ALMA MATER STUDIORUM - UNIVERSITÀ DI BOLOGNA

Consorzio Spinner

Dottorato di Ricerca in Automatica e Ricerca Operativa

Settore Concorsuale di afferenza: 01/A6 - RICERCA OPERATIVA

Settore Scientifico disciplinare: Mat/09 - RICERCA OPERATIVA

Ciclo XXVII

Models and Algorithms for the Optimization of Real-World Routing and Logistics Problems

Stefano Novellani

Coordinatore Dottorato
Prof. Daniele Vigo

Supervisore
Prof. Manuel Iori

Supervisore
Prof. Daniele Vigo

Esame finale anno 2015

Contents

Acknowledgments

Ringrazio le persone che hanno viaggiato al mio fianco in questa parte di percorso. Ringrazio la famiglia, gli amici, i colleghi. In maggior parte per il sostegno, i sorrisi, le leggerezze, il filosofeggiare, la qualità dei discorsi. Ringrazio il mio supervisore Manuel Iori, per la disponibilità, il tempo, la pazienza. Ma soporattutto per i suggerimenti, le esortazioni, i proficui scambi d'idee, l'avermi insegnato -bene o male- a fare questo lavoro. Ringrazio Mauro Dell'Amico, Daniele Vigo, Eleni Hadjiconstantinou, Thomas Stützle e Anand Subramanian per i consigli, le collaborazioni, le opportunità, tutte innumerevoli.

List of Figures

List of Tables

Preface

Logistics involves planning, managing, and organizing the flows of goods from the point of origin to the point of destination in order to meet some requirements. Logistics and transportation aspects are very important and represent a relevant costs for producing and shipping companies, but also for public administration and private citizens. The optimization of resources and the improvement in the organization of operations is crucial for all branches of logistics, from the operation management to the transportation. As we will have the chance to see in this work, optimization techniques, models, and algorithms represent important methods to solve the always new and more complex problems arising in different segments of logistics. Many operation management and transportation problems are related to the optimization class of problems called *Vehicle Routing Problems* (VRPs). In this work, we consider several real-world deterministic and stochastic problems that are included in the wide class of the VRPs, and we solve them by means of exact and heuristic methods. In Chapter 1 we briefly describe the VRPs, their characteristics, the most used solving techniques, and we highlight some subclasses of the VRPs related to our work. In the following chapters we treat three classes of real-world routing and logistics problems. In Part II we deal with one of the most important tactical problems that arises in the managing of the bike sharing systems, that is the Bike sharing Rebalancing Problem (BRP). In Chapter 2 we formally define the BRP, for which we supply a set of formulations and inequalities included in branch-and-cut algorithms. In Chapter 3 we tackle the BRP with a destroy and repair algorithm including novel properties to reduce the complexity of the components of the algorithm. In the same chapter, we also solve the generalization of the BRP where maximum duration constraints for the routes are considered, the so-called one-commodity Pickup and Delivery VRP (1-PDVRP), for which we propose new inequalities included in a branch-and-cut framework. We also show the adaptation of the destroy and repair algorithm to solve the 1-PDVRP. In Chapter 4 we define the stochastic BRP (SBRP), the version of the BRP with stochastic demands. We provide different models for the SBRP and solve them with L-Shaped methods and branch-and-cut algorithms. Moreover, we present heuristic algorithms for the SBRP that account for the negative and pos-

itive correlations between the nodes in the solution construction. In Part III we propose models and algorithms for real-world earthwork optimization problems. In Chapter 5 we present a novel two-phase earthwork optimization model for highway construction, considering many real-world features for the first time. In Chapter 6 we present a Decision Support System (DSS) for highway construction developed for the Autostrada Pedemontana Lombarda highway project in collaboration with one of the major construction companies in Europe. The DSS includes easy-to-use graphical and optimization components and it can be applied to several highway construction projects. In Part IV we consider the optimization of the 3D Printing (3DP) process. In Chapter 7 we describe the 3DP process and we highlight several optimization issues in 3DP. Among those, we define the problem related to the tool path definition in the 3DP process, the 3D Routing Problem (3DRP), which is a generalization of the arc routing problem. We present an ILP model and several heuristic algorithms to solve the 3DRP.

Part I

Introduction

Chapter 1

Introduction to Routing Problems

We can say, without exaggeration, that one of the most studied class of problems in optimization and logistics is the class of the *Vehicle Routing Problems* (VRPs). In this work, many facets of the VRPs are considered: we propose models and algorithms for real-world new routing and logistics problems. We thus report in this chapter a brief description of the most classic characteristics of the family of the VRPs, the most used solving techniques, and some highlights on the problem related to those tackled in the following of this PhD thesis.

1.1 The Vehicle Routing Problems

The vehicle routing problem has been introduced for the first time by Dantzig and Ramsel [12] in 1959 for a real-world application for gasoline delivery to gas stations and it was then called *truck dispatching problem*. By citing Irnish, Toth, and Vigo [26], the family of VRPs can be summarized as "determine a set of vehicle routes to perform all (or some) transportation requests with the given vehicle fleet at minimum cost; in particular, decide which vehicle handles which request in which sequence so that all vehicles routes can be feasibly executed".

The VRPs are normally defined on graphs that are made of nodes and arcs (and/or edges) that model the road network. Thus, the family of VRPs can be divided into problems where the requests are set on the nodes (the *node routing problems*, that are the one normally referred as to VRPs) and problems where the requests are set on the arcs (and/or edges), the so-called *Arc Routing Problems* (ARPs) (see, e.g., Dror [18], Wøhlk [50], and Corberán and Laporte [9]). In this thesis we tackle versions of both problems, in particular we consider VRPs (node routing problems) in Chapters 2–6, and a generalization of the ARPs in Chapter 7. Also for this reason we discuss

now the most common version of the VRPs and the most used techniques to solve them.

The relevance of the VRPs and the wide space they have in the international academic and industrial community in all its facets and variants is mostly due to its practical and economical importance. Indeed, the algorithmic results yielded in more than 50 years of academic and industrial research on the VRPs allowed considerable cost and time savings, improvement in planning and organizing activities, and, in practice, allowed to produce software tools, decision support systems, phone applications, etc. to respond to real-world problems. In the following chapters we present many algorithms and a decision support system made to respond to real-world problems of public administrations, private citizens, and private companies.

To give an idea of the wideness of the family of VRPs, in the following we catalog the characteristics of the most important versions of the VRPs treated in the literature. We follow the classification delineated in Toth and Vigo [48].

1.1.1 The Network Characteristics

When considering the characteristics of the network we can refer to problems where the tasks to be performed are identified on nodes (node routing problems), on arcs (or edges) (ARPs), or on both as in the *general routing problem* (see, e.g., Orloff [34]). In the case of node routing problem we have traveling costs or times on the links (arcs or edges), that can be symmetric (in case of edges) or asymmetric (in case of arcs). The traveling cost between two nodes is considered, most of the times, as the shortest distance between them, even if in real-world problems costs and times strongly depend on the hour of the day in which an arc (edge) is traveled. These types of data, such as traveling times, can become known during the traveling operations and this is the case of the dynamic VRP. If those data can be described by a random variable with a given distribution probability, then it is the case of the stochastic VRP.

1.1.2 The Type of Transportation Requests

The type of requests can be very diverse, in the following are reported some characteristics of the requests.

- Delivery and collection: different type of demands representing deliveries and collections can be given, we can divide them into: distribution of goods from a depot to customers, which is the classic Capacitated Vehicle Routing Problem (CVRP) (see, e.g., Toth and Vigo [48]); collection of goods from customers to the depots (see, e.g., Sankaran and Ugabe[43], Gouden et al. [24]); and pickup and delivery, where both distribution and collection are required (see, e.g., Berbeglia et al. [3]).

- Visit and vehicle scheduling: in this case nor delivery or collection is consider as the request, but simply the fact of visiting a customer or a location is required (see, e.g., Desrosier et al. [15]).
- Point-to-point transportation: we are considering here pickup and delivery problems where the origin and destination of a customer are paired (see, e.g., Parragh [35]).
- Repeated requests: sometimes it occurs that a certain set of requests can be repeated in the time horizon, these are called repeated requests. This is the case of the periodic VRP (see, e.g., Cordeau et al [10]) and of the inventory routing problems (see, e.g., Bertazzi and Speranza [4]).
- Split and non-split services: if splitting services is allowed, requests can be satisfied with more than one visit of one or more vehicle. We thus have the split delivery VRP (see, e.g. Dror and Trudeau [19]).
- Combined shipment and multi-modal service: this is the case where the services can be performed by making use of consolidation centers, hubs, and cross-docking (see, e.g., Perboli et al. [36]).
- Routing with profits and service selection: in this case a complete service to all the request is not possible due to the constraints, thus we must decide a set of request to satisfy such that the maximum profit is gained (see, e.g. Fillet et al. [20]).
- Dynamic and stochastic routing: it is the case where uncertainty is a relevant factor. In such a case the problem can be seen as dynamic, if part of the data are revealed during the operations, or stochastic, if the uncertain data are described with random variables and having a probability distribution.

1.1.3 Intra-route Constraints

We refer here to constraints whose feasibility can be checked on single routes. These occur in the following cases:

- Loading constraints: loading constraints can depend on the maximum capacity of the vehicle (as in the CVRP), but they can depend also on weight, space, and volume (see, e.g., Iori et al. [25]), the order of loading (see, e.g., Cordeau et al. [11]), but also considering the compatibility of different goods (see, e.g., Xu et al. [51]), and the use of several compartments on a vehicle (see, e.g., Derigis et al. [13]).
- Route length: route length constraint considers not to exceed a maximum duration in time or distance for the routes (see, e.g., Christophides et al. [7]).

- Multiple use of vehicles: contemplating that vehicles can perform more than one route over the planning horizon (see, e.g., Taillard et al. [46]).
- Time windows and scheduling aspects: constraints related to schedules are very important in many VRP variants, accounting for travel, service, and waiting times together with the time windows in which services must be performed (see, e.g., Dasaulniers et al. [14]).

1.1.4 Fleet Characteristics

We can consider the use of vehicles with different characteristics such as dimension, costs, speed, the possibility of loading just a set of goods, the possibility of visiting just a set of customers, the possibility of using transshipment (see, e.g., Drexel [17]). The vehicles (homogeneous or not) can start or end their journeys in different depots (see, e.g., Renard et al. [40]), and the routes can also be opened (vehicles can start or end their journeys in visiting nodes) (see, e.g., Li et al. [30]).

1.1.5 Inter-route Constraints

Inter-route constraints are constraints whose feasibility, and thus the feasibility of the solution, depend on how the routes are combined. In this class we have balancing constraints between the maximum and the minimum route duration (see, e.g., Bodin et al. [6]). Another constraint can be related to the synchronization issues among the routes (see, e.g., Drexel [16]).

1.1.6 The Objective Function

The classic objective function for the VRPs is to minimize the traveling costs expressed in distances or times of the routes, but other single or multi-objectives can be evaluated. For example the customer satisfaction, the min-max objective (minimizing the objective of the longest route), minimizing the number of vehicles used, and using non-linear cost functions.

1.2 Solution Methods for the VRPs

Since the very first introduction in 1959, many solution methods have been used and developed to solve and tackle the VRPs. These solving methodologies can be divided into two classes of optimization algorithms: the exact algorithms, that are proven to converge to the optimal solution, if it exist, and the heuristic algorithms, that can obtain very good solutions in short computational times. In the following we report some classic methodologies and algorithms used to solve VRPs, in accordance with the Toth and Vigo [48], in particular see, e.g, [44], [37], and [29]. The classification is far from

being exhaustive, for the interested reader we refer to Toth and Vigo [47] and [48].

1.2.1 Exact Algorithms

To give an idea of the exact algorithms for the VRPs, we report here some of the classic exact algorithms to solve the Capacitated VRP (CVRP).

- Branch-and-bound. Branch-and-Bound (B&B) algorithms are enumerating algorithms that decompose the problem in subproblems easier to solve creating a branching tree. The branching is performed on the variables and informations on bounds are used in order not to explore the entire branching tree to obtain the optimal solution. The B&B algorithms for the CVRP are based on the relaxation of some set of constraints of the classic 2-index formulation for the CVRP (see, e.g. Toth and Vigo [48]). These relaxations result in a Assignment Problems or Shortest Spanning Tree problems on whose solutions is performed the tree search. In addition, different branching techniques and reduction strategies are applied, and the introduction of cutting planes can be attempted.

- Column Generation. The Column Generation (CG) methods are based on the classic set partitioning and set covering formulations (see, e.g., Balinski and Quandt [1]). THe CG method starts from a small subset of routes, each one represented by a variables, and solves the linear relaxation of the corresponding reduced model deriving the optimal dual variables associated with the constraints. Thanks to the dual information, the CG problem (called pricing) tries to find the route, not in the subset, that has the most negative reduced cost or proved that no such a route exists. In this latter case the solution of the linear relaxation on the subset of routes is the solution of the complete model and the algorithm terminates. Otherwise the route with smaller reduced cost that the pricing algorithm found is added to the subset of routes and a new iteration is performed. Recently, the CG methods have been extended to branch-and-cut-and-price algorithms, which are hybrid methods between the CG and the branch-and-cut algorithms, that are presented in the following.

- Branch-and-cut. The Branch-and-Cut (B&C) algorithms involve running a B&B algorithm and using cutting planes to tighten a relaxation of the problem considered. Given the solution of the relaxation, if this is feasible for the relaxed constraints then the solution found is optimal, otherwise we have to separate the obtained solution from the optimal one by identifying violated inequalities. To do so, exact and heuristic separation algorithms can be used. We then include the generated constraint to the relaxed model and reiterate (see, e.g., Laporte

et al. [28]). The branching strategies are normally applied on variables with fractional values.

1.2.2 Heuristic Algorithms

As reported by Toth and Vigo [48], very sophisticated exact algorithms have been developed, but only instances with around 100 customers can be solved to optimality and in high computing times. On the other hand, real-life problems and instances need good solutions in short times: this is the reason why, since the first appearance of the VRPs, heuristic algorithms have been designed and used to solve them. Without intending to furnish with an exhaustive classification of the heuristic algorithms, in the following we give and overview on the constructive heuristics, the two-phase heuristics, the improvement heuristics, and the metaheuritics.

- Constructive Heuristics. Constructive heuristics are algorithms that build step by step a solution, and are normally used to provide a starting solution for the improvement heuristics. The constructive heuristics can be divided into sequential constructive heuristics and parallel constructive heuristics. In the first case the solution is built one route at the time, while in the second case more routes are build in parallel. The most known constructive heuristic for the VRP is the savings algorithm (see, e.g., Clarke and Wright [8]), that builds routes made of single vertices and then tries to merge them by using a savings criterion. This algorithms can be used in both parallel and sequential way.

- Two-phase Heuristics. The two-phase heuristics can be divided into route-first cluster-second algorithms and cluster-first route-second algorithms (see, e.g., Gillet and Miller [22]). In the first type one route is built and then separated into several routes in order to obtain a feasible solution, in the second case the nodes are grouped in clusters and then the routes are formed with respect to the clusters.

- Improvement Algorithms. Improvement algorithms are algorithms that perform inter-route and intra-route moves on a solution to obtain better quality solutions. Some examples can be the swap between two nodes on a solution, the k-opt exchange (see, e.g., Lin [31]), etc. These moves can also be seen as destroy and repair operators that destroy and repair the solution. In the adaptive large neighborhood search (see, e.g., Ropke and Pisinger [42]) destroy and repair moves are selected randomly in a roulette wheel mechanism obtaining very good results.

- Metaheuristic Algorithms. The metaheuristics for the VRPs can be classified into local search methods and population based heuristics.

- Local Search Methods. Local search methods explore the solution space by moving, at each iteration, from one solution to another in its neighborhood trying to escape from local optima. Just to name some metaheuritic algorithms based on local searches we have: simmulated annealing (see, e.g., Kirckpatrick et al. [27]), tabu search (see, e.g., Glover [23]), iterated local search (see, e.g., Lourenço et al. [32]), variable neighborhood search (see, e.g., Mladenović and Hansen [33]).
- Population Based Heuristics. Population based heuristics starts from a population of solutions that evolves during the iterations of the algorithm, for example by combining the population solutions to obtain better ones. These algorithms normally take the inspiration from natural concepts. Just to name some population based algorithms, we have: ant colony optimization (see, e.g., Reiman et al. [39]), genetic algorithms (see, e.g., Prins [38]), scatter search, and path relinking (see, e.g., Glover [23] and Resende et al. [41]).

1.2.3 Hybridization and Unified Methods

In the last years, two research directions have emerged: a large use of hybrid methods and the research for unified algorithms. In the first case a hybridization of different techniques and paradigms such as local search, population based search, and exact algorithms has occurred (see, e.g., Subramanian et al. [45]). In the second case, the creation of unified methods attempts at assessing many different VRPs, moving from problem-specific methods to more flexible approaches that could solve a wide range of problems (see, e.g., Vidal et al. [49]).

1.3 Pickup and Delivery Problems

In this thesis we widely treat Pickup and Delivery Problems (PDPs), in particular we consider PDPs for good transportation. For this reason we give a short recall to the PDPs and e brief classification following the one delineated by Battarra et al. [2]. PDPs is a class of VRPs where commodities have to be transported from different origins to different destinations. We can have:

1.3.1 Many-to-Many

Many-to-Many (M-M) problems represent the class in which each commodity have multiple origins and destinations and in which any location can be the origin and destination of multiple commodities. The problem of redis-

tributing bikes in a bike sharing system, that we also tackle in the following, lays in this class.

1.3.2 One-to-Many-to-One

One-to-Many-to-one (1-M-1) problems represent the class in which some commodities must be delivered from a depot to the customers and some others must be collected from customers and carried back to the depot.

1.3.3 One-to-One

The one-to-one (1-1) class of problems is the one where each commodity and request is given with an origin-destination pair.

1.4 Stochastic Vehicle Routing Problems

A particular class of problems, part of the wider class of VRPs, has received the interest of many researcher recently: the class of Stochastic Vehicle Routing Problems (SVRPs). In the SVRPs not all informations are deterministic, and hence some level of uncertainty has to be accounted for. The uncertainty can derive from diverse events end on one or more set of data: we can thus have stochastic demands, stochastic customers, and stochastic traveling times/costs (see, e.g., Gendreau et al. [21]).

One of the most common way of modeling and solving stochastic problems is Stochastic Programming (SP). In SP the stochastic parameters are formulated as random variables given with a probability distribution. The SP models are obtained by determining the informational process. Based on this process the SP models are defined in two or more stages depending to when random variables become known. This problems are normally modeled in two-stage formulation with recourse, represented by a first stage decisions to be made a priori, and the recourse decisions to be made after the realization of the stochastic variables, i.e. when the information is available. The recourse decisions are related to the outcome of the stochastic parameters and can lead to an increase of the costs depending on the recourse function, that gives the average recourse cost after a first stage decision, or impose the first stage decisions to be changed. Both linear and integer SP models are normally solved by using the L-Shaped method and the Integer L-Shaped method (see, e.g., Birge and Louveaux [5]).

Another method to model the stochastic problems is to include probabilistic constraints, which means that some constraints can be satisfied within a certain probability. We refer the reader to Birge and Louveaux [5] for more details.

Bibliography

[1] M. L. Balinski and R. E. Quandt. On an integer program for a delivery problem. *Operations Research*, 12:300–304, 1964.

[2] M. Battarra, J.-F. Cordeau, and M. Iori. *Vehicle Routing: Problems, Methods, and Applications*, Chapter 6: Pickup-and-Delivery Problems for Goods Transportation. In [48], 2014.

[3] G. Berbeglia, J.-F. Cordeau, I. Gribkovskaia, and G. Laporte. Static pickup and delivery problems: a classification scheme and survey. *Top*, 15:1–31, 2007.

[4] L. Bertazzi and M. G. Speranza. Inventory routing problems: an introduction. *EURO Journal on Transportation and Logistics*, 1:307–326, 2012.

[5] J. R. Birge and F. Louveaux. *Introduction to stochastic programming*. Springer Science & Business Media, 2011.

[6] L. Bodin, V. Maniezzo, and A. Mingozzi. Street routing and scheduling problems. In *Handbook of Transportation Science*, pages 395–432. Springer, 1999.

[7] N. Christofides, A. Mingozzi, P. Toth, and C. Sandi. The travelling salesman problem, combinatorial optimisation, 1979.

[8] G. Clarke and J. W. Wright. Scheduling of vehicles from a central depot to a number of delivery points. *Operations research*, 12:568–581, 1964.

[9] Á. Corberán and G. Laporte. (eds.) *Arc Routing: Problems, Methods, and Applications*. SIAM, 2014.

[10] J.-F. Cordeau, M. Gendreau, and G. Laporte. A tabu search heuristic for periodic and multi-depot vehicle routing problems. *Networks*, 105–119, 1997.

[11] J.-F. Cordeau, M. Iori, G. Laporte, and J.-J. Salazar González. A branch-and-cut algorithm for the pickup and delivery traveling salesman problem with lifo loading. *Networks*, 55:46–59, 2010.

[12] G. B. Dantzig and J. H. Ramser. The truck dispatching problem. *Management science*, 6:80–91, 1959.

[13] U. Derigs, J. Gottlieb, J. Kalkoff, M. Piesche, F. Rothlauf, and U. Vogel. Vehicle routing with compartments: applications, modelling and heuristics. *OR spectrum*, 33:885–914, 2011.

[14] G. Desaulniers, O. B- G. Madsen, and S. Ropke. *Vehicle Routing: Problems, Methods, and Applications*, Chapter 5: The Vehicle Routing Problem with Time Windows. In [48], 2014.

[15] J. Desrosiers, Y. Dumas, M. M. Solomon, and F. Soumis. Time constrained routing and scheduling. *Handbooks in operations research and management science*, 8:35–139, 1995.

[16] M. Drexl. Synchronization in vehicle routing-a survey of vrps with multiple synchronization constraints. *Transportation Science*, 46:297–316, 2012.

[17] M. Drexl. Applications of the vehicle routing problem with trailers and transshipments. *European Journal of Operational Research*, 227:275–283, 2013.

[18] M. Dror. *Arc routing: theory, solutions, and applications*. Springer Science & Business Media, 2000.

[19] M. Dror and P. Trudeau. Savings by split delivery routing. *Transportation Science*, 141–145, 1989.

[20] D. Feillet, P. Dejax, and M. Gendreau. Traveling salesman problems with profits. *Transportation science*, 39:188–205, 2005.

[21] M. Gendreau, O. Jabali, and W. Rei. *Vehicle Routing: Problems, Methods, and Applications*, Chapter 8: Stochastic Vehicle Routing Problems. In [48], 2014.

[22] B. E. Gillett and L. R. Miller. A heuristic algorithm for the vehicle-dispatch problem. *Operations research*, 22:340–349, 1974.

[23] F. Glover. Heuristics for integer programming using surrogate constraints. *Decision Sciences*, 8:156–166, 1977.

[24] B. L Golden, A. A. Assad, and E. A. Wasil. Routing vehicles in the real world: applications in the solid waste, beverage, food, dairy, and newspaper industries. In *The vehicle routing problem*, 245–286. SIAM, 2001.

[25] M. Iori, J.-J. Salazar-González, and D. Vigo. An exact approach for the vehicle routing problem with two-dimensional loading constraints. *Transportation Science*, 41:253–264, 2007.

[26] S. Irnich, P. Toth, and D. Vigo. *Vehicle Routing: Problems, Methods, and Applications*, Chapter 1: The Family of Vehicle Routing Problems. In [48], 2014.

[27] S. Kirkpatrick, C.D. Gelatt Jr, M.P. Vecchi, and A. McCoy. Optimization by simulated annealing. *Science*, 220:671–679, 1983.

[28] G. Laporte, Y. Nobert, and M. Desrochers. Optimal routing under capacity and distance restrictions. *Operations research*, 33:1050–1073, 1985.

[29] G. Laporte, S. Ropke, and T. Vidal. *Vehicle Routing: Problems, Methods, and Applications*, Chapter 3: Heuristics for the Vehicle Routing Problems. In [48], 2014.

[30] F. Li, B. Golden, and E. Wasil. The open vehicle routing problem: Algorithms, large-scale test problems, and computational results. *Computers & Operations Research*, 34:2918–2930, 2007.

[31] S. Lin. Computer solutions of the traveling salesman problem. *Bell System Technical Journal, The*, 44:2245–2269, 1965.

[32] H. R. Lourenço, O. C. Martin, and T. Stützle. *Iterated local search*. Springer, 2003.

[33] N. Mladenović and P. Hansen. Variable neighborhood search. *Computers & Operations Research*, 24:1097–1100, 1997.

[34] C. S. Orloff. Routing a fleet of m vehicles to/from a central facility. *Networks*, 4:147–162, 1974.

[35] S. N. Parragh, K. F. Doerner, and R. F. Hartl. A survey on pickup and delivery problems. *Journal für Betriebswirtschaft*, 58:21–51, 2008.

[36] G. Perboli, R. Tadei, and D. Vigo. The two-echelon capacitated vehicle routing problem: Models and math-based heuristics. *Transportation Science*, 45:364–380, 2011.

[37] M. Poggi and E. Uchoa. *Vehicle Routing: Problems, Methods, and Applications*, Chapter 3: New Exact Algorithms for the Capacitated Vehicle Routing Problems. In [48], 2014.

[38] C. Prins. A simple and effective evolutionary algorithm for the vehicle routing problem. *Computers & Operations Research*, 31:1985–2002, 2004.

[39] M. Reimann, K. Doerner, and R. F. Hartl. D-ants: Savings based ants divide and conquer the vehicle routing problem. *Computers & Operations Research*, 31:563–591, 2004.

[40] J. Renaud, G. Laporte, and F. F. Boctor. A tabu search heuristic for the multi-depot vehicle routing problem. *Computers & Operations Research*, 23:229–235, 1996.

[41] M. G. C. Resende, C. C. Ribeiro, F. Glover, and R. Martí. Scatter search and path-relinking: Fundamentals, advances, and applications. In *Handbook of metaheuristics*, 87–107. Springer, 2010.

[42] S. Ropke and D. Pisinger. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation science*, 40:455–472, 2006.

[43] J. K. Sankaran and R. R. Ubgade. Routing tankers for dairy milk pickup. *Interfaces*, 24:59–66, 1994.

[44] F. Semet, P. Toth, and D. Vigo. *Vehicle Routing: Problems, Methods, and Applications*, Chapter 2: Classical Exact Algorithms for the Capacitated Vehicle Routing Problems. In [48], 2014.

[45] A. Subramanian, E. Uchoa, and L. S. Ochi. A hybrid algorithm for a class of vehicle routing problems. *Computers & Operations Research*, 40:2519–2531, 2013.

[46] E. D. Taillard, G. Laporte, and M. Gendreau. Vehicle routeing with multiple use of vehicles. *Journal of the Operational research society*, 1065–1070, 1996.

[47] P. Toth and D. Vigo. (eds.) *The vehicle routing problem*. SIAM, 2001.

[48] P. Toth and D. Vigo. (eds.) *Vehicle Routing: Problems, Methods, and Applications*. SIAM, 2014.

[49] T. Vidal, T. G. Crainic, M. Gendreau, and C. Prins. A unified solution framework for multi-attribute vehicle routing problems. *European Journal of Operational Research*, 234:658–673, 2014.

[50] S. Wøhlk. A decade of capacitated arc routing. In *The vehicle routing problem: latest advances and new challenges*, 29–48. Springer, 2008.

[51] H. Xu, Z.-L. Chen, S. Rajagopal, and S. Arunapuram. Solving a practical pickup and delivery problem. *Transportation science*, 37:347–364, 2003.

Part II

Models and Algorithms for the Bike sharing Rebalancing Problems

Chapter 2

The Bike Sharing Rebalancing Problem: Mathematical Formulations and Benchmark Instances

[1]

Bike sharing systems offer a mobility service whereby public bicycles, located at different stations across an urban area, are available for shared use. These systems contribute towards obtaining a more sustainable mobility and decreasing traffic and pollution caused by car transportation. Since the first bike sharing system was installed in Amsterdam in 1965, the number of such applications has increased remarkably so that hundreds of systems are now operating all over the world.

In a bike sharing system, users can take a bicycle from a station, use it to perform a journey and then leave it at a station, not necessarily the same one of departure. This behaviour typically leads to a situation in which some stations become full and others are empty. Hence, a balanced system requires the redistribution of bicycles among stations.

In this chapter, we address the Bike sharing Rebalancing Problem (BRP), in which a fleet of capacitated vehicles is employed in order to re-distribute the bikes with the objective of minimizing total cost. This can be viewed as a special one-commodity pickup-and-delivery capacitated vehicle routing problem. We present four mixed integer linear programming formulations of this problem. It is worth noting that the proposed formulations include an exponential number of constraints, hence, tailor-made branch-and-cut algorithms are developed in order to solve them.

[1] The results of this chapter appear in: M. Dell'Amico, E. Hadjicostantinou, M. Iori, and S. Novellani. The bike sharing rebalancing problem: Mathematical formulations and benchmark instances. *Omega*, 45:7–19, 2014.

The mathematical formulations of the BRP were first computationally tested using data obtained for the city of Reggio Emilia, Italy. Our computational study was then extended to include bike sharing systems from other parts of the world. The information derived from the study was used to build a set of benchmark instances for the BRP which we made publicly available on the web. Extensive experimentation of the branch-and-cut algorithms presented in this chapter was carried out and an interesting computational comparison of the proposed mathematical formulations is reported. Finally, several insights on the computational difficulty of the problem are highlighted.

Keywords: Integer programming, Routing, Traveling salesman, Vehicle scheduling.

2.1 Introduction

Bike sharing systems offer a mobility service in which public bicycles are available for shared use. These bicycles are located at stations that are displayed across an urban area. The users of the system can take a bicycle from a station, use it for a journey, leave it into a station (not necessarily the one of departure), and then pay according to the time of usage.

These systems are an important instrument used by public administrations to obtain a more sustainable mobility, decrease traffic and pollution caused by car transportation, and solve the so-called last mile problem related to proximity travels. From the first bike sharing system installed in Amsterdam in 1965, their number increased in the following years to reach, in 2014, more than 700 systems only in Europe, see, e.g., DeMaio [10] and project OBIS [23]. In North America the implementation of bike sharing systems started only in 2008, see Pucher et al. [26], but as far as we know it already counts more than 20 operating systems. In the rest of the world the number of systems is rising at a very high rate, as discussed, e.g., by Shaheen et al. [28].

Stations are made of different slots, each of which hosts a single bicycle. In modern systems, stations are connected to the Internet and display in real time the occupation status of each slot. In this way users can easily check where it is possible to pick up or drop a bicycle. The usage of the system is monitored continuously, and the collected information is used to improve the level of service.

Operating bike sharing systems has a cost that may vary greatly (depending on the system itself, the population density, the service area and the fleet size), with a consistent impact on the budget of the public administration. The setup costs for installing the system include, among others, the cost of purchasing the bikes, the slots, and the stations, and the cost of

the back-end system used to operate the equipment, see, e.g., DeMaio [10]. The daily operating costs include maintenance, insurance, possibly website hosting and electricity, and, most important, the cost due to the redistribution of bikes among the stations. Indeed, at the end of a day some stations are typically full and others are empty.

A commonly adopted rule for rebalancing is to keep each station only partly occupied, i.e., there should always be in a station some slots occupied by bicycles, to allow users to pick them up, and some free slots, to allow users to drop a bicycle at the end of their journey. Let us suppose that a desired level of occupation is present in the early morning in a given bike station, then the number of bikes may change drastically during the day from the desired level because of the users' travel behavior. This happens typically in cities characterized by a hilly territory, see, e.g., Kaltenbrunner et al. [18], where users take a bike from a station located at the top of a hill, leave it at the bottom and then take the journey back with different means of transportation. It is also common for cities located in flat areas, where some stations have large inflows or outflows at different times of the day. In the next section we report the results of the analysis of the system at the city of Reggio Emilia (Italy) used over a period of seven months.

Repositioning is usually done by means of capacitated vehicles based at a central depot, that pick up bicycles from stations where the level of occupation is too high and deliver them to stations where the level is too low. Usually a buffer of bicycles is kept at the depot, and used to allow a more flexible redistribution. The resulting optimization problem of deciding how to route the vehicles so as to perform the redistribution at minimum cost is known in the literature as the *Bike sharing Rebalancing Problem* (BRP), and has recently attracted the interest of many researchers and practitioners in the area. It can be modeled either as a *dynamic* or a *static* optimization problem. In the static version, a snapshot of the level of occupation at the stations is taken and then used to plan the redistribution. In the dynamic version, the real-time usage of the system is taken into account, and the redistribution plan is possibly updated as soon as the information required to make decisions is revealed over time.

Usually, static rebalancing is associated with a redistribution process that is performed during the night, when the system is kept closed or the demand is very low, whereas dynamic rebalancing is associated to redistributions operated during the day, when demand may be high. In the real-world case that we studied in detail the redistribution is performed during the night, and hence we focus on the static version of the problem.

In this chapter we provide several contributions. In Section 2.2 we briefly present the real-world case study that we conducted at the city of Reggio Emilia, by analyzing the travel flows, the users behavior and the resulting levels of occupation at the stations. In Section 4.3 we formally describe the BRP and discuss the related literature. In Section 4.4 we propose four

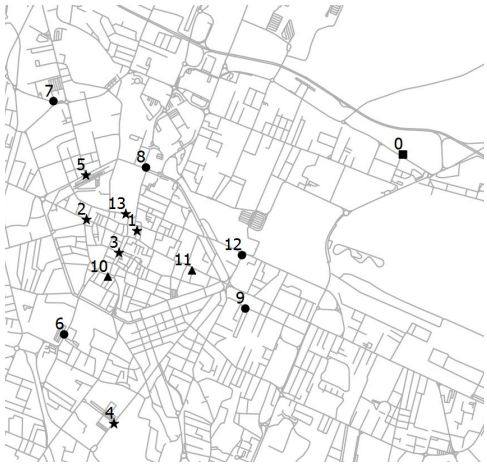


Figure 2.1: The bike sharing system of Reggio Emilia. The depot is depicted by 0. Stars, circles and triangles represent stations of the first, second and third group, respectively.

Mixed Integer Linear Programming (MILP) formulations to model the problem. All these formulations involve an exponential number of constraints, so Section 3.5.2 presents the branch-and-cut algorithms that we implemented to solve them. We present a large set of benchmark instances in Section 2.6, obtained by analyzing the usage of several bike systems around the world; we make these instances publicly available on the Internet. Extensive computational results of the branch-and-cut algorithms are reported in Section 2.7. Finally, conclusions are given in Section 2.8 and future research directions are discussed.

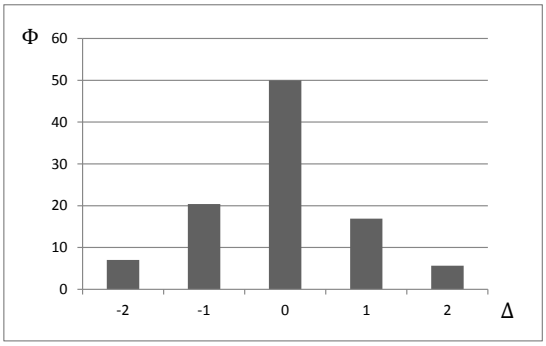
2.2 Data analysis of a real-world case

The first real-world case that we studied is the bike sharing system of Reggio Emilia, a city of around 170 thousand inhabitants located in a very flat area in northern Italy. The system, which is depicted in Figure 2.1, is quite small and now counts one depot (indicated by 0 in the figure), 13 stations and about 100 bicycles. It operates all day but is kept closed during the night, which is quite common for bike sharing systems in small/medium cities. The redistribution of the bicycles is carried out during the night, by means of a single vehicle that visits each station exactly once.

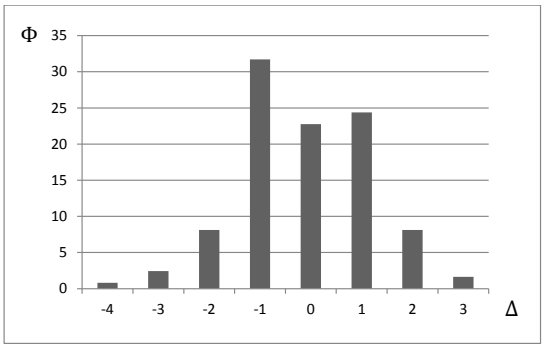
The data associated with a seven-month usage of the system was provided to us by the municipality of the city. It contains the list of journeys performed by the users in the considered period, including time and station of departure and arrival of each journey. For each station we evaluated the net flow of bicycles on a daily basis, computed as the difference between

the inflow and the outflow. This gives the difference between the bicycles available at the beginning of the day and those left at the end of the day in each station. We then plotted the distribution of the net flow over the period, see the distribution graphs in Figure 2.2. The x-axis gives the difference Δ between arrivals and departures in a station per day, and the y-axis gives the percentage of times Φ (frequence of occurrence) this number appears throughout the period that we studied. We mostly found normal-like distributions, as the one depicted in Figure 2.2-(a) for station 4, but also a bimodal one for station 5, see Figure 2.2-(b). In both cases, for the majority of the days over the observation the stations ended up with a number of bikes different from that available at the beginning of the day, and this supports the choice of performing rebalancing operations.

Furthermore, by analyzing the net flow per hour of all the stations we have been able to determine the diverse variability in usage by the customers. We could consequently divide the stations into three groups, as shown in



(a)



(b)

Figure 2.2: Net flow of bicycles per day, following (a) a normal-like distribution in station 4 and (b) a bimodal distribution in station 5 (Δ = difference between arrivals and departures, Φ = frequence of occurrence).

Figure 2.3. In this figure, the x-axis represents the hour τ of the day, and the y-axis states the cumulative number ν of bikes arriving into or departing from the station within a seven-months period. More specifically:

1. The first group, see Figure 2.3-(a), has a peak of incoming bikes between 7 and 9 am, and a smaller one between 1 and 3 pm. The peaks of outgoing bikes occur between 12 am and 2 pm and between 4 and 6 pm. These stations are all situated in the city center, with the exception of one that is located near the hospital (stations 1, 2, 3, 4, 5 and 13). This usage fits well with the behavior of users that work in the city center but live outside;

2. The second group is complementary to the first one, see Figure 2.3-(b). These stations (6, 7, 8, 9 and 12) are in the so-called park-and-ride areas, where users can leave their cars and continue their ride on a public bike;

3. The third group, see Figure 2.3-(c), contains stations where arrivals and departures follow a similar pattern during the day. These stations (10 and 11) are located close to places, such as the train station, that are used all day long.

