on a route can not exceed the capacity of the vehicle, which depends on the number of wagons attached to it (1, 2 or 3);
- Total number of the wagons used should be less than W;
- A vehicle is assigned only one configuration;
- Each vehicle must leave the depot 0 ;
- All vehicles must return to the depot $n+1$;
- The start time of the next route by the same vehicle should be after the finishing time of its previous route;

An unused vehicle is modeled by driving the "empty" route $(0, n+1)$. We consider $d_{ij} = t_{ij}$ in this problem and M is a big number.

2.5 Mathematical Model

The mathematical model is described next.

$$\min \sum_{k\in V}\sum_{r\in R}\sum_{i\in N}\sum_{j\in N} d^k_{ijr}x^k_{ijr} \tag{2.8}$$

s.t.

$$\sum_{k\in V}\sum_{r\in R}\sum_{j\in N} x^k_{ijr} = 1 \qquad \forall i \in C \tag{2.9}$$

$$\sum_{i\in C} d_i(\sum_{j\in N} x^k_{ijr}) \leq q^k \qquad \forall k \in V,\ \forall r \in R \tag{2.10}$$

$$q^k = \sum_{n=1}^{3} nqz^k_n \qquad \forall k \in V \tag{2.11}$$

$$\sum_{n=1}^{3}\sum_{k\in V} nz^k_n \leq |W| \tag{2.12}$$

$$\sum_{n=1}^{3} z^k_n = 1 \qquad \forall k \in V \tag{2.13}$$

$$\sum_{j\in N\setminus\{0\}} x^k_{0jr} = 1 \qquad \forall k \in V,\ \forall r \in R \tag{2.14}$$

$$\sum_{i\in N} x^k_{ihr} - \sum_{j\in N} x^k_{hjr} = 0 \qquad \forall h \in C,\ \forall k \in V,\ \forall r \in R \tag{2.15}$$

$$\sum_{i\in N\setminus\{n+1\}} x^k_{i,n+1,r} = 1 \qquad \forall k \in V,\ \forall r \in R \tag{2.16}$$

$$s^k_{ir} + s_i + t_{ij} - M(1-x^k_{ijr}) \leq s^k_{jr} \quad \forall i \in N\setminus\{n+1\},\ \forall j \in N\setminus\{0\},\ \forall k \in V,\ \forall r \in R \tag{2.17}$$

$$a_i \sum_{j\in N-\{0\}} x^k_{ijr} \leq s^k_{ir} \leq b_i \sum_{j\in N-\{0\}} x^k_{ijr} \qquad \forall i \in C,\ \forall k \in V,\ \forall r \in R \tag{2.18}$$

$$s^k_{0r} \geq s^k_{n+1,r-1} \qquad \forall r \in R,\ \forall k \in V \tag{2.19}$$

$$x^k_{ijr} \in \{0,1\} \qquad \forall i,j \in N,\ \forall k \in V,\ \forall r \in R \tag{2.20}$$

$$z^k_n \in \{0,1\} \qquad \forall k \in V,\ n \in \{1,2,3\} \tag{2.21}$$

$$s^k_{ir} \geq 0 \qquad \forall i \in C,\ \forall k \in V,\ \forall r \in R \tag{2.22}$$

$$q^k \geq 0 \qquad \forall k \in V \tag{2.23}$$

The objective function (2.8) minimizes the total distances of tours. The constraints (2.9) ensure that each customer is visited exactly once. Equations (2.10) and (2.11) state that the total demand on a route can not exceed the capacity of each vehicle which depends on the number of wagons attached to it. The constraint in (2.12) shows the number of wagons in total. Constraints (2.13) ensure that a vehicle is assigned only one configuration. Equations (2.14), (2.15), and (2.16) indicate that each vehicle must leave the depot 0; flow conservation constraints; and finally, all vehicles must return to the depot $n+1$. The inequalities (2.17) establish the relationship between the vehicle departure time from a customer and its immediate successor. Constraints (2.18) assert that the time windows are observed. Constraints (2.19) ensure a proper trip sequencing for the workday of a vehicle that the starting time of the next trip of the vehicle must be after the finishing time of its previous trip. Equations (2.20) and (2.21) are integer variables.

2.6 Solution Methods

A huge range of exact and approximate solution methods exist to solve *VRPs*. While exact methods provide optimal solutions, approximate methods, usually heuristic and metaheuristic approaches, yield a solution near to the optimal solution but not optimal. Exact methods for *VRP* can be applied to small instances of up to 50 customers, whereas approximate algorithms are applicable to any number of customers. Sometimes a combination of both methods can obtain a better and faster solution. [73] is a good review of the variety of approaches. Fig 2.9 presents a number of approaches to solving *VRP*.

2.6.1 Exact Methods

An exact method obtains an optimal solution. One of the main approaches used to solve variants of *VRP* is Branch and X, where X can be bound, cut, or price. Branch and X model *VRP* as an integer or mixed-integer problem and solves it using a tree search method. At each node in the search tree, an LP relaxation is solved to obtain a bound. Cuts

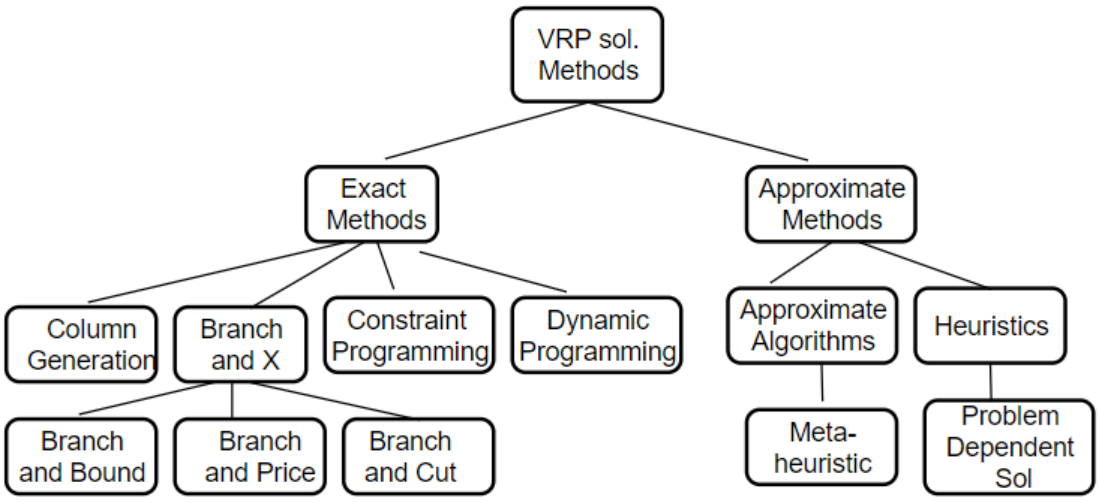


Figure 2.9: *VRP* solution methods

may be added at each node, and columns may also be added in case the number of variables is exponential, to begin with. Some of these works are Dell'Amico et al. [74] which is a branch and price approach to *VRP*, Fukasawa et al. [75] describes branch and cut and price for *CVRP*, Lysgaard et al. [76] is about branch and cut algorithm for *CVRP*, Fischetti et al. [77] presents a branch and bound algorithm for *VRP* and another branch and bound algorithm is in Laporte and Nobert [78].

An optimal solution of the linear problem is computed in the first iteration and gradually in each iteration, we improve to get closer to an integer solution, and ultimately an integer solution which is optimal for the *VRP* is obtained.

For other exact approaches, we mention Dynamic programming [79] and Column generation [80]. Dynamic programming is used for variants of *VRPs*. The idea is to break the problem into simpler sub-problems and solve them separately, then combine the solutions to sub-problems to get an overall solution.

Column generation solves the problem by starting from a feasible solution to a restricted problem where a limited number of columns are considered. The columns are gradually added to the problem and solved again until achieving an optimal solution.

Constraint programming (CP) [81] is another popular method for solving *VRP*. *CP* at heart tries to infer new constraints given the relationships between the variables and uses these constraints to reduce the search space.

2.6.2 Approximate Methods

Whereas exact methods cannot solve *VRPs* with more than 50-100 customers in a reasonable time, approximate methods can give a solution near the optimal one, fast. Constructive heuristics and local search are two approximate methods that are commonly used. A constructive heuristic is applied to a large number of start points to generate good solutions in a short time efficiently. The way they work is usually to start with an empty trip and add customers one by one. Finally, customers are embedded in various trips. In [82], we can find good examples of heuristics for VRP. On the other hand, local search can be combined with a heuristic to give a more efficient solution. They are lots of local search methods [83, 84] that usually move or exchange the clients to find a better solution. There are several neighborhood change operators like Relocate, Split-to-single, 2-opt, Cross Exchange, Combine operators, etc.

Meta-heuristic is another common approximate approach where the local search operators and a construction heuristic are combined to obtain a solution for a large search space. Other meta-heuristic methods for *VRP* are Simulated Annealing (SA) [85, 86], Deterministic Annealing (DA) [87], Tabu Search (TS) [88, 89, 90], Genetic Algorithms (GA) [91, 92], Ant Systems (AS) [93, 94], and Neural Networks (NN) [95]. The first three algorithms start from an initial solution x_1 and move at each iteration t from x_t to a solution x_{t+1} in the neighborhood $N(x_t)$ of x_t, until a stopping condition is satisfied. If $f(x)$ denotes the cost of x, then $f(x_{t+1})$ is not necessarily less than $f(x_t)$. As a result, care must be taken to avoid cycling. GA examines at each step a population of solutions. Each population is derived from the preceding one by combining its best elements and discarding the worst.

Ant systems is a constructive approach in which several new solutions are created at each iteration using some of the information gathered at previous iterations. Tabu search, genetic algorithms, and ant systems are methods that record, as the search proceeds, information on solutions encountered and use it to obtain improved solutions. Neural networks is a learning mechanism that gradually adjusts a set of weights until an acceptable solution is reached [96]. Usually, a few of them will be used together to obtain a better solution. This increases the calculation time, and the result is more effective.

Chapter 3

Elementary Shortest Path Problem with Resource Constraints

To generate all tours needed to solve the subproblem in column generation, we need to know the shortest path problem and elementary shortest path problem. In this section, we describe the shortest path problem. Label correcting algorithm is then described which is an approach to generating elementary shortest paths in section 3.2.

Elementary Shortest Path Problem: Find a path of minimum cost in a weighted directed graph $G = (V,A)$ from a source node s to a destination node t, each node on the path is visited only once. In shortest path problem, nodes can be visited more than once.

Shortest Path Problem with Resource Constraint: Find the shortest path from a source to a destination that meets a number of constraints defined based on some resources.

Elementary Shortest Path Problem with Resource Constraint: ($ESPPRC$)

Let $G = (V,A)$ be a directed network, where A is the set of arcs and $V = \{v_1, \ldots, v_n\}$ is the set of nodes, including an origin node p and a destination node d. A cost c_{ij} is associated with each arc $(v_i, v_j) \in A$. Let L be the number of resources and $d_{ij}^l \geq 0$ be the consumption of resource $l \in L$ along arc (v_i, v_j). Values d_{ij}^l are assumed to satisfy the triangle inequality for each resource l. With each node v_i and each resource l are associated two nonnegative values a_i^l and b_i^l, such that the total consumption (t_i^l) of resource l along a path from p to v_i is constrained to the interval $[a_i^l, b_i^l]$. If the consumption of resource l is lower than a_i^l when the path reaches v_i, it is set to a_i^l. Note that this notation is natural for the time resource, but also allows us to represent capacity constraints by defining intervals $[0, Q]$ on nodes, where

Q is the capacity limit. The objective is to generate a minimum cost elementary path from p to d that satisfies all the resource constraints. ESPPRC is NP-complete and can be solved by dynamic programming.

The following is a mathematical model for *ESPPRC*:

$$\min \sum_{(v_i,v_j)\in A} c_{ij}x_{ij} \tag{3.1}$$

s.t.

$$\sum_{(v_i,v_j)\in A} x_{ij} - \sum_{(v_j,v_i)\in A} x_{ji} = 0 \qquad \forall v_j \in V \setminus \{p,d\} \tag{3.2}$$

$$\sum_{(p,v_j)\in A} x_{pj} = 1 \tag{3.3}$$

$$\sum_{(v_j,d)\in A} x_{jd} = 1 \tag{3.4}$$

$$t_i^l + d_{ij}^l - t_j^l + Mx_{ij} \leq M \qquad \forall l \in \{1,\ldots,L\}, (v_i,v_j) \in A \tag{3.5}$$

$$a_i^l \leq t_i^l \leq b_i^l \qquad \forall l \in \{1,\ldots,L\}, v_i \in V \tag{3.6}$$

$$x_{ij} \in \{0,1\} \qquad \forall (v_i,v_j) \in A \tag{3.7}$$

$$t_i^l \geq 0 \qquad \forall l \in \{1,\ldots,L\}, v_i \in V \tag{3.8}$$

Where x_{ij} and t_j^l are variables representing flow and resource consumption respectively and M is a big number.

The objective function (3.1) is to minimize the total cost. Constraints (3.2) ensure that the vehicle leaves each customer after visiting. Equations (3.3) and (3.4) state that the vehicle starts from the origin p and returns to the destination d. The constraints (3.5) show that the starting time of visiting a customer must be after its immediate successor. Equations (3.7) and (3.8) are the bounds on the variables.

3.1 Dynamic Programming Algorithms

Dynamic Programming is a technique for solving complex problems by dividing them into simpler sub-problems in such a way that the optimal solution of smaller ones can be used to get the optimal solution of the overall problem.
Among various methods presented to solve SPPRC, the Label correcting algorithm is the only dynamic programming algorithm to solve ESPPRC in a graph with negative costs.

3.2 Label correcting algorithms

Label correcting algorithm is one of the popular algorithms used to find a shortest path in a weighted directed graph with negative costs (the algorithm doesn't work when there are negative cost cycles). The idea is to progressively discover the path from the origin to every other node. At the start, every node has an overestimate on the length of the path from the origin to the node, then iteratively those lengths are revised based on a shorter path.
Below we present Bellman-Ford shortest path algorithm first and then Desrochers' Algorithm [97] for solving the shortest path problem with resource constraints and then an extension of the algorithm presented in [33] will be explained that can solve the elementary shortest path problem with resource constraints.

3.2.1 Bellman-Ford Shortest Path Algorithm

The algorithm computes the shortest path from a source to all vertices and works by relaxing the edges like Dijkstra's algorithm. Although the algorithm is slower than Dijkstra's, it can be used when the graph has negative edge weights and can also determine negative cycles if there is one.
Bellman-Ford algorithm [98] overestimates the length of the path from the source vertex to all other vertices which is infinity for all vertices except the resource at the start and then it starts relaxing these overestimate of lengths by discovering the new paths which are shorter than the previous ones. This process is followed for all the vertices $V-1$ times to find the

shortest path. The steps of the algorithm are:

- Initialize the distance to the source vertex as 0 and all other vertices are at distance ($d(v)$) infinity.
- Run the algorithm for $V-1$ iterations and in each iteration do the following relaxation on the edges. For every edge (u,v) with the length of c_{uv}, if $d(v) < d(u) + c_{uv}$, then $d(v) = d(u) + c_{uv}$.
- Detecting the negative cycles if there is any update to some $d(v)$ after $V-1$ iterations. The negative cycles easily can be detected by one extra round, run the program for all edges if $d(v) > d(u) + c_{uv}$, then the negative cycle is determined.

Algorithm 1 Bellman-Ford Algorithm

Input: Directed Graph $G = (V,E)$; lengths of all edges as c_{uv}; node s as origin
Initialization
$d(s) \leftarrow 0$
for all $v \in V - \{s\}$ **do**
  $d(v) \leftarrow \infty$
end for
for all $e \in E$ **do**
  if $d(v) < d(u) + c_{uv}$ **then**
    $d(v) \leftarrow d(u) + c_{uv}$
  end if
end for

An example run of the algorithm with initialization, first iteration, and second iteration are shown in Fig 3.1, 3.1, and 3.1 respectively :

3.2.2 Desrocher Algorithm for the *SPPRC*

We briefly explain the algorithm presented by Desrchoer [97]. The algorithm works by labeling paths from the source to node v_j, each label is associated with a path and consists of the consumption of the resources and the cost of the path. Dominance rules are applied to avoid generating numerous labels. We need some notation to describe these rules.

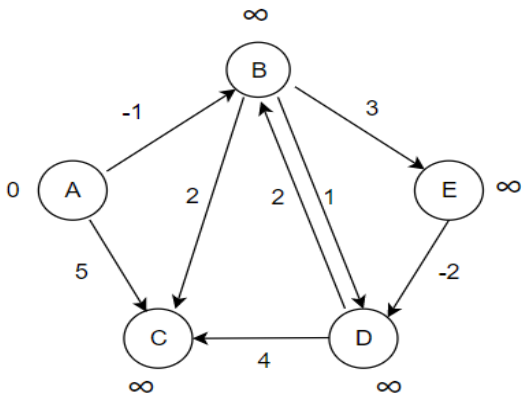


Figure 3.1: Initialization- Bellman-Ford

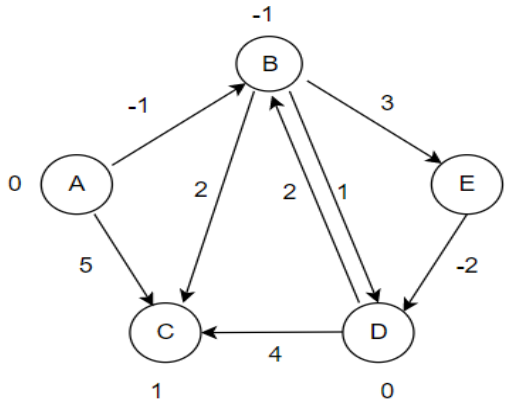


Figure 3.2: First iteration- Bellman-Ford

Associated with each path X_{pi} from the origin node p to a node v_i, is a label that is denoted by (R_i, C_i), where

$$R_i = (T_i^1, T_i^2, \ldots, T_i^l)$$

T_i^l corresponds to the quantity of resource l used by the path from origin to node i.

The cost is determined by the resource consumption,

$$C_i = C(T_i^1, T_i^2, \ldots, T_i^l)$$

The dominance rules are as follows:

Suppose X_{pi} and X'_{pi} are two distinct paths from origin p to node v_i with associated labels

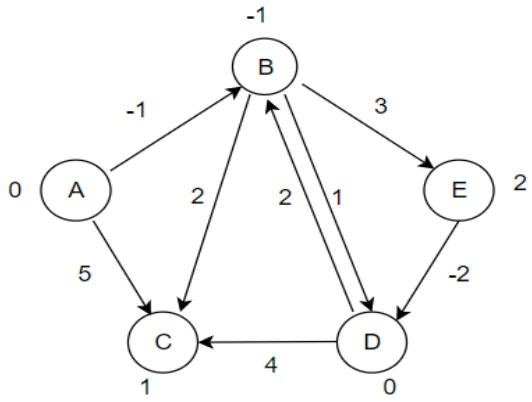


Figure 3.3: Second iteration- Bellman-Ford

(R_i, C_i) and $(R_i^{'}, C_i^{'})$ respectively, X_{pi} dominates $X_{pi}^{'}$ if and only if

$$C_i \leq C_i^{'}, \quad T_i^l \leq T_i^{'l} \quad \forall l = 1, \ldots, L$$

The optimal solution of the shortest path problem is determined by the min cost path in all non-dominated labels.

Fig 3.4 illustrates the first iteration of the algorithm applied on four nodes with one resource constraint. The resource consumption and the cost of each arc are indicated in brackets on edges and labels are presented next to each vertex. The label of the origin (node p) is $[0,0]$, then we start extending paths from node p to all possible nodes. So, the algorithm processes each node as well as checks the dominance relation at each node as follows:

- Processing of node p: There are edges (p, v_2) and (p, v_3) directed from the node p which we can use to extend the label $[0,0]$ of node p. The information on edge (p, v_2) indicates that the consumption of the resource and cost are 3, 3 respectively, so the label $[0,0]$ can use 3 units of the resource consumption and add 3 units of costs. The label at node v_2 is $[3,3]$. In the same way, the label $[0,0]$ of the node p is extended to the node v_3 and the label is updated to be $[3,3]$ at the node v_3.

- Processing of node v_2: There is a label $[3,3]$ at node v_2 and there are two edges

directed from v_2, (v_2, v_3), and (v_2, d). The label $[3,3]$ is extended to $[4,1]$ at node v_3 given the resource consumption and the cost of the edge (v_2, v_3). The label is then $[4,4]$ at node d.

- Processing of node v_3: Node v_3 can result in an extension to nodes v_2 and d. There are two labels at node v_3 that can extend and must be checked to see if we can keep them at the new node based on the dominance relation. The extension of the label $[3,3]$ to node v_2 is $[4,4]$ which is not kept because of the dominance relation that there is another label $[3,3]$ which is less in the resource consumption and the cost than the label $[4,4]$ and label $[4,1]$ is extended to be $[5,2]$ at node v_2. So, the two labels at node v_3 are extended to be $[4,4]$ and $[5,2]$ at node d.