More enhanced studies on bike traveling habits are out of the scope of this chapter, so the interested reader is referred to the relevant literature. Vogel and Mattfeld [33] present a business model for bike sharing systems. They model the repositioning activities with the help of a system dynamics approach, that they solve with a simulation tool. They conclude that "more effort spent on repositioning leads to a better corporate performance in terms of satisfied customers". Kaltenbrunner et al. [18] provide an analysis of human mobility data in the urban area of Barcelona (Spain) using the amount of available bikes in the stations of the local bike sharing program. By using data sampled from the operator's website, they detect temporal and geographic mobility patterns within the city. Lin and Yang [19] address the strategic planning of public bicycle sharing systems. These authors propose a model that attempts to determine the number and locations of bike stations, the users' travel paths, and the network structure of bike trips connecting the stations.

The determination of the most suitable bike trips has been studied by Souffriau et al. [29], for a problem arising in East Flanders (Belgium), and motivated by the problem that a cyclist deals with when looking for a nice route of a certain length. They propose a mathematical model and a metaheuristic. The metaheuristic obtained good computational results and was then embedded into a web-based cycle route planner. We also mention that a similar problem was faced in a different context by Boctor et al. [7], who studied the assignment of trips to vehicles in petrol stations replenishment problems.

2.3 Problem description and previous work

We are given a complete digraph $G = (V, A)$, where the set of vertices $V = \{0, 1, \ldots, n\}$ is partitioned into the depot, vertex 0, and the stations, vertices $\{1, 2, \ldots, n\}$. Each station i has a request q_i, which can be either positive or negative. If $q_i \geq 0$ then i is a *pickup* vertex, where q_i bikes should be removed; if $q_i < 0$ then i is a *delivery* vertex, where q_i bikes should be supplied, for $i \in V \setminus \{0\}$. The bikes removed from pickup vertices can either go to a delivery vertex or back to the depot. Bikes supplied to delivery vertices can either come from the depot of from pickup vertices. A fleet of m identical vehicles of capacity Q is available at the depot to transport the bikes. A traveling cost c_{ij} is associated with each arc $(i, j) \in A$.

The BRP involves determining how to drive at most m vehicles through the graph, with the aim of minimizing the total cost and ensuring that the following constraints are not violated: (i) each vehicle performs a route that starts and ends at the depot, (ii) each vehicle starts from the depot empty or with some initial load (i.e., with a number of bikes that vary from 0 to Q), (iii) each station is visited exactly once and its request is completely fulfilled by the vehicle visiting it, and (iv) the sum of requests of the visited stations plus the initial load is never negative or greater than Q in the route performed by a vehicle.

In our study, each request q_i is computed as the difference between the number of bikes present at station i when performing the redistribution, and the number of bikes in the station in the final required configuration. Note that we impose that a station with request $q_i = 0$ must be visited, even if this implies that no bike has to be dropped off or picked up there. This case arises, for example, when the driver of the vehicle is supposed to check that the station is correctly working. The case in which stations with null requests have to be skipped can be simply obtained by removing in a preprocessing phase those stations from the set of vertices.

The fact that each vehicle is allowed to start its route with some bikes enlarges the space of feasible BRP solutions, and allows to obtain a more flexible redistribution plan. Note also that we do not impose the sum of redistributed bikes to be null, and hence there can be a positive or a negative flow of bikes on the depot. This consideration is useful to model cases in which some bikes enter or leave the depot for maintenance.

The traveling cost c_{ij} is computed in our case as the shortest length of a path in the road network connecting i and j, for $i, j \in V$. It is important to work on a directed graph, because all bike sharing systems we are aware of are located in urban areas, and thus one-way streets typically have a strong impact on the choice of the routes performed by the vehicles during the redistribution.

The notation defined for the BRP is summarized in Table 2.1.

The BRP belongs to the wide class of *Pickup and Delivery Vehicle Rout-*

V	set of vertices
A	set of arcs
n	number of stations
m	number of vehicles
Q	vehicle capacity
q_i	demand at vertex i
c_{ij}	cost of the arc (i, j)

Table 2.1: BRP notation summary.

ing Problems (PDVRPs), where a fleet of vehicles is used to transport requests from the depot and/or some vertices to the depot and/or other vertices in the network. In particular, some *pickup customers* require a certain amount of freight to be picked up by a vehicle and then transported elsewhere, whereas some *delivery customers* require a certain amount of freight to be delivered there. The BRP generalizes the well-known *Capacitated Vehicle Routing Problem* (CVRP), where customers are either all pickup vertices or all delivery vertices, but not both, and thus it is a strongly NP-hard problem.

Detailed reviews and classifications of the PDVRPs have been proposed by Berbeglia et al. [6] and Parragh et al. [24, 25]. More recent surveys have been presented by Battarra et al. [3], for what concerns the transportation of freight, and Doerner and Salazar-González [11], for the transportation of people. Following the notation introduced in Berbeglia et al. [6], the BRP can be classified as a *Many-to-Many* (M-M) vehicle routing problem, in which a request has multiple origins (in our case pickup stations) and multiple destinations (in our case delivery stations).

The most basic problem in the class of M-M PDVRPs is probably the *One-commodity Pickup and Delivery Traveling Salesman Problem* (1-PDTSP), which calls for routing a single capacitated vehicle to meet M-M requests. The 1-PDTSP was formally introduced by Hernández-Pérez and Salazar-González [15], who presented mathematical formulations and branch-and-cut algorithms. The mathematical formulations were given for both the symmetric and asymmetric version of the problem, and were based on the use of a binary variable x_{ij} taking value one if the edge or arc (i, j) was used, 0 otherwise, and a non-negative variable f_{ij} giving the flow of the single commodity on edge or arc (i, j). Computational results were provided only for the symmetric formulation, that was solved by means of a branch-and-cut algorithm. The results of this approach were later improved by Hernández-Pérez and Salazar-González [17], with an enhanced branch-and-cut algorithm working again on the symmetric formulation but exploiting only the binary variable x_{ij}. To address larger instances, Hernández-Pérez and Salazar-González [16] proposed two simple heuristics, whereas a well-performing variable neighborhood descent metaheuristic was introduced by Hernández-Pérez et al. [14]. All these works considered the case in which

vehicles can leave the depot with some load, but the sum of all requests is equal to zero.

The 1-PDTSP has also attracted the interests of other researchers. Wang et al. [34] studied some variants of the 1-PDTSP that involve unit-load requests. They developed polynomial-time exact algorithms for the case of distribution on a path, and for the cases of distribution on a tree having $Q = 1$ or $Q = +\infty$. Martinovic et al. [20] presented a simulated annealing approach, whereas Zhao et al. [35] developed a genetic algorithm. The results of these two metaheuristics were later improved by Hosny and Mumford [13] by means of a *Variable Neighborhood Search* (VNS), but at the expense of very high computing times. A more recent VNS has been proposed by Mladenović et al. [22], that used a binary tree to efficiently perform feasibility checks and obtained very good computational results.

The multiple-vehicle case, known as the *One-commodity Pickup and Delivery Vehicle Routing Problem* (1-PDVRP), was formally introduced by Shi et al. [30], who studied the case in which a maximum duration limit is imposed on each route. They presented a genetic algorithm and a three-index mathematical formulation making use of a binary variable x_{ijk}, taking value 1 if vehicle k travels along edge or arc (i, j), 0 otherwise, and a non-negative variable f_{ij} representing the flow of the single commodity on edge or arc (i, j). Computational experiments were performed only for the genetic algorithm, that was tested on a set of randomly generated instances with symmetric costs.

The BRP is a generalization of the 1-PDTSP which involves more than one vehicle. Indeed, in the next sections we use successful properties and algorithmic ideas adapted for the BRP from [15] and [17]. Moreover, the BRP can be seen as a special case of the 1-PDVRP, because it does not consider maximum route duration.

As previously mentioned, the issue of rebalancing a bike sharing system has attracted the interest of many researchers in recent years. Benchimol et al. [5] study the case in which the redistribution is performed by a single capacitated vehicle, split deliveries are allowed (i.e., a vertex may be visited more than once) and the sum of all requests is balanced to zero. They present complexity results, lower bounding techniques, and approximation algorithms, but do not provide computational results.

In a recent paper, Chemla et al. [8] focus again on the single vehicle case, allowing split deliveries. Vertices with null demand are not forced to be visited, but can be used by vehicles as temporary buffer when transporting bikes among other stations. In their paper, the authors obtain a lower bound by solving the relaxation of the problem in which a limit is imposed on the maximum number of times that a vertex can be visited. They also obtain an upper bound by using a tabu search algorithm and run their bounding procedures on instances obtained by modifying those randomly created in [15].

In another recent paper, Raviv et al. [27] define the *static bicycle repositioning problem* in which they do not only minimize total traveling costs of a fleet of heterogeneous vehicles, but also users' dissatisfaction that is linked with a convex function to the inventory level of each station. The authors present two MILP formulations, an arc-indexed and a time-indexed one, each of them represents a different version of the problem. They allow limited or unlimited split delivery, limited or unlimited transshipment, dwelling with a maximum duration of each route and with synchronized transshipment. Exact and heuristics methods to speed up the computational times for solving the formulations are also presented in this paper.

Contardo et al. [9] present a formulation for the dynamic version of the bike rebalancing problem, allowing the use of more than one vehicle. Their objective function is the maximization of the serviced demand. An arc flow formulation working on a discretized time horizon is proposed and solved with with Dantzig-Wolfe and Benders decomposition. Extensive tests on several randomly created instances are presented.

2.4 MILP formulations for the BRP

In this section we present four *Mixed Integer Linear Programming* (MILP) formulations of the BRP. Computational evidence obtained from preliminary experiments suggested to us that it would be better to adopt two-index formulations for the BRP.

2.4.1 Formulation F1

The starting point of the aforementioned two-index formulations is the well-known *Multiple Traveling Salesman Problem* (m-TSP), see, e.g., Bektas [4], in which at most m uncapacitated vehicles based at a central depot have to visit a set of vertices, with the constraint that each vertex is visited exactly once. By defining a binary variable x_{ij}, taking value 1 if arc (i, j) is used by a vehicle, 0 otherwise, the m-TSP can be modeled as:

$$(m\text{-TSP}) \qquad \min \sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij} \tag{2.1}$$

$$\sum_{i \in V} x_{ij} = 1 \qquad j \in V \setminus \{0\} \tag{2.2}$$

$$\sum_{i \in V} x_{ji} = 1 \qquad j \in V \setminus \{0\} \tag{2.3}$$

$$\sum_{j \in V} x_{0j} \leq m \tag{2.4}$$

$$\sum_{j \in V \setminus \{0\}} x_{0j} = \sum_{j \in V \setminus \{0\}} x_{j0} \tag{2.5}$$

$$\sum_{i \in S} \sum_{j \in S} x_{ij} \leq |S| - 1 \qquad S \subseteq V \setminus \{0\}, S \neq \emptyset \quad (2.6)$$

$$x_{ij} \in \{0, 1\} \qquad i, j \in V. \quad (2.7)$$

Objective function (2.1) minimizes the traveling cost. Constraints (2.2) and (2.3) impose that every vertex but the depot is visited exactly once. Constraints (2.4) and (2.5) ensure, respectively, that at most m vehicles leave the depot, and that all vehicles that are used return to the depot at the end of their route. Constraints (2.6) are the classical *subtour elimination constraints*, see, e.g., Gutin and Punnen [12], that impose the connectivity of the solution. These constraints increase exponentially, so, as usual, we proceed in branch-and-cut fashion, by invoking a separation procedure to identify the violated ones from the non-violated ones, and then adding to the model only the violated constraints in an iterative way. The details will be given in the next section.

To guarantee the feasibility of a solution with respect to the BRP, we need to include additional constraints in the m-TSP formulation, so as to ensure that demands are satisfied and vehicle capacities are not exceeded. This may be done in different ways.

The first option is to define an additional continuous variable θ_j, representing the load of a vehicle after it visited vertex j, for $j \in V$. The load should be updated along the route by taking into consideration the fact that, if the vehicle travels along arc (i, j), then θ_j should be equal to $\theta_i + q_j$. The aforementioned restriction can be imposed by the non-linear constraints:

$$\theta_j \geq (\theta_i + q_j)\, x_{ij} \qquad i \in V, j \in V \setminus \{0\} \quad (2.8)$$

$$\theta_i \geq (\theta_j - q_j)\, x_{ij} \qquad i \in V \setminus \{0\}, j \in V \quad (2.9)$$

$$\max\{0, q_j\} \leq \theta_j \leq \min\{Q, Q + q_j\} \qquad j \in V. \quad (2.10)$$

Constraints (2.8) and (2.9) impose that, if x_{ij} takes value 1, then $\theta_j = \theta_i + q_j$, and hence model the flow conservation independently of the sign of q_j. Constraints (2.10) simply give lower and upper bounds on the loads.

These constraints generalize the Miller-Tucker-Zemlin subtour elimination constraints for the TSP, see Miller et al. [21], to the case where vertices have demands unrestricted in sign. They can be linearized with the standard "big M" method, obtaining:

$$\theta_j \geq \theta_i + q_j - M(1 - x_{ij}) \qquad i \in V, j \in V \setminus \{0\} \quad (2.11)$$

$$\theta_i \geq \theta_j - q_j - M(1 - x_{ij}) \qquad i \in V \setminus \{0\}, j \in V. \quad (2.12)$$

Constraints (2.8) and (2.9) are linearized to (2.11) and (2.12), respectively. In our implementation we set the "big M" equal to $\min\{Q, Q + q_j\}$ in (2.11), and equal to $\min\{Q, Q - q_j\}$ in (2.12).

Our *formulation F1* for the BRP is thus given by (2.1)–(2.7), (2.10)–(2.12). Note that subtour elimination constraints must be kept in the formulation, because (2.10)–(2.12) are not enough to ensure connectivity of the solution when some demands are non-positive and some are non-negative.

2.4.2 Formulation F2

Our second formulation also builds upon the m-TSP, but makes use of an additional variable f_{ij} giving the *flow* over arc (i, j), i.e., the load on the vehicle traveling along arc (i, j), if any, for $(i, j) \in A$. To impose feasibility with respect to the BRP, it is then enough to include in the m-TSP:

$$\sum_{i \in V} f_{ji} - \sum_{i \in V} f_{ij} = q_j \quad j \in V \setminus \{0\} \quad (2.13)$$

$$\max\{0, q_i, -q_j\} x_{ij} \leq f_{ij} \leq \min\{Q, Q + q_i, Q - q_j\} x_{ij} \qquad (i, j) \in A. \quad (2.14)$$

Constraints (2.13) model the balance of the flows on the arcs entering and leaving a given vertex. Constraints (2.14) impose lower and upper bounds on the flows on each arc, and make these bounds as tight as possible by considering whether or not an arc is traveled by a vehicle. Indeed, for what concerns the lower bound, if arc (i, j) is used, then f_{ij} should be at least greater than q_i if $q_i > 0$ (because q_i has just been collected) or greater than $-q_j$ if $q_j < 0$ (because q_j has to be supplied to the next vertex). Similar considerations apply to the upper bound.

Formulation F2 of the BRP is thus given by (2.1)–(2.7), (2.13)–(2.14).

2.4.3 Formulation F3

We use the approach proposed by Hernández-Pérez and Salazar-González [15, 17] for the 1-PDTSP in order to develop another formulation of the BRP. We start again from the m-TSP, but ensure that solutions satisfy demands, without exceeding the vehicle capacity, by adding to the model the following family of constraints:

$$\sum_{i \in S} \sum_{j \in S} x_{ij} \leq |S| - \max \left\{ 1, \left\lceil \frac{\left| \sum_{i \in S} q_i \right|}{Q} \right\rceil \right\} \quad S \subseteq V \setminus \{0\}, S \neq \emptyset. \quad (2.15)$$

Constraints (2.15) are similar to the *generalized subtour elimination constraints* for the CVRP. They state that, for each subset S of vertices, the number of arcs with both tail and head in S should not exceed the cardinality of S minus the minimum number of vehicles required to serve S. An estimation of the minimum number of vehicles is simply obtained by computing the absolute value of the sum of the demands, dividing it by the vehicle capacity and then rounding up the result. This value can be zero when the sum of the q_j is null. In such a case the value 1 is used instead,

because at least one vehicle is needed (notice that S does not contain the depot).

Our *formulation F3* is thus given by (2.1)–(2.5), (2.7), and (2.15). Note that constraints (2.15) are exponentially many, and hence, as already observed for (2.6), we need a separation procedure that identifies violated constraints from non-violated ones. Also note that these constraints are valid for formulations F1 and F2, and may be used to improve their convergence to the optimum.

2.4.4 Formulation F4

We propose a fourth formulation of the BRP that builds upon the two-commodity flow model originally presented by Baldacci et al. [2] for the symmetric CVRP. This formulation first requires to modify the graph by adding a copy of the depot, defined in the following by a new vertex $n+1$. We define $\widetilde{V} = V \cup \{n+1\}$ the resulting set of vertices, and $\widetilde{A} = A \cup \{(j, n+1) : j \in \widetilde{V}\} \cup \{(n+1, j) : j \in \widetilde{V}\}$ the resulting set of arcs. Let us also define $\widetilde{V}_0 = \widetilde{V} \setminus \{0, n+1\}$ and $Q_{tot} = \sum_{j \in \widetilde{V}_0} q_j$. We set $q_{n+1} = 0$, $c_{n+1,j} = +\infty$, and $c_{j,n+1} = c_{j0}$ for $j \in \widetilde{V}$, knowing that $c_{0,n+1} = 0$, and then we set $c_{j0} = +\infty$ for $j \in \widetilde{V}$.

We use three sets of variables. The first two have already been used in formulation F2: x_{ij} takes value 1 if arc (i, j) is used, 0 otherwise, for $(i, j) \in \widetilde{A}$, and f_{ij} gives the flow over arc (i, j), i.e., the load of the vehicle passing over arc (i, j), if any, for $(i, j) \in \widetilde{A}$. The third variable is another continuous variable g_{ji} that gives the residual space of the vehicle traveling along arc (i, j), if any, for $(i, j) \in \widetilde{A}$.

In this formulation, vehicles leave the initial depot, vertex 0, visit customers, and then end their route in the final depot, vertex $n + 1$. Variable f_{ij} gives the load along the route performed by the vehicle, starting from the initial depot, vertex 0, following the vertices visited by the route and then ending in the final depot, vertex $n + 1$. In opposite way, g_{ji} gives the free space on the vehicle, starting from $n + 1$, retracing back the route and then ending 0. The resulting model is then:

(Formulation F4)

$$\min \sum_{i \in \widetilde{V}} \sum_{j \in \widetilde{V}} c_{ij} x_{ij} \tag{2.16}$$

$$\sum_{i \in \widetilde{V}} x_{ij} = 1 \qquad j \in \widetilde{V}_0 \tag{2.17}$$

$$\sum_{i \in \widetilde{V}} x_{ji} = 1 \qquad j \in \widetilde{V}_0 \tag{2.18}$$

$$\sum_{j \in \widetilde{V}} x_{0j} \leq m \tag{2.19}$$

$$\sum_{j\in\widetilde{V}_0} x_{0j} = \sum_{j\in\widetilde{V}_0} x_{j,n+1} \tag{2.20}$$

$$\sum_{i\in S}\sum_{j\in S} x_{ij} \leq |S| - 1 \qquad S \subseteq \widetilde{V}_0, S \neq \emptyset \tag{2.21}$$

$$f_{ij} + g_{ji} = Q x_{ij} \qquad (i,j) \in \widetilde{A} \tag{2.22}$$

$$\sum_{i\in\widetilde{V}}(f_{ji} - g_{ij}) - \sum_{i\in\widetilde{V}}(f_{ij} - g_{ji}) = 2q_j \qquad j \in \widetilde{V}_0 \tag{2.23}$$

$$\sum_{j\in\widetilde{V}_0} f_{0j} \geq \max\{0, -Q_{tot}\} \tag{2.24}$$

$$\sum_{j\in\widetilde{V}_0} f_{j,n+1} \geq \max\{0, Q_{tot}\} \tag{2.25}$$

$$\sum_{j\in\widetilde{V}_0} g_{j0} \leq \min\{mQ, mQ + Q_{tot}\} \tag{2.26}$$

$$\begin{aligned} \max\{0, q_i, -q_j\} x_{ij} &\leq f_{ij} \\ f_{ij} &\leq \min\{Q, Q + q_i, Q - q_j\} x_{ij} \end{aligned} \qquad (i,j) \in \widetilde{A} \tag{2.27}$$

$$\begin{aligned} (Q - \min\{Q, Q + q_i, Q - q_j\}) x_{ij} &\leq g_{ji} \\ g_{ji} &\leq (Q - \max\{0, q_i, -q_j\}) x_{ij} \end{aligned} \qquad (i,j) \in \widetilde{A} \tag{2.28}$$

$$x_{ij} \in \{0,1\} \qquad i,j \in \widetilde{V}. \tag{2.29}$$

Constraints (2.17)–(2.20) and (2.29) are equivalent to the ones discussed for formulation F1, but adapted to the new sets of vertices and arcs. Constraints (2.22) state that, if a vehicle travels along arc (i,j), then the sum of the load and the residual space on that arc is equal to the vehicle capacity. Constraints (2.23) impose that the difference between loads and residual capacities that enter and leave a vertex is double of the demand of that vertex.

The total load leaving the initial depot should be in any case non-negative, and moreover, in case Q_{tot} takes a negative value, it should be not lower than this value. This fact is imposed by constraints (2.24). Similarly, constraints (2.25) state that the total load entering the final depot is in any case non-negative, and not lower than the sum of all demands in case this is positive. Inequalities (2.26) state that the free space on the vehicles leaving the depot must be not greater than the minimum value between the total capacity of the m vehicles, and the total capacity plus Q_{tot}. Finally, constraints (2.27) and (2.28) enforce lower and upper bounds on the continuous variables.

2.5 Branch-and-Cut algorithm

The models presented in the previous section include an exponential number of constraints, hence we solve them with a *Branch-and-Cut* (B&C) algorithm. We use the B&C framework of CPLEX 12.2, that solves at every vertex of an enumeration tree the linear relaxation of a MILP model, and then invokes user-developed separation procedures for the possible addition of cuts.

At the root node of the B&C procedure we obtain an initial solution by using a simple greedy heuristic. The greedy method builds routes one at a time, using the principle of the closest neighbor. It starts from the depot and then visits the closest, yet unserved, customer. It then extends the path by moving to the next closest neighbor for which the sum of collected and delivered demands is feasible (see Section 2.5.2 below for the algorithm used to determine if a path is feasible), if any. When no extension is possible, the route is closed by going back to the depot, and possibly a new route is initialized in an iterative fashion. The model is then initialized with the generalized subtour elimination constraints for sets of one or two vertices, hence $x_{ii} = 0$ for all $i \in V$ and $x_{ij} + x_{ji} \leq 2 - \max\{1, \lceil |q_i + q_j|/Q \rceil\}$, for all $(i, j) \in A$. After testing a few branching strategies, we adopted the automatic strong branching procedure by CPLEX, that provides the best computational performance on our test bed.

In Section 2.5.1 we present some valid inequalities that we use to improve the convergence of the algorithm to the optimum, whereas in Section 2.5.2 we discuss the separation procedures that we implemented. Since inequalities and separation procedures are based only on the x_{ij} variables, they apply directly to formulations 1, 2 and 3. The adaptation to formulation 4 is trivial, and is outlined in Section 2.5.2.

2.5.1 Valid inequalities

We first propose two new simple families of *clique inequalities* with three vertices. Consider a pair of vertices i and j, and let $S(i, j) \subset V \setminus \{0\}$ be the subset of vertices whose demand results in a total greater than Q or smaller than $-Q$. Namely, $S(i, j) = \{h \in V_0, h \neq i, h \neq j : |q_i + q_j + q_h| > Q\}$. Then the following inequalities are valid for the BRP:

$$x_{ij} + \sum_{h \in S(i,j)} x_{jh} \leq 1 \qquad i, j \in V \setminus \{0\}, h \in S(i, j) \tag{2.30}$$

$$\sum_{h \in S(i,j)} x_{hi} + x_{ij} \leq 1 \qquad i, j \in V \setminus \{0\}, h \in S(i, j). \tag{2.31}$$

We note that, in (2.30), at most one among the x_{jh} variables may take the value of 1 because of the degree constraint, and that all of them are

incompatible with x_{ij} because of the capacity constraint. Thus the sum of all these variables is at most 1 in a feasible solution. A similar consideration applies to (2.31).

For the next family of inequalities we need some further notation. Let P denote a path consisting of a sequence of vertices starting from the depot. Let also $|P|$ denote the number of vertices in P, and $P(i)$ denote the index of the i-th vertex of path P, for $i = 0, 1, \ldots, |P|-1$, with $P(0) = 0$. Given P, we compute $q_P^{\min} = \min_{i=1}^{|P|-1}\{\sum_{j=1}^{i} q_{P(j)}\}$ and $q_P^{\max} = \max_{i=1}^{|P|-1}\{\sum_{j=1}^{i} q_{P(j)}\}$, that denote, respectively, the minimum and maximum of the cumulative demand along P.

As observed by Hernández-Pérez and Salazar-González [15], a path P is infeasible with respect to the capacity constraint if $q_P^{\max} - q_P^{\min} > Q$. Indeed, if the vehicle leaves the depot with an empty load, then along the path it will reach a maximum load equal to $q_P^{\max}$, that clearly cannot exceed Q. Now notice that $q_P^{\min}$ may be negative even in a feasible solution, but in this case the vehicle has to start from the depot by carrying $-q_P^{\min}$ bikes. Hence, the maximum load becomes $q_P^{\max} - q_P^{\min}$, which also cannot exceed Q.

Let $\mathcal{P}$ be the family of all infeasible paths, the following *infeasible path constraints* are thus valid inequalities for the BRP:

$$\sum_{i=0}^{|P|-2} x_{P(i),P(i+1)} \leq |P| - 2 \qquad P \in \mathcal{P}. \tag{2.32}$$

Inequality (2.32) simply states that, if path P is infeasible then not all the arcs connecting two consecutive vertices of P may belong to a feasible solution. A way to enforce them is to consider also the arcs connecting non-consecutive vertices of P. Indeed, any arc $(P(i), P(j))$ with $j \neq i + 1$ is incompatible with arcs $(P(i), P(i + 1))$ and $(P(j - 1), P(j))$, because of, respectively, out-degree and in-degree constraints. Hence we can add the corresponding variable, $x_{P(i),P(j)}$ to the left-hand side of (2.32), without affecting the right-hand side value. This process can be repeated for all arcs connecting two non-consecutive vertices in the path, with the exception of those arcs leaving the depot, because up to m of them may belong to a feasible solution. Hence, we obtain the following:

$$x_{0,P(1)} + \sum_{i=1}^{|P|-2} \sum_{j=i+1}^{|P|-1} x_{P(i),P(j)} \leq |P| - 2 \qquad P \in \mathcal{P}. \tag{2.33}$$

Inequalities (2.33) generalize, in the case of multiple vehicles, the *tournament constraints*, discussed by Ascheuer et al. [1] for the Traveling Salesman Problem. Note that these constraints can be used in directed formulations (where a variable is associated to each directed arc), but cannot be applied to undirected ones (where variables are associated to undirected edges).

2.5.2 Separation procedures

The aim of this section is to present the procedures that we use to determine whether or not the valid inequalities we proposed in Section 2.5.1 are violated by a given solution $\overline{x}$ (possibly fractional).

Separations S1 and S2. The inequalities of type (2.30) and (2.31) are separated exactly, by enumerating all possible pairs of vertices i and j, computing the set S_{ij}, and then evaluating if the sum of the $\overline{x}$ involved in the inequality exceeds one. In the following we will refer the separation of (2.30) and (2.31) as *separation S1* and *separation S2*, respectively.

Separation S3. To separate constraints (2.6) and (2.15), we first build a supporting graph $\overline{G} = (\overline{V}, \overline{A})$, where $\overline{V} = V$, $\overline{A} = \{(i,j) \in A : \overline{x}_{ij} > 0\}$, and a capacity $\overline{x}_{ij}$ is assigned to every arc $(i,j) \in \overline{A}$. Constraints (2.6) may be separated exactly as usually done for the TSP, by computing $O(n)$ max flows on $\overline{G}$, using the depot as a source and any vertex i, in turn, as a sink. If the max flow value obtained is lower than one, then i is disconnected from the depot, and hence the cut that corresponds to the set S induced by the min cut may be added to the model. For the BRP, we obtain a simple algorithmic improvement by checking, for any set S induced by a min cut, if the minimum number of vehicles required to serve S is lower than the max flow value obtained, and, in such a case, we add the corresponding (stronger) constraint (2.15). In this way the exact separation of (2.6) is used as a heuristic for separating (2.15). We refer to this procedure as *separation S3*.

Separation S4. To separate constraint (2.15), we extend the procedure developed in Hernández-Pérez and Salazar-González [15] for the 1-PDTSP to the directed case. First of all we associate a demand $q_0 = Q_{tot}$ to the depot. Then we build a new supporting graph $\overline{G}'$, obtained by adding two dummy nodes, $n+1$ and $n+2$, to $\overline{G}$. We connect $n+1$ to any $i \in V$ having $q_i > 0$ with an arc of capacity q_i/Q, and then connect any $i \in V$ having $q_i < 0$ to $n+2$ with an arc of capacity $-q_i/Q$ (in this way, the capacities associated with the arcs are always non-negative) . We then compute the max flow on $\overline{G}'$, using $n+1$ as a source and $n+2$ as a sink. The constraint (2.15) corresponding to the set S induced by the min cut is then checked, and, if violated, it is added to the model. We refer to Hernández-Pérez and Salazar-González [15] for further details.

Conforming to the fact that we are solving a problem on an asymmetric graph, we perform a second separation for (2.15). We build a second modified supporting graph $\overline{G}''$, adding again two new dummy vertices, $n+1$ and $n+2$, to $\overline{G}$. In this case however we include in $\overline{G}''$ new arcs $(n+1, i)$ of capacity $-q_i/Q$ for all $i \in V$ having $q_i < 0$, and new arcs $(i, n+2)$ of capacity q_i/Q for all $i \in V$ having $q_i > 0$. Once more, we use the max flow procedure on the resulting supporting graph, from $n+1$ to $n+2$, and check the constraints (2.15) for the sets S induced by the min cuts. In the

following, these procedures are referred to as *separation S4*.

Separation S5. The tournament constraints (2.33) are separated exactly, by generating all possible paths starting from the depot and using a depth-first strategy. We initialize vector P with $P(0) = 0$, then select the first outgoing arc having a positive value of $\overline{x}$, say, $(0, i)$, and extend the path to include the head of the selected arc by setting $P(1) = i$. The process is then reiterated. Every time we add a vertex to the path, we check if it is infeasible. If so, then we add the cut and backtrack to the previous vertex, otherwise we continue extending the path. The path extension continues as long as the sum of the involved $\overline{x}_{ij}$ is large enough to possibly lead to a violated cut, i.e., as long as it is strictly greater than $|P| - 2$. As soon as the sum becomes smaller, we backtrack to the previous vertex. Anytime we backtrack, we continue the depth-first search by selecting the next arc with positive value of $\overline{x}$ and then extend the path consequently.

Suppose a violated cut of type (2.33) has been found, then we know that the set $S = \{P(1), P(2), \ldots, P(|P|)\}$ is currently visited by a single vehicle (the one performing the path P). Now we estimate the minimum number of vehicles required to serve S using $\lceil |\sum_{i \in S} q_i|/Q \rceil$, and, if this is greater than one, we add the violated generalized subtour elimination constraint (2.15) for this set S. Constraints (2.15) are usually stronger than (2.33), but do not dominate them (because the latter involves variable $x_{0,P(1)}$, not contained in the former), and hence they are both profitable for the model solution. The current procedure is called *separation S5*.

The separation procedures (S1-S5) are invoked at every node of the enumeration tree in the order in which we described them. With regards to formulation F4, there is only a slight modification to implement. Recall that this formulation works on a modified graph $\widetilde{V}$, where $n+1$ is a copy of the original depot 0. When creating the supporting graph $\overline{G}$, we merge again 0 and $n+1$ into a unique vertex 0 (this is done by simply setting $\overline{x}_{j,0} = \overline{x}_{j,n+1}$ for all $j \in V$), and then use the separation process just described for the previous formulations. The resulting cut, if any, is then mapped back on $\widetilde{V}$. On the basis of computational evidence resulting from detailed experimentation, the separation process for all formulations is terminated as soon as a violated cut is found.

2.6 Benchmark Instances

In our attempt to solve real-world instances, we collected real data from the web sites of twenty-two bike sharing systems characterized by diverse size. The cities included in our studies are: Bari, Brescia, Bergamo, La Spezia, Parma, Rome, Torino, Treviso, and Reggio Emilia, in Italy; Dublin in Ireland; Boston, Denver, Madison, Miami, Minneapolis, and San Anto-

nio in the USA; Ottawa and Toronto in Canada; Ciudad de Mexico and Guadalajara in Mexico; Buenos Aires in Argentina and Rio de Janeiro in Brazil.

We have been in touch with the bike sharing operators of these cities, in particular with those people responsible for the bike repositioning, to obtain information regarding the depot location, the desirable level of occupation, the characteristics of the available fleet of vehicles and the type of repositioning performed. Not all of them furnished us with all the required information. In general, we can state that some of them perform the repositioning during the night, some others also during the day. From the web sites we collected the coordinates of the stations and of the depot. If no information for the depot was available, we assumed that the depot was located at the same place as one of the stations. We used a Geographical Information System to compute the shortest traveling distances c_{ij} (in meters) between each pair of points, considering the two possible directions. We then took a snapshot of the nightly level of occupation at each station, and fixed the demand of the station to be the difference between the level of occupation encountered and the desired level of occupation of the station. Indeed, according to the information provided by the majority of operators, the most common rule adopted to rebalance the stations is to fill half of the slots in each station, thus we set the desired level of occupation of a station to half its number of slots. Also note that, for those cities where the depot was allocated to a bike station, we set the demand of that station to 0.

City	Country	$\|V\|$	$\min\{q_i\}$	$\text{avg}\{q_i\}$	$\max\{q_i\}$	$\text{dev}\{q_i\}$	$\min\{c_{ij}\}$	$\text{avg}\{c_{ij}\}$	$\max\{c_{ij}\}$	$\text{dev}\{c_{ij}\}$
Bari	Italy	13	-5	-1.54	5	2.70	400	2283.97	5400	1067.62
Reggio Emilia	Italy	14	-10	-2.00	3	4.17	300	2095.05	5500	1110.44
Bergamo	Italy	15	-12	-1.00	10	5.39	100	1532.86	3200	631.94
Parma	Italy	15	-6	-1.07	4	2.76	200	3121.43	8800	1857.86
Treviso	Italy	18	-4	-0.83	3	2.23	340	3510.99	8398	2133.37
La Spezia	Italy	20	-5	0.05	6	2.93	193	2521.61	6128	1279.82
Buenos Aires	Argentina	21	-20	-0.43	20	17.15	689	4676.75	12780	2246.80
Ottawa	Canada	21	-5	-0.05	5	2.58	180	2219.15	5030	1012.63
San Antonio	U. S.	23	-4	1.74	8	3.43	98	1950.98	4808	944.61
Brescia	Italy	27	-11	-1.41	4	3.69	200	2571.65	6600	1174.49
Roma	Italy	28	-17	-2.36	18	9.45	200	5989.55	27400	7361.37
Madison	U. S.	28	-6	0.29	8	2.93	53	3085.99	9922	1780.75
Guadalajara	Mexico	41	-11	-1.07	1	1.94	60	3278.30	14728	2440.40
Dublin	Ireland	45	-11	-1.42	6	3.73	203	2190.50	4734	933.16
Denver	U. S.	51	-8	-0.69	7	3.28	211	3873.94	12000	2643.54
Rio de J.	Brazil	55	-7	-1.47	7	3.82	420	5328.35	16591	2587.97
Boston	U. S.	59	-8	-0.27	16	5.28	243	3911.82	19239	2378.22
Torino	Italy	75	-7	-0.49	9	3.73	23	2527.84	7200	1157.14
Toronto	U. S.	80	-11	0.15	12	5.17	150	2339.03	6283	1103.70
Miami	U. S.	82	-8	-2.24	9	3.99	68	4000.00	13771	3336.49
Ciudad de M.	Mexico	90	-17	-0.97	17	8.48	15	2551.94	7264	1409.34
Minneapolis	U. S.	116	-9	-0.79	5	2.87	6	6045.24	19468	3558.17

Table 2.2: Benchmark instances (notation explained in Table 2.1).

According to the data we collected, the most common vehicle capacities encountered in practice are 30, 20 and 10 bikes for each vehicle. We applied these three values to the data obtained for each city. For some instances the minimum vehicle capacity used is set to $\max_{i \in V}\{|q_i|\}$, when the number of the latter exceeds 10. For Buenos Aires, whose maximum demand was

20, we tested only 20 and 30. We generated 65 instances in total, that are summarized in Table 2.2. The values $\min\{q_i\}$, $\text{avg}\{q_i\}$, $\max\{q_i\}$, and $\text{dev}\{q_i\}$ give, respectively, the minimum, average, and maximum demand, and the standard deviation. Similar information is provided for the distances c_{ij}. The number of vertices varies from 13 to 116 stations. All test instances have variability in the number of bikes leaving the depot (since $\text{avg}\{q_i\}$ is not 0). All distances are given in meters. All instances are available online at `http://www.or.unimore.it/resources.htm`.

2.7 Computational results

All algorithms described in Sections 4.4 and 3.5.2 were coded in C/C++ and run on an Intel Core i3-2100 CPU, 3.10 GHz, 4.00 GB. We used CPLEX 12.2 as the MILP solver imposing the selection of a single processor. Formulations F1−F4 were computationally tested on the problem instances described in Section 2.6. For clarity purposes, details of the computational implementation of the formulations we tested are given in Table 2.3, where we also include the additional separation procedures described in Section 2.5.2.

	variables	core formulation	additional separations
Formulation F1	x_{ij}, θ_j	(2.1)–(2.7), (2.10)–(2.12), S3	S1, S2, S4, S5
Formulation F2	x_{ij}, f_{ij}	(2.1)–(2.7), (2.13)–(2.14), S3	S1, S2, S4, S5
Formulation F3	x_{ij}	(2.1)–(2.5), (2.7), (2.15), S3, S4	S1, S2, S5
Formulation F4	x_{ij}, f_{ij}, g_{ij}	(2.16)–(2.29), S3	S1, S2, S4, S5

Table 2.3: Implementation of formulations F1−F4.

In this section, we present the computational results of the B&C algorithm using formulations F1−F4. The B&C algorithm was terminated when either the optimal solution was found or a predetermined time-limit elapsed. The time-limit imposed was 3600 CPU seconds. Table 2.4 reports the computational results obtained for the complete set of 65 problem instances. In this table, the following information is provided for each problem instance:

- *City* $(|V|, Q)$: City name associated with a given problem instance which is uniquely identified by the number of vertices $|V|$ of the corresponding digraph and the vehicle capacity Q used to transport the bikes at the specified city;
- UB: The value of the best feasible solution found by the B&C algorithm using formulations F1 to F4;
- $Gap_1 - Gap_4$: Percentage deviation from UB of $LB_1 - LB_4$, best lower bounds obtained by the B&C algorithm using core formulations F1 to F4 including the set of additional separations for each formulation,

respectively, as shown in Table 3. $Gap_i = 100/(1 - LB_i/UB)$, for $i = 1, \ldots, 4$.

- *Time*: Computational time for running the B&C algorithm, in CPU seconds.

To compare the average performance of formulations F1−F4 over the complete set of problem instances, the following additional information is given in Table 2.4:

- *Opt*: The total number of problem instances solved to optimality by the B&C algorithm within 3600 seconds;
- $\#Nodes$: Average number of nodes (over all instances) explored by the B&C algorithm;
- $\#Cuts$: Average number of cuts generated by the B&C algorithm.

As shown in Table 2.4, the first 25 problem instances and a large number of other instances are easily solved by all formulations within a few seconds. Formulation F1 gives the worst results in terms of average gap, computational time, and number of generated cuts, whereas, formulations F2 and F4 demonstrate similar performance for all performance indicators. On the other hand, F3 performs best over all formulations with average $Gap_3 = 2.24\%$ and it optimally solves all problem instances including up to 50 vertices. Indeed, formulation F3 finds the optimal solution for 49 out of 65 instances within an average time of 15 minutes generating an average of approximately 1200 B&C nodes. However, it is worth noting that, for some of the larger instances, Gap_3 is larger than Gap_2, (e.g., Ciudad de Mexico (90, 17)) or Gap_4 (e.g., Miami (82, 10)). Further insight into the best performing formulation F3 was gained by computationally evaluating the impact of each separation procedure on the resulting percentage gaps computed at the root node of the B&C tree for the most difficult problem instances.

In Table 2.5 we test the core formulation F3, see Table 2.3, and the ones obtained by including just one of the separation procedures at a time. We denote with Gap_{core} the percentage deviation from UB of the lower bound obtained with the core version, while we use $Gap^+(Sx)$ to denote the results of the core version *plus* separation Sx.

In Table 2.6 we present the results of the full version of formulation F3 (i.e., core formulation plus all separation procedures) and those obtained by removing from it one separation procedure at a time. We denote with Gap_{full} the percentage deviation from UB of the lower bound obtained with the full version, while we use $Gap^-(Sx)$ to denote the results of the full version *minus* separation Sx.

			F1		F2		F3		F4	
Inst.	City ($\|V\|$, Q)	UB	Gap_1	Time	Gap_2	Time	Gap_3	Time	Gap_4	Time
1	Bari (13, 30)	14600	0.00	0.06	0.00	0.11	0.00	0.02	0.00	0.27
2	Bari (13, 20)	15700	0.00	0.06	0.00	0.10	0.00	0.02	0.00	0.11
3	Bari (13, 10)	20600	0.00	0.16	0.00	0.39	0.00	0.03	0.00	0.32
4	Reggio Emilia (14, 30)	16900	0.00	0.03	0.00	0.03	0.00	0.02	0.00	0.06
5	Reggio Emilia (14, 20)	23200	0.00	0.09	0.00	0.52	0.00	0.03	0.00	0.33
6	Reggio Emilia (14, 10)	32500	0.00	5.59	0.00	0.67	0.00	0.05	0.00	0.22
7	Bergamo (15, 30)	12600	0.00	0.05	0.00	0.05	0.00	0.03	0.00	0.10
8	Bergamo (15, 20)	12700	0.00	0.06	0.00	0.26	0.00	0.00	0.00	0.21
9	Bergamo (15, 12)	13500	0.00	0.27	0.00	0.74	0.00	0.11	0.00	0.95
10	Parma (15, 30)	29000	0.00	0.05	0.00	0.03	0.00	0.02	0.00	0.06
11	Parma (15, 20)	29000	0.00	0.05	0.00	0.02	0.00	0.02	0.00	0.06
12	Parma (15, 10)	32500	0.00	0.22	0.00	0.26	0.00	0.05	0.00	0.45
13	Treviso (18, 30)	29259	0.00	0.12	0.00	0.14	0.00	0.05	0.00	0.62
14	Treviso (18, 20)	29259	0.00	0.12	0.00	0.20	0.00	0.03	0.00	0.45
15	Treviso (18, 10)	31443	0.00	0.27	0.00	0.75	0.00	0.09	0.00	0.79
16	La Spezia (20, 30)	20746	0.00	0.09	0.00	0.16	0.00	0.03	0.00	0.31
17	La Spezia (20, 20)	20746	0.00	0.09	0.00	0.10	0.00	0.03	0.00	0.24
18	La Spezia (20, 10)	22811	0.00	0.16	0.00	1.45	0.00	0.09	0.00	1.34
19	Buenos Aires (21, 30)	76999	0.00	1.36	0.00	3.54	0.00	0.37	0.00	12.40
20	Buenos Aires (21, 20)	91619	0.00	23.26	0.00	3.58	0.00	16.80	0.00	7.35
21	Ottawa (21, 30)	16202	0.00	0.06	0.00	0.05	0.00	0.02	0.00	0.09
22	Ottawa (21, 20)	16202	0.00	0.06	0.00	0.06	0.00	0.02	0.00	0.09
23	Ottawa (21, 10)	17576	0.00	0.30	0.00	1.90	0.00	0.11	0.00	1.26
24	San Antonio (23, 30)	22982	0.00	0.19	0.00	2.74	0.00	0.08	0.00	4.38
25	San Antonio (23, 20)	24007	0.00	3.63	0.00	1.36	0.00	0.09	0.00	7.52
26	San Antonio (23, 10)	40149	5.36	3600.00	0.00	7.07	0.00	1.06	0.00	17.86
27	Brescia (27, 30)	30300	0.00	0.70	0.00	0.77	0.00	0.06	0.00	2.22
28	Brescia (27, 20)	31100	0.00	6.07	0.00	6.38	0.00	0.20	0.00	9.11
29	Brescia (27, 11)	35200	0.00	24.46	0.00	10.83	0.00	1.37	0.00	9.31
30	Roma (28, 30)	61900	0.00	4.27	0.00	5.48	0.00	0.84	0.00	8.46
31	Roma (28, 20)	66600	0.00	22.04	0.00	7.61	0.00	1.72	0.00	33.38
32	Roma (28, 18)	68300	0.00	16.15	0.00	3.26	0.00	0.58	0.00	4.99
33	Madison (28, 30)	29246	0.00	0.09	0.00	0.10	0.00	0.02	0.00	0.24
34	Madison (28, 20)	29839	0.00	0.31	0.00	0.44	0.00	0.05	0.00	0.84
35	Madison (28, 10)	33848	0.00	6.02	0.00	16.56	0.00	0.53	0.00	6.36
36	Guadalajara (41, 30)	57476	0.00	1.16	0.00	2.97	0.00	0.22	0.00	6.22
37	Guadalajara (41, 20)	59493	0.00	2.29	0.00	8.18	0.00	0.34	0.00	7.21
38	Guadalajara (41, 11)	64981	1.18	3600.00	0.00	39.15	0.00	1.79	0.00	17.99
39	Dublin (45, 30)	33548	0.00	435.69	0.00	180.81	0.00	6.05	0.00	280.05
40	Dublin (45, 20)	39786	2.43	3600.00	1.14	3600.00	0.00	76.72	0.00	3140.89
41	Dublin (45, 11)	54392	7.70	3600.00	1.17	3600.00	0.00	610.80	0.00	3513.55
42	Denver (51, 30)	51583	0.00	66.57	0.00	11.01	0.00	0.67	0.00	15.88
43	Denver (51, 20)	53465	0.00	410.05	0.00	372.78	0.00	25.33	0.00	1096.95
44	Denver (51, 10)	67459	1.47	3600.00	0.00	1535.62	0.00	231.52	0.00	3443.68
45	Rio de J. (55, 30)	122547	4.16	3600.00	1.57	3600.00	0.00	65.57	1.97	3600.00
46	Rio de J. (55, 20)	156140	8.57	3600.00	2.29	3600.00	0.44	3600.00	1.82	3600.00
47	Rio de J. (55, 10)	259049	28.58	3600.00	2.07	3600.00	2.23	3600.00	2.23	3600.00
48	Boston (59, 30)	65669	0.00	255.25	0.00	1136.46	0.00	28.14	0.00	975.97
49	Boston (59, 20)	71879	0.00	1899.07	1.70	3600.00	0.00	473.84	1.63	3600.00
50	Boston (59, 16)	75065	0.99	3600.00	2.59	3600.00	0.37	3600.00	2.94	3600.00
51	Torino (75, 30)	47634	0.00	464.46	0.00	919.59	0.00	13.79	0.00	1703.46
52	Torino (75, 20)	50204	2.38	3600.00	2.70	3600.00	0.00	859.69	3.50	3600.00
53	Torino (75, 10)	64797	11.30	3600.00	9.66	3600.00	9.25	3600.00	9.23	3600.00
54	Toronto (80, 30)	41549	3.54	3600.00	4.47	3600.00	1.82	3600.00	4.42	3600.00
55	Toronto (80, 20)	47898	11.81	3600.00	11.02	3600.00	11.05	3600.00	11.39	3600.00
56	Toronto (80, 12)	60763	13.15	3600.00	10.74	3600.00	11.84	3600.00	10.88	3600.00
57	Miami (82, 30)	156104	23.68	3600.00	8.43	3600.00	2.48	3600.00	9.01	3600.00
58	Miami (82, 20)	229237	38.43	3600.00	11.02	3600.00	8.66	3600.00	10.99	3600.00
59	Miami (82, 10)	415762	38.14	3600.00	6.07	3600.00	6.49	3600.00	6.12	3600.00
60	Ciudad de M. (90, 30)	88227	28.67	3600.00	23.28	3600.00	23.05	3600.00	23.74	3600.00
61	Ciudad de M. (90, 20)	116418	29.95	3600.00	23.59	3600.00	23.61	3600.00	23.63	3600.00
62	Ciudad de M. (90, 17)	109573	20.00	3600.00	9.00	3600.00	9.56	3600.00	10.73	3600.00
63	Minneapolis (116, 30)	137843	4.58	3600.00	5.82	3600.00	1.23	3600.00	5.96	3600.00
64	Minneapolis (116, 20)	186449	23.45	3600.00	18.61	3600.00	15.40	3600.00	18.41	3600.00
65	Minneapolis (116, 10)	298886	37.08	3600.00	17.65	3600.00	17.96	3600.00	18.93	3600.00
Average Gaps and Times			5.33	1330.02	2.69	1228.99	2.24	923.37	2.73	1272.84
Opt			**42**		**44**		**49**		**46**	
#Nodes			2134.32		1847.74		1240.91		1440.65	
#Cuts			2233.02		630.14		1908.28		578.09	

Table 2.4: Computational results for formulations F1−F4 on all problem instances.

City ($\|V\|$, Q)	Gap_{core}	Gap^+(S1)	Gap^+(S2)	Gap^+(S5)
Dublin (45 ,30)	1.45	1.45	1.45	1.45
Dublin (45 , 20)	4.56	4.31	3.68	4.56
Dublin (45, 11)	5.52	5.61	4.69	4.86
Denver (51, 30)	0.52	0.52	0.52	0.52
Denver (51, 20)	2.58	2.58	2.58	2.56
Denver (51, 10)	3.34	3.05	4.29	3.32
Rio de Janeiro (55, 30)	2.05	2.05	2.05	2.05
Rio de Janeiro (55, 20)	2.51	2.51	2.51	2.51
Rio de Janeiro (55, 10)	4.12	3.99	3.35	4.12
Boston (59, 30)	1.87	1.76	1.76	1.87
Boston (59, 20)	3.31	3.45	3.11	3.28
Boston (59, 16)	4.63	4.25	4.13	4.63
Torino (75, 30)	1.35	1.35	1.35	1.35
Torino (75, 20)	3.38	3.81	3.30	3.38
Torino (75, 10)	11.73	11.44	11.50	11.73
Toronto (80, 30)	5.55	5.55	5.55	5.55
Toronto (80, 20)	15.19	13.48	14.10	15.19
Toronto (80, 12)	14.40	14.87	14.97	14.40
Miami (82, 30)	3.28	3.28	3.28	3.28
Miami (82, 20)	8.91	9.17	9.29	8.91
Miami (82, 10)	6.71	7.03	6.86	6.71
Ciudad de Mexico (90, 30)	24.62	25.23	25.18	24.62
Ciudad de Mexico (90, 20)	25.43	24.96	24.94	25.43
Ciudad de Mexico (90, 17)	11.81	11.31	11.85	11.80
Minneapolis (116, 30)	2.97	2.97	2.97	2.97
Minneapolis (116, 20)	15.97	15.97	15.97	15.97
Minneapolis (116, 10)	19.44	19.14	18.83	19.44
Average gaps (%)	7.67	7.60	7.56	7.65

Table 2.5: Root node – Comparison of various separation procedures on the core formulation F3.

The results reported in Table 2.5 demonstrate that the average gap of the core formulation F3 is close to 7% but slightly greater than 25% in the worst case – Ciudad De Mexico (90, 20). The average gap Gap_{core} is only slightly improved by separately adding either separation S1, S2 or S5; however, it is worth noting that separations S1, S2 and S5 are computed very fast. The importance of separations S3 and, more clearly, S4 is evident from the results presented in Table 2.6, namely, removing S3 and S4 from the full formulation F3 leads to an increase of the percentage gap from an average value of $Gap_{full} = 7.53\%$ to $Gap^-(\text{S3}) = 8.18\%$ and $Gap^-(\text{S4}) = 14.56\%$, respectively. Notably, this gap increases from 18.52% to 36.39% for separation S4 in the case of Minneapolis (116, 10).

Based on the overall computational results presented in this section, we note that the computational difficulty of the instances grows not only with the number of vertices, as it is expected, but also when the capacity of the vehicles decreases. This behaviour was also noted for the 1-PDTSP by Hernández-Pérez and Salazar-González [15] and is due to the fact that small capacity vehicles are forced to perform long and complicated routes for pickups and deliveries. An illustrative example is given in Figure 2.4, which depicts the three optimal solutions obtained for the problem instance

City ($\|V\|$, Q)	Gap_{full}	Gap^-(S1)	Gap^-(S2)	Gap^-(S3)	Gap^-(S4)	Gap^-(S5)
Dublin (45 ,30)	1.45	1.45	1.45	3.75	3.47	1.45
Dublin (45 , 20)	3.68	3.68	4.31	6.38	12.08	3.68
Dublin (45, 11)	5.16	4.43	5.61	5.26	13.95	5.16
Denver (51, 30)	0.52	0.52	0.52	1.31	1.64	0.52
Denver (51, 20)	2.56	2.56	2.56	2.60	3.33	2.58
Denver (51, 10)	3.49	4.29	2.98	4.82	6.65	3.49
Rio de J. (55, 30)	2.05	2.05	2.05	2.05	6.45	2.05
Rio de J. (55, 20)	2.51	2.51	2.51	2.51	10.56	2.51
Rio de J. (55, 10)	3.30	3.35	3.41	3.30	21.50	3.30
Boston (59, 30)	1.76	1.76	1.76	2.75	1.68	1.76
Boston (59, 20)	3.45	3.11	3.45	4.08	3.88	3.45
Boston (59, 16)	3.87	4.13	4.25	4.48	4.33	3.87
Torino (75, 30)	1.35	1.35	1.35	2.50	1.61	1.35
Torino (75, 20)	3.81	3.30	3.81	4.68	4.41	3.81
Torino (75, 10)	11.56	11.50	11.44	12.84	15.15	11.56
Toronto (80, 30)	5.55	5.55	5.55	7.25	6.08	5.55
Toronto (80, 20)	13.30	14.10	13.48	14.62	14.90	13.30
Toronto (80, 12)	14.99	14.97	14.87	15.36	18.99	14.99
Miami (82, 30)	3.28	3.28	3.28	3.36	20.32	3.28
Miami (82, 20)	9.14	9.25	9.14	9.17	28.36	9.17
Miami (82, 10)	7.00	6.86	7.03	7.01	35.36	7.00
Ciudad de M. (90, 30)	25.46	25.18	25.23	25.51	32.58	25.46
Ciudad de M. (90, 20)	25.04	24.94	24.96	25.04	34.97	25.04
Ciudad de M. (90, 17)	11.48	11.48	11.31	11.50	24.47	11.55
Minneapolis (116, 30)	2.97	2.97	2.97	3.59	6.73	2.97
Minneapolis (116, 20)	15.97	15.97	15.97	16.50	23.42	15.97
Minneapolis (116, 10)	18.52	18.83	19.14	18.72	36.39	18.83
Average gaps (%)	7.53	7.53	7.57	8.18	14.56	7.54

Table 2.6: Root node – Comparison of various separation procedures on the full formulation F3.

arising in Reggio Emilia for the following cases:

(*i*) Q=10, three vehicle routes are needed. The first one visits stations 2, 6, 13, 5, 8 and 7, the second vehicle visits stations 1, 3, 10, 4, 9, and 11 but the third one visits only station 12. The total distance travelled is 32.5 km.

(*ii*) Q=20, two vehicle routes are needed. The first one visits stations 13, 2, 10, 4, 6, 3, 1, 5, 8 and 7 while stations 9, 11, and 12 are visited by the second vehicle. Total distance travelled is 23.2 km.

(*iii*) Q=30, a single vehicle route visits all stations (traveling for 16.9 km) in the following sequence: 7, 5, 8, 13, 2, 10, 4, 6, 3, 1, 11, 9, and 12.

2.7.1 Randomly generated instances

It is well known that TSP formulations perform differently on clustered and randomized data set. To this end, we created a set of random instances, in addition to the real-world ones, to evaluate the quality of our algorithm.

				F1		F2		F3		F4	
Name	$\|V\|$	Q	UB	Gap_1	Time	Gap_2	Time	Gap_3	Time	Gap_4	Time
*R*40_30	40	30	59429	1.45	3600.00	0.00	451.32	0.00	13.23	0.00	241.58
*R*40_20	40	20	67789	1.92	3600.00	0.00	1171.56	0.00	115.25	0.00	845.35
*R*40_10	40	10	102941	3.85	3600.00	0.00	890.31	0.00	217.83	0.00	842.11
*R*45_30	45	30	52980	0.00	70.23	0.00	42.42	0.00	2.81	0.00	115.88
*R*45_20	45	20	58204	0.00	825.68	0.00	1860.36	0.00	56.08	0.63	3600.00
*R*45_10	45	10	75223	2.42	3600.00	2.84	3600.00	0.00	591.15	3.18	3600.00
*R*50_30	50	30	60960	0.00	14.59	0.00	52.95	0.00	7.83	0.00	210.09
*R*50_20	50	20	67212	0.00	2348.79	1.19	3600.00	0.00	41.82	0.95	3600.00
*R*50_10	50	10	94465	4.55	3600.00	1.23	3600.00	0.00	248.73	0.28	3600.00
*R*55_30	55	30	67491	0.00	872.29	0.00	403.45	0.00	11.83	0.00	1155.85
*R*55_20	55	20	74429	0.00	2744.08	0.00	1143.11	0.00	33.71	0.00	1333.42
*R*55_10	55	10	103117	12.37	3600.00	1.86	3600.00	1.07	3600.00	2.17	3600.00
*R*60_30	60	30	76574	6.15	3600.00	3.08	3600.00	0.00	1025.23	3.52	3600.00
*R*60_20	60	20	89762	9.96	3600.00	4.39	3600.00	0.88	3600.00	3.27	3600.00
*R*60_10	60	10	133824	12.95	3600.00	1.90	3600.00	0.78	3600.00	1.78	3600.00
Average Gaps and Times				3.71	2618.38	1.10	2081.03	0.18	877.70	1.05	2236.28
Opt				**6**		**8**		**12**		**7**	
#Nodes				7741.20		6479.47		4664.30		4958.07	
#Cuts				2822.73		903.27		1145.80		758.67	

Table 2.7: Tests on randomly generated instances

We followed a classical random generation method adopted in the CVRP literature, see e.g. Toth and Vigo [31]. We randomly generated instances including 40, 45, 50, 55 and 60 vertices, because problems of this size are challenging to solve for real-world cases. The coordinates of the vertices are randomly generated between 0 and 100, while the depot is set in (50, 50). Distances between vertices are computed as the Euclidean ones, but perturbed by a factor δ, randomly chosen between -20% and 20% to induce asymmetry on the costs. In more detail, let b_{ij} be the Euclidean distance between the two vertices i and j, for $i, j, \in V, j > i$. The cost matrix is computed by setting $c_{ij} = b_{ij}(1 + \delta)$, for $i, j \in V$, then the resulting c_{ij} values are multiplied by 100 and rounded up to the next integer value. A check on triangularity of distances is then performed to adjust non-triangular distances. We considered 10, 20 and 30 as plausible values for Q. The demand of each vertex is randomly generated between -10 and 10 but other than zero, while the demand of the depot is set to zero.

In Table 2.7 we report the instance name, the number of vertices, the vehicle capacity, the best upper bound found and, for each formulation, the percentage gap between the best upper bound and the best lower bound, and the solution time. The average gaps and computational times are also reported. The number of instances solved to optimality is indicated by *Opt*, while $\#Nodes$ and $\#Cuts$ illustrate the average number of visited nodes and added cuts for each formulation. The time limit was set to 3600 seconds.

The results confirm the computational experience derived from solving the real-world instances. When increasing the number of vertices and the vehicle capacity the instances get harder to solve. The formulation having better results is, once again, F3. As reported in Table 2.7, 12 instances out of 15 are solved to optimality by F3, and the percentage gap between the best upper bound and the best lower bound is, on average, lower than

0.20%. The average solution time is less than 15 minutes. Overall, tests on randomly generated instances confirm the quality of the proposed algorithms and formulations.

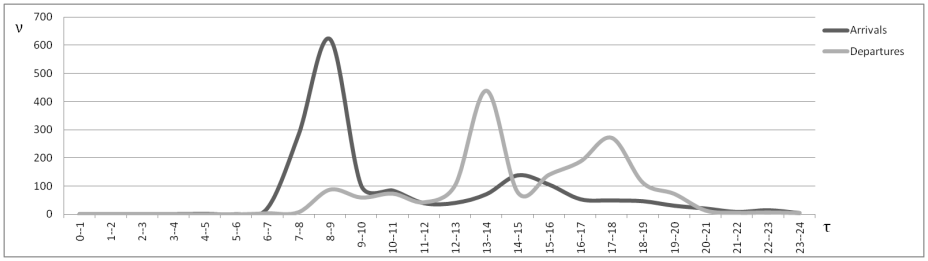
2.8 Conclusions

In this chapter we considered the Bike sharing Rebalancing Problem, which calls for the repositioning of public bikes of a bike sharing program at minimum cost, using a fleet of capacitated vehicles. We modeled the problem proposing four different formulations, which take into account different types of variables and constraints. All formulations involve an exponential number of constraints, so we solved them by using a branch-and-cut algorithm.

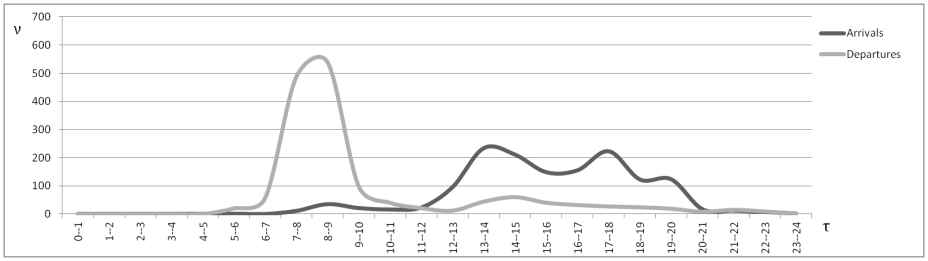
By collecting data from several web sites, we produced an interesting test bed containing 65 instances. We made the test bed publicly available, and used it to computationally evaluate the branch-and-cut algorithm. Computational evidence suggests that the best computational results are produced by formulation F3. This formulation efficiently solves instances with up to 50 vertices, but may fail in solving some larger ones in one hour of computational time on a standard PC.

Acknowledgments

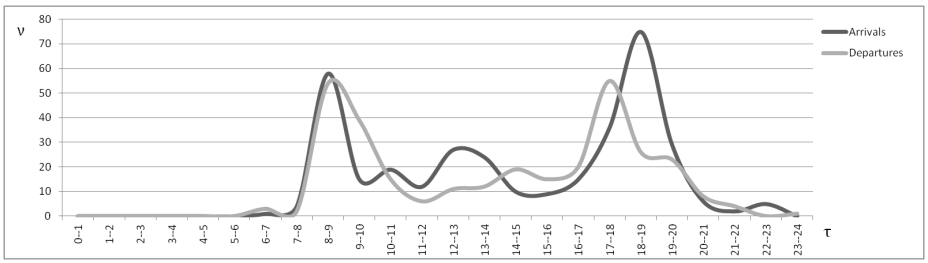
We thank the Mobility Concilorship of Reggio Emilia, and Alessandro Meggiato, Mobility Manager of the Municipality of Reggio Emilia, for the data and information supplied. We also thank Scott Mullen (General Manager of Hubway, Boston bike sharing system), Matt Verlee (Customer Relations & Low Income Outreach Manager of Denver B-Cycle), Marco Verità (of Brescia Mobilità, Brescia bike sharing system), Ben Kunde (Tech Manager, Madison B-Cycle), Caleb Cohate (San Antonio B-Cycle), Sonia Fakiel (Mejor en Bici, Buenos Aires bike sharing system) and ATB Mobilità (La BiGi, Bergamo bike sharing system) for the information they furnished us with.



(a)



(b)



(c)

Figure 2.3: Typical arrivals and departures per hour in a station of (a) the first group, (b) the second group and (c) the third group (τ = hour of the day, ν = cumulative number of bikes arriving into or departing from a station within a seven-months period).

(a)

(b)

(c)

Figure 2.4: Optimal solutions for Reggio Emilia with (a) Q=10, (b) Q=20, and (c) Q=30.

Bibliography

[1] N. Ascheuer, M. Fischetti, M. Grötschel. A polyhedral study of the asymmetric travelling salesman problem with time windows. *Networks*, 36:69–79, 2000.

[2] R. Baldacci, E. Hadjiconstantinou, A. Mingozzi, An exact algorithm for the capacitated vehicle routing problem based on a two-commodity network flow formulation, *Operations Research*, 52:723–738, 2004.

[3] M. Battarra, J.-F. Cordeau, and M. Iori. *Vehicle Routing: Problems, Methods, and Applications*, Chapter 6: Pickup-and-Delivery Problems for Goods Transportation. In [32], 2014.

[4] T. Bektas, The multiple traveling salesman problem: an overview of formulations and solution procedures, *Omega*, 34:209 – 219, 2006.

[5] M. Benchimol, P. Benchimol, B. Chappert, A. De La Taille, F. Laroche, F. Meunier, L. Robinet, Balancing the stations of a self service "Bike Hire" system, *RAIRO-Operations Reserach*, 45:37–61, 2011.

[6] G. Berbeglia, J.-F. Cordeau, I. Gribkovskaia, G. Laporte, Static pickup and delivery problems: A classification scheme and survey, *TOP*, 15:1–31, 2007.

[7] F. Boctor, J. Renaud, F. Cornillier, Trip packing in petrol stations replenishment, *Omega*, 39:86 – 98, 2011.

[8] D. Chemla, F. Meunier, R. Wolfler Calvo, Bike sharing systems: Solving the static rebalancing problem, *Discrete Optimization*, 10:120–146, 2013.

[9] C. Contardo, C. Morency, L.-M. Rousseau, Balancing a Dynamic Public Bike-Sharing System, Technical Report CIRRELT-2012-09, CIRRELT, 2012.

[10] P. DeMaio, Bike Sharing: History, Impacts, Model of Provision and Future, *Journal of Public Transportation*, 10:41–56, 2009.

[11] K. Doerner,J.-J. Salazar-González *Vehicle Routing: Problems, Methods, and Applications*, Chapter 7: Pickup-and-Delivery Problems for People Transportation. In [32], 2014.

[12] G. Gutin, A. Punnen (eds.), *The Traveling Salesman and its Variations*, Kluwer, Dordrecht, 2002.

[13] M. Hosny, C. Mumford, Solving the one-commodity pickup and delivery problem using an adaptive hybrid VNS/SA approach, in: R. Schaefer, C. Cotta, J. Kolodziej, G. Rudolph (eds.), *Parallel Problem Solving from Nature*, PPSN XI, volume 6239 of *Lecture Notes in Computer Science*, Springer, Berlin, 2010, pp. 189–198.

[14] H. Hernández-Pérez, I. Rodríguez-Martín, J.-J. Salazar-González, A hybrid GRASP/VND heuristic for the one-commodity pickup-and-delivery traveling salesman problem, *Computers & Operations Research*, 36:1639–1645, 2009.

[15] H. Hernández-Pérez, J.-J. Salazar-González, A branch-and-cut algorithm for a traveling salesman problem with pickup and delivery, *Discrete Applied Mathematics*, 145:126–139, 2009.

[16] H. Hernández-Pérez, J.-J. Salazar-González, Heuristics for the one-commodity pickup-and-delivery traveling salesman problem, *Transportation Science*, 38:245–255, 2004.

[17] H. Hernández-Pérez, J.-J. Salazar-González, The one-commodity pickup-and-delivery traveling salesman problem: Inequalities and algorithms, *Networks*, 50:258–272, 2007.

[18] A. Kaltenbrunner, R. Meza, J. Grivolla, J. Codina, R. Banchs, Urban cycles and mobility patterns: Exploring and predicting trends in a bicycle-based public transport system, *Pervasive and Mobile Computing*, 6:455–466, 2010.

[19] J.-R. Lin, T. Yang, Strategic design of public bicycle sharing systems with service level constraints, *Transportation Research Part E* 47:284–294, 2010.

[20] G. Martinovic, I. Aleksi, A. Baumgartner, Single-commodity vehicle routing problem with pickup and delivery service, *Mathematical Problems in Engineering*, Article ID 697981, 2008.

[21] C. Miller, A. Tucker, R. Zemlin, Integer programming formulations and traveling salesman problems, em J. Association Comput. Machinery 7:326–329, 1960.

[22] N. Mladenović, D. Urošević, S. Hanafi, A. Ilić, A general variable neighborhood search for the one-commodity pickup-and-delivery travelling salesman problem, *European Journal of Operational Research* 220:270–285, 1960.

[23] OBIS Project, Optimization Bike Sharing in European cities. A handbook, 2011.

[24] S. N. Parragh, K. F. Doerner, R. F. Hartl, A survey on pickup and delivery problems. Part I: Transportation between customers and depot, *Journal für Betriebswirtschaft*, 58:21–51, 2008.

[25] S. N. Parragh, K. F. Doerner, l. R. F. Hart, A survey on pickup and delivery problems. Part II: Transportation between pickup and delivery locations, *Journal für Betriebswirtschaft* 58:81–117, 2008.

[26] J. Pucher, R. Buehler, M. Seinen, Bicycling renaissance in North America? An update and re-appraisal of cycling trends and policies, *Transportation Research Part A*, 45:451–475, 2011.

[27] T. Raviv, M. Tzur, I. Forma, Static repositioning in a bike-sharing system: models and solution approaches, *EURO Journal of Transportation and Logistics*,2:187–229,2013.

[28] S. Shaheen, S. Guzman, H. Zhang, Bikesharing in Europe, the Americas, and Asia: Past, Present, and Future, *Transportation Research Record* 2143:159–167, 2010.

[29] W. Souffriau, P. Vansteenwegen, G. Vanden Berghe, D. Van Oudheusden, The planning of cycle trips in the province of East Flanders, *Omega* 39:209–213, 2011.

[30] X. Shi, F. Zhao, Y. Gong, Genetic algorithm for the one-commodity pickup-and-delivery vehicle routing problem, in: *IEEE International Conference on Intelligent Computing and Intelligent Systems*, volume 1: 175–179, 2009.

[31] P. Toth and D. Vigo. (eds.) *The Vehicle Routing Problem.* SIAM, 2002.

[32] P. Toth and D. Vigo. (eds.) *Vehicle Routing: Problems, Methods, and Applications.* SIAM, 2014.

[33] P. Vogel, D. Mattfeld, Modeling of repositioning activities in bike-sharing systems, in: *12th WCTR, Lisbon, Portugal*, 1–13, 2010

[34] F. Wang, A. Lim, Z. Xu, The one-commodity pickup and delivery travelling salesman problem on a path or a tree, *Networks* 48:24–35, 2006.

[35] F. Zhao, S. Li, J. Sun, D. Mei, Genetic algorithm for the one-commodity pickup-and-delivery traveling salesman problem, *Computers & Industrial Engineering*, 56:1642–1648, 2009.

Chapter 3

A Destroy and Repair Algorithm for the Bike sharing Rebalancing Problem

In this chapter, we deal with the Bike sharing Rebalancing Problem (BRP), the problem of driving a fleet of capacitated vehicles to redistribute bicycles among the stations of a bike sharing system that we have already described in Chapter 2. We tackle the BRP with a newly developed metaheuristic algorithm that includes the use of an effective new constructive heuristic and of seven local search procedures for which we propose some techniques to reduce the complexity of the neighborhood exploration. In addition, we adapted the algorithm for the BRP to solve the one-commodity Pickup and Delivery Vehicle Routing Problem (1-PDVRP), a problem similar to the BRP where a maximum duration is imposed for each route. Moreover, we adapted an available branch-and-cut algorithm for the BRP to solve the 1-PDVRP and we show how to effectively handle the duration constraint in a two-index formulation for the general VRP. Our metaheuristic algorithm for the BRP improved the best known solutions of instances from the literature. We also tested our algorithm on newly collected, very large real-world instances obtaining good quality solutions. Morover, we solved literature 1-PDVRP instances with our algorithm improving remarkably the quality of the solutions and obtaining some new solution that were proved to be optimal by the developed branch-and-cut algorithm.

Keywords: Bike sharing, Rebalancing, Metaheuristic, 1-PDVRP, Speed-up, Maximum duration, Branch-and-cut

3.1 Introduction

Bike sharing systems are public or private systems designed to increase the use of bicycles and decrease congestion, to solve the last mile problem, and to provide a mobility service to the users where other means of transportation are not available. These systems were born in the 1960s in Amsterdam (see, e.g., DeMaio [9]) and spread all over the world counting more than 700 operating systems and more than 200 systems planned, under construction or about to be implemented (see e.g., DeMaio and Meddin [10]).

Bike sharing systems are composed of several bike stations located in different sites of the city and each station is composed of a number of bike slots where bicycles can be returned or collected. In a balanced situation, each bike station must have a certain number of empty slots, to allow arrivals, and a certain number of full slots, to allow departures.

Let us define the *level of occupation* of a station as the number of bicycles out of the number of slots present in that station, and the *balanced level of occupation* as the level of occupation that the system operator desires in a station in a balanced situation. After a certain amount of time, the users will have moved the bicycles among the stations of the system and the level of occupation will have gone far from the balanced one. For example, the stations on the top of a hill will be typically empty and the stations at the bottom will typically be full. This situation creates inefficiency in the system such that users cannot collect or return bicycles when needed. The system operators want the stations to be brought to their balanced levels of occupation to avoid the inefficiency created by full and/or empty stations and they perform a redistribution of bicycles that is called rebalancing. The rebalancing is performed by vehicles and it is normally required every day at the end of the day, when the system is closed or when the use of the system can be considered negligible. In this case, the rebalancing is called static. Some bike sharing operators may require that the rebalancing is performed when the system is open, that is a dynamic rebalancing. In this chapter we study the static case.

The *Bike sharing Rebalancing Problem* (BRP) is a problem related to the static rebalancing of a bike sharing system and, in particular, drives a fleet of homogeneous and capacitated vehicles to rebalance the stations of a bike sharing system at minimum cost. In this chapter, we present a new constructive heuristic and a set of efficient local searches that are included into a metaheuristic algorithm for the BRP. This new algorithm provides shorter solving times for the easiest instances and better upper bounds for the hardest instances when compared to the branch-and-cut algorithm developed by Dell'Amico et al. [8] and good quality solutions for newly collected, large real-world instances. Moreover, we adapted the metaheuristic algorithm to the *one-commodity Pickup and Delivery Vehicle Routing Problem* (1-PDVRP), a variation of the BRP where a maximum duration constraint

is imposed to each route. Instances of the 1-PDVRP from the literature have been solved and the best known solutions been strongly improved. When solving the 1-PDVRP, we adapted a previously available branch-and-cut algorithm and we show how to efficaciously handle the maximum duration constraint in a two-index formulation for general VRP problems. We believe this to be an additional interesting contribution in the field of capacitated vehicle routing.

The chapter is structured as follows. In Section 4.3, we formally define the BRP and its variation with maximum duration constraint, known as the 1-PDVRP. A mathematical model is proposed for both problems and a brief literature review is reported. In Section 3.3, we describe some properties of the BRP that allow us to speed up local searches. In Section 3.4, we present the framework of the proposed algorithm for the BRP and we describe its components. In Section 3.5, we describe the adaptation of the algorithm to the 1-PDVRP and the adaptation of a branch-and-cut algorithm by Dell'Amico et al. [8] to solve the same problem. In Section 3.6, extensive computational results on newly collected real instances and on literature instances are presented. We conclude in Section 7.7.

3.2 Problem description

The *Bike sharing Rebalancing Problem* (BRP) is modeled on a complete digraph $G = (V, A)$, where $V = \{0, 1, \ldots, n\}$ is the set of vertices including the n stations and the depot (vertex 0) and A is the set of arcs between each pair of vertices. Let c_{ij} be the non-negative cost associated with the arc $(i, j) \in A$. For each vertex $i \in V$ a request q_i is given. Requests can be positive or negative. If a station has a positive request, we call it pickup station; if it has a negative request we call it delivery station. The quantities picked up at pickup stations can be used to respond to requests of delivery stations or go to the depot. Vehicles can leave the depot not necessarily empty, if needed. The objective is to drive a fleet of m identical vehicles of capacity Q available at the depot to respond to requests and to minimize the total cost. We impose that a station with request $q_i = 0$ must be visited, even if this implies that no bike has to be dropped off or picked up there. The case in which stations with null request have to be skipped can be simply obtained by removing in a preprocessing phase those stations from the set of vertices.