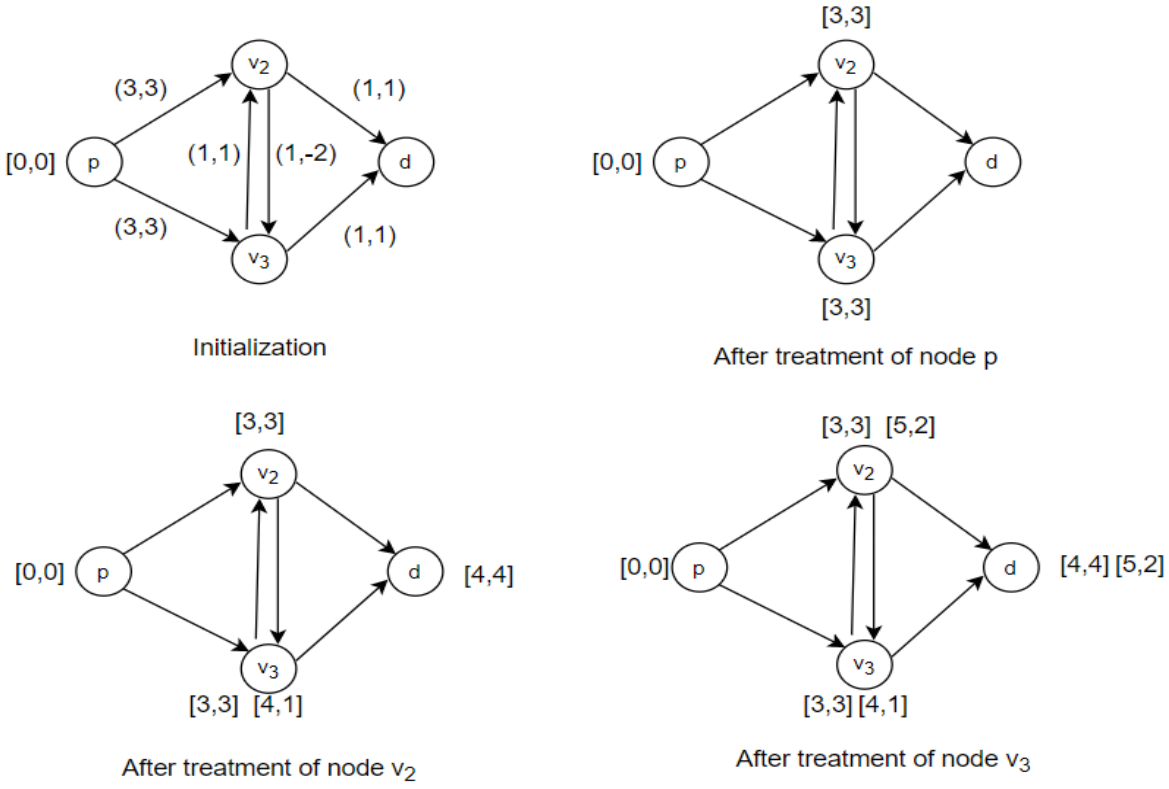


Figure 3.4: First iteration of Desrocher's algorithm

3.2.3 Label Correcting Algorithm for *ESPPRC*

Feillet et al. [33] modified Desrocher's algorithm to obtain an elementary shortest path with resource constraints in graphs which can have negative cost cycles (the overall sum of the cycle becomes negative).

A state is associated with each path X_{pi} from the origin node p to a node v_i as

$$R_i = (T_i^1, T_i^2, \ldots, T_i^L, s_{iv}, V_i^1, V_i^2, \ldots, V_i^n)$$

where,

- $T_i^l, l = 1, \ldots, L$ are the quantity of resources used on the path from p to node i.
- n binary indicator, one for each node $v_k \in V$ indicated as V_i^k
- s_{iv}, is the number of visited nodes on path X_{pi}.

$V_i^k = 1$ if node i is visited in the path X_{pi}, 0 otherwise. The label associated with this path is the same in the previous section (R_i, C_i).

Dominance Relation: If there are two distinct paths, X_{pi} and $X_{pi}^{'}$ from origin p to node v_i, X_{pi} dominates $X_{pi}^{'}$ if and only if

$$C_i \leq C_i^{'}, \quad s_{iv} \leq s_{iv}^{'}, \quad T_i^l \leq T_i^{'l} \quad \forall l = 1, \ldots, L, \quad V_i^k \leq V_i^{'k} \quad \forall k = 1, \ldots, n$$

An illustration of this new algorithm [33] is in Fig. 3.5 for the case of a single resource.
Each edge has a bracketed expression on it that contains the resource and the cost of each edge. Beside each node, labels can be seen, the first and last numbers in the square brackets are the resource consumption and cost of the path respectively, and the four intermediate numbers correspond to four nodes, indicating if the node is visited or not. The element s_{iv} of the label is not shown in Fig. 3.5 for simplicity.
The label of p is $[0;1,0,0,0;0]$ which means the consumption of the resource for the path

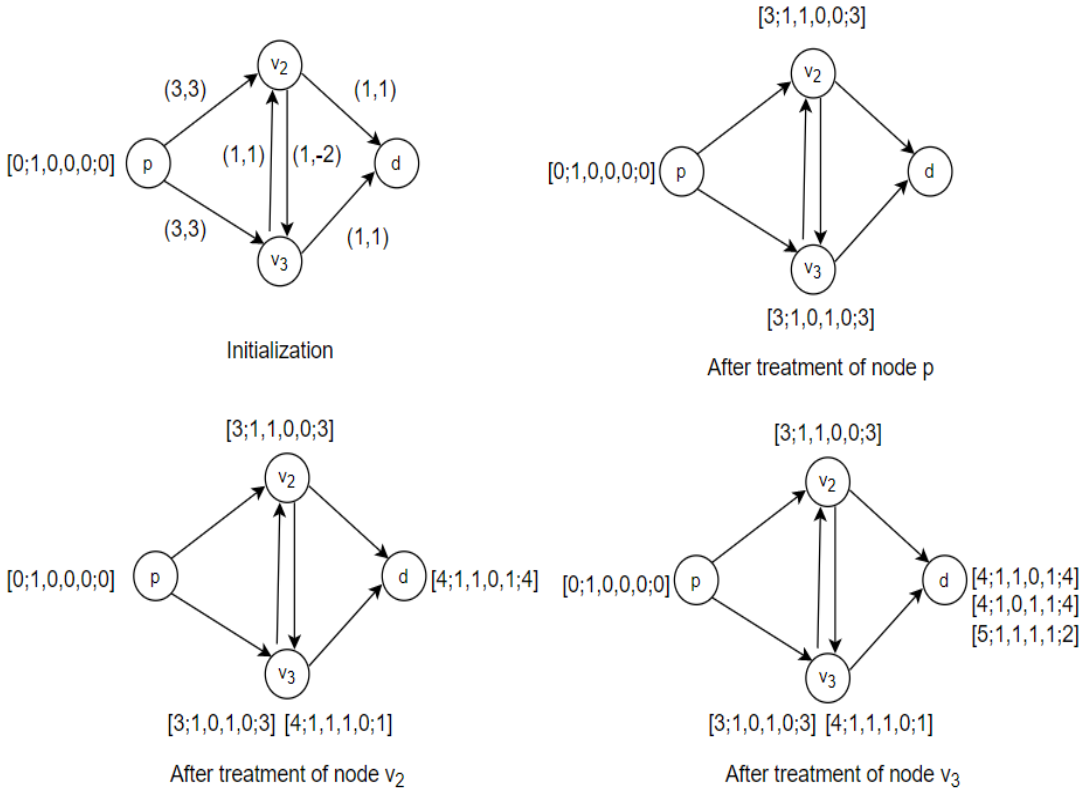


Figure 3.5: First iteration of the new definition of labels and dominance relation

is zero, the second digit which is 1 illustrates that the node p is visited and the third to fifth digits are for the other three nodes that are not visited in this path yet, and the last zero digit indicates the cost of the path is 0. This is the initialization step of the algorithm.

Continuing with the iterations, the first node which is p must be processed, so the label on p will be extended to nodes v_2 and v_3. For the extension of the path from node p to v_2, the label is updated to be $[3;1,1,0,0;3]$ which means the consumption of the resource is 3, nodes p and v_2 are visited and nodes v_3 and d are not visited in this path and the last digit shows the cost of the path. Similarly, the label on p is extended to v_3 and the new label on v_3 is $[3;1,0,1,0;3]$. We continue this process for all nodes to determine all possible paths from p to d.

Then, we need to process node v_2. There is one label at node v_3, $[3;1,1,0,0;3]$ which can be extended to nodes v_3 and d in terms of edges (v_2,v_3) and (v_2,d). The new label at node

v_3 and d are $[4;1,1,1,0;1]$ and $[4;1,1,0,1;4]$ respectively. So, there are two labels at node v_3 which both can be kept since none of them dominates another one.

The last step is to process node v_3 and extend its labels to nodes v_2 and d. None of the labels at node v_3 can be extended to v_2 because of the dominance relation on label $[3;1,0,1,0;3]$ which is updated to $[4;1,1,1,0;4]$ and all elements of this label are greater than equal label $[3;1,1,0,0;3]$ that already exists at node v_2 and the corresponding value of node v_2 is 1 on label $[4;1,1,1,0;1]$ that can not be extended to v_2, as elementary paths are created. The extension of two labels of node v_3 to node d is possible, so the new labels at node d are $[4;1,0,1,1;4]$ and $[5;1,1,1,1;2]$.

The definition of the labels is improved using the concept of unreachable node next.

Unreachable Node: For each path X_{pi} from the origin node p to a node $v_i \in V$, a node v_k is said to be unreachable if it is included in X_{pi} or if there exists a resource $l \in 1,...,L$ satisfying $T_i^l + d_{ik}^l > b_k^l$ (which means that the current value of consumption of l prevents the path from reaching node v_k) [33], where, b_k^l is the maximum value of the resource and d_{ik}^l is the total consumption of the resource in node i from node p in path x_{pi}.

The definition of the label is modified as:

$V_i^k = 1$ if node v_k is unreachable if it is already on the path or it can not be visited because of the violation of the resource constraints, and s_{iv} is the number of unreachable nodes, so $s_{iv} = \sum_{k=1}^{n} V_i^k$.

The definition of the dominance relation will stay the same and we only keep non-dominated paths. This labeling scheme is more efficient for dynamic programming. We use this labelling scheme in the next chapter in section 4.3.

Description of the Algorithm:

The algorithm finds all non-dominated paths from node p to all other nodes.

We need the following notation to describe the algorithm:

- A_i: List of labels on node v_i
- $Succ(v_i)$: Set of successors of node v_i.
- E: List of nodes waiting to be processed.
- $Extend(L_i, v_j)$: Function that returns the label resulting from the extension of label $L_i \in A_i$ towards node v_j when the extension is possible, nothing otherwise.
- $Dominated(A_j)$: Procedure that removes dominated labels in the list of labels A_j.
- F_{ij}: List of labels extended from v_i to v_j

Algorithm 2 Label Correcting Algorithm for *ESPPRC*

```
Initialization
A_p ← {(0,...,0)}
for all v_i ∈ V − {p} do
   F_ij ← ∅
end for
E = {p}
while E! = ∅ do
   Choose v_i ∈ E
   for all v_j ∈ Succ(v_i) do
      F_ij ← ∅
      for all L_i ∈ A_i do
         if V_i^j = 0 then
            F_ij ← F_ij ∪ Extend(L_i, V_j)
         end if
      end for
      A_j ← Dominated(F_ij ∪ A_j)
      if A_j has changed then
         E ← E ∪ {v_j}
      end if
   end for
   E ← E − {v_i}
end while
```

For the initialization of the algorithm, label $(0,\ldots,0)$ is assigned to the list of labels of node p (A_p) and the list of labels is empty for other nodes and node p is added to a set E. The algorithm continues until set E is empty.

In each iteration, the last node (v_i) of the set E is chosen and we check for all labels of the list A_i to see if we can extend them to each successor (v_j) of node v_i. If node v_j is not already on the path L_i (label on v_i), the path L_i is extended to v_j. After the extension of all labels on node v_i to node v_j, the dominance relation is checked to remove dominated labels between these new labels plus previous ones on node v_j, and if the list of labels at node v_j changes, the node is added to the set E. This process should be done for all successors of node v_i, then node v_i is deleted from the set E.

This algorithm is used to solve the sub-problem of the column generation in the next

chapter in section 4.3.

Chapter 4

Column Generation

Column generation was proposed to solve linear problems where there are many more variables than constraints. Even, if we were capable of generating all the columns, the simplex algorithm would still be unable to calculate all the decreased costs of zero (non-basic) variables due to memory requirements. The benefit of column generation is that by focusing only on variables that have the potential to improve the objective function, it decreases the number of variables kept in the memory. Column generation works well on problems like Vehicle routing, Airline Scheduling, Shift Scheduling, and Job shop Scheduling [2].

Some papers that use column generation for VRP are: LENT [99] uses a column generation method to solve the time dependent vehicle routing problem with soft time windows and stochastic travel times, the model allows the travel time distributions to change during a day because of traffic congestion; Kallehauge et al. [100] uses column generation for vehicle routing problem with time windows; Fukasawa et al. [75] is a combination of branch and cut and Lagrangean relaxation/column generation for Capacitated Vehicle Routing Problem; Rousseau et al. [101] is about solving small $VRPTW$s with Constraint Programming Based Column Generation; Liberatore et al. [102] used column generation for the vehicle routing problem with soft time windows. For a nice explanation of column generation, see Rousseau [2].

Column generation for VRP is described in section 4.1 starting with a simple example

we show how column generation solves it. Section 4.2 presents the column generation for $MTVRP$. The column generation is described in section 4.3 for $MTVRP-VW-TW$ which is an original contribution to this thesis.

4.1 Column Generation for VRP

To solve VRP using column generation, the problem is decomposed into two problems, the master problem and the sub-problem. Steps can be described as follow:

- The number of columns in the master problem can be large, so they are progressively introduced to the master problem. We start with the restricted master problem (RMP) which is an integer program. The linear relaxation programming of the RMP ($LRMP$) is solved at the beginning using the simplex method.

- The dual variables associated with the optimal solution of an $LRMP$ are used to define a pricing sub-problem. The solution to the sub-problem helps determine new routes. The column with the most negative reduced cost has the most potential to improve the solution of the master problem and this route is added to the $LRMP$. Columns are added until there are no columns with reduced cost. At this point, we have an optimal solution for the $LRMP$ and a lower bound for RMP.

- Pricing problem is modeled as an Elementary Shortest Path Problem with Resource Constraints ($ESPPRC$) and can be solved by dynamic programming given in section 3.2.

- Finally, a lower bound is embedded in a branch and bound algorithm to solve the master problem.

Fig.4.1 shows the flowchart of the column generation procedure.

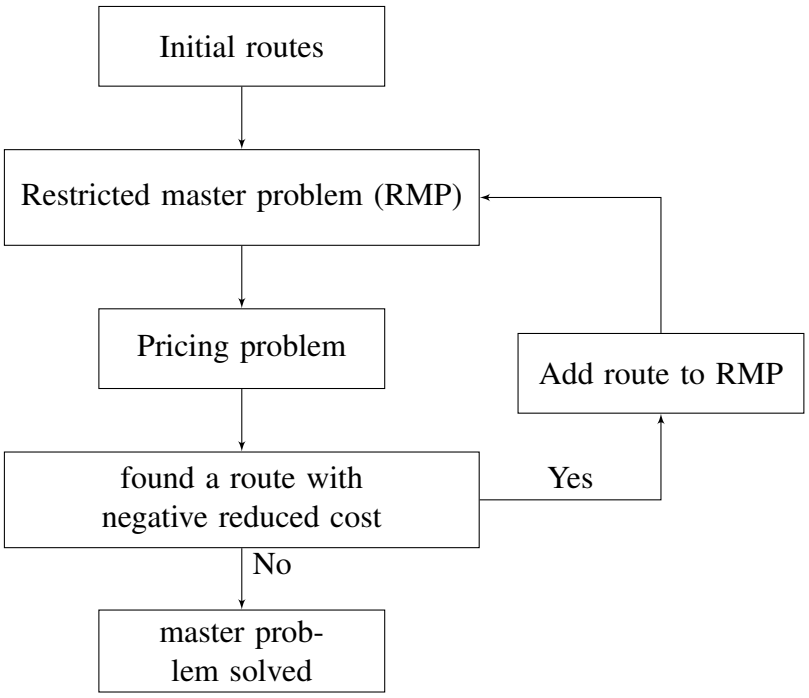


Figure 4.1: Overview of the column generation procedure

4.1.1 Master Problem and Sub-Problem for the *VRP*

The column generation approach is explained using a simple *VRP* with four customers in which each route can have a maximum of two clients. Fig. 4.2 shows a *VRP* with four clients that all must be visited and numbers on the edges are distances between customers and customers with depot as well. The distances on the missing edges are given by the triangle inequality.

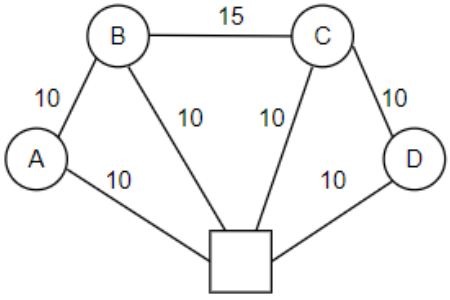


Figure 4.2: VRP Example [2]

To write the *RMP*, initially we start with a subset of routes from all the possible routes

visiting maximum of two clients. There are seven such possible routes and the total distance of each route can be calculated as:

$x_1 : Depot \rightarrow A \rightarrow Depot$ and $d_1(distance) = 10 + 10 = 20$

Table 4.1 shows the total distance of each route.

Table 4.1: All tours of VRP

Number of route	route	Visited customers	Distance (d_r)
1	x_1	$Depot - A - Depot$	20
2	x_2	$Depot - B - Depot$	20
3	x_3	$Depot - C - Depot$	20
4	x_4	$Depot - D - Depot$	20
5	x_5	$Depot - A - B - Depot$	30
6	x_6	$Depot - B - C - Depot$	35
7	x_7	$Depot - C - D - Depot$	30

The *VRP* model with these routes can be described as:

$$\min \quad 20x_1 + 20x_2 + 20x_3 + 20x_4 + 30x_5 + 35x_6 + 30x_7$$

A: $x_1 + x_5 = 1$ client A is in route x_1 and x_5

B: $x_2 + x_5 + x_7 = 1$

C: $x_3 + x_6 + x_7 = 1$

D: $x_4 + x_6 = 1$

$x_i \in \{0,1\} \quad i = 1,2,3,4,5,6,7$

where,

$$x_i = \begin{cases} 1 & \text{if route } i \text{ is in the solution} \\ 0 & \text{if route } i \text{ is not in the solution.} \end{cases}$$

To do column generation, we start with a subset of these routes, one route for each customer. so, the master problem is:

$$\begin{aligned}
&\min \quad 20x_1 + 20x_2 + 20x_3 + 20x_4 \\
&\text{A:} \quad x_1 = 1 \\
&\text{B:} \quad x_2 = 1 \\
&\text{C:} \quad x_3 = 1 \\
&\text{D:} \quad x_4 = 1 \\
&x_i \in \{0,1\} \quad i = 1,2,3,4
\end{aligned}$$

π_A, π_B, π_C, and π_D are dual variables corresponding to each constraint with value 20 each. Given the dual values, therefore we can calculate all reduced costs of routes to see if there is any route to improve the current solution or not.

If d_r is the cost of route r and a_{ir} is one if customer i is in route r, zero otherwise. The reduced cost of the routes is:

$$d_r - \sum_{i \in \{A,B,C,D\}} a_{ir}\pi_i$$

The reduced costs of the remaining routes are in Table. 4.2

Table 4.2: Reduced costs- First iteration

Number of route	route	Visited customers	Reduced cost
5	x_5	$Depot - A - B - Depot$	$30 - (20 + 20) = -10$
6	x_6	$Depot - B - C - Depot$	-5
7	x_7	$Depot - C - D - Depot$	-10

Therefore there are new routes with the potential of improving the master solution given the negative reduced costs. Route x_5 with the reduced cost of -10 is added to the master problem. The new columns in the basis are x_3, x_4, and x_5, so the total distance of routes is decreased to 70. With these new routes, the dual values corresponding to each constraint are 10, 20, 20, and 20 respectively. So, the reduced costs of the remaining routes are:

There are two routes with a negative reduced cost that can improve the solution. The

Table 4.3: Reduced costs- Second iteration

Number of route	route	Visited customers	Reduced cost
6	x_6	$Depot-B-C-Depot$	$35-(20+20)=-5$
7	x_7	$Depot-C-D-Depot$	$30-(20+20)=-10$

route x_7 is selected to be added to the master problem. So, the new basis of the master problem is routes x_5 and x_7 and the objective function decreases to 60. The new dual values are 10, 20, 10, and 20 respectively. The reduced cost of the route x_6 is $35-(20+10)=5$. So, there is no new route with the negative reduced cost to improve the master objective function.

The solution is shown in Fig. 4.3

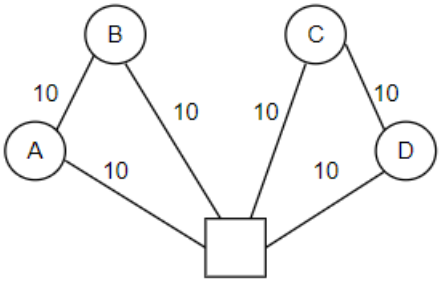


Figure 4.3: VRP optimal-Solution

Mathematical model of the Master Problem

After this example that shows how the column generation works, we describe the formulation of the master problem and *RMP* for basic *VRP*. In chapter one, capacitated vehicle routing problem was presented, and based on the same problem, the master problem will be described.

We assume that R is the set of all possible routes, starting from the depot and ending at the depot. Each route visits customers exactly once. A solution of *VRP* is a subset of these feasible routes. We select a number of these routes so that the total distance of the routes is minimized. To introduce the master problem which is a set covering problem, the following

notation and variables are used:

- Every column corresponds to a feasible route
- R: Set of all feasible routes
- C_r: The total distance of route $r \in R$
- $|V|$: The number of the vehicles

We use the following coefficients and decision variables.

$$y_r = \begin{cases} 1, & \text{if route } r \text{ is in the solution} \\ 0, & \text{otherwise} \end{cases} \tag{4.1}$$

$$a_{ir} = \begin{cases} 1, & \text{if customer } i \text{ is in route } r \\ 0, & \text{otherwise} \end{cases} \tag{4.2}$$

The master problem is:

$$\min \sum_{r \in R} C_r y_r$$

s.t.

$$\sum_{r \in R} a_{ir} y_r \geq 1 \quad \forall i \in C$$

$$y_r \in \{0,1\} \qquad \forall r \in R$$

The LP-dual of the master problem is:

$$\max \sum_{i \in C} \pi_i$$

s.t.

$$\sum_{i \in C} a_{ir} \pi_i \leq C_r \quad \forall r \in R$$

$$\pi_i \geq 0 \qquad \forall i \in C$$

Let π be an optimal solution to the LP-dual. The reduced cost of each route is:

$$\overline{C}_r = C_r - \sum_{i \in C} a_{ir} \pi_i \tag{4.3}$$

As long as there is a route with negative reduced cost, the dual problem does not have an optimal solution, therefore, any route with negative reduced cost can be added to the restricted master problem, usually, the most negative one is added to improve the primal solution.