3.2.1 Formulation for the BRP

In this section, we present a *mixed integer linear programming* (MILP) formulation to solve the BRP. The starting point is *Formulation F3* by Dell'Amico et al. [8] built on the *multiple traveling salesman problem* (m-TSP), in which at most m uncapacitated vehicles based at a central depot

have to visit a set of vertices, with the constraint that each vertex is visited exactly once. By defining a binary variable x_{ij}, taking value 1 if arc (i, j) is traveled by a vehicle, and 0 otherwise, the BRP can be modeled as (3.1)–(3.6).

$$\min z = \sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij} \tag{3.1}$$

$$\sum_{i \in V} x_{ij} = 1 \qquad j \in V \setminus \{0\} \tag{3.2}$$

$$\sum_{i \in V} x_{ji} = 1 \qquad j \in V \setminus \{0\} \tag{3.3}$$

$$\sum_{j \in V} x_{0j} \leq m \tag{3.4}$$

$$\sum_{i \in S} \sum_{j \in S} x_{ij} \leq |S| - \max \left\{ 1, \left\lceil \frac{\left| \sum_{i \in S} q_i \right|}{Q} \right\rceil \right\} \qquad S \subseteq V \setminus \{0\}, S \neq \emptyset. \tag{3.5}$$

$$x_{ij} \in \{0, 1\} \qquad i, j \in V. \tag{3.6}$$

Objective function (3.1) minimizes the traveling costs. Constraints (3.2) and (3.3) impose that every node but the depot is visited once. Constraints (3.4) ensure that at most m vehicles leave the depot. To guarantee the feasibility of a solution with respect to the BRP, we need to substitute the typical subtour elimination constraints by the family of constraints (3.5) in the m-TSP formulation, so as to ensure that requests are satisfied and vehicles capacities are not exceeded. Constraints (3.5) are similar to the *generalized subtour elimination constraints* for the CVRP. They state that, for each subset S of vertices, the number of arcs with both tail and head in S should not exceed the cardinality of S minus the minimum number of vehicles required to serve S. An estimation of the minimum number of vehicles is simply obtained by computing the absolute value of the sum of the requests, dividing it by the vehicle capacity and then rounding up the result. This value can be zero when the sum of the q_i is null; in such a case, the value one is used instead, because at least one vehicle is needed (notice that S does not contain the depot) to impose the connectivity of the solution. Constraints (3.5) are exponentially many, so the above formulation can be solved in *branch-and-cut* fashion, by invoking a separation procedure to divide the violated ones from the non-violated ones, and then adding the violated cuts to the model in an iterative way. For details we refer to Dell'Amico et al. [8] and to Section 3.5.2 below.

3.2.2 A special case of the BRP: the 1-PDVRP

The *one-commodity Pickup and Delivery Vehicle Routing Problem* (1-PDVRP) is a generalization of the BRP in which a maximum duration constraint is imposed for the routes. The problem was introduced by Shi et al. [32], where the authors impose a maximum distance MD to each route. In this chapter, we solve a more general version of the 1-PDVRP where the maximum distance MD is considered as a maximum duration in time, T, where a time t_{ij} of traveling the arc $(i, j) \in A$ is introduced and it is proportional to the cost between i and j such that $c_{ij} = \tau t_{ij}$. Moreover, we consider a service time s_i in each station that depends on the quantity to be picked up or dropped, such that $s_i = \sigma|q_i|$. The scalars τ and σ are non-negative. We solve the version of the 1-PDVRP that takes into account the traveling time t_{ij}, the service time s_i, and the maximum duration of a route T because it fits better with the BRP. The 1-PDVRP proposed by Shi et. al [32] is a particular case of our version, where $\tau = 1$, $\sigma = 0$, and $T = MD$.

When addressing the 1-PDVRP we need to express the maximum duration constraints. To do so, we define the set $\mathcal{P}$ of routes that identify paths starting and ending at the depot. We guarantee that each route does not exceed the maximum duration limit considering the traveling times of the arcs of the path P and the service times of the visited stations as follows:

$$\sum_{i \in P} \sum_{j \in P} (t_{ij} + s_j) x_{ij} \leq T \qquad P \in \mathcal{P}. \tag{3.7}$$

Note that constraints (3.7) are exponentially many, and, hence, as already observed for (3.5), we need a separation procedure that divides violated constraints from non-violated ones.

3.2.3 Prior Work

The BRP and the 1-PDVRP belong to the class of *Pickup and Delivery Vehicle Routing Problems* (PDVRP), and generalize the *Capacitated Vehicle Routing Problem* (CVRP). Thus, they are both strongly NP-hard problems.

According to the classification introduced by Berbeglia et al. [3] and followed by Battarra et al. [1] the BRP is a *Many-to-Many* (M-M) vehicle routing problem, where M-M means that the origins and destinations of the requests are multiple. The simplest M-M PDVRP is the *one-commodity Pickup and Delivery Traveling Salesman Problem* (1-PDTSP), which has been introduced by Hernández-Pérez and Salazar-González [18]. The 1-PDTSP aims at traveling a single capacitated vehicle to meet the multiple requests of a single commodity and minimize the costs. Hernández-Pérez and Salazar-González [16] presented asymmetric and symmetric formulations for the 1-PDTSP and solved the symmetric one by means of a branch-and-cut algorithm, which is then improved in Hernández-Pérez and Salazar-González [19]. Hernández-Pérez et al. [17, 15] presented also two simple

heuristics and a variable neighborhood descent. Other metaheuristic algorithms for the 1-PDTSP have been proposed, lately, by Martinovic et al. [25] (iterated modified simulated annealing), Zhao et al. [35] (genetic algorithm), Hosny and Mumford [20] (hybrid variable neighborhood search and simulated annealing approach), and Mladenović et al. [26] (variable neighborhood search).

The BRP is a generalization of the 1-PDTSP to the case of multiple vehicles, and has been formally introduced by Dell'Amico et al. [8]. They proposed four formulations for the BRP and solved them by branch-and-cut. They evaluated the algorithms on 65 real-world instances and 15 random instances solving to optimality all instances with up to 51 vertices.

The multiple vehicle case with a maximum duration limit imposed for every route, i.e., the 1-PDVRP, has been formally introduced by Shi et al. [32]. They proposed a three-index formulation for which they did not report any computational evaluation. They also presented a genetic algorithm and tested it on a set of randomly generated symmetric instances derived from those proposed by Hernández-Pérez and Salazar-González [17].

Many other papers deal with the balancing of bike sharing systems considering different aspects. We report a relevant collection of these papers by dividing them in static rebalancing with split delivery, rich static rebalancing, and dynamic rebalancing. The *static rebalancing with split delivery*, the case where customers can be visited more than once and their requests split, is examined in Benchimol et al. [2]. They present complexity results, lower bounding techniques and approximation algorithms for the single vehicle case where the sum of all requests is equal to zero. A computational evaluation of the techniques was not given. Chemla et al. [5] also considered the single vehicle case where split delivery is allowed, but they imposed a limit on the maximum number of times a vertex can be visited. Stations with null requests can be used as buffers. The authors presented a formulation and a tabu search algorithm.

Several papers on *rich static rebalancing* problems have been presented recently. We refer to rich static rebalancing problems as those where the static rebalancing problem is considered and many features, such as transshipment, different objectives, multiple depots, and inventory policies, in addition to split delivery, are accounted. Raviv et al. [28] defined the *static bicycle repositioning problem* as the problem of minimizing the traveling costs of a fleet of heterogeneous vehicles and the users' dissatisfaction that is linked to the inventory level of each station. They presented two MILP formulations allowing, respectively, limited and unlimited split delivery and transshipment, and considering also a maximum duration for each route. They also developed exact and heuristic methods to solve the presented formulations. Rainer-Harbach et al. [27] solved another rich static problem defined as the *balancing bicycle sharing system problem*, which intends to find the routes of an heterogeneous fleet of vehicles to minimize the devia-

tion from the target level of request of each station. They also considered, as secondary objectives, the duration of the routes and the number of loading actions performed. The problem allows split delivery and transshipment and it is imposed that the vehicles return empty to the depot. The authors proposed a greedy constructive heuristic and some metaheuristic algorithms. Di Gaspero et al. [12] solved a similar rich static rebalancing problem where split delivery is allowed but intermediate storages are not. The problem considers multiple depots and multiple vehicles, and a maximum duration time imposed on each vehicle. The aim is to find routes that operate pick-ups and deliveries to bring the level of occupation in each station as close as possible to the balanced one and that minimize the total traveling time. The authors introduced two different constraint programming models and proposed two branching strategies. Then they included the constraint programming models in a large neighborhood search approach obtaining good computational results. Schuijbroek et al. [34] combined the inventory policy and the routing optimization. The authors solved the problem of minimizing the makespan of the routes respecting a lower and a upper bound on the number of bikes required in each station. The stations that do not need a visit can be used as a buffer, the routes can be open and a limited number of transshipment is allowed. The authors proposed a MILP formulation, a constraint programming approach and a cluster-first route-second heuristic.

A solution method for the *dynamic rebalancing*, where requests can change during the rebalancing operations, has been presented by Contardo et al. [7]. The objective is to maximize the serviced requests by using a set of vehicles. The authors presented a formulation using discretized time and solved it heuristically with Dantzig-Wolfe and Benders decompositions.

3.3 Properties of feasible paths for the BRP

In this section, we present some properties of feasible BRP paths that allow us to speed up the computation of several procedures that are at the basis of our metaheuristic algorithm. These procedures are formally defined in Section 3.4 and they are all based on the following set of operators:

- *Remove* a vertex i from its current position in the solution;
- *Insert* a vertex i in a position;
- *Move* a vertex i from its position to another one (composition of remove and insert);
- *Swap* two vertices i and j by moving i to the position of j and j to the position of i (once again, composition of remove and insert);
- *Merge* two (partial) routes.

Our speed-up techniques build on the basic route feasibility property presented in [17], and that we recall here briefly. We recall that a path P is an ordered sequence of vertices $P(0), P(1), \ldots, P(|P|)$, while a route is a path that starts and ends at the depot, i.e., $P(0) = P(|P|) = 0$. Let us first compute the cumulative request along a route by using the following recursion

$$l_{P(0)}(P) = 0, \text{ and } l_{P(i)}(P) = \sum_{k=0}^{i} q_{P(k)} \text{ for } i = 1, \ldots, |P|. \tag{3.8}$$

In other words $l_{P(i)}(P)$ gives the value of the load on the vehicle *after* visiting vertex $P(i)$ of path P, if the vehicle left the depot with no load. Negative $l_{P(i)}(P)$ values may still lead to feasible routes, because the initial load of the vehicle is not restrained to be 0. In particular, a route is feasible if the following inequality is satisfied

$$\max_{i=0}^{|P|-1}\{l_{P(i)}(P)\} - \min_{i=0}^{|P|-1}\{l_{P(i)}(P)\} \leq Q. \tag{3.9}$$

In the following we make use of a parameter Δ_P, that we call the *amount of feasibility* of P and set as

$$\Delta_P = Q - \max_{i=0}^{|P|-1}\{l_{P(i)}(P)\} + \min_{i=0}^{|P|-1}\{l_{P(i)}(P)\}. \tag{3.10}$$

Intuitively, routes with a small Δ_P value have a tight constraint on the choice of the initial load. This parameter can be used to define some sufficient conditions, called Δ-*checks* in the following, for quick feasibility checks of the previously introduced neighborhood operators.

Property 1 *If $|q_i| \leq \Delta_P$, then vertex i can be feasibly inserted in any position of route P.*

Proof. Let P' be the route obtained after the insertion of vertex i in route P. By applying (3.9), P' is feasible if

$$\max_{k=0}^{|P'|-1}\{l_{P'(k)}(P')\} - \min_{k=0}^{|P'|-1}\{l_{P'(k)}(P')\} \leq Q. \tag{3.11}$$

To prove that (3.11) holds if the hypothesis is satisfied, let us consider two cases:

-) $q_i \geq 0$. In this case, with respect to the route P, the insertion of vertex i may reduce or keep unchanged both the max and the min terms in (3.11). The maximum difference between the two emerges when the max increases by the maximum possible quantity, i.e., by q_i, and the min does not change. We can thus bound this difference as

$$\max_{k=0}^{|P'|-1}\{l_{P'(k)}(P')\} - \min_{k=0}^{|P'|-1}\{l_{P'(k)}(P')\} \leq$$
$$\left(\max_{k=0}^{|P|-1}\{l_{P(k)}(P)\} + q_i\right) - \min_{k=0}^{|P|-1}\{l_{P(k)}(P)\} =$$
$$\max_{k=0}^{|P|-1}\{l_{P(k)}(P)\} + q_i - \min_{k=0}^{|P|-1}\{l_{P(k)}(P)\} \leq$$
$$\max_{k=0}^{|P|-1}\{l_{P(k)}(P)\} + \Delta_P - \min_{k=0}^{|P|-1}\{l_{P(k)}(P)\} = Q,$$

where the last two steps follows, respectively, from the hypothesis and from (3.10).

-) $q_i < 0$. In this case we consider instead that the maximum difference between the two terms in (3.11) is obtained when the min term decreases by q_i and the max remains unchanged, thus we can state that

$$\max_{k=0}^{|P'|-1}\{l_{P'(k)}(P')\} - \min_{k=0}^{|P'|-1}\{l_{P'(k)}(P')\} \leq$$
$$\max_{k=0}^{|P|-1}\{l_{P(k)}(P)\} - \left(\min_{k=0}^{|P|-1}\{l_{P(k)}(P)\} + q_i\right) \leq$$
$$\max_{k=0}^{|P|-1}\{l_{P(k)}(P)\} - \min_{k=0}^{|P|-1}\{l_{P(k)}(P)\} - q_i \leq$$
$$\max_{k=0}^{|P|-1}\{l_{P(k)}(P)\} - \min_{k=0}^{|P|-1}\{l_{P(k)}(P)\} + \Delta_P = Q,$$

and the thesis is verified. □

Similar proofs, reported in the Annex 3.A, lead to the following results.

Property 2 *Given a feasible route P:*

a) If $i \in P$ and $|q_i| \leq \Delta_P$, then vertex i can be feasibly removed from route P;

b) If $i \in P$ and $|q_i| \leq \Delta_P$, then vertex i can be feasibly moved in any position along route P;

c) If $i, j \in P$ and $|q_i - q_j| \leq \Delta_P$, then the swap of vertices i and j is feasible;

Moreover, given a pair of feasible routes P and R:

d) If $i \in P$, $j \in R$, and $|q_i - q_j| \leq \min\{\Delta_P, \Delta_R\}$, then the swap of vertices i and j is feasible.

Proof. See the Annex 3.A. □

When the quick Δ-checks are not enough to certify feasibility of a move, an exact check must be performed. We consider more elaborated data structures to reduce the computational complexity of the exact check. In this sense, our contribution follows a well established line of research on difficult combinatorial problems such as the vehicle routing with time windows and scheduling with release and due dates (see, e.g., Ergun and Orlin [13], Liao et al. [23], and Ibaraki et al. [21]).

We use the concept of *load windows*, that express the feasible intervals of load that can be carried on the vehicle before or after visiting a vertex along a given route, by extending the basic result in (3.9). In particular, we use $F_{P(i)}(P) = [\underline{f}_{P(i)}(P), \overline{f}_{P(i)}(P)]$ to define the *forward load window* along route P, i.e., the feasible interval for the load on the vehicle after leaving vertex $P(i)$, and supposing that $P(i)$ is the last vertex of the route before returning to the depot. This can be obtained by

$$\begin{aligned} &F_{P(i)}(P) = \left[\underline{f}_{P(i)}(P), \overline{f}_{P(i)}(P)\right] = \\ &\left[l_{P(i)}(P) - \min_{j=0}^{i}\{l_{P(j)}(P)\}, l_{P(i)}(P) + Q - \max_{j=0}^{i}\{l_{P(j)}(P)\}\right]. \end{aligned} \tag{3.12}$$

The computation of (3.12) can be performed in linear time by using a simple structure that keeps track of the minimum and maximum values without recalculating them from scratch. Note that the amount of feasibility of P can be quickly computed as $\Delta_P = \overline{f}_{P(|P|-1)}(P) - \underline{f}_{P(|P|-1)}(P)$.

We similarly define the *backward load window* along route P, as $B_{P(i)}(P) = [\underline{b}_{P(i)}(P), \overline{b}_{P(i)}(P)]$. This load window gives the feasible interval for the load on the vehicle *before* it visits vertex $P(i)$, and considering that $P(i)$ is the first vertex in the route after the depot. This can be computed by first using a recursion that gives the cumulative load $r_{P(i)}$ in the reverse route starting from the depot and visiting the vertices in the order $P(|P|), P(|P|-1), \ldots, P(i)$. Formally

$$r_{P(|P|)}(P) = 0, \text{ and } r_{P(i)}(P) = -\sum_{k=|P|}^{i} q_{P(k)} \text{ for } i = |P|-1, \ldots, 0. \tag{3.13}$$

We thus compute, again in linear time, the backward load window as

$$\begin{aligned} &B_{P(i)}(P) = \left[\underline{b}_{P(i)}(P), \overline{b}_{P(i)}(P)\right] = \\ &\left[r_{P(i)}(P) - \min_{j=i}^{|P|}\{r_{P(j)}(P)\}, r_{P(i)}(P) + Q - \max_{j=i}^{|P|}\{r_{P(j)}(P)\}\right]. \end{aligned} \tag{3.14}$$

The forward and backward windows are used to construct the following property.

Property 3 *Two disjoint and feasible routes P and R can be merged in a feasible route $P \oplus R$, where the last vertex of P before returning to the depot, $P(|P|-1)$, is followed by the first vertex of R after the depot, $R(1)$, if and only if*

$$F_{P(|P|-1)}(P) \cap B_{R(1)}(R) \neq \emptyset.$$

Proof. See the Annex 3.A. □

By applying similar reasonings, we obtain the following additional results.

Property 4 *Given a feasible BRP solution, the following moves lead to another feasible solution.*

a) Removing the first vertex after the depot or the last vertex before the depot from a route is always feasible.

b) Removing a vertex $P(i)$ from a route P is feasible if and only if $F_{P(i-1)}(P) \cap B_{P(i+1)}(P) \neq \emptyset$

c) Let I be the single-vertex route formed by vertex i, i.e., $(0, i, 0)$. The insertion of i into a route P after a node $P(p) \in P$ is feasible if and only if the two following conditions are satisfied. The first condition is that i can be feasibly inserted after the first part of P, i.e., $F_{P(p)}(P) \cap B_i(I) \neq \emptyset$. If this is true, then let

$$F_i(P) = \left[\max\{\underline{f}_{P(p)}(P), \underline{b}_i(I)\} + l_i(I), \min\{\overline{f}_{P(p)}(P), \overline{b}_i(I)\} + l_i(I)\right],$$

and so obtain the second condition, that is $F_i(P) \cap B_{P(p+1)}(P) \neq \emptyset$.

d) Let $I = (0, i, 0)$ and $J = (0, j, 0)$. The swap of a vertex $i \in P$ (let $P(h) = i$) with another vertex $j \in R$ is feasible if and only if the following conditions are satisfied for route P and for route R. The first condition for route P is $F_{P(h-1)}(P) \cap B_j(J) \neq \emptyset$. If this is valid, then let us compute

$$F_j(P) = \left[\max\{\underline{f}_{P(h-1)}(P), \underline{b}_j(J)\} + l_j(J), \min\{\overline{f}_{P(h-1)}(P), \overline{b}_j(J)\} + l_j(J)\right].$$

The second condition is then $F_j(P) \cap B_{P(h+1)}(P) \neq \emptyset$. Similarly, being $R(k) = j$, for route R we have the following condition $F_{R(k-1)}(R) \cap B_i(I) \neq \emptyset$. If this is valid we can compute

$$F_i(R) = \left[\max\{\underline{f}_{R(k-1)}(R), \underline{b}_i(I)\} + l_i(I), \min\{\overline{f}_{R(k-1)}(R), \overline{b}_i(I)\} + l_i(I)\right].$$

Then the last condition to be satisfied is $F_i(R) \cap B_{R(k+1)}(R) \neq \emptyset$.

We conclude this section by discussing the update of the load windows. Let us consider, for example, that a merging of two routes P and R has been performed in this order. Let us consider the intersection between the forward load window of the last node before the depot of route P, $P(|P|-1)$, and the first node after the depot of route R, $R(1)$, that is: $F_{P(|P|-1)}(P) \cap B_{R(1)}(R) = \left[\max\{\underline{f}_{P(|P|-1)}(P), \underline{b}_{R(1)}(R)\}, \min\{\overline{f}_{P(|P|-1)}(P), \overline{b}_{R(1)}(R)\}\right] = \left[\underline{\phi}, \overline{\phi}\right]$. It is convenient to update the existing forward and backward load windows, without recomputing them from scratch. To update the forward load window for the vertices of route R we can compute: $F_{R(i)}(P \oplus R) = \left[\underline{\phi} + l_{R(i)}(R), \overline{\phi} + l_{R(i)}(R)\right]$, $i = 1, \ldots, |R|-1$, while for the vertices of route P the forward load window does not change. To update the backward load windows for the vertices of route P we can compute: $B_{P(j)}(P \oplus R) = \left[\underline{\phi} + r_{P(j)}(P), \overline{\phi} + r_{P(j)}(P)\right]$, $j = 1, \ldots, |P| - 1$, while for the vertices of route R the backward load window does not change.

3.4 Algorithm Framework

The overall framework of the algorithm that we use to solve the BRP is reported in Algorithm 1. It mainly consist of an iterated destroy and repair mechanism, enriched with local search procedures. The principle of using solution destruction with a subsequent solution re-construction or repair has been proposed in a number of different algorithms using names such as simulated annealing (see, e.g., Jacobs and Brusco [22]), ruin-and-recreate (see, e.g., Schrimpf et al. [33]), large neighborhood search (see, e.g., Shaw [31]), iterated greedy (see, e.g., Ruiz and Stützle [30]), iterative constructive search (see, e.g., Richmond and J. E. Beasley [29]), or iterative flattening (see, e.g., Cesta et al. [4]). We first construct a feasible initial solution by using a greedy algorithm (*Savings&Losses*, described in Section 3.4.1) and we refine it by using a set of local search procedures (explained in Section 3.4.4). This solution is then iteratively destroyed and repaired by using in order *DestroyProcedure* and *RepairProcedure* (presented in Sections 3.4.2 and 3.4.3, respectively), and again improved by the local search procedures, until a *StoppingCriterion* is reached. Note that our algorithm works with routes that are feasible with respect to the capacity constraint, but accepts solutions in which the number of routes is larger than m. This number is implicitly minimized by several of the adopted local search procedures.

3.4.1 Constructive algorithm

The constructive algorithm that we developed starts from the well-known *Savings* algorithm by Clarke and Wright [6], but adapts it to the new prob-

Savings&Losses $\rightarrow (x^*, z^*)$
LocalSearches $(x^*) \rightarrow (x_{ls}, z_{ls})$
if $z_{ls} < z^*$ **then**
  $z^* \leftarrow z_{ls}$ and $x^* \leftarrow x_{ls}$
end if
repeat
  DestroyProcedure $(x_{ls}) \rightarrow (\tilde{x})$
  RepairProcedure $(\tilde{x}) \rightarrow (x')$
  LocalSearches $(x') \rightarrow (x_{ls}, z_{ls})$
  if $z_{ls} < z^*$ **then**
    $z^* \leftarrow z_{ls}$ and $x^* \leftarrow x_{ls}$
  end if
until StoppingCriterion
return x^* and z^*

Algorithm 1: Algorithm DR_BRP

lem at hand by introducing the concept of *loss of flexibility*. We start by building n routes, each containing a single vertex. We then iteratively select two routes and merge them into a single one, until no more merging is possible.

To select the pair of routes to be merged, we take into account all possible combinations. When considering the merging of routes, say, P and R, we evaluate the feasibility of the resulting route $P \oplus R$ by making use of Property 3. If feasible, then we evaluate the saving obtained in the cost function, if any, as $S_{P\oplus R} = c_{P(|P|-1),R(1)} - c_{0,R(1)} - c_{P(|P|-1),0}$. We also evaluate the loss of flexibility induced by the merging, as the difference between the amount of feasibility (as defined in Section 3.3) of the two original routes, Δ_P and Δ_R, and that of the resulting route $\Delta_{P\oplus R}$, computed as $L_{P\oplus R} = \Delta_P + \Delta_R - 2\Delta_{P\oplus R}$. In practice, a high value of $L_{P\oplus R}$ means that the size of the resulting load window would be consistently reduced with respect to the sizes of the original windows, and so the resulting route would be harder to be feasibly merged with other routes in the successive iterations. We then set an evaluation function that takes both terms into account, as

$$E_{P\oplus R} = \alpha S_{P\oplus R} + (1-\alpha) L_{P\oplus R}, \tag{3.15}$$

where α is a parameter of the algorithm that takes values between 0 and 1. A feasible merge of two routes leading to the lowest value according to (3.15), is then selected. The value taken by α in our computational experiments is discussed in Section 3.6.

We can note that only the backward load window of the first node after the depot of route P and the forward load window of the last node before the depot of route R are needed to determine the feasibility of merging the

two routes for Savings&Losses. Let us call $F_{P(|P|-1)}(P) \cap B_{R(1)}(R) = \left[\underline{\phi}, \overline{\phi}\right]$, then we can compute: $F_{P \oplus R}(P \oplus R) = \left[\underline{\phi} + l_{R(|R|-1)}(R), \overline{\phi} + l_{R(|R|-1)}(R)\right]$ and $B_{P \oplus R}(1) = \left[\underline{\phi} + r_{P(1)}(P), \overline{\phi} + r_{P(1)}(P)\right]$.
Moreover $r_{P(1)}(P) = -l_{R(|R|-1)}(R)$ and $l_{P \oplus R}(|P \oplus R| - 1)(P \oplus R) = l_{R(|R|-1)}(R) + l_{P(|P|-1)}(P)$.

3.4.2 Destroy Procedure

To insert diversification into the metaheuristic algorithm, we make use of a component called *DestroyProcedure*. It randomly selects a number of vertices one at a time, independently one from the other and with uniform probability, and removes them from the current solution. The number of vertices to be removed is also selected randomly with a uniform probability in an interval defined by the parameters π and δ; more precisely: $[\max\{3, \pi - \delta\pi\}, \pi + \delta\pi]$. Each selected vertex is removed from its current route by applying the remove operator introduced in Section 3.3. If the selected vertex is the first one (right after the depot) or the last one (right before the depot) of a route, its removal can be operated without causing infeasibility on the remaining route because of Property 4-(a). In all other cases, we check the feasibility of the remaining route by checking Properties 2-(a) and 4-(b). If the remaining route is feasible, we perform the removal of the selected vertex; if it is infeasible, we perform the removal and divide the remaining route into the two separate routes defined by the removal (which are then feasible again because of Property 4-(b)). The values taken by the parameters π and δ were set after computational tests, as described in Section 3.6. We found it computationally convenient to halve/double the number of vertices to destroy every ten iterations of the main algorithm loop.

3.4.3 Repair Procedures

After using DestroyProcedure, we have a partial solution where not every vertex is allocated to a route, and some vertices that have not been assigned to any route. To restore feasibility, we apply two repair procedures that use the insert operator to include the non-assigned vertices into existing or newly formed routes. In details:

1. *RepairInsertion.* We randomly select, with uniform probability, one of the non-assigned vertices. We evaluate the feasibility and the cost of inserting this vertex in any position in any route, plus the option of creating a new route containing only this vertex. Among the feasible options, we select the one having minimum cost, and then re-iterate with the next non-assigned vertex until all of them are assigned. To check the feasibility of the insertion we make use of Properties 1 and 4-(c) of Section 3.3.

2. *Savings&Losses.* We use the constructive algorithm of Section 3.4.1, by considering the existing routes in the partial solution and each non-assigned vertex as if it were a single-vertex route. Merging of the routes follows the previously described process.

According to preliminary computational tests, we set RepairProcedure to execute at each iteration of the algorithm either RepairInsertion or Savings&Losses, by alternating them.

3.4.4 Local Search Procedures

To improve the solution obtained by the constructive algorithm and/or after a destroy and repair phase, we implemented several local search procedures based on the operators introduced in Section 3.3.

Apart from Δ-checks, load windows, and fast update techniques previously described, we also make use of a simple but effective idea called *Dont' look bit*, that we describe here for the insert operator. If we determine that the insertion of a vertex into a route is infeasible, then we keep track of this infeasibility as long as the route is not modified by other procedures, so as to avoid useless feasibility reevaluations. A slight variation applies to the swap neighborhood. We now give the details of the local search procedures that we implemented.

1. *MoveLS.* Select a vertex, remove it from its current position and insert it into another position, either in the same route or in a different one. Feasibility is quickly checked by means of Properties 1, 2-(a), 2-(b), 4-(b), and 4-(c). If the selected vertex is the only vertex in its route, then the route is deleted.

2. *MoveConsecutiveLS.* Select at most κ consecutive vertices in a route, remove them from their current position and insert them into another position, either in the same route or in a different one. Let us call P the route from which we remove the κ vertices and R the one in which we insert them. The feasibility of the removal is checked by merging the two remaining subroutes of P as discussed in Property 3. The feasibility of the insertion is instead performed in $O(\kappa)$, by first evaluating the κ forward and backward load windows of the removed vertices as if they were a single route, and then merging them with the first part of R by using again Property 3. If this is feasible, we update the load windows and check the feasibility of merging this new subroute with the second part of R. The value κ is a parameter of the algorithm to be defined after computational tests, as described in Section 3.6.

3. *SwapLS.* Select two vertices in the same route or in different routes and swap them. The feasibility is checked by applying Properties 2-(c),

2-(d), and 4-(d).

4. *SwapCoupleLS.* It extends the previous local search by selecting two pairs of consecutive vertices and swapping them, maintaining the same order of the vertices in each pair. The process for checking feasibility is similar to the one described for MoveConsecutiveLS, but simpler because only pairs of vertices are moved in this case and no longer lists of κ vertices.

5. *SwapThreeLS.* Select three nodes, say, v_1, v_2, and v_3, belonging to either, one, two, or three routes, and swap them in the two possible ways (v_1 with v_2, v_2 with v_3, v_3 with v_1, or v_1 with v_3, v_3 with v_2, v_2 with v_1). Use Properties 2-(c), 2-(d), and 4-(d) for feasibility check.

6. *CrossingLS.* Select two routes, P and R, and two positions along the routes. Cut each route right after the selected position, and merge the first part of P with the second part of R, and the first part of R with the second part of P. Use Property 3 for feasibility check. Note that a particular case arises when a position is at the very beginning of its route and the other position is at the very end of the other route, when CrossingLS consequently attempts a merging of the two routes into a single one.

7. *CrossingThreeLS.* It extends the previous local search, by selecting three routes, and attempting the two possible ways of cutting and merging them.

The number of routes in the solution is possibly decreased by the procedures MoveLS, MoveConsecutiveLS, CrossingLS, and CrossingThreeLS. Every procedure is run in first improvement fashion, which is computationally preferable to best improvement on our instances (as defined by the parameter ι). The procedures are inserted in a *Variable Neighborhood Descent* (VND) type framework (see, e.g., Hansen and Mladenović [14]), where they are invoked one after the other according to a given order. Our implementation is a VND type framework because we do not come back to the previous neighborhood if an improvement is found. In details, we continue invoking local searches following the given order until the last one, then we come back to the first local search procedure and repeat the order until no improvement in the current solution is provided by all local search procedures. This method is similar to the piped VND (see, e.g., den Besten and Stützle [11]), where the procedure stops at the end of the last local search procedure. According to the preliminary computational tests, the order of the local searches was set to 1, 3, 5, 4, 6, 2, and 7.

In terms of complexity, we first note that a straightforward implementation of MoveLS would require $O(n^3)$ time, because the number of possible moves is $O(n^2)$ (considering each vertex to be removed and each insertion

position), and evaluating each move would require $O(n)$ for computing feasibility of the at most two routes involved by the move. With the use of the load windows we reduce this complexity to $O(n^2)$, because feasibility is checked in $O(1)$. Similar reasonings apply to the next local searches, leading to the following complexities: $O(n^2)$ for MoveConsecutiveLS considering a fixed value of κ; $O(n^2)$ for SwapLS and SwapCoupleLS; $O(n^3)$ for SwapThreeLS; $O(n^2)$ for CrossingLS and $O(n^3)$ for CrossingThreeLS.

3.5 Adaptation to the 1-PDVRP

In this section, we describe how we adapted the algorithm presented in the previous section and the branch-and-cut of Dell'Amico et al. [8], both originally developed for the BRP, to include the maximum route duration constraint and to solve the 1-PDVRP.

3.5.1 Metaheuristic algorithm for the 1-PDVRP

To solve the 1-PDVRP making use of the metaheuristic algorithm developed for the BRP, we must perform additional tests when checking the feasibility of a solution, as the feasibility for the BRP is a necessary but not sufficient condition for the feasibility of the 1-PDVRP. Thus, after checking that a solution is feasible for the BRP, we evaluate that the sum of the travel and service times of each route does not exceed the maximum duration.

The inclusion of this additional check can be performed without increasing the complexity when using the data structure as above. This inclusion is easy for every component of our algorithm so we do not describe it in details. We mention here the fact that the remove operator is always feasible because of the triangular inequality that we assumed to be valid for the cost matrix, and thus the time matrix, whilst the insert operator needs an additional check on the times of the routes. For local searches based on the merge operator, it is convenient to keep track of the cumulative time (service and travel) from the depot to any node in a route. This is useful, for example, when performing a move of CrossingLS that attempts to merge the first part of a route P with the second part of a route R. The total duration of the resulting route can be obtained by summing the total time of the first part of P, the time to travel along the connecting arc, and the total time of the second part of R (obtained by subtracting the total time of the first part of R from the total time of R).

3.5.2 Branch-and-Cut for the 1-PDVRP

To solve to optimality model (3.1)-(3.7) for the 1-PDVRP, a branch-and-cut algorithm is required since an exponential number of constraints is considered. The branch-and-cut algorithm that we used is directly derived from

the one presented in Dell'Amico et al. [8]. We developed some new valid inequalities for the different features of the problem and several separation procedures have been added to the mentioned branch-and-cut algorithm. We use the branch-and-cut framework of CPLEX 12.2, that solves at every node of an enumeration tree the linear relaxation of a MILP model, and then invokes user-developed separation procedures to possibly add cuts.

We first present some valid inequalities that we use to strengthen constraint (3.7), thus to improve the convergence of the algorithm to the optimum, and then we discuss the separation procedures that we implemented.

Valid inequalities

For the next family of inequalities we need some further notation. Let P be a path. We recall that $P(i)$ denotes the index of the i-th vertex of path P, for $i = 0, 1, \ldots, |P|$, with $P(0) = 0$. A path is called infeasible with respect to the maximum duration if $\sum_{i=0}^{|P|-1}(t_{P(i)P(i+1)} + s_{P(i+1)}) > T$. Adding $t_{P(|P|),0}$ to the left hand side the check can be improved.

Let $\mathcal{P}$ be the family of all infeasible paths, the following *infeasible path constraints* are thus valid inequalities for the BRP:

$$\sum_{i=0}^{|P|-1} x_{P(i),P(i+1)} \leq |P| - 2 \qquad P \in \mathcal{P}. \tag{3.16}$$

Inequality (3.16) simply states that, if path P is infeasible then not all the arcs connecting two consecutive vertices of P may belong to a feasible solutions. A way to enforce them is to consider the related *tournament constraints*, by taking into account also the arcs connecting non-consecutive vertices of P. Indeed, any arc $(P(i), P(j))$ with $j \neq i+1$ is incompatible with arcs $(P(i), P(i+1))$ and $(P(j-1), P(j))$, because of, respectively, out-degree and in-degree constraints. Hence, we can add the corresponding variable, $x_{P(i),P(j)}$ to the left-hand side of (3.16), without affecting the right-hand side value. This process can be repeated for all arcs connecting two non-consecutive vertices in the path, with the exception of those arcs leaving the depot, because up to m of them may belong to a feasible solution. We thus obtain the following:

$$x_{0,P(1)} + \sum_{i=1}^{|P|-1} \sum_{j=i+1}^{|P|} x_{P(i),P(j)} \leq |P| - 2 \qquad P \in \mathcal{P}. \tag{3.17}$$

Inequalities (3.16) and (3.17) enforce the maximum route duration constraints, but are computationally very weak. To strengthen them in some favored cases, we take advantage of a new type of constraint, similar to the well-known capacity cut constraint, which we call *time-packing constraint*.

To express the time-packing constraint family of inequalities we need to introduce a lower bound on the time needed to reach a node and serve it. Let us define $lb_i(V) = \min_{j \in V : |q_j + q_i| \leq Q} \{t_{ji}\} + s_i, i \in V$. If we compute the lower bound in a subset $S \subseteq V$ we have $lb_i(S) = \min_{j \in S : |q_j + q_i| \leq Q} \{t_{ji}\} + s_i, i \in V$. Thus we can write the inequality (3.18), which computes a bound on the number of vehicles needed to serve the customers in S while respecting the maximum time constraint. Note that the equation also considers the time needed to return to the depot 0. If the summation of all the lower bounds is greater than the maximum time T then more routes are needed to serve the subset S.

$$\sum_{i \in S} \sum_{j \in V \setminus S} x_{ij} \geq \left\lceil \frac{\sum_{i \in S} lb_i(V) + lb_0(S)}{T} \right\rceil \quad S \subseteq V \setminus \{0\}, S \neq \emptyset, S \cap \{0\} = \emptyset. \tag{3.18}$$

Separation procedures

The aim of this subsection is to present the procedures that we use to determine if the valid inequalities that we proposed are violated by a given possibly fractional solution $\overline{x}$.

To separate constraints (3.18), we first build a supporting graph $\overline{G} = (\overline{V}, \overline{A})$, where the set of vertices is $\overline{V} = V \cup \{n+1\}$, where $\{n+1\}$ is a dummy node, and the set of arcs is $\overline{A} = A' \cup A''$. We have $A' = \{(i,j) \in A : \overline{x}_{ij} > 0\}$, and a capacity $\overline{x}_{ij}$ is assigned to every arc $(i,j) \in A'$; moreover $A'' = \{(n+1, i) : i \in V_0\}$. To each arc $(n+1, i) \in A''$ is associated a capacity equal to $lb_i(V)/T$. We then compute the max flow on $\overline{G}$, using $n+1$ as a source and 0 as a sink. The constraint (3.18) corresponding to the set S induced by the min-cut is then checked, and, if violated, it is added to the model.

The tournament constraints (3.17) are separated exactly, by generating all possible paths starting from the depot and using a depth-first strategy. We initialize path P with $P(0) = 0$, then select the outgoing arc having the largest value of $\overline{x}$, and extend the path to include the head of the selected arc. Every time we add a node to the path, we check if it is infeasible. If so, then we add the cut and backtrack to the previous node, otherwise we continue extending the path. The path extension continues as long as the sum of the involved $\overline{x}_{ij}$ is large enough to possibly lead to a violated cut, i.e., as long as it is strictly greater than $|P| - 2$. When it becomes smaller, we backtrack to the previous node. Anytime we backtrack, we continue the depth-first search by selecting the next arc with positive value of $\overline{x}$ and then extend the path consequently.

Suppose a violated cut of type (3.17) has been found, then we know that the set $S = \{P(1), P(2), \ldots, P(|P|)\}$ is currently visited by a single vehicle (the one performing the path P). This procedure can also be used

to separate the constraints (3.7). Indeed, when an infeasible path is found we can impose a constraint (3.7) where the route is the path found.

The separation procedures are invoked at every node of the enumeration tree in the order in which we described them. On the basis of computational evidence on the instances we tested, we stop the separation process as soon as we find a violated cut, if any.