Sub-Problem

To determine the column to be added, we need a sub-problem to find the route with the most negative reduced cost. The cost of each route which is the sum of the traveling time of all edges on the route can be calculated as:

$$C_r = \sum_{i \in N} \sum_{j \in N} c_{ij} x_{ij}$$

And the reduced cost ($\overline{C}_r$) is:

$$\overline{C}_r = C_r - \sum_{i \in C} a_{ir} \pi_i$$

By using this information we can write the pricing sub-problem which gives the column with the minimum reduced cost. So sub-problem can be expressed as:

$$\min \sum_{i \in N} \sum_{j \in N} c_{ij} x_{ij} - \sum_{i \in C} a_{ir} \pi_i \tag{4.4}$$

s.t.

$$\sum_{i \in C} \sum_{j \in N} d_i x_{ij} \leq Q \tag{4.5}$$

$$\sum_{j \in N} x_{0j} = 1 \tag{4.6}$$

$$\sum_{i \in N} x_{ih} - \sum_{j \in N} x_{hj} = 0 \quad h \in C \tag{4.7}$$

$$\sum_{i \in N} x_{i,n+1} = 1 \tag{4.8}$$

$$x_{ij} \in \{0,1\} \qquad i,j \in N \tag{4.9}$$

The sub-problem which is a shortest path problem can be solved using dynamic programming [33] to find all feasible routes with a negative reduced cost.

Initialization

To start the process we generate one route per customer and these routes will be the initial input to the master problem.

4.2 Column generation for *MTVRPTW*

The mathematical model was described in Chapter 2. Now, the column generation approach is applied to solve the problem.

4.2.1 Master Problem and sub-Problem for the multi-trip vehicle routing problem

Let us summarize the mathematical model for *MTVRPTW* next.

$$\min \sum_{k\in V}\sum_{r\in R}\sum_{i\in N}\sum_{j\in N} d_{ijr}^{k} x_{ijr}^{k} \tag{4.10}$$

s.t.

$$\sum_{k\in V}\sum_{r\in R}\sum_{j\in N} x_{ijr}^{k} = 1 \quad \forall i \in C \tag{4.11}$$

$$\sum_{i\in C} d_i \sum_{j\in N} x_{ijr}^{k} \leq q^{k} \quad \forall k \in V,\ r \in R \tag{4.12}$$

$$\sum_{j\in N} x_{0jr}^{k} = 1 \quad \forall k \in V,\ r \in R \tag{4.13}$$

$$\sum_{i\in N} x_{ihr}^{k} - \sum_{j\in N} x_{hjr}^{k} = 0 \quad \forall h \in C,\ k \in V,\ r \in R \tag{4.14}$$

$$\sum_{i\in N} x_{i,n+1,r}^{k} = 1 \quad \forall k \in V,\ r \in R \tag{4.15}$$

$$s_{ir}^{k} + s_i + t_{ij} - M(1 - x_{ijr}^{k}) \leq s_{jr}^{k} \quad \forall i \in N - \{n+1\},\ j \in N - \{0\},\ \forall k \in V,\ r \in R \tag{4.16}$$

$$a_i \sum_{j\in N-\{0\}} x_{ijr}^{k} \leq s_{ir}^{k} \leq b_i \sum_{j\in N-\{0\}} x_{ijr}^{k} \quad \forall i \in C,\ k \in V,\ r \in R \tag{4.17}$$

$$s_{0r}^{k} \geq s_{n+1,r-1}^{k} \quad \forall r \in R,\ k \in V \tag{4.18}$$

$$x_{ijr}^{k} \in \{0,1\} \quad \forall i, j \in N,\ k \in V \tag{4.19}$$

$$s_{ir}^{k} \geq 0 \quad \forall i \in C,\ k \in V,\ r \in R \tag{4.20}$$

$$q^{k} \geq 0 \quad \forall k \in V \tag{4.21}$$

The objective function (4.10) is to minimize the total distance of all tours. Constraints state that each customer must be visited once (4.11), the sum of demands of customers visited during a route must not exceed the capacity of the vehicle (4.12), all routes must start and end at the depot (4.13) and (4.16), flow conservation constraint (4.14), the relationship between the vehicle departure time from a customer and its immediate successor (4.16), the time window constraints for each customer (4.17), and finally a proper route sequencing for the workday of a vehicle (4.18).

For column generation, a few changes are needed when the vehicle can perform more than one route in a day. In our case, a tour represents all the routes that a vehicle can make during a workday. So, a tour is a collection of routes.

4.2.2 Master problem for *MTVRP*

We use the following notation to describe the master problem for *MTVRP*.

- Tour: all routes a vehicle services during a day
- Every column corresponds to a tour (with multiple routes)
- Ω: The set of all feasible tours
- d_w: The total distance of tour $w \in \Omega$
- $|V|$: The number of the vehicles

$$a_{iw} = \begin{cases} 1, & \text{if customer } i \text{ is in tour } w \\ 0, & \text{otherwise} \end{cases} \tag{4.22}$$

The following decision variables are introduced:

$$y_w = \begin{cases} 1, & \text{if tour } w \text{ is in the solution} \\ 0, & \text{otherwise} \end{cases} \tag{4.23}$$

The master problem formulation which is set covering formulation is:

$$\min \sum_{w\in\Omega} d_w y_w \tag{4.24}$$

s.t.

$$\sum_{w\in\Omega} a_{iw} y_w \geq 1 \quad i \in C \tag{4.25}$$

$$\sum_{w\in\Omega} y_w \leq |V| \tag{4.26}$$

$$y_w \in \{0,1\} \quad w \in \Omega \tag{4.27}$$

$$\tag{4.28}$$

Here, each column corresponds to a tour and a solution is a subset of Ω. The LP relaxation of the master problem is solved by the simplex method and the dual variables associated with the constraints of the optimal solution are used to define a pricing subproblem that can give us the columns with a negative reduced cost as shown next.

4.2.3 Subproblem of the $MTVRP$

The pricing problem is an elementary shortest path problem with resource constraints and is constructed using the dual variables of the optimal solution to the master problem and the constraints that are not used in the master problem. Suppose that $\pi_i, i \in C$ are the dual variables associated with the covering constraints (4.25) and the dual variable μ is associated with constraint (4.26), then the objective is to find the column with the minimum reduced cost ($\overline{d}_w$):

$$d_w - \sum_{i\in d} a_{iw}\pi_i - \mu = \overline{d}_w - \mu$$

where $\overline{d}_w = d_w - \sum_{i\in d} a_{iw}\pi_i$.

4.3 Column generation for $MTVRP-VW-TW$

Multi-trip vehicle routing problem with a variable number of wagons and time windows was described in section 2.4 and the mathematical model was presented in section 2.5. In this section, we solve the problem using column generation. The master problem, pricing subproblem, and the techniques for solving the pricing subproblem will be explained. This contribution is new and extends the methodology presented in section 4.2 for $MTVRP$.

4.3.1 Master problem for $MTVRP-VW-TW$

We note the following:

- Tour: all routes a vehicle can have during a day;
- Every column corresponds to a tour;
- Ω: the set of all feasible tours;
- d_w: the total distance of tour $w \in \Omega$;
- W: the total number of wagons;
- $|V|$: the number of the vehicles;
- Numbers of the used wagons should be less than W;
- n_w: the number of wagons used for a vehicle that is used in tour w and $n_w \in \{1,2,3\}$;

$$a_{iw} = \begin{cases} 1, & \text{if customer } i \text{ is in tour } w \\ 0, & \text{otherwise} \end{cases} \tag{4.29}$$

We use the following decision variables:

$$y_w = \begin{cases} 1, & \text{if tour } w \text{ is chosen} \\ 0, & \text{otherwise} \end{cases} \tag{4.30}$$

The master problem is:

$$\min \sum_{w \in \Omega} d_w y_w \tag{4.31}$$

s.t.

$$\sum_{w \in \Omega} a_{iw} y_w \geq 1 \quad \forall i \in C \tag{4.32}$$

$$\sum_{w \in \Omega} y_w \leq |V| \tag{4.33}$$

$$\sum_{w \in \Omega} n_w y_w \leq |W| \tag{4.34}$$

$$y_w \in \{0,1\} \qquad \forall w \in \Omega \tag{4.35}$$

$$\tag{4.36}$$

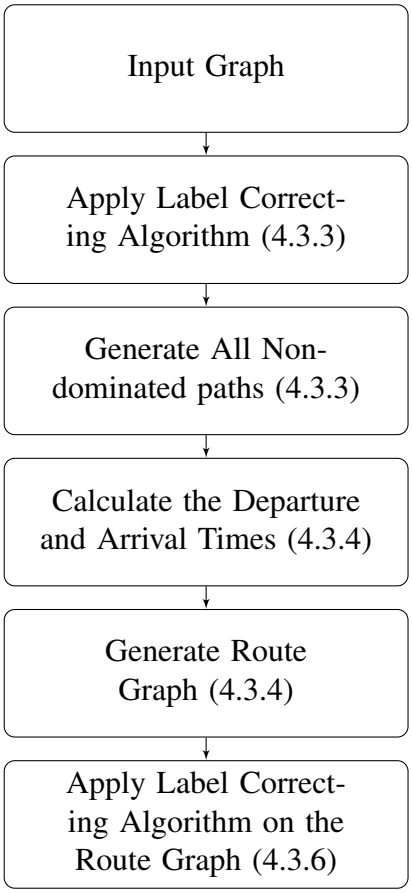
The objective function (4.31) is to minimize the total distances of all tours. Constraints (4.32) ensure that each customer is visited at least once. The constraint (4.33) states that the number of all tours must be less than the number of vehicles. Constraint (4.34) states that the number of wagons used in all the vehicles must be less than the total number of wagons.

The solution is a subset of Ω. As the number of columns is exponential in the number of customers, we solve the restricted master problem (RMP) with a limited number of columns for the initial solution and the columns are progressively added into *RMP*. The LP-relaxation of *RMP* (*LRMP*) is solved with an *LP* solver to obtain the dual variables associated with the optimal solution of the *RLMP*, these dual values are sent to the sub-problem to determine new tours with negative reduced cost and these new tours are added to the master problem, the process will continue until there are no more tours with negative reduced cost. This guarantees an optimal solution to the *RLMP*.

4.3.2 Method for constructing tours/columns

Once the LP relaxation of the restricted master problem is solved, the dual values are provided to determine the objective function of the sub-problem which is an elementary shortest path problem with resource constraints from the artificial start node to the artificial end node in the route graph. The route graph is constructed using a label correcting algorithm which is explained in the following sections. The label correcting algorithm is applied to the route graph to generate all non-dominated tours. Therefore the solution of the sub-problem is determined, once we have all non-dominated tours in the route graph. If there is a new tour with the negative reduced cost from three sub-problems with different capacities which are made from three route graphs, it is added to the master problem. The algorithm iterates until no new tour with the negative reduced cost can be generated.

The following flowchart shows the process of generating all non-dominated tours for three different capacities separately.



4.3.3 Generating all non-dominated paths

To solve the subproblem, all non-dominated routes must be generated, and the label correcting algorithm is used to create all these routes.

We need to generate elementary paths, so we keep track of previously visited nodes. A path p from an origin node o to a node j is labeled with $R_p = (C_p, t_p^1, ..., t_p^l, s_{pv}, V_p^1, ..., V_p^n)$, where $L = \{1, ..., l\}$ is the set of resources and we have two resources t_p^1 and t_p^2. Time consumption, t_p^1 is the time used in the path till customer v_j which is calculated as $t_p^1 = t_p^1 + t_{ij} + s_j$, where v_i and v_j are two adjacent customers on the route and s_j represents the service time for the customer v_j. Load consumption, t_p^2 is the capacity used for the path and is given by $t_p^2 = t_p^2 + d_j$ where d_j is the quantity that must be delivered to the customer v_j. A time interval $[a_i, b_i]$ representing the time window is associated with each customer v_i and a load interval $[0, Q]$, where Q is the capacity of the vehicle. C_p is the length of the path and is made negative by replacing the distance t_{ij} of each arc with $t_{ij} - a$, such that $a > max_{(i,j)\in A} t_{ij}$. We make the distances negative so that vehicles leave the depot otherwise, it would be optimal to stay at the depot, s_{pv} is the number of unreachable nodes and $V_p^i = 1$ if node i is unreachable, 0 otherwise. The following dominance relation is used to determine non-dominated routes:

Dominance relation: If there are two paths from origin o to node j, p and $p\prime$, with labels R_p and $R_{p\prime}$, respectively, then path p dominates $p\prime$ if and only if $C_p \leq C_{p\prime}, s_{pv} \leq s_{p\prime v}, t_p^k \leq t_{p\prime}^k, \forall k = 1, ..., l, V_p^i \leq V_{p\prime}^i, \forall i = 1, ..., n$ [1].

That is, path p dominates $p\prime$ if (a) it is no longer, (b) it does not consume more resources for every resource considered and (c) every unreachable node is also unreachable for path $p\prime$ [1].

Using this relation will keep only the labels that are for non-dominated elementary paths.

To implement the label correcting algorithm for our problem, we need to create a label $(C_p, t_p^1, t_p^2, s_{pv}, V_p^1, ..., V_p^n)$ which represents a path p from the depot to the customer j. Using these labels all feasible non-dominated routes will be generated.

During the extension of the path, we need to see if the current time consumption (t_p^1) plus the distance d_{ij} is less than a_i, then t_p^1 is replaced by a_i and the extension of the path continues. Each time that we extend one node or the node is unreachable, s_{pv} increases by one. We also eliminate the partial routes when we are extending the paths. So, at the end of the algorithm, we will have all non-dominated routes.

4.3.4 Creation of the route graph

After creating all non-dominated routes, each of them can be looked at as a node in a new graph that we call the route graph. The route graph includes these routes as the nodes and two artificial nodes for the start and end of the vehicle workday. To create the route graph, the following rules must be applied to determine if there is an edge between nodes r and $r\prime$:

- Route r and $r\prime$ must not visit the same customer.
- The feasibility of servicing route $r\prime$ after route r is determined through departure time windows as explained below.

To satisfy the second condition, the latest departure and arrival times and the earliest departure and arrival times need to be calculated. To have an edge $(r, r\prime)$ the latest departure of route $r\prime$ must be greater than the latest arrival of route r. There are edges between the artificial start node and all routes and from all routes to the artificial end node.

There are two time windows for each route node r which are the earliest and latest departure times [$\underline{t}_0^r$,$\bar{t}_0^r$] and earliest and latest arrival times [$\underline{t}_{n+1}^r$,$\bar{t}_{n+1}^r$]. Routes must be started and completed in these intervals. These time windows are determined as shown below.

Latest departure and arrival times

If the route r is shown as a sequence $(0 = i_0, i_1, i_2, \ldots, i_{n_r}, i_{n_r+1} = n+1)$ where the first and last points are the depot as well as other customers (n_r ones) in the middle, first the

latest feasible time $\bar{t}^r_{i_j}$ of each customer must be calculated using a back-ward sweep of route r starting from i_{n_r+1} to i_0. Therefore:

$$\bar{t}_{i_{n_r+1}} \leftarrow b_{i_{n_r+1}}$$

$$\bar{t}^r_{i_j} \leftarrow \min\{\bar{t}^r_{i_{j+1}} - t_{i_j i_{j+1}} - s_{i_j}, b_{i_j}\}, \quad \forall j = i_{n_r}, \ldots, i_0$$

Finally, we will have $\bar{t}^r_{i_0}$ which is the latest departure of the route r and again in a similar way we obtain the latest arrival time of the route as well as the latest feasible schedules $(\bar{t}^r_{i_j})$ at each customer using a forward sweep, so each $\bar{t}^r_{i_j}$ can be calculated as:

$$\bar{t}^r_{i_j} \leftarrow \max\{\bar{t}^r_{i_{j-1}} + t_{i_{j-1} i_j} + s_{i_j}, a_{i_j}\}, \quad \forall j = i_1, \ldots, i_{n_r+1}$$

Earliest departure and arrival times

Suppose we calculated the latest departure time $(\bar{t}^r_0)$, the earliest departure time$(\bar{t}^r_{n+1})$ and the latest feasible schedules to begin service at each customer$(\bar{t}^r_{i_j})$ in the route r, two cases can happen:

Case 1: there is no waiting time (vehicle doesn't arrive before time windows) in the latest feasible time of customers, then we can shift the latest times by the minimum of $(\bar{t}^r_{i_j})$ and a_{i_j}, so we can calculate it for each route as:

$$\delta^r = min_{j=0,\ldots,n_r+1}(\bar{t}^r_{i_j} - a_{i_j})$$

By deducting these units of the latest departure and arrival times, the earliest departure and arrival can be obtained. So, we have:

$$\underline{t}^r_0 = \bar{t}^r_0 - \delta^r$$

and,

$$\underline{t}^r_{n+1} = \bar{t}^r_{n+1} - \delta^r$$

Having all the latest departure and arrival times as well as the earliest departure and arrival times, it is possible to write the time windows for all routes as: $[\underline{t}^r_0, \bar{t}^r_0]$ and $[\underline{t}^r_{n+1}, \bar{t}^r_{n+1}]$.

Case 2: If there are some waiting times in the latest feasible time for customers. Then, we can not leave the depot earlier, when the latest arrival times are before the time windows.

So, the earliest departure and arrival times will be the same as the latest departure and arrival times respectively.

$$\underline{t}^r_0 = \bar{t}^r_0$$

and,

$$\underline{t}^r_{n+1} = \bar{t}^r_{n+1}$$

In case 2, the time windows for departure and arrival will be a single point. So, the route r' can be served after route r if $\underline{t}^r_{n+1} + \delta^{r'} \leq \bar{t}^{r'}_0$

Given all this information, now to create the route graph, we must note that there must not be any common customer between routes r and r' and the latest arrival time of the route r must be less than the latest departure time of the next route r'. If both the conditions are met, then there is an edge from r to r'. Below we show the route graph for a small example problem from [1].

Example

A simple instance of five customers and two nodes 0 and 6 associated with the depot is in Table 4.4. Indices 1 to 5 are the five customers. There are the coordinates and time windows associated with each customer, listed in the same row. Travel time between any two nodes is the same as the distances and servicing time of all customers is $s = 10$.

The label correcting algorithm is applied on this instance and all the non-dominated

Table 4.4: Customers with time windows [1]

Node	x	y	Time window
0	40	50	$[0,\infty]$
1	25	85	$[591,874]$
2	22	75	$[73,350]$
3	22	85	$[473,588]$
4	20	80	$[418,913]$
5	20	85	$[40,390]$
6	40	50	$[0,\infty]$

routes are shown in Table 4.6. The calculation of the latest and earliest departure and arrival times for the first route $(0,3,1,6)$ is as follows:

$r = (0,3,1,6)$

$i_0 = 0, i_1 = 3, i_2 = 1, i_3 = 6$

$t_{03} = 39.55, t_{31} = 3, t_{16} = 38.07$

$s_3 = 10, s_1 = 10$

Next, we calculate the latest departure and arrival times. Table (4.5) shows the calculation of the latest departure.

Table 4.5: Latest departure calculation

$\bar{t}_6^1$	$\bar{t}_1^1$	$\bar{t}_3^1$	$\bar{t}_0^1$
∞	$min(\bar{t}_6^1 - t_{16} - s_1, b_1)$ =$min(\infty - 38.07 - 10,874)$ =874	$min(\bar{t}_1^1 - t_{13} - s_3, b_3) =$ $min(874 - 3 - 10,588) = 588$	$min(\bar{t}_3^1 - t_{30} - s_1, b_0) =$ $min(588 - 39.55 - 0,\infty) = 548.65$

The backward sweep gave us the latest departure time $\bar{t}_0^1 = 548.65$, a forward sweep gives the latest arrival time:

$$\bar{t}_0^1 = 548.65$$

$$\bar{t}_3^1 = max\{548.65 + 39.35, 473\} = 588$$

$$\bar{t}_1^1 = max\{588 + 3.00 + 10,591\} = 601$$

Table 4.6: Time windows and cost of each route [1]

Route	Departure	Arrival	Cost
0, 3, 1, 6	[538.65, 548.65]	[639.07, 649.07]	-82.82
0, 4, 3, 6	[421.57, 522.35]	[536.57, 637.35]	-82.46
0, 4, 1, 6	[537.88, 639.07]	[820.88, 922.07]	-82.05
0, 2, 5, 6	[42.20, 143.50]	[319.20, 420.50]	-81.94
0, 2, 6	[42.20, 113.80]	[319.20, 390.80]	-20.02
0, 4, 6	[381.95, 464.05]	[876.95, 959.05]	-9.52
0, 1, 6	[552.93, 639.07]	[35.93, 922.07]	-5.48
0, 3, 6	[433.65, 522.35]	[548.65, 637.35]	-2.92
0, 5, 6	[0.00, 90.62]	[349.69, 440.31]	-1.00

$$\bar{t}_6^1 = max\{601 + 38.07 + 10, 0\} = 649.07$$

The calculations for the other tours are done similarly, so we have the latest departure and arrival times of all tours which are in Table 4.6, the second column.

Now, we need to calculate the earliest departure and arrival times. Observing the latest feasible schedule, we see that there is no waiting time in this schedule, so case 1 arises and the minimum shift must be calculated. For the first tour r_1, we have:

$$\delta^1 = min\{548.65 - 0, 588 - 473, 601 - 591, 649.07 - 0\} = 10$$

We know, $\underline{t}_0^r = \bar{t}_0^r - \delta^r$, and we have $\underline{t}_0^1 = \bar{t}_0^1 - \delta^1 = 548.65 - 10 = 538.65$. In the same way, $\underline{t}_6^1 = \bar{t}_6^1 - \delta^1 = 649.07 - 10 = 639.07$. So, the time windows of tour one for departure and arrival at the depot are:

$$[\underline{t}_0^1, \bar{t}_0^1] = [538.65, 548.65]$$

and,

$$[\underline{t}_6^1, \bar{t}_6^1] = [639.07, 649.07]$$

The same calculation is done for all tours to get the earliest departure and arrival time. The third column of Table 4.6 displays these numbers.

Using this information, the route graph is constructed. We also check whether there are no common customers between the two routes and if two consequent routes are feasible based on the departure time windows, an edge is then drawn between these two routes. Fig.4.4 shows the route graph for this example. Each node indicates a route and numbers in the circle are customers visited in the route, numbers on each edge are the cost of the route, a tour must be found for a vehicle by visiting these routes, then the tour is a multi trip for the vehicle. The elementary shortest path obtained is shown with a dotted line.

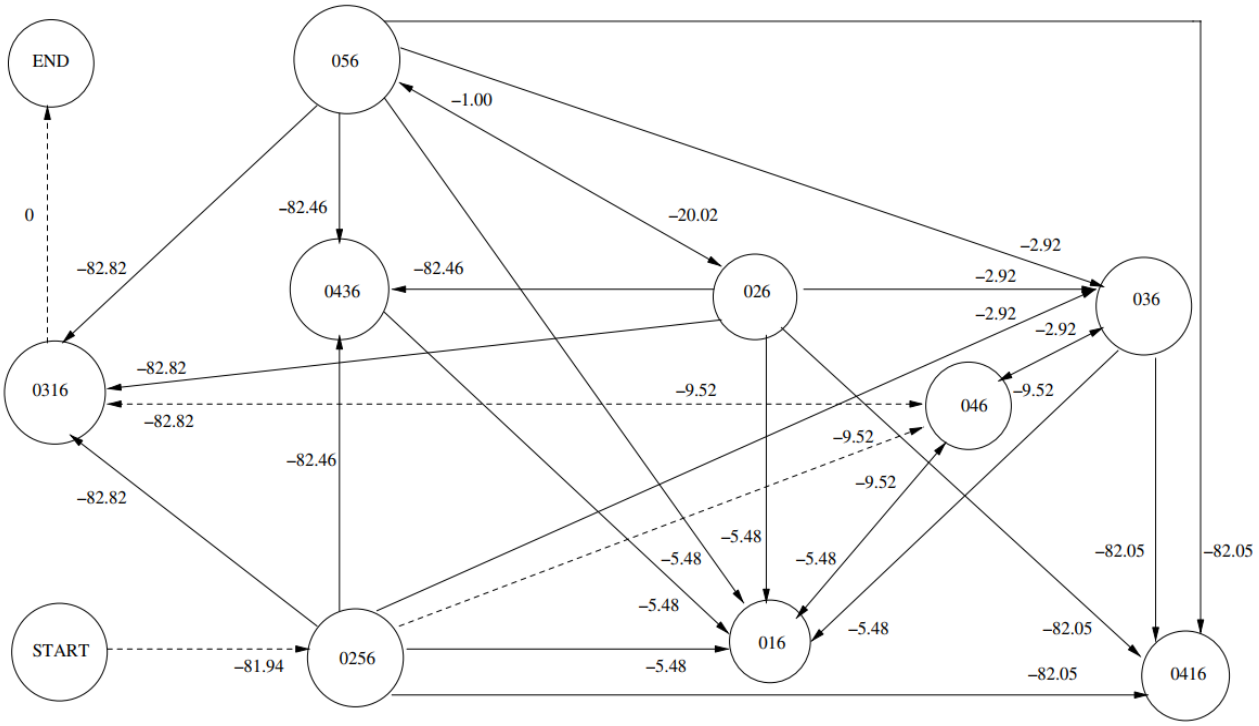


Figure 4.4: Route graph

4.3.5 Sub-problem for $MTVRP-VW-TW$

The subproblem is defined on the route graph $G^T = (V^T, A^T)$ where V^T is the set of all non-dominated routes generated by the label correcting algorithm [33] plus two artificial nodes for the start and end of the tour. A^T is the set of edges in the route graph with the time windows on each route, $[\underline{t}^r_0, \overline{t}^r_0]$ and $[\underline{t}^r_{n+1}, \overline{t}^r_{n+1}]$.

To formulate the subproblem on the route graph, dual variables associated with the master problem constraints are needed. Let π_i be dual the variables associated with constraints (4.32) in the master problem and μ_0 and μ_1 are the dual variables associated with (4.33) and (4.34) constraints respectively.

Let $c_{rs} = d_s$, where d_s is the total distance of route s, the reduced cost of arc (r,s) is:

$$\overline{c}_{rs} = c_{rs} - \sum_{i \in V_s} \pi_i$$

Using the binary variable X_{rs} which is one if the route (r,s) is used and zero otherwise and the continuous variable T_r which is the departure time of the route r, we formulate the subproblem as follows:

$$\min \sum_{(r,s)\in A^T} \overline{c}_{rs} X_{rs} - \mu_0 - \mu_1 \tag{4.37}$$

s.t.

$$\sum_{(r,h)\in A^T} X_{rh} - \sum_{(h,s)\in A^T} X_{hs} = 0 \quad \forall h \in V^T \tag{4.38}$$

$$\sum_{r\in A^T} X_{0r} = 1 \tag{4.39}$$

$$\sum_{r\in A^T} X_{r,n+1} = 1 \tag{4.40}$$

$$T_r + (\bar{t}^r_{n+1} - \bar{t}^r_0) - M(1 - X_{rs}) \leq T_s \quad \forall (r,s) \in A^T \tag{4.41}$$

$$\underline{t}^r_0 \leq T_r \leq \bar{t}^r_0 \quad \forall r \in V^T \tag{4.42}$$

$$X_{rs} \in \{0,1\} \quad \forall (r,s) \in A^T \tag{4.43}$$

$$T_r \geq 0 \quad \forall (r,s) \in A^T \tag{4.44}$$

$$\tag{4.45}$$

The objective function (4.37) is the reduced cost of the tour. Constraint (4.38) indicates that the vehicle must leave a route and go to the next one. Constraints (4.39) and (4.40) ensure that the tour starts and ends at the depot. The inequalities (4.41) establish the relationship between the vehicle departure time from a route and its immediate successor. Constraints (4.42) assert that the time windows of routes are observed.

4.3.6 Solving the pricing subproblem

For three different capacities of the vehicles, we have three route graphs. Consequently, three subproblems will be solved. The label correcting algorithm can be applied to each of them again to find the new tour with the negative reduced cost to be added to the master problem.

Now, the label correcting algorithm can be implemented to determine all non-dominated tours of the route graph.

To implement the label correcting algorithm on the route graph, a label L_p which is associated with a path p from the start node in the route graph to a route node r needs to be created :

$$L_p = (C_p, T_p^r, s_{pv}, V_p^1, \ldots, V_p^{n^T})$$

C_p is the cost of the path p. When a route node is added to the path, the cost of the route node will be added to the cost of the path.

T_p^r is the departure time of route r.

s_{pv} is the number of unreachable route nodes.

$V_p^{r'}$ indicates that the route node r' is unreachable if the value is 1, 0 otherwise.

There is no capacity resource for the label corresponding to each path of the route graph. In fact, the only resource consumption is the time and the time constraints $\underline{t}_0^r \leq T_r \leq \bar{t}_0^r$ need to be checked to see if the path is extendable or not.

To extend a path, an arc (r,s) must be added keeping the feasibility of departing the depot to service route s before the latest departure time of route node s. If s is added, then the cost and time consumed are added too. A few points must be noted:

- Start of service for route r must be in $[\underline{t}_0^r, \bar{t}_0^r]$ and be completed in $[\underline{t}_{n+1}^r, \bar{t}_{n+1}^r]$
- Traveling time is $\bar{t}_{n+1}^r - \bar{t}_0^r$
- Service time of route $s(s_s)$ is the time that route s needs for preparation and any waiting time, so $s_s = \delta^s$
- The time consumed on the route s is the duration of route s plus the setup time of route s plus any waiting time before departure
- Possibility of servicing route s after route r is given by $T_r + (\bar{t}_{n+1}^r - \bar{t}_0^r) \leq T_s$

Using the extend and dominate function in the label correcting algorithm, all non-dominated tours will be generated. Papers [33], [1] and [42] are used to have an algorithm to generate all non-dominated tours. The algorithm first generate all non-dominated routes and then generate the rout graph and finally create all non-dominated tours.The algorithm is described next.

Description of the Algorithm:

The algorithm finds all non-dominated tours on the route graph from the origin node p (depot).

We need the following notation to describe the algorithm:

- $G = (N, A)$: The input graph.
- N: Set of customers and vertices 0 and $n+1$ as the depot.
- A: Set of all edges between vertices in N
- H_i: List of labels on node v_i
- $Succ(v_i)$: Set of successors of node v_i.
- E: List of nodes waiting to be processed.
- $Extend(L_i, v_j)$: Function that returns the label resulting from the extension of label $L_i \in H_i$ towards node v_j when the extension is possible, nothing otherwise.
- $Dominated(A_j)$: Procedure that removes dominated labels in the list of labels H_j.
- F_{ij}: List of labels extended from v_i to v_j
- $Routes$: To save all non-dominated routes.
- $(Ld-time)_k$: latest departure time of $R_k \in Routes$.
- $(La-time)_k$: latest arrival time of $R_k \in Routes$.

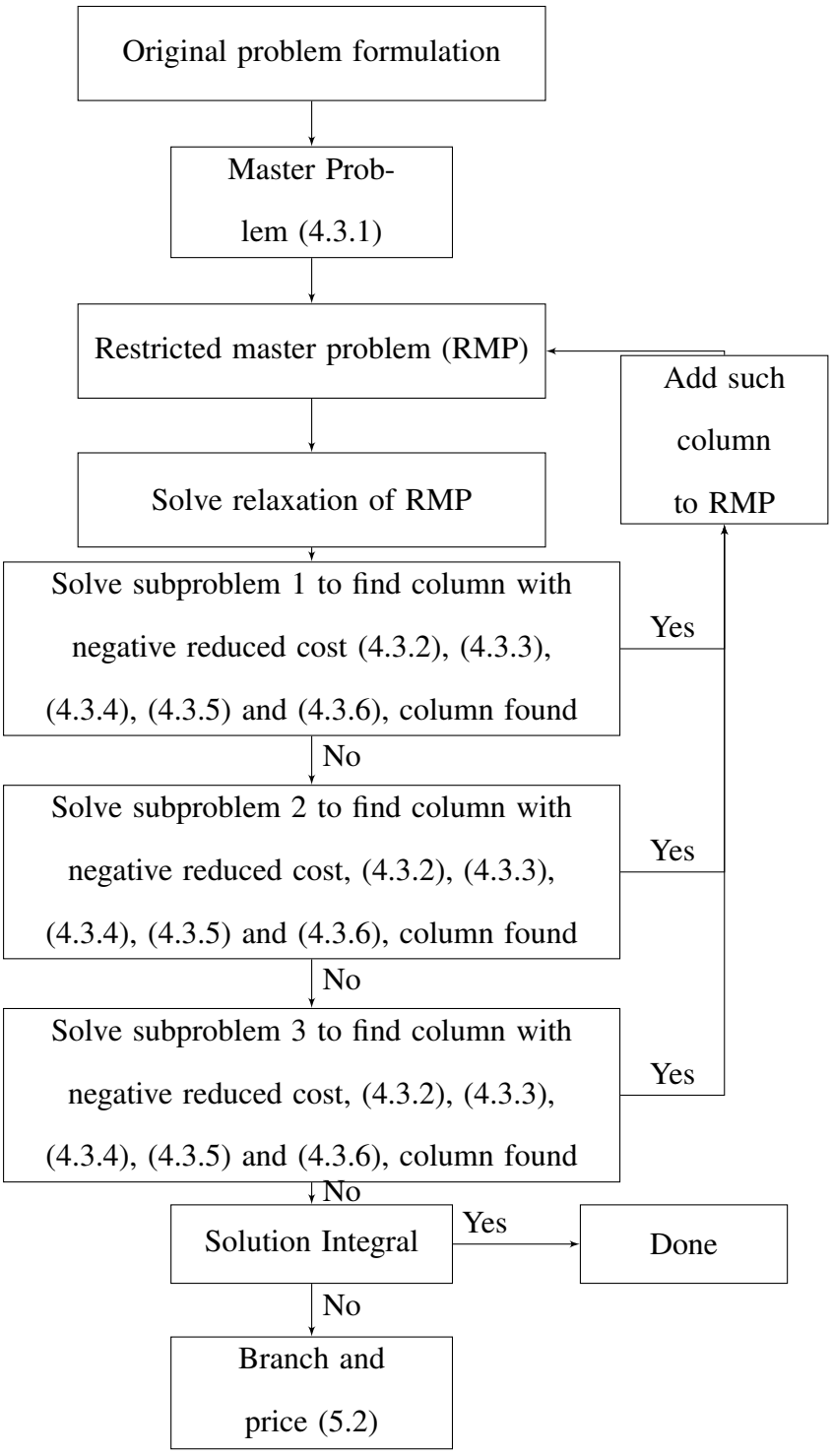
- $(Ed-time)_k$: earliest departure time of $R_k \in Routes$.
- $(Ea-time)_k$: earliest arrival time of $R_k \in Routes$.
- $G^T = (V^T, A^T)$: Route graph
- V^T: Set of all non-dominated routes which are vertices in the route graph.
- A^T: Set of edges in the route graph
- H_k^T: List of labels on node R_k
- $SuccT(R_k)$: Set of successors of route R_k.
- E^T: List of routes waiting to be processed.
- $ExtendT(L_k^T, R_k)$: Function that returns the label resulting from the extension of label $L_k^T \in H_i^T$ towards node R_h when the extension is possible, nothing otherwise.
- $DominatedT(H_h^T)$: Procedure that removes dominated labels in the list of labels H_h^T.
- FT_{kh}: List of labels extended from R_k to R_h

Algorithm 3 Generating all non-dominated tours

Input: $G(N,A)$ # All notation used are written above
output: all non-dominated tours
Initialization # Generate all non-dominated routes
$H_p \leftarrow \{(0,...,0)\}$
for all $v_i \in V - \{p\}$ **do**
  $F_{ij} \leftarrow \emptyset$
end for
$E = \{p\}$
while $E! = \emptyset$ **do**
  Choose $v_i \in E$
  for all $v_j \in Succ(v_i)$ **do**
    $F_{ij} \leftarrow \emptyset$
    for all $L_i \in H_i$ **do**
      if $V_i^j = 0$ **then**
        $F_{ij} \leftarrow F_{ij} \cup Extend(L_i, V_j)$
      end if
    end for
    $H_j \leftarrow Dominated(F_{ij} \cup H_j)$
    if H_j has changed **then**
      $E \leftarrow E \cup \{v_j\}$
    end if
  end for
  for all $L_i \in H_i$ **do**
    if L_i is not extended to any $v_j \in Succ(v_i)$ **then**
      $Routes \leftarrow Extend(L_i, n+1)$
    end if
  end for
  $E \leftarrow E - \{v_i\}$
end while
for all $R_k \in Routes$ **do** # Generate the route graph
  Add R_k to V^T
  Calculate $(Ld-time)_k$, $(La-time)_k$,$(Ed-time)_k$, and$(Ea-time)_k$.
end for
for all $R_k, R_h \in Routes$ **do**
  if R_k and R_h don't have common customer and $(La-time)_k \leq (Ld-time)_h$ **then**
    Add an edge from vertex R_k to vertex R_h in A^T
  end if
end for
for all $R_k \in Routes$ **do**
  Add an edge from vertex R_k to p and from p to R_k
end for

Initialization # Generate all non-dominated tours
$H_k^T \leftarrow \{(0,...,0)\}$
for all $R_k \in V^T - \{p\}$ **do**
  $FT_{kh} \leftarrow \emptyset$
end for
$E^T = \{p\}$
while $E^T != \emptyset$ **do**
  Choose $R_k \in E^T$
  for all $R_h \in SuccT(R_k)$ **do**
    $FT_{kh} \leftarrow \emptyset$
    for all $L_k^T \in H_k^T$ **do**
      if $R_k^h = 0$ **then**
        $FT_{kh} \leftarrow FT_{kh} \cup ExtendT(L_k^T, R_h)$
      end if
    end for
    $H_h^T \leftarrow DominatedT(FT_{kh} \cup H_h^T)$
    if H_h^T has changed **then**
      $E^T \leftarrow E^T \cup \{R_h\}$
    end if
  end for
  $E^T \leftarrow E^T - \{R_k\}$
end while

There are three sub-problems based on the various capacity. All tours will be generated for these three sub-problems using algorithm 3. First, we will see if there is a new tour with a negative reduced cost in the sub-problem with one wagon, if so, the column will be added to the master problem, if not, the second sub-problem will be checked. If there is a new tour with a negative reduced cost to be added to the master problem. If not, we will check the third sub-problem which uses three wagons. We keep solving the sub-problems until all tours with the negative reduced cost are found and added to the master problem. The following flowchart shows the procedure of the algorithm.



4.3.7 Summary

In this chapter, we describe the column generation for VRP, $MTVRPTW$, and $MTVRP-VW-TW$. The first two are the background material, and for clarity one example of col-

umn generation for VRP is presented. The novel work in this thesis is column generation for $MTVRP-VW-TW$, and the experimental evaluation of it is in chapter 7. To do column generation for $MTVRP-VW-TW$, we need to generate all non-dominated paths using label corrected algorithm (3.2) for three different capacities of vehicles. Then the route graphs for these three capacities can be constructed. Each node of the route graph is a route from the generation of non-dominated paths. So each route graph is made of all possible routes as nodes and the possibility of having two consecutive routes for a vehicle as edges. Again, the label correcting algorithm is applied on any of these route graphs to generate all non-dominated paths for this new graph (route graph). The result is multi trips for a single vehicle and it gives us the solution to the subproblems to determine if a new tour with the negative reduced cost exists. We check that any of the three subproblems have a tour with a negative reduced cost. If there is any one, will be added to the master problem.

Chapter 5

Branch and Price

In the first section, branch and bound is explained and next we explain branch and price and apply it to our problem MTVRP-VW-TW.

5.1 Branch and Bound

Branch and bound algorithms are methods for global optimization for nonconvex problems. They are non-heuristic, in the sense that they maintain a provable upper and lower bound on the (globally) optimal objective value. Branch and bound algorithms can be slow, however in the worst case, they require effort that grows exponentially with problem size, but in some cases, the methods converge with much less effort [103].

In the following, we explain the basic idea of branch and bound, how the algorithm works, flowchart, and examples for the algorithm are from [3].

5.1.1 Basic idea

In fact, the branch and bound break the problem into the smaller sized problems to solve as illustrated by the following mixed integer linear program (*MILP*) :

$$J^* = \min_{(x,y)} \quad c^T x + d^T y$$

$$\text{s.t.} \quad (x,y) \in X$$

Where X is feasible solutions of the problem,

$$X = \{(x,y) \in \mathbb{R}^n_+ \times \mathbb{Z}^p_+ : Ax + By \geq b\}$$

$\mathbb{R}^n_+$ and $\mathbb{Z}^p_+$ are the space of n dimensional vectors of positive real numbers and the space of p dimensional vectors of positive integer numbers respectively. X can be decomposed into smaller sets, $X = X_1 \cup X_2 \cup \cdots \cup X_K$, and let

$$J^k = \min\{c^T x + d^T y : (x,y) \in X_k\} \quad k = 1, \ldots, K$$

Then, $J^* = max_k J^k$

5.1.2 Decomposition of the mixed-integer linear problem

Consider the LP-relaxation of the initial problem (*MILP*) as:

$$\begin{aligned} J_R = \min_{(x,y)} \quad & c^T x + d^T y \\ \text{s.t.} \quad & (x,y) \in P \end{aligned}$$

Where, $P = \{(x,y) \in \mathbb{R}^n_+ \times \mathbb{R}^p_+ : Ax + By \geq b\}$.

Let $(x^R, y^R) \in P$ be a feasible solution of the *LP*. We check whether all variables of y are integers, if not, a fractional y_j is taken, and then, we decompose the problem based on bounds on integer variables as shown next.

Let y^R_j be a variable with the fractional value, therefore, these two problems must be solved:

$$P^1 := P \cap \{y : y_j \leq \lfloor y^R_j \rfloor\}$$
$$P^2 := P \cap \{y : y_j \geq \lceil y^R_j \rceil\}$$

The following figure shows how decomposition works.

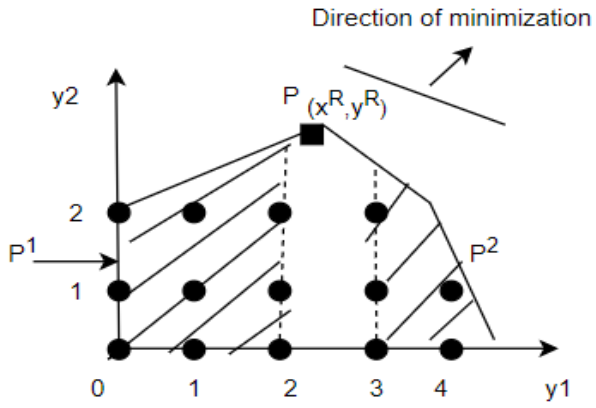


Figure 5.1: Branching at a node

So, two new *LP* problems for P^1 and P^2 are solved.

5.1.3 Enumeration

The root of the enumeration tree is the *LP* solution and the problems obtained from the decomposition add two new child nodes to the tree, a left branch and a right branch.

5.1.4 Bounds

For the given *MILP* let (x^*, y^*) be the optimal solution of the problem with objective value J^* and let J_R be the objective value of the relaxation of *MILP*, and $(\bar{x}, \bar{y})$ be any integer feasible solution with the objective value $\bar{J}$, we have:

- J_R provides a lower bound on J^*, so $J_R \leq J^*$
- $\bar{J}$ provides an upper bound on J^*, so $J^* \leq \bar{J}$

Consequently,

$$J_R \leq J^* \leq \bar{J}$$

5.1.5 Pruning

Nodes in the enumeration tree are pruned in the following ways:

- Pruning by optimality: A solution of the LP is an integer, the decomposition and adding cuts can not improve the solution. We check if it can improve the best solution so far, if it improves, update the best lower bound and prune the node.

- Pruning by bound: A solution J^i at a node i is worse than the best upper bound that we have, $J^i \geq \overline{J}$, then we do not explore node i.

- Pruning by infeasibility: A *LP* is infeasible at a node, then the node i is not explored.