3.6 Computational Results

In this section, we present the computational results for the metaheuristic and the branch-and-cut algorithms that we implemented to solve the BRP and the 1-PDVRP. To evaluate the quality of the metaheuristic algorithm when solving the BRP, we used the instances by Dell'Amico et al. [8] and, according to their method, we collected some new larger instances in terms of number of stations. The new instances use real data of bike sharing systems of the following cities: Brisbane, Milano, Lille, Toulouse, Sevilla, Valencia, Bruxelles, Lyon, Barcelona and London. Some characteristics of the new instances are reported in Table 3.1, where we depict the name of the city and country in which the bike sharing system is located, and the number of vertices of each instance. We then present the minimal, average, maximum value and the standard deviation of the request and of the transportation costs. To evaluate the performance of our algorithm on the 1-PDVRP we used the instances proposed by Shi et al. [32].

According to the number of vertices $|V|$, we divided both the BRP and the 1-PDVRP instances into three sets: for $|V| \leq 50$, $50 < |V| < 100$ and $|V| \geq 100$. We will refer to the three sets as *small-size* instances, *medium-size* instances, and *large-size* instances, respectively.

We determined the parameter setting by tuning them with the *irace Package* [24], which automatically configures optimization algorithms by finding the most appropriate settings given a set of instances. We used the default irace Package setting which sets using a maximum number of 1000 algorithms runs during the tuning. The stopping criterion for the algorithm runs is set to ten, 600, and 3600 CPU seconds for small-size, medium-size, and large-size instances, respectively. We decided to tune our algorithm on a modified set of instances, derived by the medium-instances, the Dublin instances, and the Minneapolis instances. We generated the modified instances by perturbing the distances c_{ij} choosing randomly in $[c_{ij} - 0.1 \cdot c_{ij}, c_{ij} + 0.1 \cdot c_{ij}]$, $(i, j) \in A$ and by perturbing the demand q_i of 10% of the nodes choosing randomly in $[\max\{q_i - 2, -Q\}, \min\{q_i + 2, Q\}]$, $i \in V \setminus \{0\}$. Both BRP and 1-PDVRP instances have been tested with the obtained set of parameters presented in Table 3.2. In the same table can be found the parameters used for testing the instances. We recall that α and δ can take real values in the interval $[0, 1]$, π can take an integer value in

the interval $[2, 15]$, ι can take value 0 for first improvement and 1 for best improvement, and κ can take an integer number in the interval $[3, |V|])$. All the tests reported in the following have been performed on a Intel Core i3-2100 with 3.10 GHZ by using randomly generated seed values.

City	Country	$\|V\|$	$\min\{q_i\}$	$\text{avg}\{q_i\}$	$\max\{q_i\}$	$\text{dev}\{q_i\}$	$\min\{c_{ij}\}$	$\text{avg}\{c_{ij}\}$	$\max\{c_{ij}\}$	$\text{dev}\{c_{ij}\}$
Brisbane	Australia	150	-15	1.27	17	5.91	2	3769.07	13959	2129.98
Milano	Italy	184	-18	0.88	17	8.40	6	3313.63	8126	1422.46
Lille	France	200	-20	-0.42	16	6.72	163	7450.82	18877	4655.73
Toulouse	France	240	-13	-1.49	12	6.16	6	3943.35	12459	1904.32
Sevilla	Spain	258	-20	-1.56	10	6.89	2	4652.58	13510	2305.15
Valencia	Spain	276	-20	-1.50	14	7.64	134	4241.34	13413	1892.71
Bruxelles	Belgium	304	-13	-0.18	16	8.24	211	5844.93	19032	2726.70
Lyon	France	336	-20	-0.70	17	7.85	18	4817.44	47657	3571.38
Barcelona	Spain	410	-17	-2.56	19	9.84	2	4699.91	13671	2229.90
London	U. K.	564	-28	-0.58	29	9.91	2	5833.37	19412	3100.69

Table 3.1: New BRP instances.

Parameter	Value
α	0.7335
π	4
δ	0.6582
ι	1
κ	35

Table 3.2: Parameters used for computational tests.

3.6.1 Small-size Instances

To evaluate the algorithm on the set of small-size instances we ran it for ten seconds taking note of the time when it found the best solution. Results are reported in Table 3.3, where the instances are depicted by the name of city, the number of vertices and the vehicle capacity. The table also reports the value of the optimal solution (opt) obtained by the branch-and-cut presented in Dell'Amico et al. [8] and the time needed to get the optimal solution ($t_{B\&C}$). The last five columns refer to algorithm DR_BRP and report the average solution value (avg) obtained by running the algorithm ten times for a ten seconds time limit, the percentage gap between the optimal and the average solution value ($\%gap_{avg} = 100 \cdot (avg - opt)/opt$), the minimal solution value among the ten trials (min), the percentage gap between the minimal solution value and the optimal one ($\%gap_{min} = 100 \cdot (min - opt)/opt$), and the average time needed to obtain the best solution by the metaheuristic algorithm(t_b).

One can note that our metaheuristic algorithm DR_BRP found the optimal solution in all ten trials for 37 out of 41 instances in times that are competitive with the ones needed by the branch-and-cut algorithm and obtained the optimal solution at least once for the remaining four instances

of the set. The average gap of 0.04 % from the optimal solution within an average time of 0.66 seconds is extremely promising.

Instance			B&C		DR_BRP (10 sec)				
City	$\|V\|$	Q	opt	$t_{B\&C}$	avg	$\%gap_{avg}$	min	$\%gap_{min}$	t_b
Bari	13	30	14600	0.02	14600.0	0.00	14600	0.00	0.00
Bari	13	20	15700	0.02	15700.0	0.00	15700	0.00	0.00
Bari	13	10	20600	0.03	20600.0	0.00	20600	0.00	0.00
Reggio Emilia	14	30	16900	0.02	16900.0	0.00	16900	0.00	0.00
Reggio Emilia	14	20	23200	0.03	23200.0	0.00	23200	0.00	0.00
Reggio Emilia	14	10	32500	0.05	32500.0	0.00	32500	0.00	0.01
Bergamo	15	30	12600	0.03	12600.0	0.00	12600	0.00	0.01
Bergamo	15	20	12700	0.00	12700.0	0.00	12700	0.00	0.01
Bergamo	15	12	13500	0.11	13500.0	0.00	13500	0.00	0.00
Parma	15	30	29000	0.02	29000.0	0.00	29000	0.00	0.00
Parma	15	20	29000	0.02	29000.0	0.00	29000	0.00	0.00
Parma	15	10	32500	0.05	32500.0	0.00	32500	0.00	0.00
Treviso	18	30	29259	0.05	29259.0	0.00	29259	0.00	0.04
Treviso	18	20	29259	0.03	29259.0	0.00	29259	0.00	0.04
Treviso	18	10	31443	0.09	31443.0	0.00	31443	0.00	0.08
La Spezia	20	30	20746	0.03	20746.0	0.00	20746	0.00	0.03
La Spezia	20	20	20746	0.03	20746.0	0.00	20746	0.00	0.03
La Spezia	20	10	22811	0.09	22811.0	0.00	22811	0.00	0.12
Buenos Aires	21	30	76999	0.37	76999.0	0.00	76999	0.00	0.03
Buenos Aires	21	20	91619	3.58	91619.2	0.00	91619	0.00	4.20
Ottawa	21	30	16202	0.02	16202.0	0.00	16202	0.00	0.00
Ottawa	21	20	16202	0.02	16202.0	0.00	16202	0.00	0.00
Ottawa	21	10	17576	0.11	17576.0	0.00	17576	0.00	0.02
San Antonio	23	30	22982	0.08	22982.0	0.00	22982	0.00	0.00
San Antonio	23	20	24007	0.09	24007.0	0.00	24007	0.00	0.05
San Antonio	23	10	40149	1.06	40149.0	0.00	40149	0.00	0.37
Brescia	27	30	30300	0.06	30300.0	0.00	30300	0.00	0.07
Brescia	27	20	31100	0.20	31100.0	0.00	31100	0.00	0.09
Brescia	27	11	35200	1.37	35200.0	0.00	35200	0.00	0.39
Roma	28	30	61900	0.84	61900.0	0.00	61900	0.00	0.36
Roma	28	20	66600	1.72	66670.0	0.11	66600	0.00	3.46
Roma	28	18	68300	0.58	68300.0	0.00	68300	0.00	0.00
Madison	28	30	29246	0.02	29246.0	0.00	29246	0.00	0.04
Madison	28	20	29839	0.05	29839.0	0.00	29839	0.00	0.04
Madison	28	10	33848	0.53	33848.0	0.00	33848	0.00	0.20
Guadalajara	41	30	57476	0.22	57476.0	0.00	57476	0.00	2.14
Guadalajara	41	20	59493	0.34	59493.0	0.00	59493	0.00	1.48
Guadalajara	41	11	64981	1.79	64981.0	0.00	64981	0.00	2.41
Dublin	45	30	33548	6.05	33595.4	0.14	33548	0.00	2.29
Dublin	45	20	39786	76.72	39817.2	0.08	39786	0.00	3.73
Dublin	45	11	54392	610.80	55000.6	1.12	54392	0.00	5.28
Avg.				17.25		0.04		0.00	0.66

Table 3.3: Results on small-size instances

3.6.2 Medium-size Instances

The computational results for the medium-size instances of the BRP are reported in Table 3.4. Some information on the branch-and-cut algorithm of Dell'Amico et al. [8] is reported, such as the best lower and upper bounds, the percentage gap between them ($gap = 100 \cdot (UB - LB)/UB$), and the time needed to get these values in a time limit set to one hour. Then we present z_{avg} (under multicolumn Avg), the average solution obtained by running the algorithm ten times for ten minutes, and the percentage gaps between z_{avg} and the lower and upper bounds from the branch-and-cut, $\%gap_{LB} = 100 \cdot (z_{avg} - LB)/LB$ and $\%gap_{UB} = 100 \cdot (z_{avg} - UB)/UB$, respectively. In Table 3.4 are also presented the minimal values z_{min} (under Min) obtained by the algorithm on the ten trials and the gaps between

z_{min} and the bounds of the branch-and-cut computed in the same form as for Avg. We also show the average time needed to obtain the best solution with a time limit set to 600 seconds in t_b.

One can notice that the algorithm can procure the optimal solution for seven instances out of eight for which an optimal solution was given. To notice the improvement on the upper bound obtained for the instances Ciudad de Mexico (90,30) and Ciudad de Mexico (90,20), where we improve the solution in average of 17.49 % and 18.72%, respectively, and in the best case of 18.08% and 18.98%, respectively.

We can see that on average the metaheuristic algorithm finds solutions 2.68% above the lower bound obtained by the branch-and-cut and algorithm and improves the upper bound by 2.81%; the best solution cost is only 2.28% above the lower bound and improves the upper bound by 3.17%.

Instance			B&C				DR_BRP (10 min)						
							Avg				*Min*		
City	$\|V\|$	Q	LB	UB	$\%gap$	t	z_{avg}	$\%gap_{LB}$	$\%gap_{UB}$	t_b	z_{min}	$\%gap_{LB}$	$\%gap_{UB}$
Denver	51	30	51583	51583	0.00	0.67	51583	0.00	0.00	0.48	51583	0.00	0.00
Denver	51	20	53465	53465	0.00	25.33	53465	0.00	0.00	24.68	53465	0.00	0.00
Denver	51	10	67459	67459	0.00	231.52	67459	0.00	0.00	102.99	67459	0.00	0.00
Rio de J.	55	30	122547	122547	0.00	65.57	122582.1	0.03	0.03	198.66	122547	0.00	0.00
Rio de J.	55	20	155446	156140	0.44	3600.00	155992.7	0.35	-0.09	304.13	155517	0.05	-0.40
Rio de J.	55	10	253690	259049	2.07	3600.00	257412.5	1.47	-0.63	241.12	257147	1.36	-0.73
Boston	59	30	65669	65669	0.00	28.14	65669	0.00	0.00	16.16	65669	0.00	0.00
Boston	59	20	71879	71879	0.00	473.84	72057.2	0.25	0.25	178.16	71916	0.05	0.05
Boston	59	16	74790	75065	0.37	3600.00	75318.8	0.71	0.34	280.50	75085	0.39	0.03
Torino	75	30	47634	47634	0.00	13.79	47634	0.00	0.00	246.44	47634	0.00	0.00
Torino	75	20	50204	50204	0.00	859.69	51026	1.64	1.64	251.83	50438	0.47	0.47
Torino	75	10	58814	64797	9.23	3600.00	62031.6	5.47	-4.27	303.85	61717	4.94	-4.75
Toronto	80	30	40794	41549	1.82	3600.00	41783.5	2.43	0.56	201.49	41390	1.46	-0.38
Toronto	80	20	42621	47898	11.02	3600.00	46876.6	9.98	-2.13	269.37	46631	9.41	-2.65
Toronto	80	12	54238	60763	10.74	3600.00	58878.7	8.56	-3.10	321.39	58539	7.93	-3.66
Miami	82	30	152229	156104	2.48	3600.00	154344.7	1.39	-1.13	335.05	154038	1.19	-1.32
Miami	82	20	209379	229237	8.66	3600.00	215167.1	2.76	-6.14	265.13	214250	2.33	-6.54
Miami	82	10	390536	415762	6.07	3600.00	402746.8	3.13	-3.13	300.28	397921	1.89	-4.29
C. de M.	90	30	67894	88227	23.05	3600.00	72797.5	7.22	-17.49	213.91	72279	6.46	-18.08
C. de M.	90	20	88952	116418	23.59	3600.00	94621.6	6.37	-18.72	254.22	94319	6.03	-18.98
C. de M.	90	17	99714	109573	9.00	3600.00	104213.7	4.51	-4.89	359.59	103658	3.96	-5.40
Avg.					5.17	2309.45		2.68	-2.81	222.35		2.28	-3.17

Table 3.4: Results on medium-size instances

3.6.3 Large-size Instances

In Table 3.5, we report the results for the large-size BRP instances. The tests have been performed similarly to those for the medium-size instances making use of the same parameter values, but with a time limit of 1800 seconds. The columns depicted in Table 3.5 are the same used in Table 3.4, even if the column reporting the times of the branch-and-cut algorithm is not necessary anymore since the time limit of one hour is always reached.

One can notice that the branch-and-cut algorithm is not giving good bounds for the large-size instances, on the other hand we can comment that in most of the cases we can get heuristic solutions with a value closer to the

lower bound than to the upper bound, which is improved by a 33.92%, in average.

Instance			B&C (1 h)			DR_BRP (30 min)						
						Avg				*Min*		
City	$\|V\|$	Q	LB	UB	$\%gap$	z_{avg}	$\%gap_{LB}$	$\%gap_{UB}$	t_b	z_{min}	$\%gap_{LB}$	$\%gap_{UB}$
Minneapolis	116	30	136148	137843	1.23	139874.3	2.74	1.47	745.28	138467	1.70	0.45
Minneapolis	116	20	157736	186449	15.40	166797.0	5.74	-10.54	1003.72	166150	5.33	-10.89
Minneapolis	116	10	246133	298886	17.65	264335.2	7.40	-11.56	946.37	262936	6.83	-12.03
Brisbane	150	30	108275	158043	31.49	115949.2	7.09	-26.63	742.81	115120	6.32	-27.16
Brisbane	150	20	132419	196739	32.69	146930.0	10.96	-25.32	957.83	146313	10.49	-25.63
Brisbane	150	17	147236	234210	37.14	160385.6	8.93	-31.52	966.44	160015	8.68	-31.68
Milano	184	30	145245	227983	36.29	168931.2	16.31	-25.90	871.18	167493	15.32	-26.53
Milano	184	20	187175	295994	36.76	219558.6	17.30	-25.82	823.76	218249	16.60	-26.27
Milano	184	18	203716	299630	32.01	236394.9	16.04	-21.10	1048.32	234423	15.07	-21.76
Lille	200	30	164149	231244	29.01	178133.5	8.52	-22.97	666.67	176976	7.81	-23.47
Lille	200	20	191630	440350	56.48	215007.8	12.20	-51.17	280.09	213090	11.20	-51.61
Toulouse	240	30	166653	404792	58.83	190146.8	14.10	-53.03	1147.86	188995	13.41	-53.31
Toulouse	240	20	190739	427959	55.43	231062.0	21.14	-46.01	1398.52	228674	19.89	-46.57
Toulouse	240	13	256036	461125	44.48	308982.7	20.68	-32.99	868.98	307874	20.25	-33.23
Sevilla	258	30	194805	461011	57.74	227911.7	16.99	-50.56	898.40	225076	15.54	-51.18
Sevilla	258	20	240210	516734	53.51	281492.2	17.19	-45.52	1054.31	279990	16.56	-45.82
Valencia	276	30	245979	673479	63.48	292262.6	18.82	-56.60	789.54	287854	17.02	-57.26
Valencia	276	20	302368	698436	56.71	370057.4	22.39	-47.02	1064.40	367201	21.44	-47.43
Bruxelles	304	30	255259	502920	49.24	315487.7	23.60	-37.27	851.97	311097	21.88	-38.14
Bruxelles	304	20	301100	580594	48.14	379141.4	25.92	-34.70	835.66	376387	25.00	-35.17
Bruxelles	304	16	348303	626721	44.42	426992.4	22.59	-31.87	1038.33	424432	21.86	-32.28
Lyon	336	30	300950	580437	48.15	364083.4	20.98	-37.27	879.61	360009	19.62	-37.98
Lyon	336	20	356787	668971	46.67	437588.1	22.65	-34.59	1136.03	433959	21.63	-35.13
Barcelona	410	30	311774	983627	68.30	545633.1	75.01	-44.53	458.27	543929	74.46	-44.70
Barcelona	410	20	449060	1088850	58.76	774818.4	72.54	-28.84	1538.57	771507	71.80	-29.14
Barcelona	410	19	429327	1089070	60.58	805513.2	87.62	-26.04	1411.27	800622	86.48	-26.49
London	564	30	385748	1304850	70.44	705232.0	82.82	-45.95	1572.97	699571	81.35	-46.39
London	564	29	363299	1339890	72.89	725468.0	99.69	-45.86	1447.33	718026	97.64	-46.41
Avg.					45.85		27.78	-33.92	980.16		26.83	-34.40

Table 3.5: Results on large-size instances

3.6.4 Local Searches Evaluation

In this subsection we furnish with an insight on the contribution of the local search procedures with respect to the complete DR_BRP algorithm. We produced seven versions of our algorithm, by removing one of the seven local search procedures. Each version was run ten times on the medium-size instances, but disregarding the Denver instances because they are too easy and thus not interesting for this study. In Table 3.6 we report for each evaluated instance the percentage gap, $\%gap_{(\cdot)}$, between the average solution value obtained by the complete DR_BRP algorithm and the average solution value reached without the local search procedure $(\cdot)$. The index of the local search procedure in Table 3.6 is the one already used in Section 3.4.4 (e.g., $\%gap_1$ gives the percentage gap between DR_BRP and DR_BRP without MoveLS). By looking at the average values, reported in the last line of the table, one can notice that all local search procedures that we implemented have a positive contribution to the behavior of DR_BRP. Indeed each removal leads to slightly worse solutions. In particular, MoveConsecutiveLS (index 2) and CrossingLS (index 6) have the largest impact on the overall algorithm, as their removal worsens the average solution value by

0.74% and 0.18%, respectively.

City	$\|V\|$	Q	$\%gap_1$	$\%gap_2$	$\%gap_3$	$\%gap_4$	$\%gap_5$	$\%gap_6$	$\%gap_7$
Rio de Janeiro	55	30	0.03	-0.36	-0.01	-0.02	-0.02	0.00	-0.01
Rio de Janeiro	55	20	0.00	-0.15	0.02	-0.04	0.00	-0.10	0.00
Rio de Janeiro	55	10	-0.03	-0.17	0.03	-0.07	-0.06	-0.04	-0.01
Boston	59	30	0.00	-0.07	0.00	0.00	0.00	0.00	0.00
Boston	59	20	-0.03	-0.07	-0.03	0.03	0.02	-0.03	0.00
Boston	59	16	-0.08	-1.79	-0.04	-0.36	-0.20	-0.03	-0.13
Torino	75	30	0.00	-0.42	-0.02	0.00	0.00	-0.01	-0.01
Torino	75	20	-0.66	-1.01	-0.21	-0.13	0.24	-0.20	0.09
Torino	75	10	-0.23	-0.59	-0.13	-0.09	0.09	-0.44	0.04
Toronto	80	30	0.15	-1.32	0.07	-0.08	-0.14	-0.33	0.05
Toronto	80	20	0.49	-1.89	0.39	0.17	0.41	-0.12	0.05
Toronto	80	12	-0.33	-1.92	-0.50	-0.40	-0.15	-1.00	-0.43
Miami	82	30	0.05	-0.09	0.00	0.04	-0.06	-0.14	0.05
Miami	82	20	-0.11	-0.28	-0.17	-0.18	-0.40	-0.22	-0.18
Miami	82	10	0.07	0.28	0.18	-0.09	-0.30	0.34	0.26
Ciudad de Mexico	90	30	0.12	-1.59	-0.09	0.09	0.10	-0.32	0.06
Ciudad de Mexico	90	20	0.06	-0.83	0.07	-0.05	-0.03	-0.24	-0.07
Ciudad de Mexico	90	17	-0.20	-1.11	-0.16	-0.09	-0.09	-0.33	-0.01
Avg.			-0.04	-0.74	-0.03	-0.07	-0.03	-0.18	-0.01

Table 3.6: Evaluation of local search components.

3.6.5 1-PDVRP Instances

The metaheuristic algorithm developed from the BRP has been slightly modified to include the maximum duration constraint typical of the 1-PDVRP and we use it to solve the instances available in the literature for the 1-PDVRP (see, Shi et al. [32]).

To test the quality of our algorithm, we decided to solve the instances for the 1-PDVRP setting as time limit the same times needed by the algorithm presented by Shi et al. [32] to terminate. The PC on which we performed our tests is faster compared to the one used by Shi et al. [32]. In particular, our PC is equipped with a 3.10 GHz CPU while theirs with a 1.60 GHz CPU. Hence, to perform a fair set of tests we assumed as time limit for each instance the time needed by the algorithm of Shi et al. [32] divided by 3.10 and multiplied by 1.60. In Table 3.7, we present the results of these tests. In details, we show the average solution obtained in the ten trials within the time limit computed as previously explained (z_{avg}), and the average time needed to find the best solution (t_b). Then we recall the best average solution of Shi et al. [32] ($best_{SZG}$) and the gap between z_{avg} and $best_{SZG}$ ($\%\overline{gap}_{SZG} = 100 \cdot (z_{avg} - best_{SZG})/z_{avg}$). Moreover we replicate the minimal solution obtained by our algorithm (z_{min}) and the gap between z_{avg} and z_{min} ($\%gap_{min} = 100 \cdot (z_{avg} - z_{min})/z_{avg}$).

One can notice that DR_BRP improves upon SZG algorithm by about 10%. In two cases (n40q10A and n400q10A), our algorithm needs more time to reach an improved solution. Our algorithm reaches solution values only 3.66% above the best known ones, when using the SZG time limits.

In the following, we reproduce the experiments according to what we did for the BRP instances. In Table 3.8, we present the results on the small-size

Inst.	$\|V\|$	Q	z_{avg}	t_b	$best_{SZG}$	$\%\overline{gap}_{SZG}$	z_{min}	$\%\overline{gap}_{min}$
n20q10A	20	10	5001	0.10	5515.4	-10.29	5001	0.00
n30q10A	30	10	6547.1	0.20	6906	-5.48	6503	0.67
n40q10A	40	10	7619.3	0.32	7476.4	1.88	7407	2.79
n50q10A	50	10	7478.9	0.54	9263.5	-23.86	7283	2.62
n60q10A	60	10	9445.1	0.58	9931.6	-5.15	8939	5.36
n100q10A	100	10	12946.6	0.37	14379.3	-11.07	12317	4.86
n200q10A	200	10	20423.4	12.89	23331.7	-14.24	19341	5.30
n300q10A	300	10	26069.2	43.60	29805.3	-14.33	24763	5.01
n400q10A	400	10	35588	49.53	34574.4	2.85	33764	5.13
n500q10A	500	10	34553.6	121.306	39872.5	-15.39	32890	4.81
Avg.				22.94		-9.51		3.66

Table 3.7: Comparison with the SZG algorithm with proportionate time limits.

instances for the 1-PDVRP. We recall that we ran the algorithm ten times for ten seconds and set the parameters equally to the values used for the instances of the BRP. In Table 3.8, we report the name of the instance, the number of the vertices, and the vehicle capacity. We then include the value of the optimal solution (*opt*) obtained thanks to the newly developed branch-and-cut algorithm for the 1-PDVRP. In the columns underlying the DR_BRP multicolumn are shown the average and the minimum values of the ten trials of our metaheuristic and their gaps with respect to the optimal solution. In the columns under the multicolumn SZG we report the best average solution (*best*) presented by Shi et al. [32] and the gaps relative to avg and min, specifically $\%\overline{gap} = 100 \cdot (avg - best)/best$ and $\%gap = 100 \cdot (min - best)/best$. In the last column we show the time needed by Shi et al. to get their solutions.

One can see that with our metaheuristic we can solve all the small-size instances to optimality every time for n20q10A, and at least once for the other instances, within an average time of 3.01 seconds. Moreover, when comparing our results to the algorithm presented by Shi et al. [32] it is clear that we can improve the solutions of more than 9% in average with an acceptable increase of computational times.

				SZG				DR_BRP				
Inst.	$\|V\|$	Q	opt	$best$	$\%\overline{gap}$	$\%gap$	t	avg	$\%gap_{avg}$	min	$\%gap_{min}$	t_b
n20q10A	20	10	5001	5515.4	-9.33	-9.33	0.65	5001	0.00	5001	0.00	0.10
n30q10A	30	10	6503	6906	-5.69	-5.84	0.92	6512.8	0.15	6503	0.00	1.91
n40q10A	40	10	7407	7476.4	-0.02	-0.93	1.14	7474.8	0.92	7407	0.00	5.60
n50q10A	50	10	7283	9263.5	-21.18	-21.38	1.52	7301.9	0.26	7283	0.00	4.43
Avg.					-9.05	-9.37	1.06		0.33		0.00	3.01

Table 3.8: Results on small-size instances for the 1-PDVRP.

In Table 3.9, we depict the results of our algorithm on the medium-size and large-size instances of the 1-PDVRP. Since only one instance, n60q10A, could be considered as a medium-size instance we show its results with those derived by the large-size instances. Instance n60q10A has been run for ten

minutes, while the other instances for 30 minutes. All instances have been run ten times by using the parameters used for the instances of the BRP.

Similarly to the previous tables, in Table 3.9 we show the name of the instances, the number of vertices, and the vehicles capacity. We then present the average value of the solution computed on the ten trials (z_{avg}), the best value among them (z_{min}), and the average time needed to obtain the best solution (t_b). In the remaining part of the Table we show the best average solution presented by Shi et al. [32], the gaps $\%\overline{gap}_{SZG} = 100 \cdot (z_{avg} - best_{SZG})/best_{SZG}$ and $\%gap_{SZG} = 100 \cdot (z_{min} - best_{SZG})/best_{SZG}$, and the needed computing times (t_{SZG}).

One can notice that our algorithm is able to improve every solution, in particular we improve the solution of 11.91%, in average, and of 12.85% considering the best solutions, within about 15 minutes, in average.

Inst.	$\|V\|$	Q	z_{avg}	z_{min}	t_b	$best_{SZG}$	$\%\overline{gap}_{SZG}$	$\%gap_{SZG}$	t_{SZG}
n60q10A	60	10	8941.1	8939	140.87	9931.6	-9.97	-9.99	2.01
n100q10A	100	10	12476.9	12317	855.75	14379.3	-13.23	-14.34	14.85
n200q10A	200	10	19619.5	19341	1181.46	23331.7	-15.91	-17.10	47.00
n300q10A	300	10	25092.3	24763	808.87	29805.3	-15.81	-16.92	100.25
n400q10A	400	10	34209	33951	1237.71	34574.4	-1.06	-1.80	156.47
n500q10A	500	10	33706.5	33108	1203.76	39872.5	-15.46	-16.97	240.06
Avg.					904.74		-11.91	-12.85	93.44

Table 3.9: Results on medium-size and large-size instances for the 1-PDVRP.

3.7 Conclusions

In this chapter, we solved the Bike sharing Rebalancing Problem proposing an effective metaheuristic algorithm in which are implemented a new constructive heuristic and a set of local searches made more efficient by some speed-up techniques. A related problem where a maximum duration constraint is considered, the one-commodity Pickup and Delivery Vehicle Routing Problem, has also been tackled by adapting the developed metaheuristic and the branch-and-cut algorithm. The branch-and-cut algorithm used considers in an innovative and efficacious way the duration constraint including inequalities and separation procedures that can be applied to general VRP problems with maximum duration constraints. We evaluated our algorithms on newly collected real-world and literature instances for both BRP and 1-PDVRP. We improved strongly the solutions reported in the literature getting new optima and new best known solutions in effective times.

3.A Annex

Proof of Property 2

a) Let P' be the route obtained by removing vertex i from P. Route P' is feasible if

$$\max_{k=0}^{|P'|-1}\{l_{P'(k)}(P')\} - \min_{k=0}^{|P'|-1}\{l_{P'(k)}(P')\} \leq Q. \tag{3.19}$$

To prove that this is always satisfied, let us consider two cases:

-) $q_i \geq 0$. With respect to the original route P, the removal of the vertex may either reduce or keep unchanged both the max and min terms in (3.19). The maximum difference between the two terms arises when the max remains unchanged and the min decreases by the largest possible quantity, i.e., by q_i. We can thus state that:

$$\begin{aligned}
&\max_{k=0}^{|P'|-1}\{l_{P'(k)}(P')\} - \min_{k=0}^{|P'|-1}\{l_{P'(k)}(P')\} \leq \\
&\max_{k=0}^{|P|-1}\{l_{P(k)}(P)\} - \left(\min_{k=0}^{|P|-1}\{l_{P(k)}(P)\} - q_i\right) = \\
&\max_{k=0}^{|P|-1}\{l_{P(k)}(P)\} - \min_{k=0}^{|P|-1}\{l_{P(k)}(P)\} + q_i \leq \\
&\max_{k=0}^{|P|-1}\{l_{P(k)}(P)\} - \min_{k=0}^{|P|-1}\{l_{P(k)}(P)\} + \Delta_P = Q,
\end{aligned}$$

where the last two steps follow, respectively, from the hypothesis and from Equation (3.10).

-) $q_i < 0$. In this case we consider instead that the largest difference between the two terms in (3.19) is attained when the max increases by $-q_i$ and the min remains unchanged, thus:

$$\begin{aligned}
&\max_{k=0}^{|P'|-1}\{l_{P'(k)}(P')\} - \min_{k=0}^{|P'|-1}\{l_{P'(k)}(P')\} \leq \\
&\left(\max_{k=0}^{|P|-1}\{l_{P(k)}(P)\} - q_i\right) - \min_{k=0}^{|P|-1}\{l_{P(k)}(P)\} = \\
&\max_{k=0}^{|P|-1}\{l_{P(k)}(P)\} - \min_{k=0}^{|P|-1}\{l_{P(k)}(P)\} - q_i \leq \\
&\max_{k=0}^{|P|-1}\{l_{P(k)}(P)\} - \min_{k=0}^{|P|-1}\{l_{P(k)}(P)\} + \Delta_P = Q.
\end{aligned}$$

b) Directly form Property 1 and Property 2.a.

c) Let P' be the route obtained by swapping vertices i and j in P. First suppose that i precedes j in P. Then the proof follows the footsteps of that of point a), by considering two cases, $q_i \geq q_j$ and $q_i < q_j$, and showing that (3.19) is always satisfied when the hypothesis holds. The same applies when j precedes i.

d) Let P' and R' be the routes obtained from, respectively, P and R, after the swap of i with j. Suppose without loss of generality that $\Delta_P \leq \Delta_R$, and let us concentrate on P. Similarly to the proof of point a), we consider two cases, $q_i \geq q_j$ and $q_i < q_j$. Then one can check that Equation (3.19) is satisfied on both cases, when the hypothesis holds, by applying a similar consideration to the one above on the largest difference between the max and min values attained in the equation. Note that the fact that the swap is feasible does not imply that the single removal or insertion of a vertex is feasible, i.e., $|q_i|$ and/or $|q_j|$ could be greater than Δ_P. The same reasoning applies to route R and that concludes the proof.

□

Proof of Property 3
We are given two disjoint and feasible routes P and R, to be merged in the order $(P \oplus R)$. Recall that, by using (3.12), the last forward load window of P can be expressed as

$$\begin{aligned} F_{P(|P|-1)}(P) = \\ \big[l_{P(|P|-1)}(P) - \min_{i=0}^{|P|-1}\{l_{P(i)}(P)\}, \\ l_{P(|P|-1)}(P) + Q - \max_{i=0}^{|P|-1}\{l_{P(i)}(P)\}\big]. \end{aligned} \tag{3.20}$$

Now notice that the first backward load window of route R can be expressed as the difference between the last forward load window of the route and the cumulated demand along the route, that is

$$B_{R(1)}(R) = F_{R(|R|-1)}(R) - l_{R(|R|-1)}(R). \tag{3.21}$$

The extreme values of $F_{R(|R|-1)}(R)$ can be expressed again by using (3.12), obtaining

$$\begin{aligned} F_{R(|R|-1)}(R) = \\ \big[l_{R(|R|-1)}(R) - \min_{i=0}^{|R|-1}\{l_{R(i)}(R)\}, \\ l_{R(|R|-1)}(R) + Q - \max_{i=0}^{|R|-1}\{l_{R(i)}(R)\}\big]. \end{aligned} \tag{3.22}$$

Thus combining (3.21) and (3.23) we get

$$B_{R(1)}(R) \;=\; \left[-\min_{i=0}^{|R|-1}\{l_{R(i)}(R)\}, Q - \max_{i=0}^{|R|-1}\{l_{R(i)}(R)\}\right]. \tag{3.23}$$

Let us define, for the sake of simplicity, $F_{P(|P|-1)}(P) = F = \left[\underline{f}, \overline{f}\right]$ and $B_{R(1)}(R) = B = \left[\underline{b}, \overline{b}\right]$. Their intersection is given by

$$F \cap B = \left[\max\{\underline{f}, \underline{b}\}, \min\{\overline{f}, \overline{b}\}\right].$$

We first show the "if" part of the thesis, that is, if the intersection of the two load windows is not empty, then the combined route is feasible. The two load windows can be basically seen as two intervals, thus their intersection is not empty if

$$\max\{\underline{f}, \underline{b}\} \leq \min\{\overline{f}, \overline{b}\}. \tag{3.24}$$

There are four cases. If $\max\{\underline{f}, \underline{b}\} = \underline{f}$ and $\min\{\overline{f}, \overline{b}\} = \overline{f}$, then (3.24) is satisfied because P is feasible for hypothesis. The same holds when the max and min values are attained by $\underline{b}$ and $\overline{b}$, respectively, because of the feasibility of R. Let us now concentrate on the case where $\max\{\underline{f}, \underline{b}\} = \underline{f}$ and $\min\{\overline{f}, \overline{b}\} = \overline{b}$.

For this case (3.24) summarizes to $\underline{f} \leq \overline{b}$, that using (3.21) and (3.23) corresponds to

$$l_{P(|P|-1)}(P) - \min_{i=0}^{|P|-1}\{l_{P(i)}(P)\} \leq Q - \max_{i=0}^{|R|-1}\{l_{R(i)}(R)\},$$

which can be rewritten as

$$\max_{i=0}^{|R|-1}\{l_{R(i)}(R) + l_{P(|P|-1)}(P)\} - \min_{i=0}^{|P|-1}\{l_{P(i)}(P)\} \leq Q. \tag{3.25}$$

Route $P \oplus R$ is feasible if the following condition is satisfied

$$\max_{i=0}^{|P\oplus R|-1}\{l_{P\oplus R(i)}(P \oplus R)\} - \min_{i=0}^{|P\oplus R|-1}\{l_{P\oplus R(i)}(P \oplus R)\} \leq Q. \tag{3.26}$$

To prove this, first notice that we can rewrite

$$\max_{i=0}^{|P\oplus R|-1}\{l_{P\oplus R(i)}(P \oplus R)\} = \tag{3.27}$$
$$\max\{\max_{i=0}^{|P|-1}\{l_{P(i)}(P)\}, \max_{i=0}^{|R|-1}\{l_{R(i)}(R) + l_{P(|P|-1)}(P)\}\},$$
$$\min_{i=0}^{|P\oplus R|-1}\{l_{P\oplus R(i)}(P \oplus R)\} = \tag{3.28}$$
$$\min\{\min_{i=0}^{|P|-1}\{l_{P(i)}(P)\}, \min_{i=0}^{|R|-1}\{l_{P(i)}(R) + l_{P(|P|-1)}(P)\}\}.$$

Because $\overline{f} \geq \overline{b}$, we have that $l_{P(|P|-1)}(P) + Q - \max_{i=0}^{|P|-1}\{l_{P(i)}(P)\} \geq Q - \max_{i=0}^{|R|-1}\{l_{R(i)}(R)\}$, and thus

$$\max_{i=0}^{|R|-1}\{l_{R(i)}(R) + l_{P(|P|-1)}(P)\} \geq \max_{i=0}^{|P|-1}\{l_{P(i)}(P)\},$$

which combined with (3.28) gives

$$\max_{i=0}^{|P\oplus R|-1}\{l_{P\oplus R(i)}(P\oplus R)\} = \max_{i=0}^{|R|-1}\{l_{R(i)}(R) + l_{P(|P|-1)}(P)\}. \tag{3.29}$$

Similarly, from $\underline{f} \geq \underline{b}$, it follows

$$l_{P(|P|-1)}(P) - \min_{i=0}^{|P|-1}\{l_{P(i)}(P)\} \geq -\min_{i=0}^{|R|-1}\{l_{R(i)}(R)\},$$

and hence

$$\min_{i=0}^{|P|-1}\{l_{P(i)}(P)\} \leq \min_{i=0}^{|R|-1}\{l_{R(i)}(R) + l_{P(|P|-1)}(P)\},$$

which combined with (3.29) gives us that

$$\min_{i=0}^{|P\oplus R|-1}\{l_{P\oplus R(i)}(P\oplus R)\} = \min_{i=0}^{|P|-1}\{l_{P(i)}(P)\}. \tag{3.30}$$

By including (3.29) and (3.30) into (3.25) we can thus conclude that (3.26) is satisfied. The remaining case, $\max\{\underline{f}, \underline{b}\} = \underline{b}$ and $\min\{\overline{f}, \overline{b}\} = \overline{f}$, is specular to the one just discussed, and this concludes the first part of the proof.