5.1.6 Choosing a fractional variable to branch

If none of the pruning cases happen which means the *LP* has a feasible solution, $J^i \leq \overline{J}$ and $y^i \notin \mathbb{Z}_+^P$, then we need to branch at the current node. But, which non-integer variable must be chosen, there are many selection strategies, and few of them are:

- The variable with maximum integer infeasibility: branch on a variable that has the fractional part close to 0.5.

- The variable with minimum integer infeasibility.

- Strong branching: select the variable that can improve the objective function value more. One way is to perform a few iterations of the dual simplex to determine which variable gives the most improvement.

5.1.7 Node selection

Each time a node cannot be pruned, two branches are created and we need to decide which node must be selected to be solved next. Different search algorithms on trees can be applied to determine the next node [104]. Some choices are:

- Depth-first search: when children are created, they are added to a list of unexplored nodes and the last node added to the list is selected to explore.

- Breadth-first search: select the first node added to the list to be explored.

- Best-bound search: is to select a node whose parent has the best bound.

5.1.8 Pseudocode

The pseudocode for branch and bound is shown in Algorithm 4.

Algorithm 4 Branch and Bound

```
Initialization
L ← {0}
J ← ∞
k ← 0
while L is not empty do
  select node i of L and delete it from L
  solve LP-relaxation at node i
  k ← k + 1
  if LP-relaxation is infeasible then
    if k=1 then
      The problem is infeasible
    end if
    prune node i by infeasibility
  else
    Let (x^i, y^i) be the solution and J_R^i objective value of LP-relaxation at node i
    if J_R^i ≥ J̄ then
      prune the node by bound
    else
      if all variables of solution are integers, update then
        update the upper bound, J̄ ← J_R^i
        prune node i by optimality
      else
        generate two branches at node i and add two new nodes to the list L
      end if
    end if
  end if
end while
```

5.1.9 Example

The branch and bound algorithm is illustrated for the following *IP* problem:

$$J^* = \min_y \quad -1y_1 - 2y_2$$

s.t

$$\frac{-1}{2}y_1 + y_2 \leq \frac{11}{5}$$

$$4y_1 + 5y_2 \leq 26$$

$$7y_1 + 3y_2 \leq 32$$

$$y \in \mathbb{Z}_+^2$$

We use the most fractional variable to branch and node is selected using the breadth-first search. The tree and the solution of each node are in Fig. 5.2

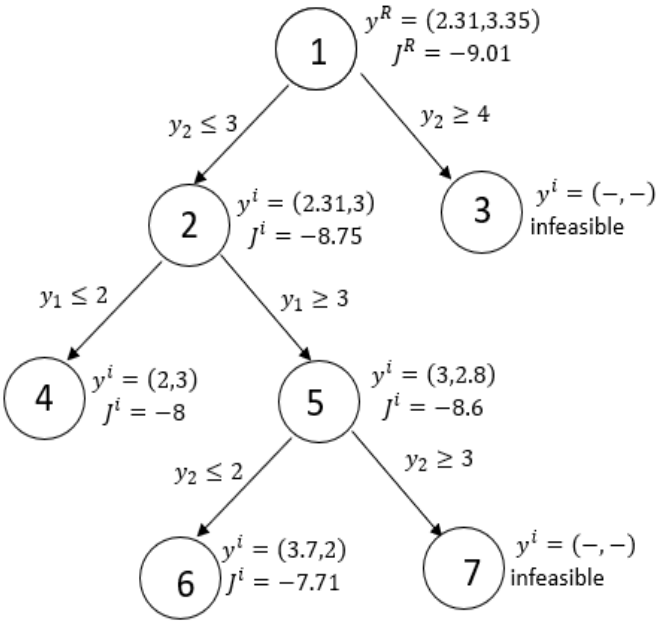


Figure 5.2: Branch and bound example [3]

The first iteration of the algorithm is the root that has the objective value of -9.01 and variables that have fractional values. The only node to branch is the root. The most fractional variable y_2 is chosen and two branches are created. so, node 1 is deleted from the

list and nodes 2 and 3 are added. The next node to treat is node 2. LP-relaxation is solved at node 2, the objective value is -8.75 and there is one fractional variable y_1. y_1 is selected and two branches are generated, nodes 4 and 5 are added to the list and node 2 is deleted. Node 3 is infeasible, so it is pruned and the node is deleted from the list. Now, node 4 should be treated that has an integer solution, the best bound is updated. Node 4 is also deleted, so the only node left in the list is 5 which has a feasible solution and one fractional variable y_2 to branch on. Then, two branches are made and nodes 6 and 7 are added to the list while node 5 is removed. Solving LP-relaxation at node 6, the objective value is greater than the best bound that we have so far, so it is pruned by bound and node 6 is removed from the list. The last node 7 has the LP-relaxation infeasible, that node is deleted from the list. There is no more node to check. So the algorithm stops and the optimal integer solution found is $(2,3)$ with the objective value -8.

5.2 Branch and price algorithm

The branch and price algorithm is a combination of the column generation and branch and bound algorithm in which the column generation is executed at each node of the search tree. Branch and price is shown in Fig. 5.3. The LP at each node is solved using column generation.

5.2.1 Branch and price algorithm for $MTVRP-VW-TW$

We describe initialization, the search strategy, branching strategy, and the upper bound that we use in our implementation of branch and price for the multi-trip VRP with variable wagons and time windows.

Initialization

At the root of the search tree, the *RLMP* is initialized with tours made of a single customer visit. The number of columns thus corresponds to the number of customers. For internal nodes in the search tree, the algorithm initializes the *RLMP* with the set of columns

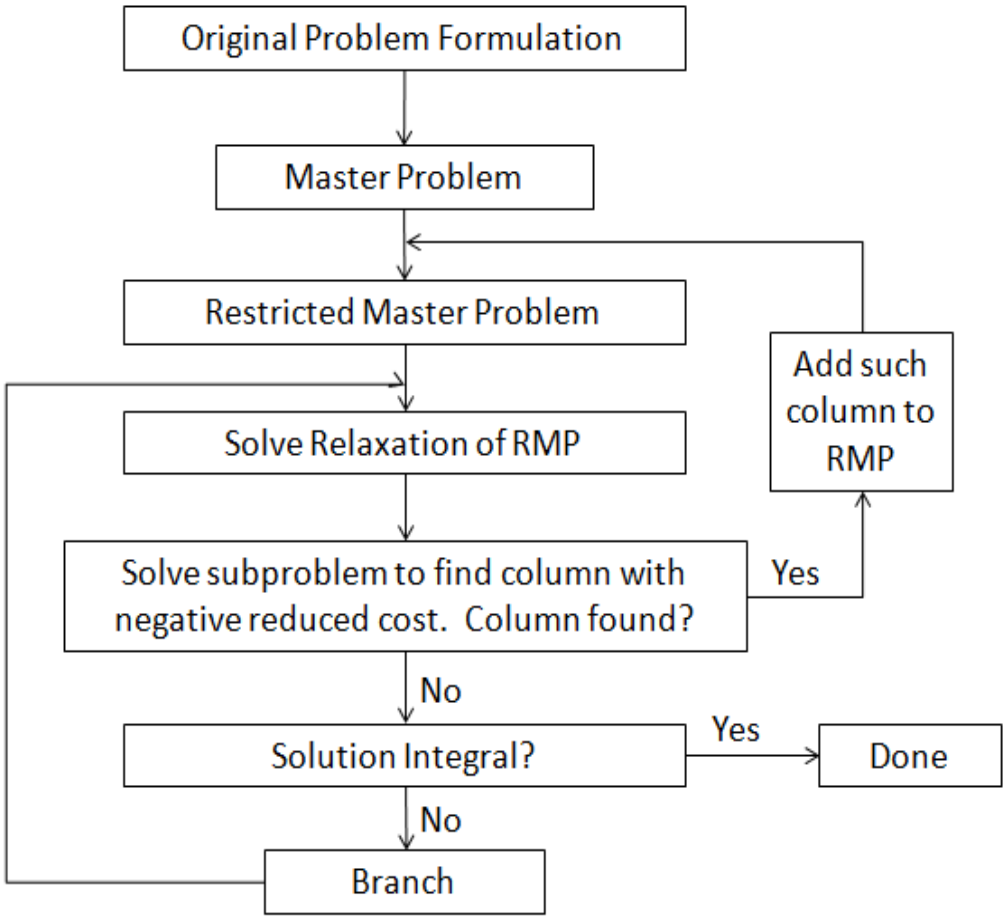


Figure 5.3: Branch and price algorithm

in the parent node considered, after removing columns that are infeasible due to branching [42]. The minimum capacity for all vehicles that service a single customer, is one wagon.

Search strategy

The branch and price tree is explored using depth first search.

Lower and upper bound

The solution to *RLMP* at the root node gives a lower bound for the problem. We solve the master problem using CPLEX. The integer solution of the master problem given by the CPLEX 12.8 is used as the upper bound.

Branching strategy

Two branching strategies are used. We branch on the number of vehicles and on the arcs.

Branching on the number of vehicles: We sum the value of variables of the optimal solution of *RLMP*, so $k = \sum_{w \in \Omega'} y_w$, where $\Omega^{'} \subseteq \Omega$. If k is fractional, two branches are created. For each branch, one additional constraint is added to the master problem, these two constraints are:

$$\sum_{w \in \Omega'} y_w \leq \lfloor k \rfloor$$

and

$$\sum_{w \in \Omega'} y_w \geq \lfloor k+1 \rfloor.$$

The dual variable value corresponding to the new constraint is added to the subproblem and the column generation is done again for this new child node.

Branching on arcs: The branch on an arc happens when the flow on an arc (i, j) is fractional. We calculate the flow on any arc that is in some column. The sum of the y_w, $w \in \Omega^{'}$ on the columns that include an arc (i, j), will give the flow on the arc. The arc with fractional value is taken. So, these branches will be:

- Left branch: $x_{ij} = 1$, which means the customer j must be visited immediately after customer i in all tours of *RLMP* and the route graph as well. To enforce it, all columns in the *RLMP* and the route graph that contains arc (i,k) with $k \neq j$ and (k, j) with $k \neq i$ must be deleted. Also, if $x_i = \sum_{w \in \Omega'} a_{iw} y_w$, then the decision variables for vertices i and j are set to one in *RLMP*, $x_i = 1$ and $x_j = 1$.

- Right branch: $x_{ij} = 0$, which means the customer j must not follow the customer i immediately. So all tours including the arc (i, j) in *RLMP* and the route graph must be removed.

Branch and price process

We start at the root and if *RLMP* is feasible, all possible columns from three subproblems will be added to the *RLMP*. The LP solution of the root is set as the lower bound and also integer solution given by the CPLEX is used as the upper bound. Two branches on the number of the vehicles are created and we use a stack in implementation DFS, so nodes are added to the front of the stack. For each node, column generation is used again to find the LP- solution of *RLMP*. We check if the node must be pruned or kept, if the node is not pruned, then we update the upper bound and create two new branches. After processing all nodes, we see if the sum of the value of variables of the best bound is an integer but variable values are not, then the branching on nodes is used to generate nodes. Then, we calculate the flows on all the arcs. In the same way, we continue with the last node added to the head of the list to see if the node must be pruned or kept after using column generation. If we keep the node, two branches are created and the upper bound is updated. The process continues until the stack is empty.

Chapter 6

Resource management in device-to-device communications

This chapter is on device to device communication in wireless networks. The system model is described briefly in section 6.1.1. The branch-n-cut algorithm is described in section 6.2. A combinatorial algorithm for solving the Lagrangian relaxation is in section 6.3. An iterative rounding algorithm and the proof of the quality of the approximation is in section 6.4.

6.1 Device to device communication

Interference minimization or maximization of sum-rate is a mature research problem in the field of wireless and cellular communication. In recent days, special attention is given to this area due to the introduction of a new mode of personal communication known as device to device (D2D) communication underlay to a cellular network [105].

Here we study the knapsack based model for resource allocation for D2D communication first proposed in [106]. Given a set of cellular users, and a set of D2D pairs (from the cellular users), the model allows for radio resource of a cellular user to be used by at most one D2D pair (1-1 resource). This radio allocation if used will generate some interference (which can be quantified given the system model) and provide some sum-rate. The model seeks to find those allocations which simultaneously meet the sum-rate and the interference requirements from the systems. For a pair of cellular user and a D2D receiver, the model specifies the signal interference noise ratio (SINR) at a cellular user when the base

station transmits, and the SINR at the D2D receiver. The SINR at the cellular user and D2D receiver is considered as the minimum level of detail needed to examine interference [107].

One can also model the resource allocation problem as a minimum knapsack problem with side constraints. If a target sum-rate is required then the objective is to minimize the total interference. The knapsack constraint models the requirement that the target sum-rate is met, and side constraints model the 1-1 resource requirement, this model was first studied in [59].

6.1.1 System model

Before an explanation of the system model, we need some definitions, which are given next.

Channel gain A channel is a link (wired or wireless) between the transmitting and receiving antennas. Channel gain is the difference between the transmitter's output power and the power received at the receiver end.

Interference: interference happens when two or more waves merge to generate one wave. Interference may be constructive or destructive. Interference can block reception, may result in a momentary loss of a signal, or may impact the quality of the sound or image that a piece of equipment produces.

Sum-rate: is the total rate of transmission over several simultaneous transmissions (over different channels).

SINR: Describes the ratio between the gain of a channel and the total interference and noise encountered by the channel. The exact formulas are given later.

The system model explained below is from [6, 106]. Let D be the set of D2D pairs and C the set of cellular users. G_{bc} is the channel gain between the base station b and cellular user $c \in C$. G_{dc} is the channel gain between a transmitter for a D2D pair $d \in D$ and the cellular user $c \in C$. G_{tr} denotes the channel gain between D2D transmitter t and D2D receiver r

(for D2D pair d). G_{bd} is the channel gain between the base station b and d's receiver. The transmission power of cellular user c and D2D pair d are P^c and P^d respectively. Variable $x_{(c,d)} \in \{0,1\}$ indicates whether a D2D pair (d) uses radio resources from a cellular user (c), and $s_{(c,d)}$ represents the sum rate. The total sum rate of the cellular users which share resources with some D2D pair is given by,

$$S = \sum_{c \in C} \sum_{d \in D} x_{(c,d)} s_{(c,d)}$$

The sum rate for a D2D pair (c,d) is given by Shannon's formula.

$$s_{(c,d)} = B \log_2 (1 + \gamma_{(c,d)}) + B \log_2 (1 + \gamma_{(d,b)})$$

where $\gamma_{(c,d)}$ is the SINR at cellular user with signal transmitting from base station given by,

$$\gamma_{(c,d)} = \frac{P^c G_{bc}}{T + P^d G_{dc}}$$

and $\gamma_{(d,b)}$ is the SINR at D2D receiver end.

$$\gamma_{(d,b)} = \frac{P^d G_{tr}}{T + P^c G_{bd}}$$

The interference for pair (c,d) is given by,

$$I_{(c,d)} = P^d G_{dc} + P^c G_{bd}$$

By assumption, each cellular user can share resources with at most one D2D pair and vice versa. Therefore,

$$\sum_{c \in C} x_{(c,d)} \leq 1, \forall d \in D$$

Similarly,

$$\sum_{d \in D} x_{(c,d)} \leq 1, \forall c \in C.$$

If I be the total interference allowed then

$$\sum_{c\in C}\sum_{d\in D} x_{(c,d)} I_{(c,d)} \leq I.$$

The model can now be written as,

$$\max \sum_{c\in C}\sum_{d\in D} x_{(c,d)} s_{(c,d)} \tag{6.1}$$

$$\sum_{c\in C}\sum_{d\in D} x_{(c,d)} I_{(c,d)} \leq I \tag{6.2}$$

$$\sum_{c\in C} x_{(c,d)} \leq 1 \qquad \forall d \in D \tag{6.3}$$

$$\sum_{d\in D} x_{(c,d)} \leq 1 \qquad \forall c \in C \tag{6.4}$$

$$x_{(c,d)} \in \{0,1\} \qquad \forall c \in C, \forall d \in D \tag{6.5}$$

The objective function (6.1) maximizes the total sum rate. Constraint (6.2) ensures that the interference is less than the target interference. Finally, constraints (6.3) and (6.4) indicate that each D2D and the user cellular can share resources with at most one user cellular and D2D respectively. We can think of cellular users and D2D pairs as two sides of a bipartite graph (V,E), and (c,d) as an edge in this graph. Let $\mathcal{M}$ be the set of matchings in the bipartite graph. Then, by a simple change of variable $(c,d) = e$ we can write the above model as:

$$\max \sum_{e\in E} x_e s_e \tag{6.6}$$

$$\sum_{e\in E} x_e I_e \leq I \tag{6.7}$$

$$\{x_e | x_e = 1\} \in \mathcal{M} \tag{6.8}$$

$$x_e \in \{0,1\} \quad \forall e \in E \tag{6.9}$$

where constraint (6.8) states that the set of edges in $\{e : x_e = 1\}$ form a matching and constraint (6.7) is the knapsack constraint. Therefore, we say that model is a knapsack with a matching side constraint. We write constraint (6.8) as $\sum_{e\in\delta(v)} x_e \leq 1, \forall v \in V = C \cup D$ at times where $\delta(v)$ is the set of edges incident on v (either a cellular user or a D2D pair).

6.2 Branch and cut

Branch and cut is derived from branch and bound when linear inequalities known as cuts are added at every search node of search tree [108]. The cuts are satisfied by all the feasible solutions and reduce the search space. We use linear inequalities known as cover-cuts [109] that are generated from equation (6.7) and the current LP solution as explained next.

Cuts need to be generated in each iteration and will be added to the relaxation of our model at each node of the branch and bound tree which reduces the integrality gap. The problem is solved again using branch and bound until the solution is an optimal integer solution of our IP.

6.2.1 Cover Cuts

A cover S is a set of pairs (c,d) where $c \in C, d \in D$ such that the total interference on the pairs in S is larger than the allowed interference

$$\sum_{(c,d)\in S} I_{(c,d)} > I \tag{6.10}$$

S is a minimal cover if no proper subset of S is also a cover. Any feasible solution can contain only at most $|S| - 1$ elements from a minimal cover S. Therefore, for any feasible solution x,

$$\sum_{(c,d)\in S} x_{(c,d)} \leq |S| - 1 \tag{6.11}$$

Equivalently, for a minimal cover S, and a feasible solution x^*:

$$\sum_{(c,d)\in S}(1-x^*_{(c,d)}) \geq 1 \tag{6.12}$$

For a cut to be valid, the LP solution x^* must also satisfy the cover condition. Violating the above condition demonstrates the existence of a cover-cut.

$$\sum_{(c,d)\in S}(1-x^*_{(c,d)}) < 1 \tag{6.13}$$

So, given x^*, if there is an S such that

$$\sum_{(c,d)\in S}(1-x^*_{(c,d)}) < 1 \tag{6.14}$$

$$\sum_{(c,d)\in S} I_{(c,d)} > I \tag{6.15}$$

Then inequality (6.11) can be added to the node in the search tree during branch and bound. Existence of a minimal cover S that satisfies equation (6.14) is determined by solving the following knapsack problem.

$$\max \sum_{c\in C}\sum_{d\in D} y_{(c,d)} I_{(c,d)} \tag{6.16}$$

$$\sum_{c\in C}\sum_{d\in D} y_{(c,d)}(1-x^*_{(c,d)}) < 1 \tag{6.17}$$

Where, $y_{c,d} \in \{0,1\}$ is the decision variable in the knapsack problem.

If we find a solution to the knapsack problem above, where the optimal value exceeds the interference capacity I, then we add (6.11) as a new constraint to our *MIP*.

6.3 Lagrangian relaxation

Let us recall the IP formulation. The input is a bipartite graph $B= (V,E)$ and an interference-cap I. $V(E)$ is the set of vertices (edges). The two sides of the partition are the D2D pairs on one side and the cellular users on the other side. Each edge e, has an

interference I_e and a sum-rate s_e. The set of edges incident on $v \in V$ is denoted $\delta(v)$. The goal is to find a maximum weight matching such that the total sum of the interferences on the edges in the matching is at most I.

$$IP = \max \sum_{e \in E} s_e x_e \tag{6.18}$$

$$\sum_{e \in \delta(v)} x_e \leq 1, \forall v \in V \tag{6.19}$$

$$\sum_{e \in E} I_e x_e \leq I \tag{6.20}$$

$$x_e \in \{0,1\}, \forall e \in E. \tag{6.21}$$

We obtain the following Lagrangian relaxation ($LR(\lambda)$) by moving the total interference-capacity constraint in the objective function.

$$LR(\lambda) = \max \sum_{e \in E} s_e x_e + \lambda (I - \sum_{e \in E} I_e x_e) \tag{6.22}$$

$$\sum_{e \in \delta(v)} x_e \leq 1, \forall v \in V \tag{6.23}$$

$$x_e \in \{0,1\}, \forall e \in E. \tag{6.24}$$

For a fixed $\lambda \geq 0$, an optimal solution to the integer program (IP), gives an objective value (for $LR(\lambda)$) which is an upper bound on the objective function of the IP. But, we do not know the optimal solution to IP. Fortunately, an optimal solution to $LR(\lambda)$ for a fixed λ can be obtained by solving a weighted maximum matching follows. Let us rewrite the objective function as:

$$\max \sum_{e \in E} (s_e - \lambda I_e) x_e + \lambda\, I \tag{6.25}$$

Any e for which $s_e - \lambda I_e \leq 0$ is not in any optimal solution. So, for a fixed λ we work with the subgraph B' with only those edges $e : s_e - \lambda I_e > 0$. Weight of an edge e is $w_e =$

$s_e - \lambda I_e > 0$. A maximum weight matching in B' is an optimal solution to $LR(\lambda)$ given λ. Therefore, our goal is to find λ such that the value of the $LR(\lambda)$ is minimized. This problem for minimizing over λ is the Lagrangian upper bound problem, $LU = \min_{\lambda \geq 0} LR(\lambda)$.

$$LU = \min_{\lambda > 0} \left(\max \sum_{e \in E} (s_e - \lambda I_e) x_e \right) + \lambda I \tag{6.26}$$

$$\sum_{e \in \delta(v)} x_e \leq 1, \forall v \in V \tag{6.27}$$

$$x_e \in \{0,1\}, \forall e \in E. \tag{6.28}$$

As an example consider the complete bipartite graph over $2n$ nodes, where a first edge e has interference of $I_e = 1$ and a sum-rate of $s_e = 1$, every other edge has a interference of $1/n^2$ and and sum-rate of 0. The total interference requirement $I = 1$. The integer solution in which the $x_1 = 1$ and $x_{i>1} = 0$ is an optimal solution with objective value 1. The fractional solution $x_1 = 1/n^2, x_{i \geq 2} = 1$ is a feasible solution to the LP relaxation with objective value $1/n^2$. This example has an unbounded integrality gap.

For each edge e, $s_e/I_e \in \{1,0\}$. Therefore, interesting values of λ for this example are $0,1$. For $\lambda = 0$, B' contains all the edges with weight $w_e = s_e$. A matching of size n, gives an optimal solution to $LR(0)$ of value 1. For the second case when $\lambda = 1$, only the first edge is in B', as $s_e - I_e < 0$ for all the other edges. A maximum weight matching includes the edge with sum-rate 1. The optimal solution value to LU is $0 + \lambda I = 1$. This example also illustrates that not only do we obtain an integral solution, at times, but this solution also satisfies the sum-rate constraint. Hence, we obtain a solution to the IP.

Next, we illustrate how to obtain the optimal solution to LU without resorting to gradient descent. The Lagrangian parameter λ can only take values in the set $\{s_e/I_e : e \in E\}$. Suppose $\lambda < s_e/I_e$ for all the edges in B'. Increasing λ by a value δ changes the objective function as $\delta I - \sum_e e \in E' I_e$ where E' is the set of edges in B' and the graph has the uniform interference. For $\delta > 0$ this net gain is negative when $I < \sum_e e \in E' I_e$. State other way, an

increase in δ reduces the objective function value. Since we want λ that minimizes $LR(\lambda)$, it is sufficient to consider $\lambda = s_e/I_e$ for some $e \in E$.

For each choice of λ and fixed number of edges we need to solve a weighted matching problem which for bipartite graph takes $O(|E|\sqrt{|V|})$ [110]. As there are $|E|$ choices for λ the total running time is $O(|E|^2\sqrt{|V|})$. This can be reduced to $O(|E|\sqrt{|V|}\log|E|)$ by sorting $\{s_e/c_e : e \in E\}$ and performing a binary search over λ. Therefore, we have the following Theorem.

Theorem 6.1. *Lagrangian upper bound problem LU on a bipartite graph $B = (V,E)$ can be solved optimally in time $O(|E|\sqrt{|V|}\log|E|)$ for the uniform interference and fixed number of edges.*

Since a maximum weight matching can be constructed for any graph in polynomial time, we can also solve the Lagrangian upper bound problem LU optimally on any graph.

6.4 Iterative rounding algorithm

Saha et al. [5] studied the problem of interference minimization when the sum-rates were arbitrary, and the interference was uniform and gave a polynomial time algorithm. The computational complexity of the case when the interferences are arbitrary, and the sum-rates are uniform is not known. In this section, we give an iterative rounding algorithm that in each round will solve an LP and set the value of one of the variables. The number of rounds is at most the number of edges and the work done in each round is polynomial. The algorithm will work with extreme point solutions of the LP. We will show that a variable with a value in a prescribed set exists in each round. We will prove a bound on the approximation ratio.

The input is a bipartite graph $B = (V,E)$ and an interference limit I. $V(E)$ is the set of vertices (edges). The two sides of the partition are the D2D pairs on one side and the cellular users on the other side. Each edge e has an interference I_e and a sum-rate s_e. The set of edges incident on $v \in V$ is denoted $\delta(v)$. The goal is to find a maximum weight matching

such that the total sum of the interference on the edges in the matching is at most I. This gives us the following integer program.

$$IP = \max \sum_{e \in E} s_e x_e \tag{6.29}$$

$$\sum_{e \in \delta(v)} x_e \leq 1, \forall v \in V \tag{6.30}$$

$$\sum_{e \in E} I_e x_e \leq I \tag{6.31}$$

$$x_e \in \{0,1\}, \forall e \in E. \tag{6.32}$$

The linear programming relaxation (LP) is:

$$LP = \max \sum_{e \in E} s_e x_e \tag{6.33}$$

$$\sum_{e \in \delta(v)} x_e \leq 1, \forall v \in V \tag{6.34}$$

$$\sum_{e \in E} I_e x_e \leq I \tag{6.35}$$

$$x_e \geq 0, \forall e \in E. \tag{6.36}$$

6.4.1 Algorithm

We give a procedure (Algorithm 5) that will construct in polynomial time (depending on the time needed to solve a linear program) an integral solution such that

- the interference is no more than interference capacity I,
- the sum-rate is at least a quarter of the maximum sum-rate in an optimal LP solution.

Algorithm 5 relies on an optimal solution to the LP relaxation and is iterative. Given an optimal solution, x^* to the LP, one of the two things happens: in each step a reduced problem (smaller in size) is obtained, which is solved iteratively. Note that when we remove

an edge, we delete its endpoints and any edge incident on the endpoints. The algorithm uses the following cases.

- An edge e exists such that $x_e^* = 0$ is removed from the graph B. This gives a reduced problem B'.
- If an edge e exists such that $x_e^* = 1$ then e is added to the solution. The edge e any edges incident on e are removed from B to obtain a reduced problem B' with interference bound $I' = I - I_e$.
- For all edges $0 < x_e^* < 1$. In this case, the graph induced by edges in $\{e \mid x_e^* > 0\}$ is a "near" cycle. We decompose this graph into two matchings and select the matching with the larger total sum rate and return the edges selected from algorithm 6 which is a greedy algorithm for knapsack problem as the solution.

  We prove the following Theorem.

Algorithm 5

1: $F = \{\}$.
2: Solve the LP relaxation for graph B. If the value of LP optimal solution is 0 then GOTO END.
3: If there is e such that $x_e = 0$ then remove edge e from B. GOTO 2.
4: If there is e such that $x_e = 1$ then add e to F and remove edge e and edges incident on e from B and $I = I - I_e$. GOTO 2.
5: If $0 < x_e < 1$ for all the edges then decompose the graph into two matching and select the matching with lager total sum-rate.
6: Use the greedy algorithm 6 on the selected matching, add edges selected from the algorithm to F and remove those edges and edges incident on them from B and $I = I - \sum_{e \in G} I_e$. GOTO 2.
7: END

Theorem 6.2. *Algorithm 5 computes a solution with the sum-rate at least a quarter of the sum-rate of the optimal LP solution and the total interference less than equal to the target interference.*

Proof of Theorem 6.2 relies on the following Lemmas 6.4, 6.5, and 6.6. The proof of Lemma 6.4 in turn relies on the structure of the extreme point solutions captured by the following Lemma.

Lemma 6.3 ([111]). *[Rank Lemma] Let $P = \{x \mid Ax \geq b, x \geq 0\}$ and let x be an extreme point solution to P such that each component $x_i > 0$. If C is any maximal set of linearly independent tight constraints ($A[i,:]x = c[i]$) then $|C| = |X|$.*

The intuition behind the rank lemma can be summarized as follows: because x is an extreme point solution the columns of A are linearly independent. Also, the column rank of A equals its row rank. Therefore any maximal set of linearly independent tight constraints equals the number of variables. For a proof of the Rank Lemma, see [111, Chapter 2]. First, we note that in any optimal solution (extreme point or not) to the LP, constraint (6.7) is satisfied at equality in any optimal solution and may be linearly dependent (or independent). As a direct application of the rank lemma, we get Lemma 6.4.

Lemma 6.4 ([111]). *Let x^* be a basic feasible solution to the LP such that $0 < x^*_e$ for all $e \in E$ and $W = V_1 \cup V_2$ be the set of vertices such that $\sum_{e \in \delta(v)} x^*_e = 1$ and $\sum_{e \in E} I_e x^*_e = I$, then*

1. *$|W| + 1 \geq |E|$. If constraint (6.7) is in the maximal set of linearly independent tight constraints then $|W| + 1 = |E|$. In the other case $|W| = |E|$.*

We now prove if $0 < x^*_e < 1$ for every edge then the bipartite graph is near a cycle. One example of such a structure is shown in section 6.4.2.

Lemma 6.5 (Structure). *Let x^* be an optimal solution to the LP defined by the graph $B' = (V', E')$ such that $0 < x^*_e < 1$ for all $e \in E'$. Then, the number of vertices of degree ≥ 3 is at most 2.*

Proof. Given an optimal solution x^*, call a constraint in the LP tight if it is satisfied at equality. Given x^* an extreme point solution let C be the set of tight constraints. Note that $\sum_{e \in E'} x^*_e = C$ (is tight), else we can reduce the objective function value. If W is the set

of vertices at which the matching constraint is tight ($\sum_{e\in\delta(v)} x_e = 1$) then by Lemma 6.4, $|W|+1 \geq |E|$.

Let number of vertices in W with degree at least 3 be k. We know

$$2|W|+2 \geq 2|E| \geq \sum_{v\in W} \delta(v) \geq 3k + (|W|-k)2 = (2|W|+k). \tag{6.37}$$

The first inequality in Equation (6.37) follows i) in Lemma 6.4. The second inequality follows as $W \subseteq V$. The third inequality is due to the assumption that $\delta(v) \geq 3$ for k vertices $\in W$. We infer, $k \leq 2$, i.e. there are $|W|-2$ vertices with degree ≤ 2.

As $x_e < 1$ for all $e \in E$, every vertex in W has degree at least 2. Therefore there are $|W|-2$ vertices with degree 2. There are at most two vertices of degree more than 2. Suppose there are only degree three vertices (either 0, 1, or 2), then the graph is either a cycle, cycle plus one edge, or a cycle plus two edges. The set of edges E is nearly a cycle and We call such a graph a "near cycle." We assume that the sum rates are low enough compared to the value of the optimal solution to LP, so we ignore these extra edges in the analysis in the next part. There cannot be any vertex of degree four in W else equation (6.37) is not satisfied. □

We focus only on the even cycle in the rest of the section. Next, we see an example of the structure and then how this structure is used in Algorithm 5. The edges in the cycles can be partitioned into two sets M_1, M_2 each of which is a matching such that for $e \in M_1, x^*_e < 1/2$.

6.4.2 Structure

As a consequence of lemmas 6.4 and 6.5, we have seen that the edges in any optimal solution x^* in which every variable takes on a fractional value form a near cycle. In each iteration, we make sure that the interference at each edge in the current solution at most the remaining interference capacity I_r. This is achieved by removing edges e for which $I_e > I_r$ after step 4 of algorithm 5.

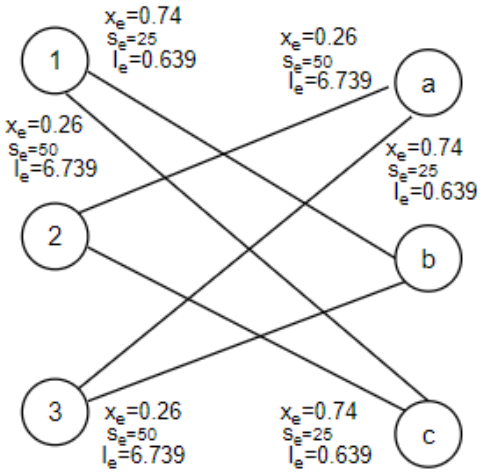


Figure 6.1: Even Cycle

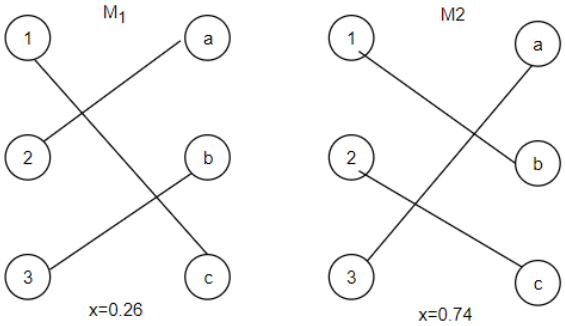


Figure 6.2: Decomposition into Matchings

Figure 6.1 shows an example of such a cycle in an optimal solution of the LP. M_1, M_2 are two matchings that this cycle is decomposed to. The edges in M_1 take value x and the edges in M_2 take value $1-x$ as shown in figure 6.2.

We now prove Theorem 6.2.

6.4.3 Proof of Theorem 6.2

Proof of Theorem 6.2. Let B be the graph at the start of Algorithm 5 and B' the reduced graph obtained after the step 3/4/5/6. Let x^* be the optimal solution to LP for graph B. If x' is the optimal solution to the LP for B'. We perform induction on the number of iterations.

BASE CASE: If there is only a single iteration then three possibilities arise. The sub-problem B' is empty in each case.

- An edge $x_e^* = 0$ is removed. The value of the objective function does not change. The LHS in the sum-rate constraint does not change.
- An edge $x_e^* = 1$ is added to the solution, $I = I - I_e$.
- For all the edges $0 < x_e^* < 1$. By Lemma 6.5 the graph is a near cycle. We decompose the cycle into two matchings. Select the matching with the larger total sum rate and use Algorithm (6). This gives the total sum rate at least a quarter of the optimal LP solution, which is proved in equation (6.46) next, and the total interference is less than equal to the target interference.

  Let S^* be the optimal solution to the *LP* problem with the variable values x_e^*. Let M_1 be the matching that $x_e^* < \frac{1}{2}$ for all $e \in M_1$ and, M_2 be the matching that $x_e^* \geq \frac{1}{2}$ for all $e \in M_2$. If S^* be the optimal solution of LP-relaxation (6.29). By definition, we have:

$$S^* = \sum_{i=1}^{2} S^*(M_i)$$

  We select the matching with the larger total sum rate between $S^*(M_1)$ and $S^*(M_2)$. If the matching $M_i,\ i \in \{1,2\}$ has the highest total sum-rate, then $S^*(M_i) \geq \frac{S^*}{2}$.

We can select a subset of edges from M_i in the solution. This is modeled as the following knapsack problem

$$KS = \max \sum_{e \in M_1} s_e x_e \tag{6.38}$$

$$\sum_{e \in M_1} I_e x_e \leq I_r \tag{6.39}$$

$$x_e \in \{0,1\}, \forall e \in M_i. \tag{6.40}$$

The LP relaxation of the KS is:

$$KS - LP = \max \sum_{e \in M_1} s_e x_e \tag{6.41}$$

$$\sum_{e \in M_1} I_e x_e \leq I_r \tag{6.42}$$

$$x_e \geq 0, \quad \forall e \in M_i. \tag{6.43}$$

Let S_o^* be the optimal solution of $KS - LP$, therefore $S_o^* \geq S^*(M_i)$ as the optimal solution for matching M_i is a feasible solution for LP of knapsack. Therefore,

$$S_o^* \geq S^*(M_i) \geq \frac{S^*}{2} \tag{6.44}$$

We use Algorithm 6 which computes an integer solution S_{OI} to maximum knapsack that is at least half in value of the optimal solution to the LP relaxation (KS-LP). By Lemma 6.6,

$$S_{OI} \geq \frac{S_o^*}{2} \tag{6.45}$$

From equations (6.44) and (6.45), we get:

$$S_{OI} \geq \frac{S_o^*}{2} \geq \frac{S^*(M_i)}{2} \geq \frac{S^*}{4}. \tag{6.46}$$

INDUCTIVE STEP: The restriction of x^* to B' is denoted x^r. By induction hypothesis B' has a solution x' which is integral and satisfies i) $\sum_{e\in E'} s_e x'_e \geq 1/4 \sum_{e\in E} s_e x^r_e$ and ii) $\sum_{e\in E} x_e I_e \leq I'$ where I' is the interference in B'. We use $I(x)$ to mean $\sum_{e\in E} I_e x_e$ and $s(x)$ to mean $\sum_{e\in E} x_e$. We examine the two cases.

- If some $e : x_e = 0$ was removed in B then $I' = I$ and $x^r = x^* - \{x^*_e\}$. Note that by induction hypothesis $I(x') \leq I' = I$, $s(x') \geq s(x^*)/4$ and x' is a solution to B.

- If some $e : x_e = 1$ and edges incident on e was removed in B then $I' = I - I_e$. The sum-rate for the solution to B is $s(x') + s_e \geq 1/4 s(x^*)$. The knapsack constraint is $I(x') + I_e \leq I' + I_e \leq I$.

□

Greedy algorithm for max knapsack:

What remains is to show that the density ordered greedy returns an integer solution with an objective value of at least half the value of the optimal solution to the LP relaxation. For completeness, we provide proof from [112]. The steps in the density ordered greedy Algorithm 6 are as follows:

- We sort the edges of matching M_1 in decreasing order of density where the density of element i is $\frac{s_i}{I_i}$.

- select edges, one by one with the highest density, until no more edges can be selected due to the target interference. Add all the selected edges to G.

- for the first element ($a \notin G$) with the highest density such that choosing it and adding it to G violates the interference constraint, a fractional part of a is selected as below:

$$x_a = \frac{I_c - \sum_{j<a} I_j}{I_a}$$

The optimal solution of LP is

$$S_o^* = S(G) + x_a s_a \tag{6.47}$$

Where $S(G) = \sum_{i \in G} s_i$.

Let $I(G) = \sum_{i \in G} I_i$.

The greedy algorithm is as follows:

Algorithm 6 Greedy algorithm for knapsack

$G \leftarrow \emptyset$
while $I(G \cup argmax_{i \notin G} \frac{s_i}{I_i}) \leq I_c$ **do**
  $G \leftarrow G \cup argmax_{i \notin G} \frac{s_i}{I_i}$
end while
$a \leftarrow argmax_{i \notin G} \frac{s_i}{I_i}$
$G \leftarrow argmax\{S(G), s_a\}$
return G

Lemma 6.6 ([112])**.** *Algorithm 6 computes a solution with the sum-rate at least half the sum-rate of the optimal knapsack LP solution.*

Proof of Lemma 6.6. Let S_o^* be the optimal solution of LP which is given in equation (6.47) and S_{OI} be the IP solution returned by algorithm 6. We have:

$$S_o^* = S(G) + x_a s_a \leq S(G) + s_a \leq 2max\{S(G), s_a\} = 2S_{OI}$$

Therefore,

$$\frac{S_o^*}{2} \leq S_{OI}$$

□

6.5 Contributions

Following are the original contributions in this chapter.

- Branch and cut algorithm.
- Lagrangian relaxation and Theorem 6.1.
- Iterative rounding algorithm (algorithm 5), Lemma 6.5, and Theorem 6.2.

Chapter 7

Computational results

In this chapter, first the computational results of the branch and price algorithm from chapter 5 for the $MTVRP-VW-TW$ are discussed. We also perform a sensitivity analysis on vehicle capacities. The results of the empirical study of the branch and cut algorithm and iterative rounding for the resource allocation problem in chapter 6 are presented and discussed next.

7.1 Experimental Evaluation for $MTVRP-VW-TW$

7.1.1 Mathematical model test

The mathematical model of MTVRP-VW-TW with the objective function of the duration of the longest route is solved with CPLEX and C++, using modified Solomon instances [113]. The algorithm is implemented in C++. The instances are a subset of Solomon set of CVRPTW test problems. They are type of $C1, C2, R1, R2$ with 25 customers. We modify the number of vehicles and capacity in type $C1, R1$ to 25 vehicles and capacity of 200 changed to 15 vehicles, 30 wagons, 4 trips and capacity of 150 for each wagon. For type $C2$, 25 vehicles and capacity of 700 is changed to 15 vehicles, 30 wagons, 4 trips and capacity of 500 for each wagon. For type $R2$ with capacity of 1000 changed to 15 vehicles, 30 wagons, 4 trips and capacity of 700 for each wagon. Table 7.1, 7.2, and 7.3 show the results.

Table 7.1: 25 Customers

Instance	Gap(in %)	CPU Time(Sec)	Objective	Best Bound
C1	66.4	14403	890	450
R1	67.2	14402	665	292
R2	72.5	14382	1033	308

Table 7.2: 50 Customers

Instance	Gap(in %)	CPU Time(Sec)	Objective	Best Bound
C1	89	14339	569	408
R1	76	14420	680	408
R2	68	14059	1065	414

Table 7.3: 100 Customers

Instance	Gap(in %)	CPU Time(Sec)	Objective	Best Bound
C1	66	14402	1137	630
R1	81	14404	3340	630
R2	68	14402	1562	630

All the instances have a large integrality gap and it takes a lot of time to solve the mathematical model.

7.1.2 Branch and price test

Instances

Solomon's (100 customer) instances [113] for Euclidean $VRPTW$ are modified. We use any the first 10 customers of Solomon's instances to test the algorithm. The Euclidean distance between two customer locations determines the travel time for these instances.
The instances that we consider (Solomon, 1987) are of six different types $C1, C2, R1, R2, RC1, RC2$. Each data set has eight to twelve, 100-node problems. Sets $C1$, $C2$ have clustered customers whose time windows were generated based on a known solution. Problem sets $R1$, $R2$ have customers location generated uniformly randomly over a square. Sets $RC1$, $RC2$ have a combination of randomly placed and clustered customers. Sets of type 1 have narrow time windows and small vehicle capacity. Sets of type 2 have large time windows and large

vehicle capacity. Therefore, the optimal solutions of type 2 problems have very few routes and significantly more customers per route.

For these types of instances, our algorithm was able to find optimal solutions for $R1$, $R2$, $RC1$ and $RC2$. It must be noted that the branch and price algorithm is an exact algorithm that could not solve more than 40-50 customers so far. In fact, only few instances for 50 customers. We solved for multi trip problem, even less customers can be handled as the route graph can't be generated for a larger number of customers. The multi trip problem with the varying capacity is studied for the first time in this thesis and it's the first time the branch and price algorithm is given.

To use Solomon's instances, we modify them as follows:

- The solution to $VRPTW$ instances are routes with one trip. We need to adjust them for the multi trip and different capacity use. We use 10 customers of the 25 in the instance. Instances of type 1 that have the capacity of 200 for each vehicle now have a capacity of 50 per wagon. Which will be 100 and 150 in case two or three wagons are attached. Instances of type 2 with the capacity of 1000 per vehicle have modified capacity of 100 per wagon.
- The number of vehicles is limited to 10.
- The number of wagons is 45. This is large enough to attach up to three wagons for each vehicle.
- The maximum number of routes in a tour is limited to three for these instances.