The "only if" part is proved similarly, noticing that $F \cap B = \emptyset$ means that $\max\{\underline{f}, \underline{b}\} > \min\{\overline{f}, \overline{b}\}$. Once again we have four cases, $\underline{f} > \overline{f}$ and $\underline{b} > \overline{b}$, which are impossible, and $\underline{f} > \overline{b}$ and $\underline{b} > \overline{f}$, which are specular. We concentrate on the case $\underline{f} > \overline{b}$, and use the consequent facts that $\overline{f} \geq \overline{b}$ and $\underline{f} \geq \underline{b}$. By applying similar steps to the ones in the "if" part we can show that

$$\max_{i=0}^{|P\oplus R|-1}\{l_{P\oplus R(i)}(P\oplus R)\} - \min_{i=0}^{|P\oplus R|-1}\{l_{P\oplus R(i)}(P\oplus R)\} > Q,$$

and thus $P \oplus R$ is infeasible. □

Bibliography

[1] M. Battarra, J.-F. Cordeau, and M. Iori. *Vehicle Routing: Problems, Methods, and Applications*, Chapter 6: Pickup-and-Delivery Problems for Goods Transportation. *Vehicle Routing: Problems, Methods, and Applications*, Toth P. and Vigo D. (eds.) SIAM, 2014.

[2] M. Benchimol, P. Benchimol, B. Chappert, A. De La Taille, F. Laroche, F. Meunier, and L. Robinet. Balancing the stations of a self service "Bike Hire" system. *RAIRO-Operations Reserach*, 45:37–61, 2011.

[3] G. Berbeglia, J.-F. Cordeau, I. Gribkovskaia, and G. Laporte, Static pickup and delivery problems: A classification scheme and survey. *TOP*, 15:1–31, 2007.

[4] A. Cesta, A. Oddi, and S. F. Smith. Iterative flattening: A scalable method for solving multi-capacity scheduling problems. *Proceedings of the Seventeenth National Conference on Artificial Intelligence*. AAAI Press / The MIT Press, 742–747, 2000.

[5] D. Chemla, F. Meunier, and R. Wolfler Calvo. Bike sharing systems: Solving the static rebalancing problem. *Discrete Optimization*, 10:120–146, 2013.

[6] G. Clarke and J. W. Wright. Scheduling of vehicles from a central depot to a number of delivery points. *Operations research*, 12:568–581, 1964.

[7] C. Contardo, C. Morency, and L.-M. Rousseau. Balancing a dynamic public bike-sharing system. Technical Report CIRRELT-2012-09, CIRRELT, 2012.

[8] M. Dell'Amico, E. Hadjicostantinou, M. Iori, and S. Novellani, The bike sharing rebalancing problem: Mathematical formulations and benchmark instances. *Omega*, 45:7–19, 2014.

[9] P. DeMaio, Bike Sharing: History, Impacts, Model of Provision and Future. *Journal of Public Transportation*, 10:41–56, 2009.

[10] P. DeMaio and R. Meddin. Bike-sharing world map. [Online]. Available: http://bike-sharing.blogspot.it/, 2014.

[11] M. den Besten and T. Stützle. Neighborhoods revisited: An experimental investigation into the effectiveness of variable neighborhood descent for scheduling. in *Proceedings of the 4th Metaheuristics International Conference*, Porto, Portugal, 2:545–550, 2001

[12] L. Di Gaspero, A. Rendl, and T. Urli. Balancing bike sharing systems with constraint programming. *Constraints*, 2014 (submitted).

[13] Ö. Ergun and J. B. Orlin. Fast neighborhood search for the single machine total weighted tardiness problem. *Operations Research Letters*, 34:41–45, 2006.

[14] P. Hansen and N. Mladenović. Variable neighborhood search: Principles and applications. *European Journal of Operational Research*, 130:449–467, 2001.

[15] H. Hernández-Pérez, I. Rodríguez-Martín, and J.-J. Salazar-González. A hybrid GRASP/VND heuristic for the one-commodity pickup-and-delivery traveling salesman problem. *Computers & Operations Research*, 36:1639–1645, 2009.

[16] H. Hernández-Pérez and J.-J. Salazar-González. A Branch-and-Cut Algorithm for a Traveling Salesman Problem with Pickup and Delivery. *Discrete Applied Mathematics*, 145:126–139, 2004.

[17] H. Hernández-Pérez and J.-J. Salazar-González. Heuristics for the one-commodity pickup-and-delivery traveling salesman problem. *Transportation Science*, 38:245–255, 2004.

[18] H. Hernández-Pérez and J.-J. Salazar-González. The One-Commodity Pickup-and-Delivery Travelling Salesman Problem. *Lecture Notes in Computer Science*, 2570:89–104, 2003.

[19] H. Hernández-Pérez and J.-J. Salazar-González. The one-commodity pickup-and-delivery traveling salesman problem: Inequalities and algorithms. *Networks*, 50:258–272, 2007.

[20] M. Hosny and C. Mumford. Solving the one-commodity pickup and delivery problem using an adaptive hybrid VNS/SA approach. *Parallel Problem Solving from Nature, PPSN XI*, ser. Lecture Notes in Computer Science, R. Schaefer, C. Cotta, J. Kolodziej, and G. Rudolph, Eds. Springer, Berlin, 6239:189–198, 2010.

[21] T. Ibaraki, S. Imahori, M. Kubo, T. Masuda, T. Uno, and M. Yagiura. Effective local search algorithms for routing and scheduling problems with general time-window constraints. *Transportation Science*, 39:206–232, 2005.

[22] L. W. Jacobs and M. J. Brusco. A local search heuristic for large set-covering problems. *Naval Research Logistics*, 42:1129–1140, 1995.

[23] C.-J. Liao, H.-H. Tsou, and K.-L. Huang. Neighborhood search procedures for single machine tardiness scheduling with sequence-dependent setups. *Theoretical Computer Science*, 434:45–52, 2012.

[24] M. López-Ibáñez, J. Dubois-Lacoste, T. Stützle, and M. Birattari. The irace package, iterated race for automatic algorithm configuration. IRIDIA, Université Libre de Bruxelles, Belgium, Tech. Rep. TR/IRIDIA/2011-004, 2011. [Online]. Available: http://iridia.ulb.ac.be/IridiaTrSeries/IridiaTr2011-004.pdf

[25] G. Martinovic, I. Aleksi, and A. Baumgartner. Single-Commodity Vehicle Routing Problem with Pickup and Delivery Service. *Mathematical Problems in Engineering*, 2008.

[26] N. Mladenović, D. Urošević, S. Hanafi, and A. Ilić. A general variable neighborhood search for the one-commodity pickup-and-delivery travelling salesman problem. *European Journal of Operational Research*, 220:270–285, 2012.

[27] P. Papazek, G. Raidl, M. Rainer-Harbach, and B. Hu. A pilot/vnd/grasp hybrid for the static balancing of public bicycle sharing systems. *Computer Aided Systems Theory-EUROCAST 2013*. Springer, 372–379, 2013.

[28] T. Raviv, M. Tzur, and I. Forma. Static repositioning in a bike-sharing system: models and solution approaches. *EURO Journal on Transportation and Logistics*, 2:187–229, 2013.

[29] A. J. Richmond and J. E. Beasley. An iterative construction heuristic for the ore selection problem. *Journal of Heuristics*, 10:153–167, 2004.

[30] R. Ruiz and T. Stützle. A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. *European Journal of Operational Research*, 177:2033–2049, 2007.

[31] P. Shaw. Using constraint programming and local search methods to solve vehicle routing problems. *Principles and Practice of Constraint Programming - CP98*, ser. Lecture Notes in Computer Science, M. J. Maher and J.-F. Puget, (eds.), Springer, 1520:417–431, 1998.

[32] X. Shi, F. Zhao, and Y. Gong, Genetic algorithm for the one-commodity pickup-and-delivery vehicle routing problem. *IEEE International Conference on Intelligent Computing and Intelligent Systems, 2009*, 1:175–179, 2009.

[33] G. Schrimpf, J. Schneider, H. Stamm-Wilbrandt, and G. Dueck. Record breaking optimization results using the ruin and recreate principle. *Journal of Computational Physics*, 159:139–171, 2000.

[34] J. Schuijbroek, R. Hampshire, and W.-J. van Hoeve. Inventory rebalancing and vehicle routing in bike sharing systems. 2013 (working paper).

[35] F. Zhao, S. Li, J. Sun, and D. Mei. Genetic algorithm for the one-commodity pickup-and-delivery traveling salesman problem. *Computers & Industrial Engineering*, 56:642–1648, 2009.

Chapter 4

The Bike sharing Rebalancing Problem with Stochastic Demands

In this chapter we deal with stochastic versions of the Bike sharing Rebalancing Problem (BRP), the problem of driving a fleet of vehicles to redistribute bicycles among stations of a bike sharing system. In the stochastic BRPs the demands of the stations are represented by random variables, each one given with its probability distribution. We define two versions of the BRP with stochastic demands, considering different decisions to be made before the realization of the random variables. We modeled the problems as stochastic programming models and tackled them by using different solving approaches, including the L-Shaped method and branch-and-cut algorithms. Moreover, we developed constructive heuristic algorithms based on the novel use of positive and negative correlations between the bike stations, based on their stochastic demands. The algorithms have been tested on newly collected real-world instances and an extensive comparison between the different approaches is given.

Keywords: Stochastic Programming, Routing, Bike Sharing, Branch-and-cut, Correlations

4.1 Introduction

Bike sharing systems are public or private systems used to solve the last mile problem and to provide an additional mean of transportation to the users moving across city centers. They appeared for the first time in Amsterdam in the 60s and they multiplied consistently in the last two decades in every continent. Bike sharing systems are made of several bike stations located in different places in a city, town, university campus, etc. Each station is made

of a number of slots where users can collect or drop off a bicycle. Users can collect a bicycle, use it for a certain amount of time, and drop it off at a station of their choice. Most of the times, users start and finish their trips in two different stations, and they do not necessarily do the return trip using the same mean of transportation. For instance, this is a typical situation when the starting bike station is located on the top of a hill and the ending bike station is at the bottom. Each station is associated with a number, that is called *balanced level of occupation*, that is the number of bicycles to be present in the station to make the station balanced. Note that if all the stations are balanced then the system is balanced. Given a balanced situation, after a certain amount of time, the system may become unbalanced and this leads to inefficiencies for the users such as empty or full stations. To rebalance the system we must bring the current situation of each station to its balanced level of occupation. The demand of each station is defined as the difference between the current level of occupation and the balanced one. We consider stochastic demands representing different scenarios, each one with its probability. The goal of this chapter is to rebalance an unbalanced situation of a bike sharing system by using a fleet of capacitated vehicles at minimum cost where demands are stochastic. We define the Stochastic Bike sharing Rebalancing Problem and propose Stochastic Programming models and exact and heuristic methodologies to solve them. In particular, we present dedicated L-Shaped algorithms, branch-and-cut algorithms, and some heuristic algorithms making use of the correlation between the nodes, based on their demands. We computationally test our algorithms on real-world newly collected instances. We also define a similar problem in which the load of each vehicle leaving the depot has to be decided in advance and does not depend on the scenario. We compare and comment the objective function of the two problems.

In Section 4.2 we produce a literature review, in Section 4.3 we formally define the problems, for which we propose several formulations and exact algorithms in Sections 4.4 and 4.5. Heuristic algorithms based on correlations between stations are proposed in Section 4.6. Computational results are reported in Section 4.7.

4.2 Literature Review

The bike sharing systems and their issues have been treated in many works lately, including strategical and tactical decisions, by defining and solving several sets of optimization problems, evaluating different levels of detail and various types of mathematical models and algorithms. One of the tactical problems that has been identified and solved is the one concerning the redistribution of the bicycles among the stations that we call *Bike sharing Rebalancing Problem* (BRP). Its deterministic static and dynamic versions

have been tackled. For a complete literature review on the static version we refer the reader to the previous chapters, where recent works on exact and metaheuristic algorithms are described. A dynamic version of the problem is solved heuristically by Contardo et al. [4].

The deterministic versions of the problem have been studied and tackled in many recent works and some papers that we describe in the following made use of historical data. Instead, according to our knowledge, stochastic versions of the BRP have not been considered yet. Regue and Recker [11] also solved a dynamic problem, where the routing and the inventory problems are both accounted and the stochastic part is considered by using historical data by means of a demand forecasting model. Saharidis et al. [12] define a mathematical formulation to design the bike sharing infrastructure, deciding the location and the size of the bike stations, incorporating a hourly demand estimation. Wang and Wang [15] provide an analysis on bike repositioning strategies proposing a simulation study where real time and historical data are considered.

Stochastic optimization and in particular stochastic vehicle routing problems have been studied in several works and with diverse methodologies in the last years, we refer the interested reader to Birge and Louveaux [2] and to King and Wallace [5]. The stochastic vehicle routing problems where the stochastic variables are the demands (i.e., the vehicle routing problems with stochastic demands) have been modeled and solved with a chance constrained programming approach (see, e.g., Gouden and Stewart [13]), with a robust optimization approach (see, e.g, Bertsimas and Simchi-Levi [1]), and with stochastic programming with recourse (see, e.g., Laporte et al. [6]). The reader can have an idea of the heterogeneity of the works on the stochastic vehicle routing problems in Gendreau et al [6]. Besides that, to our knowledge, problems with positive and negative stochastic demands of the same product (i.e., one-commodity pickup and delivery VRPs/TSPs with stochastic demands) have been tackled only by Louveaux and Salazar-González in [7] and [8], where they study the TSP version and a particular attention is payed to the vehicle capacity. In the following we define, model and solve two versions of the family of one-commodity pickup and delivery VRPs with stochastic demands.

4.3 Problems description

The *Stochastic Bike sharing Rebalancing Problem* (SBRP) is modeled on a complete digraph $G = (V, A)$, where $V = \{0, 1, \ldots, n\}$ is the set of vertices including the n customers and the depot (vertex 0) and A is the set of arcs between each pair of vertices. For each vertex $i \in V \backslash \{0\}$ a request $\tilde{q}_i^{\omega}$ is given for every scenario $\omega \in \Omega$, where Ω is the set of all the possible scenarios and p^{ω} is the probability of the scenario ω, knowing that $\sum_{\omega \in \Omega} p^{\omega} = 1$. Requests

can be positive or negative, we refer to as a pickup vertex and delivery vertex if the request is positive or negative, respectively. The quantities picked up at pickup vertices can be used to respond to demand of delivery vertices or go to the depot, so as to some quantities can come from the depot, if needed. We impose that a station with null request must be visited. In the SBRP the demands of different vertices and scenarios can be not respected by paying a penalty k_i for each unit of slack and surplus in vertices $i \in V$. These quantities of exceeding and lacking demands are limited by some considerations on the dimension H_i of each station and the balanced level of occupation d_i of each station. The objective of the SBRP is to drive the fleet of m identical vehicles of capacity Q available at the depot to a set of routes in order to respond to the scenario demands, if possible, or by paying penalties, and minimizing the sum of the traveling costs and the penalty costs.

The *Stochastic Bike sharing Rebalancing Problem with Fixed starting flows* (SBRPF) is a similar problem, where the number of bikes to load on each vehicle leaving the depot have to be determined before the realization of the demands.

4.4 Formulations for the SBRP

In this section we present a set of formulations to solve the SBRP. We adopted the terminology used in Birge and Louveaux [2].

4.4.1 Deterministic Equivalent Program

The first formulation we introduce in (4.1)-(4.12) is the *Deterministic Equivalent Program* (DEP). We define variables $x_{ij}, (i,j) \in A$, that can take value 1 if arc (i,j) is used and 0 otherwise. The cost of traveling arc (i,j) is c_{ij}. Variable f_{ij}^{ω} represents the quantity flowing on the arc $(i,j) \in A$ in scenario $\omega \in \Omega$. Variables s_j^{ω} and y_j^{ω} are the slack and surplus variables used to pay a cost k_j for each unit of slack and surplus when the flows do not respect the demand at vertex $j \in V \setminus \{0\}$ in scenario $\omega \in \Omega$.

$$\text{(DEP)} \quad \min \sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij} + \sum_{\omega \in \Omega} p^{\omega} \sum_{j \in V \setminus \{0\}} k_j (y_j^{\omega} + s_j^{\omega}) \tag{4.1}$$

$$\sum_{i \in V} x_{ij} = 1 \qquad j \in V \setminus \{0\} \tag{4.2}$$

$$\sum_{i \in V} x_{ji} = 1 \qquad j \in V \setminus \{0\} \tag{4.3}$$

$$\sum_{j \in V} x_{0j} \leq m \tag{4.4}$$

$$\sum_{i \in S} \sum_{j \in S} x_{ij} \leq |S| - 1 \qquad S \subseteq V \setminus \{0\}, S \neq \emptyset \tag{4.5}$$

$$f_{ij}^{\omega} \leq Q x_{ij} \qquad (i,j) \in A, \omega \in \Omega \tag{4.6}$$

$$\sum_{i \in V} f_{ji}^{\omega} - \sum_{i \in V} f_{ij}^{\omega} = \tilde{q}_j^{\omega} + y_j^{\omega} - s_j^{\omega} \qquad j \in V \setminus \{0\}, \omega \in \Omega \tag{4.7}$$

$$y_j^{\omega} \leq d_j \qquad j \in V \setminus \{0\}, \omega \in \Omega \tag{4.8}$$

$$s_j^{\omega} \leq H_j - d_j \qquad j \in V \setminus \{0\}, \omega \in \Omega \tag{4.9}$$

$$y_j^{\omega}, s_j^{\omega} \geq 0 \qquad j \in V \setminus \{0\}, \omega \in \Omega \tag{4.10}$$

$$f_{ij}^{\omega} \geq 0 \qquad (i,j) \in A, \omega \in \Omega \tag{4.11}$$

$$x_{ij} \in \{0,1\} \qquad (i,j) \in A. \tag{4.12}$$

The objective function (4.1) of the DEP considers two components, one that aims at minimizing the traveling costs and a second component that pays a penalty k_j for every unit of demand that is not fulfilled, multiplied by the probability of the related scenario. Constraints (4.2) and (4.3) impose that one and only one arc can enter and exit every vertex but the depot. Constraint (4.4) set that the maximum number of arcs leaving the depot is not grater than the available vehicles. Subtours are avoided thanks to constraints (4.5). In (4.6) we state that the flows can be at most the capacity of the vehicles if the corresponding arc is used, 0 otherwise. Constraints (4.7) assert that the difference between the exiting and entering flows in a vertex must be equal to the demand of a certain vertex and scenario corrected by the surplus and the slack variables. In constraints (4.8) and (4.9) a bound is set to the surplus and slack variables. In particular, we state that we cannot exceed the capacity of the station and that we cannot collect more bikes than what we have in the station. To make easier to understand constraints (4.8) and (4.9), we report, in Figure 4.1, the possible balanced level of occupation d_j with respect to a negative or non-negative demand, given that l_j is the current level of occupation of a station $j \in V \setminus \{0\}$. One can correctly say that the current level of occupation l_j is the stochastic variable (more properly l_j^{ω}), although because the balanced level of occupation is fixed for each station, the demand can be considered as the stochastic variable without loss of correctness, being

$$q_j^{\omega} = l_j^{\omega} - d_j, \qquad j \in V \setminus \{0\}, \omega \in \Omega. \tag{4.13}$$

We then impose the flow, slack and surplus variables to be positive in (4.10) and (4.11). In (4.12) the x variables are declared booleans.

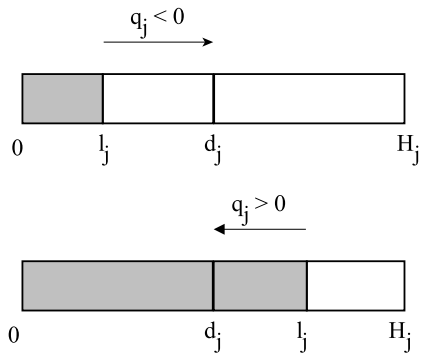


Figure 4.1: Picture of the parameters of a station.

4.4.2 Two-stage Formulation

In this subsection we propose a two-stage Stochastic Linear Programming (SLP) model with simple recourse for the SBRP. As typical for the two-stage SLP models we can separate the DEP into two stages, the first one takes into account the deterministic part of the problem, the decisions to take here and now, whilst the second stage model represents a set of models, one for each scenario, that take as input the decisions made at the first stage and make them fit with the request of the various scenarios after the realization. To perform this fitting a cut is produced and inserted into the first stage model. Both models must be solved in an iterative way: when the inequalities produced by the second stage models do not cut the solution of the first stage then the optimal solution has been found.

$$\text{(1-Stage)} \qquad \min \sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij} + E(z(\omega, y, s)) \tag{4.14}$$

$$\sum_{i \in V} x_{ij} = 1 \qquad j \in V \setminus \{0\} \tag{4.15}$$

$$\sum_{i \in V} x_{ji} = 1 \qquad j \in V \setminus \{0\} \tag{4.16}$$

$$\sum_{j \in V} x_{0j} \leq m \tag{4.17}$$

$$\sum_{i \in S} \sum_{j \in S} x_{ij} \leq |S| - 1 \qquad S \subseteq V \setminus \{0\}, S \neq \emptyset \tag{4.18}$$

$$x_{ij} \in \{0, 1\} \qquad (i, j) \in A. \tag{4.19}$$

The objective function (4.14) of the first stage model contemplates the traveling costs and the expected value of the solution of the second stage. This second component of (4.14) is the expected value of the solutions of the second stage models and substitutes the second component of (4.1), that defined the costs due to the penalties. Constraints (4.15)-(4.19) are similar to (4.2)-(4.5) and (4.12).

$$\text{(2-Stage)} \qquad \min z(\omega, y, s) = \sum_{j \in V \setminus \{0\}} k_j (y_j^\omega + s_j^\omega) \tag{4.20}$$

$$f_{ij}^\omega \leq Q x_{ij} \qquad (i,j) \in A \tag{4.21}$$

$$\sum_{i \in V} f_{ji}^\omega - \sum_{i \in V} f_{ij}^\omega = \tilde{q}_j^\omega + y_j^\omega - s_j^\omega \qquad j \in V \setminus \{0\} \tag{4.22}$$

$$y_j^\omega \leq d_j \qquad j \in V \setminus \{0\} \tag{4.23}$$

$$s_j^\omega \leq H_j - d_j \qquad j \in V \setminus \{0\} \tag{4.24}$$

$$y_j^\omega, s_j^\omega \geq 0 \qquad j \in V \setminus \{0\} \tag{4.25}$$

$$f_{ij}^\omega \geq 0 \qquad (i,j) \in A \tag{4.26}$$

The second stage formulation includes a model for each scenario $\omega \in \Omega$. Constraints (4.22) impose that the difference between the outgoing and the incoming flows coming from the first stage model respects the demands and, if not possible, doing it by using slack and surplus variables. Every unit of slack and surplus is payed into the objective function (4.20) with a cost k_j. The slack and surplus variables are bounded in (4.23) and (4.24). Then we have the non-negativity constraints.

The most common way to solve a two-stage stochastic model is represented by the L-Shaped method (see, e.g., Birge and Louveaux [2]), which solves the first stage LP, takes the linking variables between the two stages (x_{ij}) and uses them as fixed into all the second stage models. The dual problem of the second stage problem is then solved for each $\omega \in \Omega$: if at least one dual of the second stage problem is unbounded we need a feasibility cut, i.e., the first stage solution is not feasible for every scenario and we must impose its feasibility with a cut and solve again the first stage model.When all duals are feasible we could need to introduce an optimality cut, if it does not cut the solution then this is optimal, otherwise the cut that imposes the solution vector to change or the penalty value (η, the variable that considers the expected value of the second stage model) to increase. In this latter case we must recompute the first stage model with the new cut.

Then the dual problem of the second stage problem can be written as in

(4.27)-(4.38).

$$\text{(2-Stage Dual) } \max z_D(\omega, y, s) = Q \sum_{i \in V} \sum_{j \in V} x_{ij} \mu_{ij}^{\omega} + \sum_{j \in V \setminus \{0\}} q_j^{\omega} \phi_j^{\omega} + \sum_{j \in V \setminus \{0\}} d_j \pi_j^{\omega} + \sum_{j \in V \setminus \{0\}} (H_j + d_j) \nu_j^{\omega} \tag{4.27}$$

$$\mu_{jj}^{\omega} \leq 0 \qquad j \in V \tag{4.28}$$

$$\mu_{0j}^{\omega} - \phi_j^{\omega} \leq 0 \qquad j \in V \setminus \{0\} \tag{4.29}$$

$$\mu_{j0}^{\omega} + \phi_j^{\omega} \leq 0 \qquad j \in V \setminus \{0\} \tag{4.30}$$

$$\mu_{ij}^{\omega} + \phi_i^{\omega} - \phi_j^{\omega} \leq 0 \qquad (i,j) \in A, i, j \neq 0, i \neq j \tag{4.31}$$

$$\pi_0^{\omega} \leq k_0 \tag{4.32}$$

$$\pi_j^{\omega} - \phi_j^{\omega} \leq k_j \qquad j \in V \setminus \{0\} \tag{4.33}$$

$$\nu_0^{\omega} \leq k_0 \tag{4.34}$$

$$\nu_j^{\omega} + \phi_j^{\omega} \leq k_j \qquad j \in V \setminus \{0\} \tag{4.35}$$

$$\mu_{ij}^{\omega} \geq 0 \qquad (i,j) \in A \tag{4.36}$$

$$\nu_j^{\omega}, \pi_j^{\omega} \geq 0 \qquad j \in V \setminus \{0\} \tag{4.37}$$

$$\phi_j^{\omega} \, free \qquad j \in V \setminus \{0\}. \tag{4.38}$$

The overall formulation of the L-Shaped method, that we call *L-Shaped Formulation*, can be written as in (4.39)-(4.47), where η is the variable that takes into account the penalties caused by the second stage models and the derived optimality cuts which are made explicit in (4.40). The feasibility cuts are shown in (4.41). $|T_{opt}|$ is the number of iterations providing optimality cuts, while $|T_{feas}|$ is the number of iterations giving feasibility cuts, and $|T_{opt}| + |T_{feas}|$ is the number of the iteration needed to obtain the optimal solution.

$$\text{(L-Shaped) } \min \sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij} + \eta \tag{4.39}$$

$$\eta \geq \sum_{\omega \in \Omega} p^{\omega} \Big(- Q \sum_{(i,j) \in A} \mu_{ij}^{\omega,t} x_{ij} - \sum_{i \in V \setminus \{0\}} \phi_i^{\omega,t} q_i^{\omega} - \sum_{i \in V \setminus \{0\}} \pi_i^{\omega,t} d_i - \sum_{i \in V \setminus \{0\}} \nu_i^{\omega,t} (H_i - d_i) \Big) \qquad t \in T_{opt} \quad (4.40)$$

$$Q \sum_{(i,j) \in A} \hat{\mu}_{ij}^{\omega,t} x_{ij} \geq - \sum_{i \in V \setminus \{0\}} \hat{\phi}_i^{\omega,t} q_i^{\omega,t} - \sum_{i \in V \setminus \{0\}} \hat{\pi}_i^{\omega,t} d_i - \sum_{i \in V \setminus \{0\}} \hat{\nu}_i^{\omega,t} (H_i - d_i) \qquad t \in T_{feas} \quad (4.41)$$

$$\sum_{i \in V} x_{ij} = 1 \qquad j \in V \setminus \{0\} \quad (4.42)$$

$$\sum_{i \in V} x_{ji} = 1 \qquad j \in V \setminus \{0\} \quad (4.43)$$

$$\sum_{j \in V} x_{0j} \leq m \quad (4.44)$$

$$\sum_{i \in S} \sum_{j \in S} x_{ij} \leq |S| - 1 \qquad S \subseteq V \setminus \{0\}, S \neq \emptyset \quad (4.45)$$

$$x_{ij} \in \{0, 1\} \qquad (i,j) \in A \quad (4.46)$$

$$\eta \geq 0. \quad (4.47)$$

Branch-and-cut Implementation

We can solve the model described in (4.39)-(4.47) in a branch-and-cut fashion and we call this implementation *One-cut B&C*. In the DEP, the L-Shaped Formulation, and in the One-cut B&C we insert in the model the subtour elimination constraints (see, e.g., (4.45)) only when violated. To do so we firstly use a heuristic procedure to detect a violated cut, this procedure is called max-back (see, e.g., Naddef and Thienel [9]). If no violated cut is found we use the well known max-flow algorithm to find the most violated one. After the seventh node of the branching tree we use only the max-flow procedure if the current solution is not integer. In the One-cut B&C also the the constraints (4.40) and (4.41) are inserted in a branch-and-cut fashion when violated. This evaluation is performed at each branching node considering also fractional variables. In this case $|T_{opt}| + |T_{feas}|$ represents the number of inserted cuts derived by the scenarios.

4.4.3 Multi-cut Formulation

Another way of solving the two-stage problem is to insert more than one optimality cut for the entire set of scenarios, i.e., to insert one optimality

cut for each scenario that needs its component of costs to increase at a particular iteration. This method is called multi-cut L-shaped method, so we introduce here the *Multi-cut Formulation*, defined in (4.48)–(4.56). We need a variable η^ω for each scenario $\omega \in \Omega$. The summation of these new variables is evaluated into the objective function (4.48) and the new optimality constraints are represented in (4.49). $|T_{opt}| \cdot |\Omega| + |T_{feas}|$ represents the number of iterations needed by the multi-cut L-Shaped algorithm to converge.

$$\text{(Multi-cut)} \quad \min \sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij} + \sum_{\omega \in \Omega} \eta^\omega \tag{4.48}$$

$$\eta^\omega \geq p^\omega \Big(- Q \sum_{(i,j) \in A} \mu_{ij}^{\omega,t} x_{ij} - \sum_{i \in V \setminus \{0\}} \phi_i^{\omega,t} q_i^\omega - \sum_{i \in V \setminus \{0\}} \pi_i^{\omega,t} d_i - \sum_{i \in V \setminus \{0\}} \nu_i^{\omega,t} (H_i - d_i) \Big) \qquad t \in T_{opt}, \omega \in \Omega \tag{4.49}$$

$$Q \sum_{(i,j) \in A} \hat{\mu}_{ij}^{\omega,t} x_{ij} \geq - \sum_{i \in V \setminus \{0\}} \hat{\phi}_i^{\omega,t} q_i^{\omega,t} - \sum_{i \in V} \hat{\pi}_i^{\omega,t} d_i - \sum_{i \in V} \hat{\nu}_i^{\omega,t} (H_i - d_i) \qquad t \in T_{feas} \tag{4.50}$$

$$\sum_{i \in V} x_{ij} = 1 \qquad j \in V \setminus \{0\} \tag{4.51}$$

$$\sum_{i \in V} x_{ji} = 1 \qquad j \in V \setminus \{0\} \tag{4.52}$$

$$\sum_{j \in V} x_{0j} \leq m \tag{4.53}$$

$$\sum_{i \in S} \sum_{j \in S} x_{ij} \leq |S| - 1 \qquad S \subseteq V \setminus \{0\}, S \neq \emptyset \tag{4.54}$$

$$x_{ij} \in \{0, 1\} \qquad (i,j) \in A \tag{4.55}$$

$$\eta^\omega \geq 0 \qquad \omega \in \Omega. \tag{4.56}$$

Branch-and-cut Implementation

Similarly to the L-Shaped Formulation, also the Multi-cut Formulation can be solved in a branch-and-cut fashion: we call this *Multi-cut B&C*. For

both the Multi-cut Formulation and for the Multi-cut B&C we separate constraints (4.54) only when violated. As for the single cut case we use a heuristic procedure called max-back (see, e.g., Naddef and Thienel [9]) up to the seventh node of the branching tree or if the solution is integer. If no violated cut is found we use max-flow algorithm to find the most violated one. Moreover, in the Multi-cut B&C the constraints (4.49) and (4.50) are inserted in a branch-and-cut fashion when violated. This evaluation is performed at each branching node considering also fractional variables. In this case $|T_{opt}| \cdot |\Omega| + |T_{feas}|$ represents the number of inserted cuts derived by the scenarios.

4.5 Formulation for the SBRPF

In stochastic programming models we need to make some decision here and now, in the SBRP this is the case of the route design, while some other decision have to be made after the realization of the demand, in the SBRP these are represented by the definition of the flows of bicycles and of the penalties for the demands we will not respond to. The SBRPF is the stochastic version of the BRP where the flows leaving the depot on each vehicle are variables that represent decisions to be made here and now. Having this information, the bike sharing company will know the routing information and the number of bikes to load on the vehicles leaving the depot. To solve the SBRPF we just present the related deterministic equivalent program because we are interested in comparing the objective function values obtained with respect to the SBRP ones. The formulation is reported in (4.57)–(4.70)

$$\text{(SBRPF)} \quad \min \sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij} + \sum_{\omega \in \Omega} p^{\omega} \sum_{j \in V \setminus \{0\}} k_j (y_j^{\omega} + s_j^{\omega}) \tag{4.57}$$

$$\sum_{i \in V} x_{ij} = 1 \qquad j \in V \setminus \{0\} \tag{4.58}$$

$$\sum_{i \in V} x_{ji} = 1 \qquad j \in V \setminus \{0\} \tag{4.59}$$

$$\sum_{j \in V} x_{0j} \leq m \tag{4.60}$$

$$\sum_{i \in S} \sum_{j \in S} x_{ij} \leq |S| - 1 \qquad S \subseteq V \setminus \{0\}, S \neq \emptyset \tag{4.61}$$

$$f_{0j} \leq Q x_{0j} \qquad j \in V, \omega \in \Omega \tag{4.62}$$

$$f_{ij}^{\omega} \leq Qx_{ij} \qquad (i,j) \in A | i \neq 0, \omega \in \Omega \tag{4.63}$$

$$\sum_{i \in V} f_{ji}^{\omega} - \sum_{i \in V \setminus \{0\}} f_{ij}^{\omega} - f_{0j} = \tilde{q}_j^{\omega} + y_j^{\omega} - s_j^{\omega} \quad j \in V \setminus \{0\}, \omega \in \Omega \tag{4.64}$$

$$y_j^{\omega} \leq d_j \qquad j \in V \setminus \{0\}, \omega \in \Omega \tag{4.65}$$

$$s_j^{\omega} \leq H_j - d_j \qquad j \in V \setminus \{0\}, \omega \in \Omega \tag{4.66}$$

$$y_j^{\omega}, s_j^{\omega} \geq 0 \qquad j \in V \setminus \{0\}, \omega \in \Omega \tag{4.67}$$

$$f_{0j} \geq 0 \qquad j \in V \tag{4.68}$$

$$f_{ij}^{\omega} \geq 0 \qquad (i,j) \in A | i \neq 0, \omega \in \Omega \tag{4.69}$$

$$x_{ij} \in \{0,1\} \qquad (i,j) \in A. \tag{4.70}$$

In the SBRPF Formulation, the objective function and the constraints in (4.57)–(4.61), (4.65)–(4.67), and (4.70) are the same as (4.1)–(4.5), (4.8)–(4.10), and (4.12), while from (4.6), (4.7), and (4.11) we derived (4.62)–(4.64), (4.68), and (4.69). In constraint (4.62) we impose the vehicle leaving the depot to have the same value for all the scenarios and we bound this value at the capacity of the vehicles. All the other flows are bounded by the capacity of the vehicles, but they are scenario dependent, as states constraint (4.63). Constraint (4.64) represents the flow conservation constraint for each vertex but the depot, where the demand must be respected with flows or with surplus and slack variables, and where the flow coming from the depot is the same for all the scenarios. In constraints (4.68) and (4.69) we impose the non-negativity of the flow variables.

4.6 Heuristic algorithm based on correlations

One-commodity pickup and delivery problems, as the ones we are tackling in this chapter, consider positive demands of some vertices to be used for responding to negative demands of other vertices, without having an origin-destination pair. If demands are stochastic, solution methods can take advantage of the negative and positive correlations between vertices, if any. For instance, it is common to have some vertices where the demand is negative for a relevant number of scenarios, while some other vertices will manifest positive demands for a relevant number of scenarios, thus the cor-

relation between one vertex of the first set and a vertex of the second set will be negative. It is reasonable to look for the arcs of the optimal solution among those connecting negatively correlated vertices because coupling them can let the load of the vehicle far from the lower and upper bounds 0 and Q, and hence more vertices can be served with the same vehicle. Of course distances contribute relevantly to the objective function, and it also reasonable to seek the arcs of the optimal solution among those with a small distance cost. These considerations justify the simple heuristic algorithms we propose that consider distances, penalties, and negative correlations in the evaluating function. The two algorithms proposed are based on the well known closest neighbor algorithm (see, e.g., Toth and Vigo [14]) and savings algorithm (see, e.g, Clarke and Wright [3]).

4.6.1 Closest neighborhood with correlations

The first heuristic algorithm we propose to solve the SBRP is modeled on the classical heuristic for solving vehicle routing problems known as closest neighborhood, in which at every iteration the unused closest vertex to the last vertex of the current route is added at the end of the current route, if the insertion is feasible. The version of the closest neighborhood we introduce, called *Closest neighborhood with correlations*, takes into account not only the distance between the two vertices added to the penalties needed to make the insertion feasible for all the scenarios, but also the correlation between the current route and the vertex we are evaluating for the insertion. In particular, the correlation is expressed as the Pearson-Bravais correlation coefficient (see, e.g., Pearson [10]). Let us say that we have two points $i, j \in V \setminus \{0\}, i \neq j$, then the correlation between them is computed as follows:

$$\rho_{ij} = \frac{\sigma_{ij}}{\sigma_i \sigma_j} = \frac{\sum_{\omega \in \Omega}(q_i^\omega - \overline{q}_i)(q_j^\omega - \overline{q}_j)}{\sqrt{\sum_{\omega \in \Omega}(q_i^\omega - \overline{q}_i)^2}\sqrt{\sum_{\omega \in \Omega}(q_j^\omega - \overline{q}_j)^2}}$$

Where $\overline{q}_i = \sum_{\omega \in \Omega} q_i^\omega, i \in V \setminus \{0\}$ and being $-1 \leq \rho_{ij} \leq 1$, where if $\rho_{ij} = -1$ there is a perfect negative correlation between i and j, while i and j are perfectly positively correlated if $\rho_{ij} = 1$. If two vertices are negatively correlated there is a good chance that in many scenarios an increase in the demand of one vertex can correspond to a decrease in the demand of the other vertex, and this means that, if not to expensive, the arc (i, j) could be in the optimal solution. Otherwise, positive correlations suggest that two vertices have similar behaviors among the scenarios, and hence, probably, they will not be close in the optimal solution.

When we are building the solution with the closest neighborhood algorithm we have, on one hand, a vertex to be evaluated for the insertion and, on the other hand, a route being built. Thus, instead of considering a corre-

lation between two vertices, we consider the correlation between the current route R and the vertex $j \notin R$. The correlation is then computed as follows:

$$\rho_{Rj} = \frac{\sigma_{Rj}}{\sigma_R \sigma_j} = \frac{\sum_{\omega\in\Omega}(\sum_{i\in R}(q_i^\omega - \overline{q}_i))(q_j^\omega - \overline{q}_j)}{\sqrt{\sum_{\omega\in\Omega}(\sum_{i\in R}(q_i^\omega - \overline{q}_R))^2}\sqrt{\sum_{\omega\in\Omega}(q_j^\omega - \overline{q}_j)^2}}$$

Where $\overline{q}_R = \sum_{\omega\in\Omega}\sum_{i\in R} q_i^\omega$. Also in this case we have $-1 \leq \rho_{Rj} \leq 1$, where -1 corresponds to perfect negative correlation and 1 to perfect positive correlation.