Experimental results

In the following, the result of the branch and price algorithm is presented. The program was implemented in the optimization programming language OPL, using ILOG CPLEX. CEDAR, the Compute Canada cluster was used for experimentation, with a limit of 4 hours

on each solve and a maximum memory requirement of 40GB.

The results are in Table (7.4) where the columns are as follows:

- Instance: the type of the instance. For example $RC102$ is the second instance of type $RC1$.
- Gap: is the gap between LP-relaxation value at the root and the optimal integer value in %.
- CPU time: is calculated as differences between the time recorded at the root and at the end of the algorithm in seconds.
- Obj: the total distance.
- Iter: number of iterations used to solve *RLMP* by CPLEX.
- Cols: number of columns generated during the branch and price algorithm.
- Node: number of nodes explored in the search tree.
- Route: max number of routes used in all tours.
- Tour: number of tours used to visit customers.

Table 7.4: Branch and Price for 10 Customers

Instance	**Gap**	**CPU time (sec)**	**Obj**	**Iter**	**Cols**	**Node**	**Route**	**Tour**
RC102	994.364	869.56	563.562	20	17	3	2	2
RC103	994.364	899.163	563.562	20	17	3	2	2
RC105	994.548	398.842	545.237	18	17	1	2	2

Continued on next page

Table 7.4 – continued from previous page

Instance	Gap	CPU time (sec)	Obj	Iter	Cols	Node	Route	Tour
RC106	993.178	111.626	682.168	13	12	1	2	3
RC107	994.189	202.428	581.069	11	10	1	2	2
RC108	994.121	670.193	587.86	17	14	3	2	2
RC202	992.434	470.885	756.612	5	4	1	1	2
RC203	992.434	470.885	756.612	5	4	1	1	2
RC204	993.722	872.675	627.785	3	2	1	1	2
RC206	794.258	139.613	724.979	8	7	1	2	1
RC207	991.067	446.75	893.255	15	14	1	2	1
RC208	797.627	800.939	388.128	3	2	1	1	1
R102	993.936	102.048	606.438	7	6	1	2	2
R103	993.936	106	606.438	7	6	1	2	2
R108	993.326	1287.93	667.381	16	13	3	2	2
R110	994.023	171.832	597.728	10	9	1	2	2
R111	993.629	2010.17	637.106	14	12	3	2	2
R202	991.33	455.858	867.046	5	4	1	2	1
R203	991.33	458.102	867.046	5	4	1	2	1
R204	993.722	881.522	627.785	3	2	1	1	2
R205	992.355	102.946	764.518	3	2	1	2	1
R206	993.992	581.478	600.814	3	2	1	1	2
R207	993.992	593.218	600.814	3	2	1	1	2

Continued on next page

Table 7.4 – continued from previous page

Instance	Gap	CPU time (sec)	Obj	Iter	Cols	Node	Route	Tour
R208	993.993	612.737	600.683	3	2	1	1	2
R209	992.61	345.444	739.023	8	7	1	1	2
R210	993.672	1068.08	632.814	3	2	1	1	2
R211	996.263	443.173	373.736	2	1	1	1	1

7.1.3 Analysis

The mathematical model was tested and it can take hours, or days to give solutions with gaps around 60% this is why we need to use a method like the branch and price which is an exact method to have the optimal solution of the problem.

During experimentation, we notice that the algorithm explores more nodes if similar capacity (one wagon) is used for all tours. Giving the algorithm the option to check more tours with two wagons and three before starting a new branch makes it faster and explores less number of nodes. The upper bound that the CPLEX provides us with it is a good upper bound close to the optimal solution.

The algorithm solves type 2 of instances where the time windows are wide in horizon more. It is faster and uses fewer iterations. For type C instances branch and price could not find a solution to the time cut-off limit. All the instances listed in Table 7.4 were solved optimally within a time limit of four hours.

7.2 Experimental Evaluation for D2D

7.2.1 Algorithms

We study the performance of four different algorithms in this section. The first one is the default branch and bound solver in Gurobi. The solution space here is partitioned into disjoint subsets. These subsets are represented as the nodes of a branching tree. The algorithm systematically explores this tree's nodes; it evaluates and discards subtrees where a better solution cannot be obtained. The second and the third algorithms are the branch and cut algorithms. In the second algorithm, we study only the cover-cuts described in Section 6.2 and we disable any cuts introduced by Gurobi. Cuts are added at each node in the branch and bound search tree. The third algorithm uses cover cuts and the default cuts that are a part of the Gurobi solver [114]; notably,

- mixed-integer rounding (MIR) cuts: given a constraint, apply integer rounding on the coefficients of integer variables and the right hand side of a constraint.
- generalized upper bounds (GUB) cover cuts: a GUB constraint for a set of binary variables is sum of variables less than or equal to one.
- Chvatal-Gomory strong (StrongCG) cuts: given a constraint, divide all coefficients and right hand side by a positive constant, then truncate coefficients.

Finally, we evaluate the fourth algorithm; the iterative rounding algorithm described in section 6.4.

7.2.2 Data

We focus on three different types of instances; random instances, instances from network simulator, and computationally hard instances. We start with 50 D2D devices and 50 cellular users and go up to 100 D2D devices and 100 cellular users (CU) for each kind. The interference capacity is the average of all interferences across D2D pairs and CUs.

Random Instances: We use the random instances from [5]. Here, the sum rate on edge is uniformly generated in the range [0-50]. The interference is a constant value across all D2D and CU pairs from [1-50].

Real Instances: We use the network instances given in Hassan et al. [115]. The sum rate and the interference values depend on the following parameters. The cell radius is 1000, the D2D pair distance is 15 meters, the base station transmission power is 46 dBm, additive white gaussian noise (AWGN) is -174 dBm, bandwidth is 180 kHz, the carrier frequency is 1.7 GHz, and CU/D2D transmit power to 20 dBm.

Hard Instances: We use the profit-ceiling instance from Pisinger [4] to model the knapsack constraint. The instances are known to be computationally hard for Knapsack. The interferences are randomly generated between 1 and 50. The sum-rate is generated by the formula, $s_e = 3\lceil c_e/3 \rceil$ for every D2D and CU pair e.

7.2.3 Methodology

We use the JuMP library [116] in Julia [117] to program branch and cut algorithms. We use Gurobi as the backend, which can be replaced with any other open-source or commercial solver. To evaluate the benefits of adding cover-cuts (from section 6.2), we initialize the Gurobi solver with the pre-solve reductions turned off. The number of threads is one, no internal cuts are used (MIR, GUC, strong CG), heuristics, and the generation of equivalent models is also turned off at the start. We implement the iterative rounding algorithm slightly differently than presented in Section 6.4. We pick a single variable with the largest value to round each time. It can be proved (we omit the proof here) that Theorem 6.2 still holds for the modified algorithm. Though it needs more LP iterations, this change leads to a smaller violation of the knapsack constraint in practice and the bound in Theorem 6.2 holds.

Some of the models that we solve are large (10,000 or more variables). The basic branch and bound can take hours on these instances, so we use CEDAR, the Compute Canada

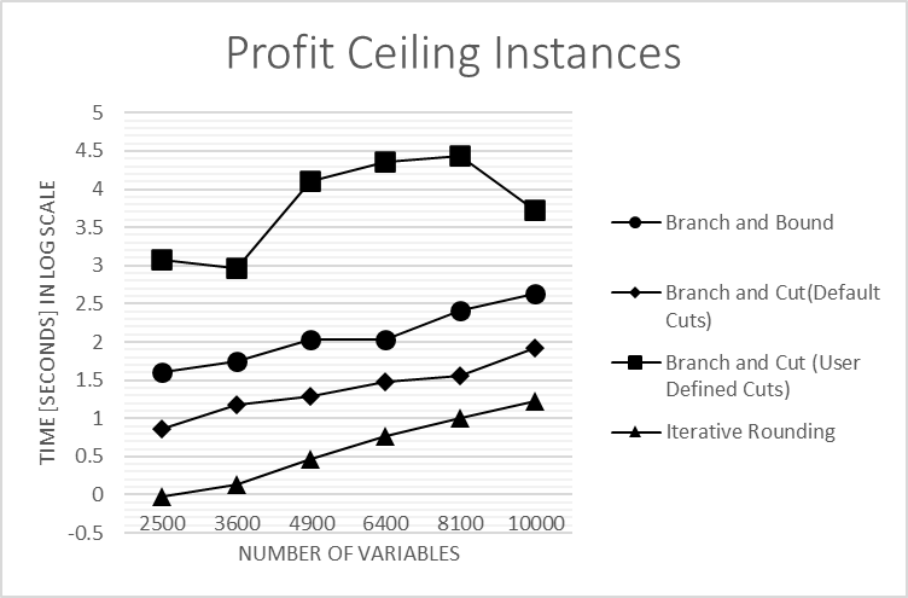


Figure 7.1: Profit Ceiling Instances [4]

cluster, with a limit of 30 hours on each solve and a maximum memory requirement of 12GB.

7.2.4 Results

Figures 7.1, 7.2, and 7.3 show the time it takes to solve the instance optimally for the first three algorithms for each instance and an algorithm as a function of the problem's size. For the iterative rounding algorithm, we get solutions that are very close to the optimal solution (within a small additive constant). The algorithms are branch and bound, branch and cut with built-in cuts from Gurobi (MIR, strong CG, GUB), branch and cut (using the cover cuts discussed in section 6.2), and the iterative rounding algorithms. The branch and cut with built-in cuts from Gurobi is referred to as Branch and Cut, whereas branch and cut with cover cuts is referred to as Branch and Cut (User) in the legend.

We see that branch and cut using cover cuts in section 6.2 generally resulted in the faster discovery of the optimal solution for random and network simulated instances. These instances are easy and no cuts were added. The optimal solution was found at the root node (signifying easy instances). The other algorithms took about the same time compared to

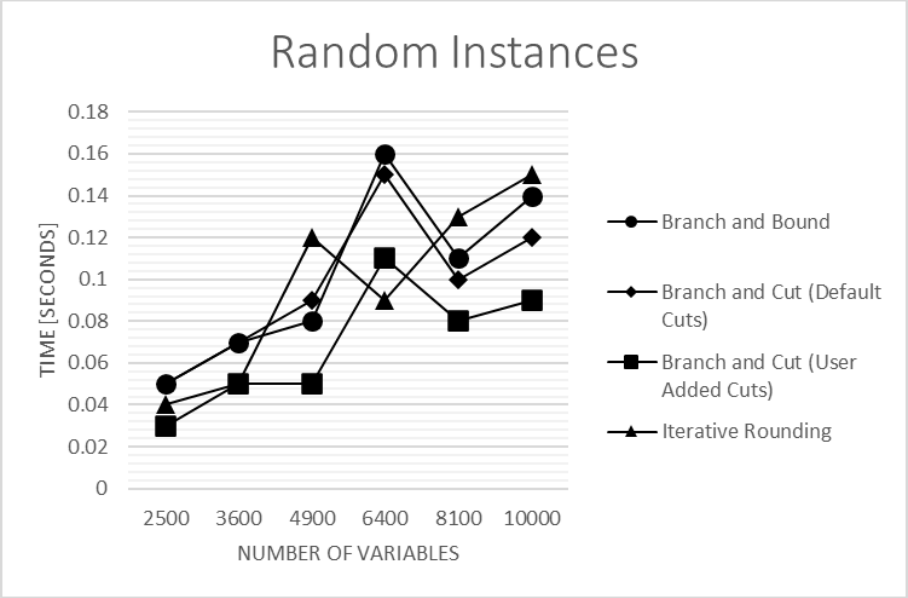


Figure 7.2: Random Instances [5]

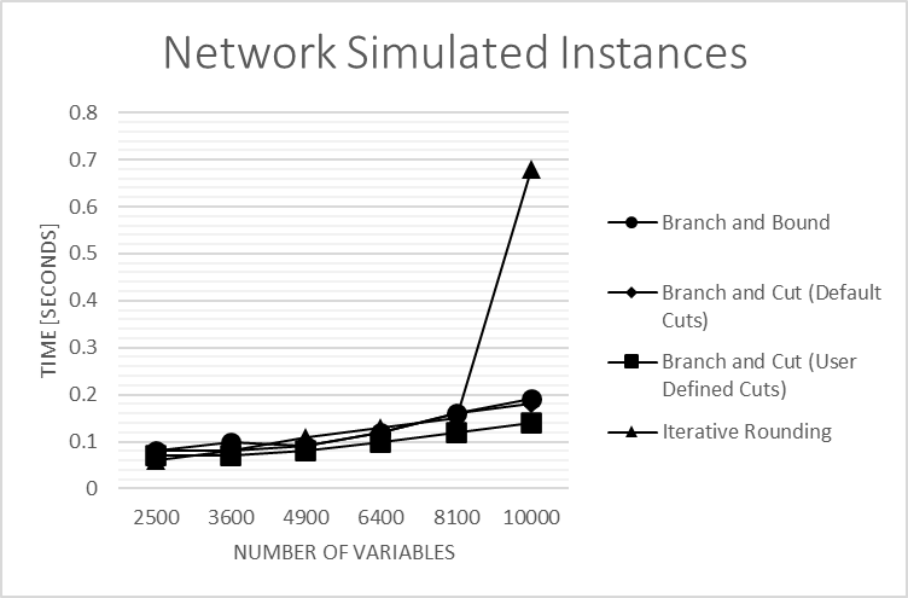


Figure 7.3: Real Networks Instances [6]

Table 7.5: Branch and Cut for Hard Instances

Number of Variables	Gurobi Configuration	Cover Cuts	Nodes Explored	Cutting Planes
2500	1 thread	325	371	User 25
	optimized	1966	3262	User 316 MIR 1 GUB 1
3600	1 thread	160	209	User 4
	optimized	2362	4239	User 111 CC 1 MIR 4 GUB 2
4900	1 thread	327	1570	User 6
	optimized	3773	6940	User 290 MIR 1
6400	1 thread	4763	5816	User 573
	optimized	4241	7992	User 233 MIR 2 StrongCG 1
8100	1 thread	5927	6712	User 668
	optimized	4104	5596	User 32 MIR 1
10000	1 thread	507	712	User 11
	optimized	5617	11492	User 397 GUB 1

the faster externally added cuts. The times for the profit ceiling instances are plotted on the log scale since branch and cut (using cover-cuts) resulted in significantly more time when compared to the other algorithms.

Figure 7.1 shows that for the hard instances from [4], iterative rounding (which finds "near" optimal solutions) took less time compared to the other algorithms.

Gurobi's built-in branch and cut (using MIR, strong CG, GUB) computed the optimal solutions faster than branch and bound with cover-cuts alone on the hard instances.

Branch and cut (using user cover cuts) took a lot of time on hard instances to obtain the optimal solution. We can infer from this that although there was a reduction in the integrality gap, more nodes are explored compared to the other algorithms.

Next, we examine the total number of cuts added during the branch and bound process. We discuss the two cases when only user cover cuts are added and when cover cuts are added along with the built-in Gurobi cuts. For random and network simulated instances, the integrality gap is non-existent. Therefore no additional cuts are needed. For computationally hard instances, the number of cuts added is in Table 7.5. Not all the cover cuts that the solver discovers are added to the nodes due to design implementation level issues in JuMP and Gurobi. The third column, labeled cover cuts, lists the total number of cover cuts discovered during the entire search process, whereas the last column, called Cutting Planes, lists the number of cuts that the solver actually used.

In summary, iterative rounding is fast and effective on computationally hard instances from [4], whereas branch and cut using cover cuts (from Section 6.2 is effective on random instances in [5], and the instances generated from simulated network parameters in [6]. Furthermore, the network instances from [6] are not computationally hard for branch-n-cut framework when modeled as maximization of the sum-rate subject to a cap on the interference. The run-time on network instances is less than 200 milliseconds which is within the limit stated in [118] as the resource allocations have to be computed in real time.

Chapter 8

Conclusion and future work

In this chapter, we conclude with a summary of the results and future directions for research are presented.

8.1 Summary

8.1.1 MTVRP-VW-TW

In this thesis, a new type of *VRP*, multi-trip vehicle routing problem with a variable number of wagons and time windows, is defined. The problem is to serve demands of clients in a specific interval of time (time windows) while vehicles can do multiple trips in a day and the capacity of the vehicle can be set at the beginning of the day by adding up to three wagons for each vehicle. First, a mathematical model of the problem is developed and then we develop a branch and price algorithm to solve the problem.

The approach used to solve the problem, is column generation which is embedded in a branch and bound algorithm. Master problem for the column generation is an LP-relaxation that can be solved by any *LP* solver and the dual variables corresponding to the optimal solution of the master problem are used by the subproblem to determine the best new column to be added to the master problem. As we have three kinds of vehicles to service customers, three different subproblems have to be solved.

Subproblem itself is an elementary shortest path problem which is solved by dynamic

programming. To this aim, we use a label correcting algorithm. As we have multiple trips, the label correcting algorithm is applied twice, the first time for generating all non-dominated routes and using these routes as new nodes to create a route graph. The route graph is created from all these non-dominated routes and the time windows for each of these routes as well as consideration of a few other rules to satisfy time windows and demand of customers. Again, the label correcting algorithm is used on the route graph to determine all non-dominated tours. Given the capacity of vehicles, we have three route graphs and all of these tours are examined in the following order, the first subproblem is checked and if there is any new tour with a negative reduced cost is added to the master problem, otherwise, the second and third subproblem are checked respectively.

Column generation is used at each node of the search tree. If there is a better upper bound, the node is kept, otherwise, it is pruned. We used two different rules for the branching, one on the number of the vehicles and another one on the arcs.
We implemented the branch and price algorithm for $MTVRP-VW-TW$ on some of Solomon's instances to show the effectiveness of the algorithm. It can compute us the optimal integer solution for limited customers.

8.1.2 D2D resource allocation

In this thesis, we studied the D2D resource allocation model which maximizes the sum-rate while capping the interference. We gave the first branch and cut algorithm and the first iterative rounding algorithm. Our technique works for the case when the objective is to minimize the interference subject providing a guaranteed sum-rate for D2D communications as well. We performed a detailed empirical evaluation of the branch and cut algorithm on instances that simulate small cell, and on computationally hard instances. Our results show that the branch and cut algorithm is efficient in practice for simulated network instances. For computationally hard instances the gain is small. We consider the linear program-

ming relaxation with cover-cuts and give an iterative rounding algorithm. We evaluate the performance of the iterative rounding algorithm on simulated network instances and computationally hard instances. The results show that iterative rounding computes near optimal solutions and is fast in practice.

We prove that the iterative rounding method provides a near optimal solution, in the sense the objective function value is at least a quarter and the knapsack constraint violation is less than the interference capacity (both are multiplicative factors).

8.2 Future work

We are interested in combining the column generation approach with metaheuristics to develop a faster solution for $MTVRP-VW-TW$. We will extend the model for split delivery with time windows, multiple wagons, and multiple trips per vehicle. The model can be extended to a multi-objective problem as well. In addition to these two types of the VRPs, the model can be extended to other variants of the VRPs. We can develop heuristic, exact and approximate approaches to solve the model.

Bibliography

[1] N. Azi, M. Gendreau, and J.-Y. Potvin, "An exact algorithm for a single-vehicle routing problem with time windows and multiple routes," *European journal of operational research*, vol. 178, no. 3, pp. 755–766, 2007.

[2] L.-M. Rousseau, "Introduction to column generation and hybrid methods for homecare routing," https://school.a4cp.org/summer2017/slidedecks/Introduction-to-Column-Generation-and-hybrid-methods-for-Homecare-Routing.pdf, 2017.

[3] "Milp algorithms: branch-and-bound and branch-and-cut."

[4] D. Pisinger, "Where are the hard knapsack problems?" *Computers & Operations Research*, vol. 32, no. 9, pp. 2271–2284, 2005.

[5] P. R. Saha, S. Choudhury, and D. R. Gaur, "Interference minimization for device-to-device communications: A combinatorial approach," in *2019 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, 2019, pp. 1–6.

[6] Y. Hassan, F. Hussain, S. Hossen, S. Choudhury, and M. M. Alam, "Interference minimization in D2D communication underlaying cellular networks," *IEEE Access*, vol. 5, pp. 22 471–22 484, 2017.

[7] J. H. R. G. B. Dantzig, "The truck dispatching problem," *Management Science*, vol. 6, no. 1, pp. 80–91, 1959.

[8] T.Caric and H.Gold, *Vehicle Routing problem*. In Tech, 2008.

[9] P. Gandotra and R. K. Jha, "Device-to-device communication in cellular networks: A survey," *Journal of Network and Computer Applications*, vol. 71, pp. 99–117, 2016.

[10] B. Fleischmann, "The vehicle routing problem with multiple use of vehicles," *Fachbereich Wirtschaftswissenschaften, Universität Hamburg*, 1990.

[11] É. D. Taillard, G. Laporte, and M. Gendreau, "Vehicle routeing with multiple use of vehicles," *Journal of the Operational research society*, vol. 47, no. 8, pp. 1065–1070, 1996.

[12] J. Brandao and A. Mercer, "A tabu search algorithm for the multi-trip vehicle routing and scheduling problem," *European journal of operational research*, vol. 100, no. 1, pp. 180–191, 1997.

[13] J. C. S. Brandão and A. Mercer, "The multi-trip vehicle routing problem," *Journal of the Operational research society*, vol. 49, no. 8, pp. 799–805, 1998.

[14] G. Nagy and S. Salhi, "The many-to-many location-routing problem," *Top*, vol. 6, no. 2, pp. 261–275, 1998.

[15] A. M. Campbell and M. Savelsbergh, "Efficient insertion heuristics for vehicle routing and scheduling problems," *Transportation science*, vol. 38, no. 3, pp. 369–378, 2004.

[16] R. J. Petch and S. Salhi, "A multi-phase constructive heuristic for the vehicle routing problem with multiple trips," *Discrete Applied Mathematics*, vol. 133, no. 1-3, pp. 69–92, 2003.