When considering if appending vertex j at the end of the current route R, $j \notin R$, we then use the following evaluating function:

$$E_C(R, j) = \alpha \left[\frac{c_{R(|R|),j}}{\max_{(i,h)\in A} c_{ih}} + \frac{\sum_{\omega\in\Omega} p^\omega k_j (y_j^\omega + s_j^\omega)}{Q|V|}\right] + (1-\alpha)\rho_{Rj} \tag{4.71}$$

Being $R(|R|)$ the last vertex of the current route R, and where α is a rational number between 0 and 1. In the component of the evaluating function multiplied by α we consider a normalized distance cost and normalized penalties of the various scenarios. In the second component of the evaluating function we consider the correlation of the insertion as we computed in (4.71). When the insertion of every vertex at the end of the current route is more expensive than opening a new route or when no more vertex is feasibly inserted then a new route is open. The algorithm is terminated when all the vertices are assigned to a route.

4.6.2 Savings with correlations

The second heuristic algorithm we propose to solve the SBRP is based on the well known savings algorithm, in which the iterative cheapest and feasible merging of two routes is applied. In the version of the algorithm we propose, that we call *Savings with correlations*, we consider the iterative merging of two routes if it is feasible for all scenarios (where the concept of feasibility includes the payment of a penalty if non responding to some demands of some vertices in a scenario). Besides that, we also take into account the correlation between the two routes, let us say R and P, that we are considering for merging, which is computed as follows:

$$\rho_{RP} = \frac{\sigma_{RP}}{\sigma_R \sigma_P} = \frac{\sum_{\omega\in\Omega}(\sum_{i\in R}(q_i^\omega - \overline{q}_R))(\sum_{j\in P}(q_j^\omega - \overline{q}_P))}{\sqrt{\sum_{\omega\in\Omega}(\sum_{i\in R}(q_i^\omega - \overline{q}_R))^2}\sqrt{\sum_{\omega\in\Omega}(\sum_{j\in P}(q_j^\omega - \overline{q}_P))^2}}$$

Where $\overline{q}_R = \sum_{\omega\in\Omega}\sum_{i\in R} q_i^\omega$ for route R. Thus the evaluating function used when merging two routes results in:

$$E_S(R,P) = \alpha(\kappa + \sigma) + (1-\alpha)\rho_{RP} \tag{4.72}$$

Being:

$$\kappa = \frac{c_{R(|R|-1),P(1)} - c_{R(|R|-1),0} - c_{0,P(1)}}{\max_{(i,h)\in A} c_{ih}}$$
$$\sigma = \frac{\sum_{\omega\in\Omega} p^{\omega} k_{P(1)} (y^{\omega}_{P(1)} + s^{\omega}_{P(1)})}{Q|V|}$$

Where $\alpha \in \mathbb{R}$ is a number between 0 and 1, $R(|R|-1)$ is the last vertex of route R before the depot, and $P(1)$ is the first vertex of route P after the depot. In the first part of the evaluating function in (4.72), we consider in κ the normalized savings of merging the two routes R and P in this direction and in σ the possible normalized penalties to pay if not respecting the demand of $P(1)$. The second part of (4.72) accounts for the correlation between R and P.

In both algorithms the solution does not necessarily respect the maximum number of vehicles, and thus routes, available, even if the algorithms try to minimize the number of routes.

4.7 Computational Results

In this section we present the computational tests on the different methodologies used to solve the SBRP and the SBRPF. The algorithms were run on on Intel Core i3-2100 CPU, 3-10 GHz, 4.00 GB, and we used CPLEX 12.6 as the MILP solver by imposing the selection of a single processor. The tests have been performed on instances derived by real-world data. We firstly describe how we collected those data.

4.7.1 Instances

To produce computational evaluation of the algorithms proposed to solve the two problems here defined, the SBRP and the SBRPF, we collected a set of real-world instances. Firstly we considered a set of usage data of 7 month concerning the bike sharing system of the city of Reggio Emilia, in Northern Italy. These data has been given to us by the operator of the system. In addition to this, in order to have a larger number of testing instances, we collected some data of Capital Bike, the bike sharing system installed in Washington and Arlington, in the USA. The Capital Bike site furnishes with complete data on the usage of the system for many years. We made use of the usage data of the third quarter of 2014. We also collected the geographic coordinates and the capacity of each station and used historical

data to determine the demands of each scenario. To this extent, we clarify that some correction on data has been necessary because the rebalancing is already applied to the system and some demands appear to be too large with respect to the station capacity. From these data we derived several instances by selecting arbitrarily sets of stations counting from 20 to 66 stations each. We selected randomly a station among all to be used as the depot.

4.7.2 Results

We report here some results of tests performed on the collected instances. Firstly we tested the heuristic algorithms developed: in Table 4.1 we report the results obtained by testing the closest neighborhood with correlations. The first column shows the name of the instance, where RE stands for Reggio Emilia, while W stands for Washington; the numbers reports the number of vertices ($|V|$) and the capacity of the vehicles (Q), respectively. The best lower bound obtained with the exact algorithms is reported (bold if it is the optimal solution value) and the percentage gap between this value and the solution value obtained thanks to the closest neighborhood with correlations is shown with respect to the different values of α used in the evaluating function. Similarly, the results for the savings with correlations are reported in Table 4.2.

In Tables 4.1 and 4.2 the best gap obtained is highlighted. One can notice that the heuristic algorithms do not present very good gaps with respect to the best known lower bound. These gaps are large because the algorithms move in the space of feasible solutions and it is not easy to build competitive solutions that are feasible for all the scenarios (172, in the case of Reggio Emilia, and 91 in the case of Washington). Besides that, one can observe that the savings with correlations shows better results with respect to the closest neighborhood with correlations algorithm. Moreover, it can be noticed that for some level of α the solution values are improved while considering the correlations with respect to the classical heuristics, indeed values of α around to 0.9 for the closest neighborhood with correlations and around 0.8 for the savings with correlations algorithm show the best results.

We now consider the results obtained thanks to the exact algorithms we developed. In Table 4.3 we report the results on the collected test instances while running the DEP Formulation. In particular, we report the name of the instance, the best lower bound (LB) and upper bound (UB), if any, and we compute and show the percentage gap between those two values. In the last column we report the time, in seconds, needed to terminate the algorithm within a time limit set to one hour. Similarly, in Tables 4.4, 4.5, 4.6, and 4.7 the results on the same instances given by the L-Shaped Formulation, the Multi-cut Formulation, the One-cut B&C implementation, and the Multi-cut B&C implementation are reported.

One can see that only the One-cut B&C and the Multi-cut B&C algo-

		α										
Inst.	LB	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
RE(16,20)	**16900.0000**	39.43	43.67	31.58	31.58	38.55	35.00	30.45	21.76	21.03	**19.91**	25.55
W(20,25)	**81576.3736**	77.56	55.57	55.57	53.28	52.95	50.40	50.11	49.41	49.41	49.21	**49.16**
W(20,15)	**77306.7802**	80.92	80.93	**77.28**	**77.28**	80.83	80.98	81.15	81.15	80.97	77.33	80.91
W(30,20)	**101515.3967**	75.53	66.51	65.38	63.27	59.76	58.63	60.42	**54.53**	67.17	67.42	67.48
W(30,35)	**120517.5714**	75.92	68.60	66.33	64.33	62.82	60.92	60.51	58.35	55.62	54.12	**50.86**
W(40,40)	101718.0349	83.69	77.05	79.14	79.14	79.14	79.47	78.77	75.32	**74.59**	77.53	77.65
W(40,35)	103300.4636	87.86	85.62	85.63	85.63	85.63	**85.50**	85.63	86.68	86.53	86.51	86.62
W(50,20)	121691.0000	87.91	85.83	85.80	83.32	**83.07**	85.40	85.40	87.46	83.80	84.79	84.91
W(66,15)	**131371.9231**	89.33	79.20	64.28	57.04	54.39	49.98	49.60	47.72	43.19	38.80	**34.85**
Avg.		77.58	71.45	67.89	66.10	66.35	65.15	64.67	62.49	62.48	**61.74**	62.00

Table 4.1: Percentage gap between the solution of the closest with correlations and the best lower bound for all instances by changing the parameter α.

		α										
Inst.	LB	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
RE(16,20)	**16900**	44.41	37.41	25.88	17.96	17.96	15.50	9.63	9.63	**6.11**	8.65	8.65
W(20,25)	**81576.3736**	62.08	67.51	55.40	52.42	52.58	51.34	48.58	45.59	**43.81**	44.20	60.15
W(20,15)	**77306.7802**	77.95	78.00	81.00	80.87	77.56	77.35	80.79	77.04	76.97	76.88	**71.35**
W(30,20)	**101515.3967**	68.73	61.39	58.11	63.89	60.90	55.71	57.76	48.00	**21.16**	44.87	44.67
W(30,35)	**120517.5714**	65.02	68.17	67.15	63.89	62.88	59.78	57.85	55.02	51.01	50.81	**50.32**
W(40,40)	101718.0349	74.52	75.42	78.25	78.81	75.02	74.05	77.33	76.89	**72.83**	75.48	75.64
W(40,35)	103300.4636	85.85	84.26	83.91	85.13	85.14	85.00	86.04	86.12	**83.32**	84.59	84.58
W(50,20)	121691.0000	85.02	85.70	85.52	82.59	**82.34**	85.06	83.88	84.83	83.13	83.14	82.82
W(66,15)	**131371.9231**	89.62	85.82	84.19	82.37	82.37	72.81	49.02	47.75	43.99	40.39	**28.13**
Avg.		72.58	71.52	68.82	67.55	66.31	64.07	61.21	58.98	**53.59**	56.56	56.26

Table 4.2: Percentage gap between the solution of the savings with correlations and the best lower bound for all instances by changing the parameter α.

rithms can find a feasible solution for all the evaluated instances, in particular the Multi-cut B&C is the algorithm, among all those presented, that found the smallest percentage gaps, that are 4.88% in average. On the other hand the L-Shaped Formulation and the Multi-cut Formulation can get the optimal values of six instances out of nine, while for $W(40, 40)$, $W(40, 35)$, and $W(50, 20)$ they could not find a feasible solution. The DEP Formulation could find a feasible solution only for the first five instances, but furnished with the best lower bound for $W(40, 40)$ and $W(40, 35)$, among all. The best upper bound for $W(50, 20)$ is given by the Multi-cut Formulation. The best solving times are obtained by the Multi-cut B&C algorithm that needs 1248.43 seconds to terminate, in average. Also the Multi-cut Formulation, which made use of a multi-cut L-Shaped algorithm, offers promising solving times, that is in average 1252.45 seconds. To conclude, the Multi-cut B&C algorithm seems the best performing one among the ones we tested on our test instances.

DEP				
Inst.	LB	UB	$\%gap$	$t(sec)$
RE(16,20)	16900.0000	16900.0000	0.00	9.88
W(20,25)	81576.3736	81576.3736	0.00	424.29
W(20,15)	77306.7802	77306.7802	0.00	31.79
W(30,20)	101516.0000	101516.0000	0.00	412.12
W(30,35)	120517.5714	120517.5714	0.00	1277.20
W(40,40)	101718.0349	-	-	3600.00
W(40,35)	103300.4636	-	-	3600.00
W(50,20)	117067.0879	-	-	3600.00
W(66,15)	109426.9021	-	-	3600.00
Avg.				1839.476

Table 4.3: Results obtained by running the DEP Formulation.

L-Shaped				
Inst.	LB	UB	$\%gap$	$t(sec)$
RE(16,20)	16900.0000	16900.0000	0.00	0.42
W(20,25)	81576.3736	81576.3736	0.00	41.41
W(20,15)	77306.7802	77306.7802	0.00	0.88
W(30,20)	101516.2967	101516.2967	0.00	2.48
W(30,35)	120517.5714	120517.5714	0.00	3.77
W(40,40)	96715.0000	-	-	3600.00
W(40,35)	92828.0000	-	-	3600.00
W(50,20)	118714.0000	-	-	3600.00
W(66,15)	131313.6264	131313.6264	0.00	3258.75
Avg.				1567.52

Table 4.4: Results obtained by using the L-shaped method to solve the L-Shaped Formulation.

In Table 4.8 we report the results for the SBRPF we performed according

Lshaped Multi-cut				
Inst.	LB	UB	$\%gap$	$t(sec)$
RE(16,20)	16900.0000	16900.0000	0.00	1.06
W(20,25)	81576.3736	81576.3736	0.00	16.13
W(20,15)	77306.7802	77306.7802	0.00	2.95
W(30,20)	101516.2967	101516.2967	0.00	9.95
W(30,35)	120517.5714	120517.5714	0.00	7.83
W(40,40)	96715.0000	-	-	3600.00
W(40,35)	92828.0000	-	-	3600.00
W(50,20)	121691.0000	-	-	3600.00
W(66,15)	131313.6264	131313.6264	0.00	434.12
Avg.				1252.45

Table 4.5: Results obtained by using the L-shaped multi-cut method to solve the Multi-cut Formulation

One-cut B&C				
Inst.	LB	UB	$\%gap$	$t(sec)$
RE(16,20)	16900.0000	16900.0000	0.00	3.60
W(20,25)	81576.3736	81576.3736	0.00	151.34
W(20,15)	77306.7802	77306.7802	0.00	3.09
W(30,20)	101516.2967	101516.2967	0.00	15.83
W(30,35)	120517.5714	120517.5714	0.00	16.15
W(40,40)	100158.2076	2502063.0000	96.00	3600.00
W(40,35)	101506.7796	187114.7802	45.75	3600.00
W(50,20)	121388.6227	3576619.0000	96.61	3600.00
W(66,15)	131313.6264	131313.6264	0.00	2541.60
Avg.			26.48	1503.51

Table 4.6: Results obtained by using the One-cut B&C algorithm with the L-Shaped method derived cuts.

Multi-cut B&C				
Inst.	LB	UB	$\%gap$	$t(sec)$
RE(16,20)	16900.0000	16900.0000	0.00	2.23
W(20,25)	81576.3736	81576.3736	0.00	26.90
W(20,15)	77306.7802	77306.7802	0.00	3.25
W(30,20)	101516.2967	101516.2967	0.00	10.30
W(30,35)	120517.5714	120517.5714	0.00	14.23
W(40,40)	100531.6473	133940.7802	24.94	3600.00
W(40,35)	101984.7100	121856.5385	16.31	3600.00
W(50,20)	121561.5534	124862.3956	2.64	3600.00
W(66,15)	131313.6264	131313.6264	0.00	378.96
Avg.			4.88	1248.43

Table 4.7: Results obtained by using the Multi-cut B&C algorithm with the L-Shaped multi-cut method derived cuts.

to what we did for the SBRP. One can notice that the instances that were easy to solve for the SBRP remain easy to solve also for the SBRPF and the instances that were hard to solve for the SBRP remain hard to solve for the SBRPF. We can observe that the objective function value does not change remarkably if compared to the SBRP one. This can lead to the conclusion that the SBRP solution is robust even if a a priori decision on the load of the vehicles is not contemplated.

SBRPF DEP				
Inst.	LB	UB	%gap	t (sec)
RE(16,20)	16922.5989	16922.5989	0.00	17.04
W(20,25)	82054.3956	82054.3956	0.00	518.55
W(20,15)	77389.1978	77389.1978	0.00	36.94
W(30,20)	101928.4835	101928.4835	0.00	785.06
W(30,35)	120742.8462	120742.8462	0.00	1051.97
W(40,40)	101901.4091	-	-	3600.00
W(40,35)	103615.1235	-	-	3600.00
W(50,20)	117564.6711	-	-	3600.00
W(66,15)	110455.1959	-	-	3600.00
Avg.				1867.73

Table 4.8: Results for the SBRPF on the test instances obtained by running the corresponding DEP Formulation.

4.8 Conclusions

In this chapter we defined and solved the SBRP, the stochastic version of one a tactical problem that arises in bike sharing systems, that is the one of redistributing bicycles among the bike stations of the system, the BRP. To solve the SBRP we proposed a deterministic equivalent program formulation, a two-stage stochastic linear programming model with simple recourse solved by means of a L-Shaped algorithm, a multi-cut L-Shaped algorithm, and two branch-and-cut algorithms. To obtain upper bounds for the SBRP we developed two heuristic algorithms that take advantage of the information on the demands on different scenarios considering the negative and positive correlations between the vertices. We tested all the algorithms on newly collected real-world instances proving the effectiveness of our methodologies, in particular, good results can be obtained with branch-and-cut algorithms where cuts are derived from the multi-cut L-Shaped method. Moreover, the proposed heuristic algorithms showed improved solutions when accounting for correlations in the evaluating functions. Finally, we defined a stochastic version of the BRP where the loads on vehicles leaving the depot must be decided before the realization of the demands. We compare the values of the solutions of the two problems, and the objective functions differs of a small amount of costs.

Bibliography

[1] D. J. Bertsimas and D. Simchi-Levi. A new generation of vehicle routing research: robust algorithms, addressing uncertainty. *Operations Research*, 44:286–304, 1996.

[2] J. R. Birge and F. Louveaux. *Introduction to stochastic programming*. Springer Science & Business Media, 2011.

[3] G. Clarke and J. W. Wright. Scheduling of vehicles from a central depot to a number of delivery points. *Operations research*, 12:568–581, 1964.

[4] C. Contardo, C. Morency, and L.-M. Rousseau. Balancing a dynamic public bike-sharing system. Technical Report CIRRELT-2012-09, CIRRELT, 2012.

[5] A. J. King and S. W. Wallace. *Modeling with stochastic programming*. Springer Science & Business Media, 2012.

[6] G. Laporte, F. Louveaux, and L. Van Hamme. An integer l-shaped algorithm for the capacitated vehicle routing problem with stochastic demands. *Operations Research*, 50:415–423, 2002.

[7] F. Louveaux and J.-J. Salazar-González. On the one-commodity pickup-and-delivery traveling salesman problem with stochastic demands. *Mathematical programming*, 119:169–194, 2009.

[8] F. Louveaux and J.-J. Salazar-González. Solving the single vehicle routing problem with variable capacity. *Transportation Science*, 2014.

[9] D. Naddef and S. Thienel. Efficient separation routines for the symmetric traveling salesman problem i: general tools and comb separation. *Mathematical Programming*, 92:237–255, 2002.

[10] K. Pearson. Notes on the history of correlation. *Biometrika*, 25–45, 1920.

[11] R. Regue and W. Recker. Proactive vehicle routing with inferred demand to solve the bikesharing rebalancing problem. *Transportation Research Part E: Logistics and Transportation Review*, 72:192–209, 2014.

[12] G. K. D. Saharidis, A. Fragkogios, and E. Zygouri. A multi-periodic optimization modeling approach for the establishment of a bike sharing network: a case study of the city of athens. In *Proceedings of the International MultiConference of Engineers and Computer Scientists*, 2, 2014.

[13] W. R. Stewart and B. L. Golden. Stochastic vehicle routing: A comprehensive approach. *European Journal of Operational Research*, 14:371–385, 1983.

[14] P. Toth and D. Vigo. *Vehicle Routing: Problems, Methods, and Applications*, SIAM, 2014.

[15] I.-L. Wang and C.-W. Wang. Analyzing bike repositioning strategies based on simulations for public bike sharing systems: Simulating bike repositioning strategies for bike sharing systems. In *Advanced Applied Informatics (IIAIAAI), 2013 IIAI International Conference on IEEE*, 306–311, 2013.

Part III

Models and Algorithms for Earthwork Optimization Problems

Chapter 5

A Two-Phase Earthwork Optimization Model for Highway Construction

[1]

One of the main activities in highway construction is earthwork, that is a complex process involving excavation, transportation, and filling of large quantities of different earth material types. Earthwork operations are costly, and undergo several constraints due to their large environmental and social impacts. Using mathematical models to produce a minimum-cost earthwork plan that satisfies all constraints is thus of great significance for enhancing the productivity of the overall construction project.

This chapter presents an earthwork optimization system based on the use of linear programming that operates in a novel two-phase approach. In the first phase an aggregate model determines the feasibility of the overall project, whereas in a second phase disaggregate models determine the actual flows of each material.

The developed quantitative method for earthwork optimization includes interesting features derived from the everyday activity of one of the major European companies in construction. The new two-phase optimization involves classical decisions on excavation, filling, quarries, dump sites, and temporary depots, but it also accounts for several novelties. One innovation is the presence of time windows imposed by a tight production schedule that let the model decide when to conclude the tasks within the time windows. Another new feature is the installation of recycling facilities, which transform the excavated materials thanks to a separation and a mixing process so

[1] The results of this chapter appear in: C. Bogenberger, M. Dell'Amico, G. Fuellerer, G. Hoefinger, M. Iori, S. Novellani and B. Panicucci. A Two-Phase Earthwork Optimization Model for Highway Construction. *to appear on Journal of Construction Engineering and Management*, 2015.

to obtain several materials apt for filling. Our model includes also a variety of materials for filling and to be processed in recycling plants. Moreover, the explicit integration of the project with the existing public road network is considered so to account for local authority restrictions, and to consider multiple paths to link the construction site to the external facilities.

Extensive computational results are obtained by running the models on a set of realistic instances, and show the efficiency of the proposed approach in solving complex earthwork problems.

Keywords: Earthwork, Mathematical Models, Linear Programming, Highway Construction.

5.1 Introduction

A highway construction project involves many steps and may take years to be completed. Before construction starts, an accurate topographic survey of the area is developed and a layout of the future road is designed, specifying the alignment (i.e., the base line of the road) and the grades (i.e., the slopes of the longitudinal road line). Once the layout of the road has been completed, actual *earthwork* can begin. Earthwork is, in a few words, the process of moving earth from those areas that are too high with respect to the road grades to those areas that are too low, and is a crucial aspect of the overall construction problem.

More in details, once the road layout is given, one determines the *cut areas*, where some earth must be dug to accomplish with the required road grades, and the *fill areas*, where instead some earth must be placed, and easily evaluate the cut and fill volumes for each area. The basic earthwork problem consists then in digging earth from cut areas and moving it to fill areas, with the goal of minimizing operational costs.

This basic problem is known in the literature as the *cut and fill problem*, and has been described in classical textbooks (see, e.g., Ahuja et al. [1]) as a typical application that can be solved by *Minimum Cost Flow* (MCF). Indeed, after associating the cut and fill areas to the nodes of a network and assigning transportation costs to its arcs, the MCF allows to find the flows that move the required volumes at minimum cost, that is, an optimal earthwork solution.

This simple model has been successively expanded to include complications arising in real-world construction problems, such as the presence of different types of earth, possible transport road constraints (e.g., accessibility and/or maximum flow restrictions), and the need of facilities such as dump sites or quarries to accommodate excess or lack of materials. Starting from Stark and Nicholls [17], *Linear Programming* (LP) became then a standard tool for solving optimization issues associated to the earthwork activities, because it allowed to easily extend the original MCF model to

include such additional constraints.

The aim of this chapter is to further advance the research on earthwork modeling, by including novel features that derive from the everyday construction activities of one of the European leaders in civil engineering. The activity that they perform for highway construction is very complex, and has been the motivation that has led us to the development of the models presented in this chapter. Previously, the construction company (Strabag AG.) relied on tools, such Excel files and mass diagrams, integrated by the long term experience of their practitioners. Thanks to the method presented in this chpater Strabag, and other construction companies, can quickly evaluate different configurations and obtain optimal solutions within seconds leaning on a fast and reliable support for decision making.

The optimal approach that we implemented to solve this challenging problem is based on two phases that contemplate an aggregate model and a disaggregate one, both based on LP. The aggregate model associates a variable with each flow of a material that travels along an arc on a given period, independently from its actual origin and destination. It is fundamental to investigate the feasibility of the project, considering the given time horizon, the available resources, and the strong capacity constraints on the arcs. Thanks to the quite small computing times that we managed to obtain for its solution, the aggregate model may be solved several times, under different network configurations, and is thus very helpful to find the best way to ensure the feasibility of the problem.

To translate the resulting aggregate solution into practical actions involving each material in each period, the disaggregate model is then invoked. This model computes precise flows by taking advantage of the fact that vehicles can transport just one material at a time. Indeed, we decompose the problem in several models, one for each period and material, where decisions made at one model do not affect decisions made on the others. Each model outputs the optimal flow paths, together with the corresponding volumes, between each material origin and destination.

Aggregate and disaggregate models can be run many times, in a *rolling horizon* perspective, to take into consideration the fact that the every day activity may diverge from the planned solution. Adjustments, as well as new information on costs or availability of resources, are easily included in the models by just modifying the input files. Our approach thus resulted in a powerful and effective decision support system. It is now being used by the partner company in a highway construction process which is currently under development in Northern Italy near the city of Milan.

5.1.1 Main Contributions

We advance the Body of Knowledge on quantitative methods for earthwork optimization. We do this by proposing a new mathematical model based on

two phases: an aggregate and a disaggregate one. Within the method we include a set of properties that were not treated or only partially treated in the literature and that are important not only for our case study. We comment them in the following:

Multiple materials: in our model we do not consider only simple earth, as in the basic cut and fill problem, but we optimize the flows of ten different materials. To our knowledge few papers accounts for multiple materials (see, e.g. Hare et al. [8] and Ammar et al. [2]), but in their works they consider a multiple set only for digging materials, where some can be used for filling and some others must be dumped, but they do not consider different types of materials for filling or to be used for recycling process. In our work we consider ten materials including three types of excavated earth, differentiated according to their quality, and four different filling materials such as asphalt, concrete and the two high quality materials coming from digging.

Recycling of the excavated materials: many papers consider to reuse excavated materials directly for filling, some of them also call it recycling (see, e.g. Ji et al. [10]). In our chapter we use two of the dug materials for filling without performing any other transformation, but we also consider a transformation process that we call recycling. The recycling activity includes two transformations of materials to be performed in two plants. The first plant is called Separation plant where a refinement of the high quality excavated material is performed, separating the rocks from the soil and crushing it at the desired level. Some of the resulting material has to be dumped, while the others two can be mixed with cement or bitumen in Mixing plants to produce, respectively, asphalt and concrete, used for filling. To our knowledge this way of recycling materials has not been tackled yet in the literature related to quantitative methods for earthwork optimization.

Time windows: in our work the process is planned according to a specific schedule, that, borrowing a terminology of production literature, we define Master Production Schedule (MPS). The MPS defines the activities to be performed in the overall period, the location where to dig and/or fill which material and the possible time windows in which they have to be completed. Time windows imposed by the MPS have already been inserted in complex construction projects, but we have not found any explicit reference in the existing Body of Knowledge in quantitative methods for earthwork optimization. Thus the explicit insertion of time windows in our model is a challenging novelty.

External network: quarries and dump sites are located outside with respect to the construction site, thus the company vehicles need to cross the public external network. Traffic restriction of heavy vehicles are normally imposed or contracted with the local administrations. This is expressed in amount of cubic meters per day on a restricted set of roads. To deal with that we consider capacity restriction on the arcs representing the external network, the availability of different paths on alternative roads, and the use

of access points from the construction site to the external network. To our knowledge just Burdett and Kozan [7] consider the problem of having an external network. To create a path from the construction site to an external site they use the Dijkstras algorithm to solve the shortest path, but this algorithm does not allow to have multiple paths from two locations: thus they consider only one path otherwise they must fix some dummy nodes. In both disaggregate models we present this problem does not arise , indeed the multi-commodity model can evaluate multiple arcs between two node, and the path-based model makes use of the labeling algorithm, that can find more than one path between two nodes.

These innovations with respect to the Body of Knowledge are motivated by the Strabag study case we considered. They appear in the model as constraints but they can be easily inactivated one by one, so to let the model to be very general and applied to many other cases.

5.2 Problem Description

Earthwork is a fundamental stage in highway construction process that involves several activities and requires careful planning and various evaluations, including environmental and structural assessments. The main earthwork activities that we identified are depicted in Figure 5.1. They include digging, filling, recycling, material disposal in dump sites, temporary allocation of materials in depots, and material acquisition from quarries.

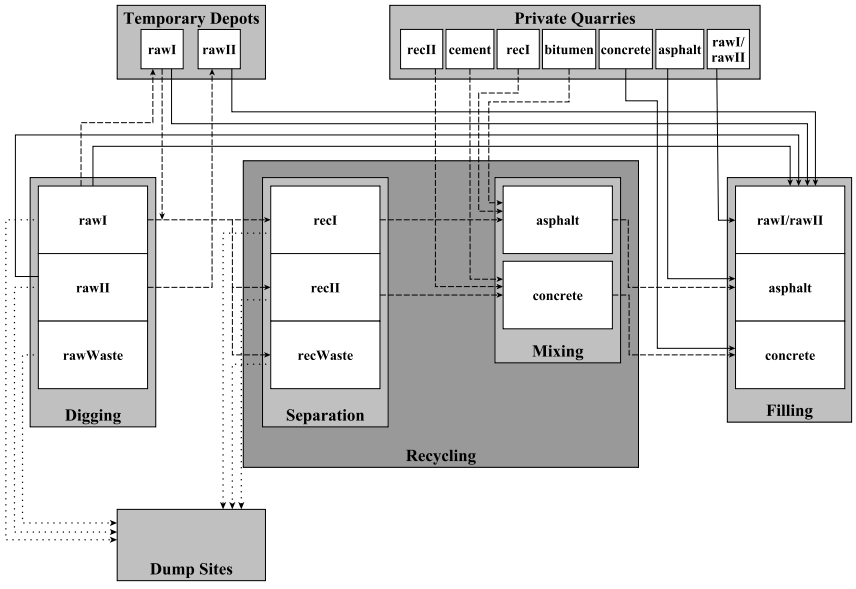


Figure 5.1: Activities flow chart.

The *digging* activities consist of cutting and moving a huge amount of

earth, typically in the order of (tens of) millions of cubic meters by means of special plant machinery such as excavators and trucks. The *filling* activities consist of carrying across the construction site a certain amount of a certain material, to, e.g., fill earth cuts and trenches, or build bridges and artificial galleries.

A fundamental step in the digging process is the on-site classification, that is, earth materials are classified depending on their geological characteristics and treated consequently. Indeed, materials excavated within the site are often unacceptable for construction and must be recycled (through a recycling process to be described below) or disposed of. According to the field activity , three types of dug earth materials can be distinguished, depending on whether they can be reused for purposes of filling, recycled, or else wasted on a dump site. In this chapter we denote them by *rawI* (high-quality soil and gravel), *rawII* (soil suitable for filling but not for recycling), and *rawWaste* (waste product without the possibility of further use). Reusing those materials, instead of throwing them directly in dump sites, is environmentally relevant and can significantly reduce the cost of earthwork. RawI and rawII materials can be used directly for filling or, if necessary, stored in temporary depots and used later on in the construction process. Temporary depots are flat areas that are rented close to the construction site, where dug materials can be stored until they are needed for filling. RawI material can also be reconstituted into other products appropriate for filling, that is, *asphalt* and/or *concrete*, by means of a recycling process.

The inclusion of recycling activities in optimizing earthwork process is, as far as we know, a novelty of this chapter. *Recycling* is a two-stage process (see again Figure 5.1). In the first stage, *separation*, rawI material is transported to the so-called *separation plants*, where it is broken and crushed into two types of recycled aggregates: *recI* and *recII*. In this stage some waste material, denoted by *recWaste*, is also produced. In the second stage, *mixing*, these recycled aggregates are carried to specialized plants where they are mixed with other materials to obtain asphalt or concrete. More precisely, in *asphalt mixing plants* recI is mixed with *bitumen* to produce asphalt, and in *concrete mixing plants* recII is mixed with *cement* to produce concrete. The waste produced during separation, denoted by *recWaste*, as well as rawWaste and possible surplus of rawI, rawII, recI, and recII are dumped into dump sites. If necessary, raw material for producing asphalt and concrete (i.e., recI and recII), as well as other materials required for filling can be acquired from private quarries.

The earthwork process must take into account the fact that recycling and mixing plants, quarries, dump sites, and temporary depots have physical capacity limitations on the total quantity of materials that they can process. Moreover, it must be considered that the aforementioned activities are bounded in time by the time windows defined by the MPS. In particular, the overall planning horizon is divided into periods (weeks in our study

cases) and each activity time window is represented by a starting and an ending period. The optimization of the earthwork process requires to consider the transportation costs, as well as the costs incurred for storing materials in temporary depots, acquiring materials from quarries, and disposing of materials into dump sites.

To take into consideration all these issues, we modeled the earthwork activities by using a directed graph, described in detail below. Briefly, the highway building site is discretized in segments, and each segment is associated with a vertex of the graph. The graph is then extended by adding vertices that correspond to quarries, dump sites, temporary depots, and recycling plants. For modeling purposes we also add a vertex for each digging and filling activity to be performed. A set of arcs is then included to represent the possible flows of materials from a vertex to another.

The arcs connecting the vertices are assigned a maximum flow capacity, that limits by above the sum of the material flows that can be carried over the arc in a given period, and can vary during the time. With respect to this point, we mention another interesting contribution of our approach, that is, the differentiation between *private* and *public* networks. The private network is in practice the highway being built, whereas the public network is the existing neighbor road infrastructure, that is forcedly affected by the construction process. Indeed, some public roads external to the building site must also be included into the graph to take into account different paths that can be used to reach dump sites and quarries, that are typically located far away from the site. These roads undergo the national transportation laws of the country where the construction is taking place, and can have very small flow capacities, due to the fact that earth transportation is typically performed by large and heavy trucks that have a large environmental and social impact on the surrounding areas.

In this framework, the goal of our optimization is to decide when and how moving, storing, recycling, buying, or wasting materials, while responding to digging and filling requests in the imposed time windows, respecting capacity conditions on nodes and arcs, and minimizing total costs.

5.3 Literature Review

Apart from the basic MCF applications already discussed in the introduction, see again Ahuja et al. [1], the first LP model to provide the earthwork allocation that minimizes the cost of the earthmoving activities, once an initial layout is given, was suggested in Stark and Nicholls [17] and then developed by Stark and Mayer [16]. The cited model solves the so-called *earthwork allocation problem*, that includes the presence of capacitated quarries and dump sites, but does not assess the presence of multiple types of earth and assumes constant unit costs for the earthwork activities.

Easa [6] attempted to combine the design of roadway grades with the basic earthwork allocation problem, limiting his study to a single type of earth and using a trial and error approach. A combination of grade selection with the earthwork allocation problem in a single LP problem was later developed by Moreb [14]. To account for the different earth material types which are usually encountered in construction practice, Mayer and Stark [13] incorporated the differing earth characteristics by creating different variables and costs for dirt, gravel, and rock material. Successively, a model that accommodates for multiple material types was also developed by Ammar et al. [2].

The basic model of Stark and Nicholls [17] has also been extended to include different cost functions. In particular, Easa [4, 5] proposed a *Mixed Integer Linear Programming* (MILP) model to incorporate stepwise unit costs, and a quadratic model to account for linear growth in movement costs. In Karimi et al. [11] uncertainties in unit cost coefficients of earthwork activities and facility capacities were considered by using a fuzzy LP model. Another extension of the basic earthwork allocation model, encompassing the project duration and a dynamic selection of quarries and dump sites, was introduced by Jayawardane and Harris [9], and solved by employing MILP. Another MILP model was used in Hare et al. [8], to capture the presence of natural obstacles that limit earth movements.

Marzouk and Moselhi [12], by using computer simulation and genetic algorithms, considered a multi-objective framework for managing earthmoving operations so as to improve productivity, efficiency, and safety. Multi-objective simulation-optimization techniques were also used in Zhang [20], where earthmoving operations are modeled through simulation and a particle swarm optimization method is used to avoid exhaustive simulation experiments of all the alternatives. Xianja et al. [19] established a bilevel programming model, where the objective of the upper level is that of minimizing the transportation cost, while the objective of the lower level is to balance the supply and demand of earthwork in the digging and filling activities. The model is solved by combining particle swarm optimization with the simplex algorithm.

To meet the need for tools that can assist project managers to plan and manage construction projects more efficiently, some research efforts in earthwork planning, scheduling, and visualization of the infrastructure construction projects have been conducted. Askew et al. [3] developed an approach to automate the earthwork planning process to create activity sets using a knowledge base and a simulation of earthwork processes. Tam et al. [18] proposed a method to automate the earthmoving planning by integrating a path-finding algorithm, a plant selection system, and a genetic algorithm. In Moselhi and Alshibani [15] genetic algorithms, LP, and geographic information systems were used to optimize the utilization of construction equipment involved in earthmoving operations. None of the cited contributions

can, however, provide a comprehensive solution to all the aspects of the construction process that is faced by Strabag and that we describe in the next sections.

5.4 Modeling Analysis and Notation

This section provides an overview of the network that we use for modeling the highway construction project, as well as the main notation that we adopt in the remaining of the chapter. To account for the fact that the MPS bounds each activity of the project in time by a time window, we consider an environment in which the overall horizon is discretized into a set of uniform periods (weeks in our study), $\mathcal{T} = \{1, 2, \ldots, T_{max}\}$. The network is represented by a graph $\mathcal{G} = (\mathcal{V}, \mathcal{A})$ defined by a collection of nodes $\mathcal{V}$ and arcs $\mathcal{A}$, and is schematized in Figure 5.2.

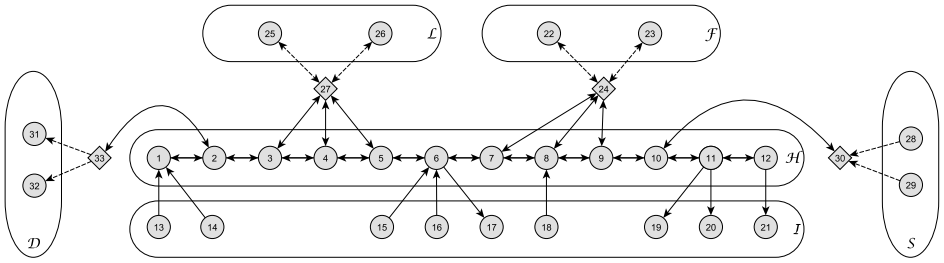


Figure 5.2: Network representation.