[17] S. Salhi and R. Petch, "A ga based heuristic for the vehicle routing problem with multiple trips," *Journal of Mathematical Modelling and Algorithms*, vol. 6, no. 4, pp. 591–613, 2007.

[18] A. Olivera and O. Viera, "Adaptive memory programming for the vehicle routing problem with multiple trips," *Computers & Operations Research*, vol. 34, no. 1, pp. 28–47, 2007.

[19] D. Cattaruzza, N. Absi, D. Feillet, and D. Vigo, "An iterated local search for the multi-commodity multi-trip vehicle routing problem with time windows," *Computers & Operations Research*, vol. 51, pp. 257–267, 2014.

[20] N. Wassan, N. Wassan, G. Nagy, and S. Salhi, "The multiple trip vehicle routing problem with backhauls: Formulation and a two-level variable neighbourhood search," *Computers & Operations Research*, vol. 78, pp. 454–467, 2017.

[21] E. B. Tirkolaee, A. A. R. Hosseinabadi, M. Soltani, A. K. Sangaiah, and J. Wang, "A hybrid genetic algorithm for multi-trip green capacitated arc routing problem in the scope of urban services," *Sustainability*, vol. 10, no. 5, p. 1366, 2018.

[22] E. Babaee Tirkolaee, P. Abbasian, M. Soltani, and S. A. Ghaffarian, "Developing an applied algorithm for multi-trip vehicle routing problem with time windows in urban waste collection: A case study," *Waste Management & Research*, vol. 37, no. 1_suppl, pp. 4–13, 2019.

[23] D. Chen, S. Pan, Q. Chen, and J. Liu, "Vehicle routing problem of contactless joint distribution service during covid-19 pandemic," *Transportation Research Interdisciplinary Perspectives*, vol. 8, p. 100233, 2020.

[24] L. Zhen, C. Ma, K. Wang, L. Xiao, and W. Zhang, "Multi-depot multi-trip vehicle routing problem with time windows and release dates," *Transportation Research Part E: Logistics and Transportation Review*, vol. 135, p. 101866, 2020.

[25] M. Desrochers, J. Desrosiers, and M. Solomon, "A new optimization algorithm for the vehicle routing problem with time windows," *Operations research*, vol. 40, no. 2, pp. 342–354, 1992.

[26] K. Halse, "Modeling and solving complex vehicle routing problems," Ph.D. dissertation, Technical University of Denmark, 1992.

[27] N. Kohl and O. B. G. Madsen, "An optimization algorithm for the vehicle routing problem with time windows based on Lagrangian relaxation," *Oper. Res.*, vol. 45, no. 3, pp. 395–406, 1997. [Online]. Available: https://doi.org/10.1287/opre.45.3.395

[28] N. Kohl, J. Desrosiers, O. B. Madsen, M. M. Solomon, and F. Soumis, "2-path cuts for the vehicle routing problem with time windows," *Transportation Science*, vol. 33, no. 1, pp. 101–116, 1999.

[29] J. Larsen, *Parallelization of the vehicle routing problem with time windows*. Citeseer, 1999.

[30] W. Cook and J. L. Rich, "A parallel cutting-plane algorithm for the vehicle routing problem with time windows," Tech. Rep., 1999.

[31] B. Kallehauge, J. Larsen, and O. B. Madsen, "Lagrangean duality and non-differentiable optimization applied on routing with time windows-experimental results," *Relatório interno IMM-REP-2000-8, Department of Mathematical Modeling, Technical University of Denmark, Lyngby, Dinamarca*, 2000.

[32] A. Chabrier, E. Danna, and C. Le Pape, "Coopération entre génération de colonnes avec tournées sans cycle et recherche locale appliquée au routage de véhicules," in *Journées nationales sur la résolution pratique de problèmes NP-complets*, 2002, pp. 83–97.

[33] D. Feillet, P. Dejax, M. Gendreau, and C. Gueguen, "An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems," *Networks: An International Journal*, vol. 44, no. 3, pp. 216–229, 2004.

[34] L.-M. Rousseau, M. Gendreau, G. Pesant, and F. Focacci, "Solving vrptws with constraint programming based column generation," *Annals of Operations Research*, vol. 130, no. 1, pp. 199–216, 2004.

[35] A. Chabrier, "Vehicle routing problem with elementary shortest path based column generation," *Computers & Operations Research*, vol. 33, no. 10, pp. 2972–2990, 2006.

[36] S. Irnich and D. Villeneuve, "The shortest path problem with k-cycle elimination (k 3): Improving a branch and price algorithm for the vrptw," *INFORMS Journal of Computing*, vol. 10, 2003.

[37] E. Danna and C. Le Pape, "Accelerating branch-and-price with local search: A case study on the vehicle routing problem with time windows, chap. 3 in column generation, g. desaulniers, j. desrosiers, and mm solomon," 2005.

[38] F. Hernandez, D. Feillet, R. Giroudeau, and O. Naud, "An exact method to solve the multitrip vehicle routing problem with time windows and limited duration," *TRISTAN*, vol. 7, pp. 366–369, 2010.

[39] R. Macedo, C. Alves, J. V. de Carvalho, F. Clautiaux, and S. Hanafi, "Solving the vehicle routing problem with time windows and multiple routes exactly using a pseudo-polynomial model," *European Journal of Operational Research*, vol. 214, no. 3, pp. 536–545, 2011.

[40] P. Munari and R. Morabito, "A branch-price-and-cut algorithm for the vehicle routing problem with time windows and multiple deliverymen," *Top*, vol. 26, no. 3, pp. 437–464, 2018.

[41] T. I. Faiz, C. Vogiatzis, and M. Noor-E-Alam, "A column generation algorithm for vehicle scheduling and routing problems," *Computers & Industrial Engineering*, vol. 130, pp. 222–236, 2019.

[42] N. Azi, M. Gendreau, and J.-Y. Potvin, "An exact algorithm for a vehicle routing problem with time windows and multiple use of vehicles," *European Journal of Operational Research*, vol. 202, no. 3, pp. 756–763, 2010.

[43] M. P. Seixas and A. B. Mendes, "A branch-and-price approach for a multi-trip vehicle routing problem with time windows and driver work hours," in *Congreso Latino-Iberoamericano de Investigación Operativa. Simpósio Brasileiro de Pesquisa Operacional*, 2012, pp. 3469–3480.

[44] S. Pace, A. Turky, I. Moser, and A. Aleti, "Distributing fibre boards: a practical application of the heterogeneous fleet vehicle routing problem with time windows and three-dimensional loading constraints," *Procedia Computer Science*, vol. 51, pp. 2257–2266, 2015.

[45] M. Gendreau, G. Laporte, C. Musaraganyi, and É. D. Taillard, "A tabu search heuristic for the heterogeneous fleet vehicle routing problem," *Computers & Operations Research*, vol. 26, no. 12, pp. 1153–1173, 1999.

[46] M. Berghida and A. Boukra, "Quantum inspired algorithm for a vrp with heterogeneous fleet mixed backhauls and time windows," *International Journal of Applied Metaheuristic Computing (IJAMC)*, vol. 7, no. 4, pp. 18–38, 2016.

[47] P. H. V. Penna, A. Subramanian, L. S. Ochi, T. Vidal, and C. Prins, "A hybrid heuristic for a broad class of vehicle routing problems with heterogeneous fleet," *Annals of Operations Research*, vol. 273, no. 1, pp. 5–74, 2019.

[48] G. Macrina, L. D. P. Pugliese, F. Guerriero, and G. Laporte, "The green mixed fleet vehicle routing problem with partial battery recharging and time windows," *Computers & Operations Research*, vol. 101, pp. 183–199, 2019.

[49] Ç. Koç, T. Bektaş, O. Jabali, and G. Laporte, "Thirty years of heterogeneous vehicle routing," *European Journal of Operational Research*, vol. 249, no. 1, pp. 1–21, 2016.

[50] E. Choi and D.-W. Tcha, "A column generation approach to the heterogeneous fleet vehicle routing problem," *Computers & Operations Research*, vol. 34, no. 7, pp. 2080–2095, 2007.

[51] A. Pessoa, E. Uchoa, and M. Poggi de Aragão, "A robust branch-cut-and-price algorithm for the heterogeneous fleet vehicle routing problem," *Networks: An International Journal*, vol. 54, no. 4, pp. 167–177, 2009.

[52] A. Bettinelli, A. Ceselli, and G. Righini, "A branch-and-cut-and-price algorithm for the multi-depot heterogeneous vehicle routing problem with time windows," *Transportation Research Part C: Emerging Technologies*, vol. 19, no. 5, pp. 723–740, 2011.

[53] K. Sundar and S. Rathinam, "An exact algorithm for a heterogeneous, multiple depot, multiple traveling salesman problem," in *2015 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 2015, pp. 366–371.

[54] A. Bettinelli, A. Ceselli, and G. Righini, "A branch-and-price algorithm for the multi-depot heterogeneous-fleet pickup and delivery problem with soft time windows," *Mathematical Programming Computation*, vol. 6, no. 2, pp. 171–197, 2014.

[55] A. Pessoa, R. Sadykov, and E. Uchoa, "Enhanced branch-cut-and-price algorithm for heterogeneous fleet vehicle routing problems," *European Journal of Operational Research*, vol. 270, no. 2, pp. 530–543, 2018.

[56] G. Fodor, E. Dahlman, G. Mildh, S. Parkvall, N. Reider, G. Miklós, and Z. Turányi, "Design aspects of network assisted device-to-device communications," *IEEE Communications Magazine*, vol. 50, no. 3, pp. 170–177, 2012.

[57] K. Doppler, M. Rinne, C. Wijting, C. B. Ribeiro, and K. Hugl, "Device-to-device communication as an underlay to LTE-advanced networks," *IEEE Communications Magazine*, vol. 47, no. 12, pp. 42–49, December 2009.

[58] M. T. Islam, A.-E. M. Taha, and S. Akl, "Reducing the complexity of resource allocation for underlaying device-to-device communications," in *2015 International Wireless Communications and Mobile Computing Conference (IWCMC)*. IEEE, 2015, pp. 61–66.

[59] M. T. Islam, A.-E. M. Taha, S. Akl, and S. Choudhury, "A two-phase auction-based fair resource allocation for underlaying D2D communications," in *2016 IEEE International Conference on Communications (ICC)*. IEEE, 2016, pp. 1–6.

[60] F. Hussain, M. Y. Hassan, M. S. Hossen, and S. Choudhury, "System capacity maximization with efficient resource allocation algorithms in d2d communication," *IEEE Access*, vol. 6, pp. 32 409–32 424, 2018.

[61] R. Zhang, X. Cheng, L. Yang, and B. Jiao, "Interference graph-based resource allocation (ingra) for D2D communications underlaying cellular networks," *IEEE Transactions on Vehicular Technology*, vol. 64, no. 8, pp. 3844–3850, 2014.

[62] P. Janis, V. Koivunen, C. Ribeiro, J. Korhonen, K. Doppler, and K. Hugl, "Interference-Aware Resource Allocation for Device-to-Device Radio Underlaying Cellular Networks," in *VTC Spring 2009 - IEEE 69th Vehicular Technology Conference*, April 2009, pp. 1–5.

[63] C. Xu, L. Song, Z. Han, Q. Zhao, X. Wang, and B. Jiao, "Interference-aware resource allocation for device-to-device communications as an underlay using sequential second price auction," in *IEEE International Conference on Communications (ICC), Ottawa, Canada*, June 2012, pp. 445–449.

[64] C. Xu, L. Song, Z. Han, Q. Zhao, X. Wang, X. Cheng, and B. Jiao, "Efficiency Resource Allocation for Device-to-Device Underlay Communication Systems: A Reverse Iterative Combinatorial Auction Based Approach," *IEEE Journal on Selected Areas in Communications*, vol. 31, no. 9, pp. 348–358, September 2013.

[65] D. Feng, L. Lu, Y. Yuan-Wu, G. Y. Li, G. Feng, and S. Li, "Device-to-Device Communications Underlaying Cellular Networks," *IEEE Transactions on Communications*, vol. 61, no. 8, pp. 3541–3551, August 2013.

[66] M. M. Solomon and J. Desrosiers, "Survey paper—time window constrained routing and scheduling problems," *Transportation science*, vol. 22, no. 1, pp. 1–13, 1988.

[67] G. Laporte, Y. Nobert, and S. Taillefer, "Solving a family of multi-depot vehicle routing and location-routing problems," *Transportation science*, vol. 22, no. 3, pp. 161–172, 1988.

[68] V. Pillac, M. Gendreau, C. Guéret, and A. L. Medaglia, "A review of dynamic vehicle routing problems," *European Journal of Operational Research*, vol. 225, no. 1, pp. 1–11, 2013.

[69] H. Psaraftis, "A dynamic-programming solutionnto the single vehicle many-to-many immediate request dial-a-ride problem," *Transportation Science*, vol. 14, no. 2, pp. 130–154, 1980.

[70] L. S. Carlo S.Sartori, "A study on the pickup and delivery problem with time windows: Matheuristics and new instances," *Computers Operations Research*, vol. 124, pp. 130–154, 2020.

[71] "Vehicle routing with pickups and deliveries," https://developers.google.com/optimization/routing/pickup_delivery.

[72] S. Onut, M. Kamber, and G. Altay, "A heterogeneous fleet vehicle routing model for solving the lpg distribution problem: A case study," in *Journal of Physics: Conference Series*, vol. 490, no. 1. IOP Publishing, 2014, p. 012043.

[73] G. Laporte, "The vehicle routing problem: An overview of exact and approximate algorithms," *European journal of operational research*, vol. 59, no. 3, pp. 345–358, 1992.

[74] M. Dell'Amico, G. Righini, and M. Salani, "A branch-and-price approach to the vehicle routing problem with simultaneous distribution and collection," *Transportation science*, vol. 40, no. 2, pp. 235–247, 2006.

[75] R. Fukasawa, H. Longo, J. Lysgaard, M. P. d. Aragão, M. Reis, E. Uchoa, and R. F. Werneck, "Robust branch-and-cut-and-price for the capacitated vehicle routing problem," *Mathematical programming*, vol. 106, no. 3, pp. 491–511, 2006.

[76] J. Lysgaard, A. N. Letchford, and R. W. Eglese, "A new branch-and-cut algorithm for the capacitated vehicle routing problem," *Mathematical Programming*, vol. 100, no. 2, pp. 423–445, 2004.

[77] M. Fischetti, P. Toth, and D. Vigo, "A branch-and-bound algorithm for the capacitated vehicle routing problem on directed graphs," *Operations Research*, vol. 42, no. 5, pp. 846–859, 1994.

[78] G. Laporte and Y. Nobert, "A branch and bound algorithm for the capacitated vehicle routing problem," *Operations-Research-Spektrum*, vol. 5, no. 2, pp. 77–85, 1983.

[79] C. Novoa and R. Storer, "An approximate dynamic programming approach for the vehicle routing problem with stochastic demands," *European journal of operational research*, vol. 196, no. 2, pp. 509–515, 2009.

[80] C. Archetti, N. Bianchessi, and M. G. Speranza, "A column generation approach for the split delivery vehicle routing problem," *Networks*, vol. 58, no. 4, pp. 241–254, 2011.

[81] B. D. Backer, V. Furnon, P. Shaw, P. Kilby, and P. Prosser, "Solving vehicle routing problems using constraint programming and metaheuristics," *Journal of heuristics*, vol. 6, no. 4, pp. 501–523, 2000.

[82] J.-F. Cordeau, M. Gendreau, A. Hertz, G. Laporte, and J.-S. Sormany, "New heuristics for the vehicle routing problem," *Logistics systems: design and optimization*, pp. 279–297, 2005.

[83] C. Groër, B. Golden, and E. Wasil, "A library of local search heuristics for the vehicle routing problem," *Mathematical Programming Computation*, vol. 2, no. 2, pp. 79–101, 2010.

[84] M. M. Silva, A. Subramanian, and L. S. Ochi, "An iterated local search heuristic for the split delivery vehicle routing problem," *Computers & Operations Research*, vol. 53, pp. 234–249, 2015.

[85] Y. Kuo, "Using simulated annealing to minimize fuel consumption for the time-dependent vehicle routing problem," *Computers & Industrial Engineering*, vol. 59, no. 1, pp. 157–165, 2010.

[86] A. S. Alfa, S. S. Heragu, and M. Chen, "A 3-opt based simulated annealing algorithm for vehicle routing problems," *Computers & Industrial Engineering*, vol. 21, no. 1-4, pp. 635–639, 1991.

[87] O. Bräysy, W. Dullaert, G. Hasle, D. Mester, and M. Gendreau, "An effective multirestart deterministic annealing metaheuristic for the fleet size and mix vehicle-routing problem with time windows," *Transportation Science*, vol. 42, no. 3, pp. 371–386, 2008.

[88] G. Barbarosoglu and D. Ozgur, "A tabu search algorithm for the vehicle routing problem," *Computers & Operations Research*, vol. 26, no. 3, pp. 255–270, 1999.

[89] S. C. Ho and D. Haugland, "A tabu search heuristic for the vehicle routing problem with time windows and split deliveries," *Computers & Operations Research*, vol. 31, no. 12, pp. 1947–1964, 2004.

[90] P. Toth and D. Vigo, "The granular tabu search and its application to the vehicle-routing problem," *Informs Journal on computing*, vol. 15, no. 4, pp. 333–346, 2003.

[91] B. M. Baker and M. Ayechew, "A genetic algorithm for the vehicle routing problem," *Computers & Operations Research*, vol. 30, no. 5, pp. 787–800, 2003.

[92] T. Vidal, T. G. Crainic, M. Gendreau, N. Lahrichi, and W. Rei, "A hybrid genetic algorithm for multidepot and periodic vehicle routing problems," *Operations Research*, vol. 60, no. 3, pp. 611–624, 2012.

[93] B. Bullnheimer, R. F. Hartl, and C. Strauss, "An improved ant system algorithm for thevehicle routing problem," *Annals of operations research*, vol. 89, pp. 319–328, 1999.

[94] ——, "Applying the ant system to the vehicle routing problem," in *Meta-heuristics*. Springer, 1999, pp. 285–296.

[95] D. Tuzun, M. A. Magent, and L. I. Burke, "Selection of vehicle routing heuristic using neural networks," *International Transactions in Operational Research*, vol. 4, no. 3, pp. 211–221, 1997.

[96] M. Gendreau, G. Laporte, and J.-Y. Potvin, "Metaheuristics for the capacitated vrp," in *The vehicle routing problem*. SIAM, 2002, pp. 129–154.

[97] M. DESROCHERS and F. SOUM, "A generalized permanent labelling algorithm for the shortest path problem with time windows," *INFOR*, vol. 206, pp. 191–212, 1988a.

[98] S. Dasgupta, C. H. Papadimitriou, and U. V. Vazirani, *Algorithms*. McGraw-Hill Higher Education New York, 2008.

[99] G. V. LENT, "Using column generation for the time dependent vehicle routing problem with soft time windows and stochastic travel times," 2018.

[100] B. Kallehauge, J. Larsen, O. B. Madsen, and M. M. Solomon, *Vehicle Routing Problem with Time Windows*. Springer US, 2005, pp. 67–98. [Online]. Available: https://doi.org/10.1007/0-387-25486-2_3

[101] L.-M. Rousseau, M. Gendreau, and G. Pesant, "Solving small vrptws with constraint programming based column generation," *Proceedings of CPAIOR'02*, 2002.

[102] F. Liberatore, G. Righini, and M. Salani, "A column generation algorithm for the vehicle routing problem with soft time windows," *4OR*, vol. 9, pp. 49–82, 03 2011.

[103] S.Boyd and J.Mattingley, "Branch and bound methods," 2007.

[104] T. Ibaraki, "Theoretical comparisons of search strategies in branch-and-bound algorithms," *International Journal of Computer & Information Sciences*, vol. 5, no. 4, pp. 315–344, 1976.

[105] M. Yang, K. Jung, S. Lim, and J. Shin, "Development of device-to-device communication in lte-advanced system," in *2014 IEEE International Conference on Consumer Electronics (ICCE)*. IEEE, 2014, pp. 448–449.

[106] M. Islam, "Radio resource allocation for device-to-device communications underlaying cellular networks," Ph.D. dissertation, Queens University, Canada, 2016.

[107] A. Iyer, C. Rosenberg, and A. Karnik, "What is the right model for wireless channel interference?" *IEEE Transactions on Wireless Communications*, vol. 8, no. 5, pp. 2662–2671, 2009.

[108] L. CACCETTA and S. P. HILL, "Branch and cut methods for network optimization," *Mathematical and Computer Modelling*, vol. 33, pp. 517–532, 2001).

[109] K. Kaparis and A. Letchford, *Cover Inequalities*, 02 2011.

[110] J. E. Hopcroft and R. M. Karp, "An $n^{5/2}$ Algorithm for Maximum Matchings in Bipartite Graphs," *SIAM Journal on Computing*, vol. 2, no. 4, p. 225–231, 1971.

[111] L. C. Lau, R. Ravi, and M. Singh, *Iterative methods in combinatorial optimization*. Cambridge University Press, 2011, vol. 46.

[112] Y. Singer, "Advanced optimization," https://people.seas.harvard.edu/~yaron/AM221-S16/lecture_notes/AM221_lecture17.pdf, 2016.

[113] "Solomon's benchmark instances," https://www.sintef.no/projectweb/top/vrptw/100-customers/.

[114] "Gurobi optimization," https://www.gurobi.com/documentation/9.5/refman/index.html.

[115] Y. Hassan, F. Hussain, S. Hossen, S. Choudhury, and M. M. Alam, "Interference minimization in d2d communication underlaying cellular networks," *IEEE Access*, vol. 15, no. 10, 2016.

[116] I. Dunning, J. Huchette, and M. Lubin, "JuMP: A modeling language for mathematical optimization," *SIAM review*, vol. 59, no. 2, pp. 295–320, 2017.

[117] J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah, "Julia: A fresh approach to numerical computing," *SIAM review*, vol. 59, no. 1, pp. 65–98, 2017.

[118] M. T. Islam, A.-E. M. Taha, S. Akl, and S. Choudhury, "A local search algorithm for resource allocation for underlaying device-to-device communications," in *2015 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2015, pp. 1–6.