Among all the nodes in $\mathcal{V}$, we identify a set $\mathcal{H}$ of nodes representing *material locations*, that is, cut and fill areas where different types of materials can be dug or filled in (e.g., nodes $1, \ldots, 12$ in Figure 5.2. In other words, $\mathcal{H}$ represents the discretization of the highway being built. The material locations might need more than one digging/filling of one or more materials to be completed in different time windows. We thus need to introduce a new set of nodes (in addition to the material location ones) to deal with the digging and filling activities of the same material with overlapping time windows to be performed on the material location. Thus, we consider a set $\mathcal{I}$ of nodes representing *tasks*, that is, digging or filling activities required for a certain material in a given area. In particular, in $\mathcal{I}$ we identify $\mathcal{I}_f$ as the set of filling tasks (e.g., nodes 17, 19, 20, and 21) and $\mathcal{I}_d$ as the set of digging tasks (e.g., nodes 13, 14, 15, 16, and 18), with $\mathcal{I}_f \cup \mathcal{I}_d = \mathcal{I}$ and $\mathcal{I}_f \cap \mathcal{I}_d = \emptyset$. Each task $i \in \mathcal{I}$ is associated with a unique location $h(i) \in \mathcal{H}$ and a unique material $m(i) \in \mathcal{M}$, being $\mathcal{M}$ the set of all earth materials involved in the construction process. Moreover, each task i has its own quantity d_i, that must be fully dug/filled in a time window $[t_s(i), t_e(i)]$ specified by its starting period $t_s(i) \in \mathcal{T}$ and its ending period $t_e(i) \in \mathcal{T}$.

The network also includes a set $\mathcal{D}$ of dump sites, representing locations where materials can be disposed of (e.g., nodes 31 and 32), and a set $\mathcal{S}$ of quarries, representing sources where materials can be acquired to respond to filling requests (e.g., nodes 28 and 29). Note that in our model it is allowed to dispose of some material in a dump site and acquire the same material from a quarry if this leads to decreased costs. Dug materials can also be stored in temporary depots (e.g., nodes 25 and 26), whose set is denoted by $\mathcal{L}$. Facilities which process materials for recycling are represented by a set of nodes $\mathcal{F}$. We distinguish among separation plants ($\mathcal{F}_r$), mixing plants for asphalt ($\mathcal{F}_a$), and mixing plants for concrete ($\mathcal{F}_c$). The union of all the dump sites, quarries, temporary depots and recycling facilities is denoted by $\mathcal{N}$.

The highway being built is accessible from the outside through a set $\mathcal{E}$ of *access points*. Access points (pictured by diamond nodes in Figure 5.2, see, e.g., nodes 24, 27, 30, and 33) represent the passage between the existing public road network and the network that is still private. The public network is simply schematized by dashed lines in Figure 5.2, but it is, in general, quite complex. It is composed by a subnetwork (represented by dashed lines in in Figure5.1) including a set of nodes, denoted by $\mathcal{B}$, and a set of arcs connecting these nodes. This information is needed to represent the different capacitated paths that can be used to reach the set $\mathcal{N}$ of outside facilities.

Connections among the nodes are represented by the overall set $\mathcal{A}$ of arcs. Note that in general there is not an arc for each pair of nodes. Indeed, movement of any material into a quarry is not permitted, therefore only leaving arcs from $\mathcal{S}$ exist and they are headed to the access points or to the public network. Similarly, movement of materials out of a dump is not permitted, thus only arcs entering in $\mathcal{D}$ exist and they come from the access points or from the public network. Plants and temporary depots are also only connected to access points and/or to the public network, but in this case the arcs are in both directions. Access points are only connected to nodes in $\mathcal{N}$, $\mathcal{B}$, and $\mathcal{H}$. All the arcs incident the access points are in both directions but those entering in $\mathcal{D}$ or leaving $\mathcal{S}$. The nodes of the public network (set $\mathcal{B}$) can only be connected to the nodes of $\mathcal{E}$, $\mathcal{N}$, and to each other. Each material location $h \in \mathcal{H}$, is connected to other material locations by arcs in both directions. Material locations can also be connected to access points with arcs in both directions and each material location is connected to at least one filling and/or digging task. No other arcs are incident to the material locations. Each filling task $i \in \mathcal{I}_f$ is connected only to its material location $h(i) \in \mathcal{H}$ by an entering arc, while each digging task $i \in \mathcal{I}_d$ is connected only to its material location $h(i) \in \mathcal{H}$ by a leaving arc. To each arc $(i, j) \in \mathcal{A}$ we associate a maximum flow capacity U_{ij}^t for each period $t \in \mathcal{T}$.

To model the recycling activity, we distinguish, in the set of all ma-

terials $\mathcal{M}$, the subset $\mathcal{M}_s$ of materials that can be stored into temporary depots, constituted by rawI and rawII, the subset $\mathcal{M}_r$ of materials exiting the separation plants, constituted by recI, recII, and recWaste, the subset $\mathcal{M}_a$ of materials entering the asphalt mixing plants, constituted by recI and bitumen, and the subset $\mathcal{M}_c$ of materials entering concrete mixing plants, constituted by recII and cement. We denote by φ_r^m the quantity of recycled aggregate $m \in \mathcal{M}_r$ that can be obtained from a unit quantity of rawI material at plant $k \in \mathcal{F}_r$. Similarly, the quantity of $m \in \mathcal{M}_a$ necessary for producing a unit quantity of asphalt at plant $k \in \mathcal{F}_a$ is denoted by φ_a^m, and the quantity of $m \in \mathcal{M}_c$ necessary for producing a unit quantity of concrete at plant $k \in \mathcal{F}_c$ is denoted by φ_c^m.

To move the materials, different types of capacitated trucks can be used. We consider four types, that can load, respectively, bitumen, concrete, cement, and all the other materials. We denote by $\mathcal{W}$ the set of types of truck, by $\mathcal{M}(w)$ the set of materials that a truck of type $w \in \mathcal{W}$ can load, and by $V_{cap}(w)$ the overall transportation capacity of the fleet of trucks of type w.

5.5 Aggregate Formulation

In the following we describe the overall aggregate LP model that we developed to optimize the earthwork activities.

Variables.

The decision variables required to formulate the problem are denoted by x_{ij}^{mt} and represent the flow (expressed in cubic meters) of material $m \in \mathcal{M}$ in period $t \in \mathcal{T}$ on arc $(i,j) \in \mathcal{A}$, and f_k^{mt}, giving, respectively, the quantity of material m stored in a temporary depot $k \in \mathcal{L}$, carried to a dump site $k \in \mathcal{D}$, or bought from a quarry $k \in \mathcal{S}$, at period t. Non-negativity constraints are imposed on both variables, that is:

$$x_{ij}^{mt} \geq 0 \qquad (i,j) \in \mathcal{A}, m \in \mathcal{M}, t \in \mathcal{T}, \tag{5.1}$$

$$f_k^{mt} \geq 0 \qquad k \in \mathcal{L} \cup \mathcal{D} \cup \mathcal{S}, m \in \mathcal{M}, t \in \mathcal{T}. \tag{5.2}$$

Objective Function.

Our aim is to provide an optimal material distribution to complete the project within the specified time, minimizing the overall costs. To each arc $(i,j) \in \mathcal{A}$ is associated a transportation cost c_{ij}. Moreover let c_k^m denote the cost for storing, loading, or acquiring a unit of material $m \in \mathcal{M}$ into a temporary depot $k \in \mathcal{L}$, into a dump site $k \in \mathcal{D}$, or from a quarry $k \in \mathcal{S}$, respectively. The objective function is then:

$$\min \qquad z_{aggr} = \sum_{(i,j)\in\mathcal{A}} \sum_{m\in\mathcal{M}} \sum_{t\in\mathcal{T}} c_{ij} x_{ij}^{mt} + \sum_{k\in\mathcal{L}\cup\mathcal{D}\cup\mathcal{S}} \sum_{m\in\mathcal{M}} \sum_{t\in\mathcal{T}} c_k^m f_k^{mt}. \tag{5.3}$$

Arc capacity constraints.

Every arc $(i,j) \in \mathcal{A}$ has a maximum capacity U_{ij}^t (in cubic meters) that may vary according to the period $t \in \mathcal{T}$. This is enforced by:

$$\sum_{m \in \mathcal{M}} x_{ij}^{mt} \leq U_{ij}^t \qquad (i,j) \in \mathcal{A}, t \in \mathcal{T}. \tag{5.4}$$

To account for the fact that some nodes can be unaccessible during some periods (e.g., because they can only be reached after the erection of an access road, or because some construction activities temporary block part of the network) a capacity zero is associated to the arcs connecting those nodes in those periods.

Vehicle capacity constraints.

Materials are moved by trucks, so the quantity of materials moved along the network is constrained by the quantity of trucks available. The fleet is composed by a set $\mathcal{W}$ of types of trucks, each of those may transport a set $\mathcal{M}(w)$ of materials and is assigned an aggregate estimation $V_{cap}(w)$ of its transportation capacity, expressed in $km \cdot m^3$. In practice, $V_{cap}(w)$ gives the total amount of material that this type of truck can move in a period, and is estimated on the basis of past activities performed by Strabag. To each arc $(i,j) \in \mathcal{A}$ is assigned a non-negative length ℓ_{ij} in km. The overall transportation capacity is imposed by:

$$\sum_{(i,j) \in \mathcal{A}} \sum_{m \in \mathcal{M}(w)} \ell_{ij} x_{ij}^{mt} \leq V_{cap}(w) \qquad t \in \mathcal{T}, w \in \mathcal{W}. \tag{5.5}$$

Digging constraints.

Recall that each task $i \in \mathcal{I}_d$ is assigned to a digging location $h(i)$ and requires a certain amount d_i of material $m(i)$ to be dug in the time window $[t_s(i),t_e(i)]$. Therefore it must be:

$$x_{i,h(i)}^{mt} = 0 \qquad i \in \mathcal{I}_d, m \in \mathcal{M} : m \neq m(i), \tag{5.6}$$

$$x_{i,h(i)}^{m(i),t} = 0 \qquad i \in \mathcal{I}_d, t \in \mathcal{T} : t < t_s(i) \text{ or } t > t_e(i), \tag{5.7}$$

$$\sum_{t=t_s(i)}^{t_e(i)} x_{i,h(i)}^{m(i),t} = d_i \qquad i \in \mathcal{I}_d. \tag{5.8}$$

Filling constraints.

As for digging, each task $i \in \mathcal{I}_f$ is assigned to location $h(i)$and has its own amount d_i of material $m(i)$ to be filled in during the time window $[t_s(i),t_e(i)]$,

therefore:

$$x^{mt}_{h(i),i} = 0 \qquad i \in \mathcal{I}_f, m \in \mathcal{M} : m \neq m(i), \tag{5.9}$$

$$x^{m(i),t}_{h(i),i} = 0 \qquad i \in \mathcal{I}_f, t \in \mathcal{T} : t < t_s(i) \text{ or } t > t_e(i), \tag{5.10}$$

$$\sum_{t=t_s(i)}^{t_e(i)} x^{m(i),t}_{h(i),i} = d_i \qquad i \in \mathcal{I}_f. \tag{5.11}$$

Flow conservation at material locations, access points, and external nodes.

To ensure the conservation flow for each material m and period t we need to impose the following constraints for material locations, access points, and external nodes:

$$\sum_{(i,j)\in\mathcal{A}} x^{mt}_{ij} = \sum_{(j,i)\in\mathcal{A}} x^{mt}_{ji} \qquad j \in \mathcal{H} \cup \mathcal{E} \cup \mathcal{B}, m \in \mathcal{M}, t \in \mathcal{T}. \tag{5.12}$$

Temporary depots constraints.

Recall that $\mathcal{M}_s = \{\text{rawI}, \text{rawII}\}$ gives the set of materials that can be stored into temporary depots during the construction process. Each depot $k \in \mathcal{L}$ has a maximum storing capacity Q_k that must be respected at each period, thus:

$$\sum_{m\in\mathcal{M}_s} f^{mt}_k \leq Q_k \qquad k \in \mathcal{L}, t \in \mathcal{T}. \tag{5.13}$$

In a rolling horizon perspective, perhaps, we might consider just a set of periods of the entire construction process, thus having an initial level of storage imposed by the evolution of the construction activities and/or a final desired level of storage. Let therefore f^{m0}_k be the initial stock of material $m \in \mathcal{M}_s$ in depot k at the beginning of the first period (i.e., the initial available volume), and $f^{m,T_{max}+1}_k$ the upper bound on the closing stock of m in k at the end of the last period. We have the following material balance constraints:

$$\sum_{(i,k)\in\mathcal{A}} x^{mt}_{ik} + f^{m,t-1}_k = \sum_{(k,i)\in\mathcal{A}} x^{mt}_{ki} + f^{mt}_k \quad k \in \mathcal{L}, m \in \mathcal{M}_s, t \in \mathcal{T}. \tag{5.14}$$

The final storage volumes are constrained by:

$$f^{m,T_{max}}_k \leq f^{m,T_{max}+1}_k \qquad k \in \mathcal{L}, m \in \mathcal{M}_s. \tag{5.15}$$

To assume, for example, that depots must be empty before and after the completion of the overall project, it is enough to set f^{m0}_k and $f^{m,T_{max}+1}_k$ to zero.

Dump sites constraints.

The amount of waste deposited in a dump site $k \in \mathcal{D}$ is limited by the total disposal capacity C_k, therefore:

$$\sum_{(i,k)\in\mathcal{A}} \sum_{m\in\mathcal{M}} \sum_{t\in\mathcal{T}} x_{ik}^{mt} \leq C_k \qquad k \in \mathcal{D}. \tag{5.16}$$

Moreover, in each period there is a maximum allowable quantity of material m, say Q_k^m, that can be disposed of in the dump site, and hence:

$$\sum_{(i,k)\in\mathcal{A}} x_{ik}^{mt} \leq Q_k^m \qquad k \in \mathcal{D}, m \in \mathcal{M}, t \in \mathcal{T}. \tag{5.17}$$

Quarries constraints.

As for the dump sites, each quarry $k \in \mathcal{S}$ has an overall capacity C_k and a maximum capacity Q_k^m for each material m in each period, therefore:

$$\sum_{(k,i)\in\mathcal{A}} \sum_{m\in\mathcal{M}} \sum_{t\in\mathcal{T}} x_{ki}^{mt} \leq C_k \qquad k \in \mathcal{S}, \tag{5.18}$$

$$\sum_{(k,i)\in\mathcal{A}} x_{ki}^{mt} \leq Q_k^m \qquad k \in \mathcal{S}, m \in \mathcal{M}, t \in \mathcal{T}. \tag{5.19}$$

Separation plants constraints.

Recall only rawI material may be carried to the separation plants (see Figure 5.1), and the exiting set of materials is $\mathcal{M}_r = \{\text{recI}, \text{recII}, \text{recWaste}\}$. It follows that the flows entering nodes $k \in \mathcal{F}_r$ for all materials $m \in \mathcal{M}$ other than rawI, as well as the flows leaving the node for all materials other than $\mathcal{M}_r$ must be zero, that is:

$$x_{ik}^{mt} = 0 \qquad (i,k) \in \mathcal{A} \colon\ k \in \mathcal{F}_r, m \in \mathcal{M} \setminus \{rawI\}, t \in \mathcal{T}, \tag{5.20}$$

$$x_{ki}^{mt} = 0 \qquad (k,i) \in \mathcal{A} \colon\ k \in \mathcal{F}_r, m \in \mathcal{M} \setminus \mathcal{M}_r, t \in \mathcal{T}. \tag{5.21}$$

The capacity of any facility represents a physical limitation that must be considered. The accumulated material inflow should not exceed the maximum allowable operating capacity Q_k of each separation plant $k \in \mathcal{F}_r$. Thus, we get:

$$\sum_{(i,k)\in\mathcal{A}} x_{ik}^{rawI,t} \leq Q_k \qquad k \in \mathcal{F}_r, t \in \mathcal{T}. \tag{5.22}$$

Each separation plant has its own and specified recipe, that states the quantities of exiting materials obtained from a unit of rawI. Formally, recall φ_r^m gives the quantity of exiting material $m \in \mathcal{M}_r$ that can be obtained from

a unit of rawI at plant $k \in \mathcal{F}_r$, and note that, for feasibility, we must have $\sum_{m \in \mathcal{M}_r} \varphi_r^m = 1$. The conservation flow at each separation plant is stated by:

$$\sum_{(k,i) \in \mathcal{A}} x_{ki}^{mt} = \varphi_r^m \sum_{(i,k) \in \mathcal{A}} x_{ik}^{rawI,t} \qquad k \in \mathcal{F}_r, m \in \mathcal{M}_r, t \in \mathcal{T}. \tag{5.23}$$

Asphalt mixing plants constraints.

Each asphalt mixing plant $k \in \mathcal{F}_a$ receives in input only $\mathcal{M}_a = \{\text{recI}, \text{bitumen}\}$, and gives in output only asphalt, thus:

$$x_{ki}^{mt} = 0 \qquad (k,i) \in \mathcal{A} \colon\ k \in \mathcal{F}_a, m \in \mathcal{M} \setminus \{asphalt\}, t \in \mathcal{T}, \tag{5.24}$$

$$x_{ik}^{mt} = 0 \qquad (i,k) \in \mathcal{A} \colon\ k \in \mathcal{F}_a, m \in \mathcal{M} \setminus \mathcal{M}_a, t \in \mathcal{T}. \tag{5.25}$$

The plant has a total production capacity Q_k for each period. Its recipe is expressed by the φ_a^m input, which gives the quantity of material $m \in \mathcal{M}_a$ necessary for producing a unit quantity of asphalt at plant $k \in \mathcal{F}_a$. For feasibility, the values of φ_a^m satisfy $\sum_{m \in \mathcal{M}_a} \varphi_a^m = 1$. We thus get:

$$\sum_{(k,i) \in \mathcal{A}} x_{ki}^{asphalt,t} \leq Q_k \qquad k \in \mathcal{F}_a, t \in \mathcal{T}, \tag{5.26}$$

$$\sum_{(i,k) \in \mathcal{A}} x_{ik}^{mt} = \varphi_a^m \sum_{(k,i) \in \mathcal{A}} x_{ki}^{asphalt,t} \qquad k \in \mathcal{F}_a, m \in \mathcal{M}_a, t \in \mathcal{T}. \tag{5.27}$$

Concrete mixing plants constraints.

These constraints are equivalent to the ones for the asphalt mixing plants. Each concrete mixing plant $k \in \mathcal{F}_c$ receives in input materials $\mathcal{M}_c = \{\text{recII}, \text{cement}\}$, gives in output concrete, and has total processing capacity Q_k for each period. Its recipe is expressed by φ_c^m, that gives the quantity of material $m \in \mathcal{M}_c$ necessary for producing a unit quantity of concrete, and is such that $\sum_{m \in \mathcal{M}_c} \varphi_c^m = 1$. We thus obtain:

$$x_{ki}^{mt} = 0 \qquad (k,i) \in \mathcal{A} \colon\ k \in \mathcal{F}_c, m \in \mathcal{M} \setminus \{concrete\}, t \in \mathcal{T}, \tag{5.28}$$

$$x_{ik}^{mt} = 0 \qquad (i,k) \in \mathcal{A} \colon\ k \in \mathcal{F}_c, m \in \mathcal{M} \setminus \mathcal{M}_c, t \in \mathcal{T}, \tag{5.29}$$

$$\sum_{(k,i) \in \mathcal{A}} x_{ki}^{concrete,t} \leq Q_k \qquad k \in \mathcal{F}_c, t \in \mathcal{T}, \tag{5.30}$$

$$\sum_{(i,k) \in \mathcal{A}} x_{ik}^{mt} = \varphi_c^m \sum_{(k,i) \in \mathcal{A}} x_{ki}^{asphalt,t} \qquad k \in \mathcal{F}_c, m \in \mathcal{M}_c, t \in \mathcal{T}. \tag{5.31}$$

The resulting aggregate model is thus to minimize (6.3), subject to (6.1)–(6.2), (6.4)–(5.31).

5.6 Disaggregate Models

The aggregate model that we described in the previous section allows to study the behavior of the system in terms of volumes of materials, so as to investigate its feasibility. Since the model is computationally efficient, it can be solved several times under different network configurations, enabling to overcome possible causes of infeasibilities. Once a feasible aggregate solution has been found, the aim is then to determine the actual flows of each material in each period, i.e., the "disaggregate" flows. In the following, we propose two models, both based on LP, that allow to convert an aggregate solution to disaggregate flows: a multi-commodity model, and a path-based model.

Note that, since vehicles may transport only one material at a time and transport is performed in one period, the disaggregate flows can be computed separately, without loosing the optimality of the solution, one for each period and each material, independently from the flows of other materials and/or other periods. Therefore, let $\bar{t}$ and $\bar{m}$, be, respectively, the period and the material for which we intend to compute the disaggregate flows. Let also $\bar{x}_{hk}^{\bar{m}\bar{t}}$ and $\bar{f}_k^{\bar{m}\bar{t}}$ denote the solution of the aggregate model, representing, respectively, the quantity of material $\bar{m}$ hauled from h to k in period $\bar{t}$, and the stock of material $\bar{m}$ at depot $k \in \mathcal{L}$ at the end of period $\bar{t}$. To ease notation, hereafter we suppress the explicit dependence on $\bar{t}$ and $\bar{m}$, writing $\bar{x}_{hk}$ for $\bar{x}_{hk}^{\bar{m}\bar{t}}$ and $\bar{f}_k$ for $\bar{f}_k^{\bar{m}\bar{t}}$.

Given an aggregate solution, obtained by solving the aggregate model, we determine the subsets of vertices and arcs that are actually used by the material $\bar{m}$ in the period $\bar{t}$. In particular we denote by $\bar{\mathcal{O}}$ and $\bar{\mathcal{D}}$ the subsets of vertices having positive outgoing aggregate flow (origins) and having positive incoming aggregate flow (destinations), respectively. The subset of vertices crossed by some aggregate flow (intermediate vertices) is denoted $\bar{\mathcal{V}}$, while the subset of arcs that are used by some aggregate flow (traversed arcs) is $\bar{\mathcal{A}}$.

A multi-commodity model.

The first disaggregation procedure that we developed is based on the solution of a multi-commodity model. To formulate the problem, we introduce continuous variables y_{ij}^{hk}, which represent the quantity of material $\bar{m}$ taken from origin $i \in \bar{\mathcal{O}}$, going to destination $j \in \bar{\mathcal{D}}$, and passing through arc $(h,k) \in \bar{\mathcal{A}}$, in period $\bar{t}$. The disaggregate multi-commodity formulation is

then given by:

$$\min \quad \sum_{i\in\bar{\mathcal{O}}}\sum_{j\in\bar{\mathcal{D}}}\sum_{(h,k)\in\bar{\mathcal{A}}} y_{ij}^{hk} \tag{5.32}$$

$$\sum_{j\in\bar{\mathcal{D}}}\sum_{(i,k)\in\bar{\mathcal{A}}} y_{ij}^{ik} = \sum_{(i,k)\in\bar{\mathcal{A}}} \bar{x}_{ik} \qquad i \in \bar{\mathcal{O}} \tag{5.33}$$

$$\sum_{i\in\bar{\mathcal{O}}}\sum_{(h,j)\in\bar{\mathcal{A}}} y_{ij}^{hj} = \sum_{(h,j)\in\bar{\mathcal{A}}} \bar{x}_{hj} \qquad j \in \bar{\mathcal{D}} \tag{5.34}$$

$$\sum_{(h,v)\in\bar{\mathcal{A}}} y_{ij}^{hv} = \sum_{(v,k)\in\bar{\mathcal{A}}} y_{ij}^{vk} \qquad i \in \bar{\mathcal{O}}, j \in \bar{\mathcal{D}}, v \in \bar{\mathcal{V}} \tag{5.35}$$

$$\sum_{i\in\bar{\mathcal{O}}}\sum_{j\in\bar{\mathcal{D}}} y_{ij}^{hk} \geq \bar{x}_{hk} \qquad (h,k) \in \bar{\mathcal{A}} \tag{5.36}$$

$$y_{ij}^{hk} \geq 0 \qquad i \in \bar{\mathcal{O}}, j \in \bar{\mathcal{D}}, (h,k) \in \bar{\mathcal{A}}. \tag{5.37}$$

The objective function (5.32) to be minimized is the total sum of the disaggregate flows. Constraints (5.33) impose the sum of disaggregate flows leaving an origin i to be equal to the sum of the aggregate flows leaving that origin. Constraints (5.34) are equivalent to (5.33), but refer to destinations. Constraints (5.35) impose flow conservation at intermediate nodes. Constraints (5.36) force the sum of the disaggregate flows on each arc to be not lower than the aggregate flow on that arc. The "$\geq$" inequality in (5.36) is used to avoid possible errors due to different precision measures and ensures that a disaggregate solution can always be found starting from an aggregate one.

A path-based model.

The second disaggregation procedure we developed is a path-based model. This formulation takes into account the various paths between a given origin and a given destination that material $\bar{m}$ can use in period $\bar{t}$. To this aim, we make use of a standard label setting algorithm for determining all the paths from each origin $i \in \bar{\mathcal{O}}$ to each destination $j \in \bar{\mathcal{D}}$ going through the arcs $(h,k) \in \bar{\mathcal{A}}$. The label setting is an algorithm that builds paths from an origin to a destination on a network and it allows to find all the possible paths by setting a label on each node of the network that specifies its predecessors, that means which preceding node on which path. All possible paths are created, therefore the number of label on each node can be exponential. Thus, the algorithm produces the set $\bar{\mathcal{P}}$ of all possible paths (for material $\bar{m}$ in period $\bar{t}$), and the set of arcs $(h,k) \in \bar{\mathcal{A}}$ used by path $p \in \bar{\mathcal{P}}$, denoted $\bar{\mathcal{A}}(p)$. To define the path-based model, we introduce new continuous variables y_p representing the quantity of material $\bar{m}$ carried on path $p \in \bar{\mathcal{P}}$, in period $\bar{t}$.

The resulting formulation is then:

$$\min \quad \sum_{p \in \bar{\mathcal{P}}} y_p \tag{5.38}$$

$$\sum_{p \in \bar{\mathcal{P}}:(h,k) \in \bar{\mathcal{A}}(p)} y_p \geq \bar{x}_{hk} \qquad (h,k) \in \bar{\mathcal{A}} \tag{5.39}$$

$$y_p \geq 0 \qquad p \in \bar{\mathcal{P}}. \tag{5.40}$$

This model allows to find the set of paths that minimizes the costs and respects the requests derived by the aggregate solution. More precisely, the objective function (6.28) is to minimize the quantities carried on the paths. Constraints (6.29) guarantee that the quantities carried on each path are not lower than the original aggregate flows. Non-negativity of y_p is stated in (6.30). The computational behavior of the two formulations is discussed in details in the next section.

5.7 Computational Results

The aggregate and disaggregate mathematical models have been implemented using the MOSEL language available with Xpress. Strabag is using our models both for the planning and operative phases of the construction of a highway in Northern Italy and is extremely satisfied with the potentiality of both aggregate and disaggregate models.

Furthermore, to clearly evaluate the quality of our algorithms and models in a more general framework, we performed extensive numerical tests on a new set of realistic instances. These new instances were obtained by applying the real-world knowledge accumulated on a hypothetical highway construction project, still based in an area located in Northern Italy. Some simplifications were introduced in the project to limit its complexity. In the following we first give some information of of the instances that we created, and then present the details of the computational results obtained by running our models with different configurations. The instances are available from the authors upon request. Our tests have been run on a PC with 3.1 GHz Intel Core i3-2100 processor, by using one thread.

5.7.1 Realistic instances

The instances that we created are based on a hypothetical highway construction among the provinces of Parma, Reggio Emilia, and Modena, in Italy, and is depicted in Figure 5.3. The highway we projected is located in parallel to the Roman road Via Emilia and presents a length of approximately 43.5 km. Henceforth, we will refer to it as *New Via Emilia*.

The New Via Emilia plan is set in an irregular territory, with altitude gaps and some rivers and streams. We discretized it in about 400 segments,

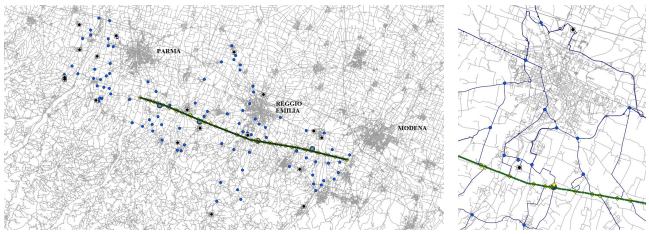


Figure 5.3: The New Via Emilia layout: on the left the overall construction site planned is depicted, while a highlight is given on the right.

each of about 100 meters length. For each segment a different amount and composition of materials to be dug and filled has been contemplated, considering the presence of bridges, viaducts, galleries, embankments, and trenches in the plan. An MPS has been drawn to engineer the various phases of the construction process. In particular, the New Via Emilia has been divided into 13 main parts, and six diverse construction phases have been established, one for digging and five for filling. The five filling phases are: 1) construction of concrete components of the skeletons of artificial and natural galleries, bridges, and viaducts; 2) construction of banks for embankments, tranches, and artificial galleries; 3) juxtaposition of the first layer of the roadway; 4) termination of the structure of bridges and viaducts; 5) asphalt paving. Time windows were assigned to each task following realistic assumptions on transportation capacities and working hours.

We then created the sets of quarries, dump sites, mixing and separation plants, temporary depots, access points, and external nodes. With this aim, we looked for real plants in the area interested by the New Via Emilia and detected 19 quarries and 19 dump sites. For simplicity, we considered each quarry and dump site capable of selling and accepting all the materials, respectively. To connect the external sites to the construction site, we situated several access points where the hypothetical building site meets the existing public network. We then connected the access points to the external plants by building paths in the public network. The resulting paths are highlighted, for a portion of the area involved by the project, in Figure 5.3. As discussed in the previous sections, several paths are typically available to reach a plant, because flow capacity on the least-cost path can be too small to guarantee the entire materials transportation. The mixing and separation plants have then been located close to those access points where wide and non-inhabited areas were available. The recipes of the separation and mixing plants were taken from real data. We also located 27 temporary depots along the building site. Capacities on arcs and nodes were again defined on the basis of realistic assumptions.

We thus obtained a first instance made of 2954 nodes (2270 of which represent tasks), 3694 arcs, and a building site characterized by 6 natu-

ral galleries, 4 bridges, one viaduct, and many kilometers in trenches. We considered a two year project, that we discretized in weeks. Material and transportation costs were estimated using real data. To have a more exhaustive test set of instances, we modified the basic New Via Emilia instance so far described by perturbing some of its characteristics. This generated a final collection of ten test instances, which is summarized in Table 5.2. The table gives the description of the modifications applied to the basic instance.

5.7.2 Aggregate model results

As previously mentioned, the aggregate model (6.3) – (5.31) has been solved using the LP solver Xpress 7.4. Different algorithms are available for solving LPs. To find the best computational performance, we considered the Dual Simplex, the Primal Simplex, and the Barrier algorithm.

Table 5.2 gives the results that we obtained on the 10 instances that we created. In particular, we report for each instance the computational time (in seconds) required by the given algorithms to reach the optimal solution.

The results of the comparison identify Barrier as the best algorithm when solving our set of instances. In average, it takes about 2 minutes with the Xpress configuration to solve an instance. The second best algorithm is the Primal Simplex, which in average needs about 17 minutes. The Dual Simplex is outperformed by the other algorithms, because its average solving time is barely three hours. The maximum values are quite small for the Barrier algorithm, but can be very high for the other two algorithms, about one hour and a half and more than 5 hours for the Primal and the Dual algorithm, respectively. The minimum values of times are not particularly high for the Primal and Barrier algorithms, while can be huge for the Dual algorithm. To note that Xpress uses the Dual Simplex algorithm by default.

Table 5.1: Characteristics of the 10 instances and computational times obtained by running the aggregate model on Xpress 7.4 by using Dual, Primal, and Barrier algorithms.

5.7.3 Disaggregate model results

The disaggregate models have been tested on 7753 disaggregate subinstances obtained by solving the 10 aggregate instances. The main characteristics of the disaggregate subinstances are reported in Table 5.3. In particular, the table shows, for each aggregate instance (Inst.), the number of disaggregate subinstances solved (#Subinst.), and the average numbers of origins, destinations, vertices, arcs, and paths touched by the flow (denoted, respectively, by $\bar{\mathcal{O}}$, $\bar{\mathcal{D}}$, $\bar{\mathcal{V}}$, $\bar{\mathcal{A}}$, and $\bar{\mathcal{P}}$). The last line gives average values over the entire set. The number 7753 comes from the fact that each aggregate instance

may generate up to $|\mathcal{M}| \times T_{max} = 1010$ subinstances, thus leading to 10100 subinstances in total, but not all materials have positive flows in all periods and so not all disaggregate models need to be run.

In Table 5.3 we also present the results obtained by running the two models developed to solve the disaggregate problem. Values T_m and T_p denote the times in seconds needed to solve the multi-commodity and the path-based models, respectively, and indicate the time spent to compute all the required sets ($\bar{\mathcal{O}}, \bar{\mathcal{D}}, \bar{\mathcal{V}}, \bar{\mathcal{A}}$, and, in addition for the path-based model, $\bar{\mathcal{P}}$) and to run the model. We then report $\Delta_T = T_m - T_p$. We can observe that, in average, the path-based model is about 20% faster than the multi-commodity one, even if an additional algorithmic component for detecting the paths has to be included.

In Figure 5.4 we report the time needed to solve each one of the 7753 models with respect to the number of the arcs considered. This shows how the performances in time of the multi-commodity decrease when increasing the number of arcs, while those of the path-based remain stable even when the number of arcs is relevant. To note that, for scaling problems, in the figure we do not report the results of two subinstances that needed 29.07 and 6.86 seconds to be solved to optimality using the multi-commodity model (and about 0.1 seconds with the path-based one).

Eventually we can conclude that the path-based model is faster than the multi-commodity model and that the path-based model is less affected by the number of arcs in the network to disaggregate, while the multi-commodity model is very sensible to the dimension of the network.

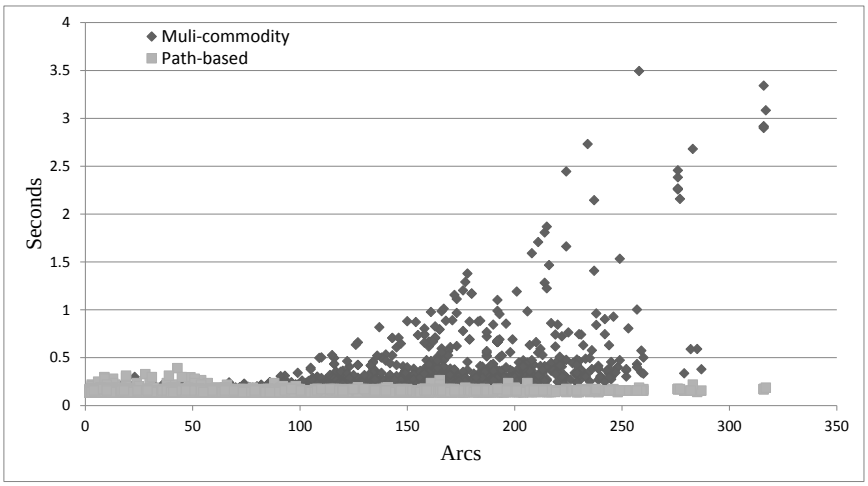


Figure 5.4: Solution time of the disaggregate models with respect to the number of arcs.

5.8 Conclusions

In this chapter we described our work of modeling and solving the earthwork optimization problem that arises when building a highway, focusing on the real-world activity performed by Strabag AG, one of the largest European companies operating in this sector. We modeled the earthwork problem by using novel a two-phase approach. In the first phase we solve an aggregate problem optimizing the overall flows of materials of the entire period of the project described by the Master Production Schedule. In the second phase we disaggregated the problem for each material and period and solved the obtained subproblems using two Linear Programming formulations. Our quantitative method includes some novelties with respect to the Body of Knowledge: we consider a set of multiple materials not only from excavations, we include a set of processes that can recycle dug material so to be used as high quality material for filling and constructing, we accounts for time windows for each activity derived from a Master Production Schedule, and we evaluate the restriction given by an network external to the construction site and the possible alternative paths. Nevertheless all these new features can be deactivated one by one so to make our methodology general and feasible to solve several types of earthwork optimization problems. To prove the quality of the proposed two-phase new algorithm we produced a set of realistic instances and solved the aggregate problem with Primal and Dual simplex and Barrier algorithms, showing that the last one outperformed the others. We also tested the disaggregate models concluding that the path-based model is more efficient than the multi-commodity one, especially when the number of arcs increases. Another reason why chosing the path-based model is that is more general than the multi-commodity because it allows the use of different cost functions The resulting approach is currently in use in Strabag, and is a powerful, flexible, and easy to replicate tool for solving real-world earthwork optimization problems. We included our optimization approach in a Decision Support System, presented in the following chapter, that combines tha presente two-phase apporach with graphical tools to visualize solutions and with a retroactive approach.

Inst.	Description	CPU Times Dual	Primal	Barrier
1	basic instance	11 321.0	1 029.2	105.8
2	reduced number of quarries and dump sites	19 106.4	465.0	128.7
3	restricted materials in quarries and dump sites	11 532.5	390.0	115.0
4	optimistic digging (45% rawI, 45% rawII, and 10% rawWaste)	8 541.4	507.8	130.9
5	pessimistic digging (25% rawI, 25% rawII, and 50% rawWaste)	12 798.6	413.2	125.2
6	balanced filling (50% rawI and 50% rawII)	12 084.5	662.0	125.5
7	enlarged time windows for each task	7 282.2	802.6	168.5
8	restricted flow capacities on the arcs of the external network	11 556.3	1 105.3	121.7
9	increased capacities for recycling plants and temporary depots	4 583.5	281.6	118.3
10	decreased capacities for recycling plants and temporary depots	10 452.8	4 626.3	131.5
Min		4 583.5	281.6	105.8
Avg		10 925.9	1 028.3	127.1
Max		19 106.4	4 626.3	168.5

Table 5.2: Characteristics of the 10 instances and computational times obtained by running the aggregate model on Xpress 7.4 by using Dual, Primal, and Barrier algorithms.

Inst.	#Subinst.	$\lvert\bar{\mathcal{O}}\rvert$	$\lvert\bar{\mathcal{D}}\rvert$	$\lvert\bar{\mathcal{V}}\rvert$	$\lvert\bar{\mathcal{A}}\rvert$	$\lvert\bar{\mathcal{P}}\rvert$	T_m	T_p	Δ_T
1	777	3.40	4.07	53.14	57.64	4.53	143.59	122.29	21.30
2	782	3.34	3.79	58.71	63.19	4.49	138.11	116.24	21.87
3	778	3.34	4.04	56.62	61.09	4.50	141.02	115.38	25.64
4	767	3.23	3.99	53.34	57.71	4.38	138.23	112.95	25.28
5	783	3.34	3.97	52.56	57.00	4.48	140.25	115.80	24.45
6	781	3.31	3.95	51.48	55.86	4.41	139.82	115.73	24.09
7	740	3.05	3.60	43.26	47.60	4.43	171.19	109.38	61.81
8	798	3.88	4.48	59.02	64.10	5.17	154.76	117.27	37.49
9	774	3.31	3.97	52.63	57.05	4.44	137.86	114.13	23.73
10	773	3.36	3.99	59.06	63.62	4.64	146.49	114.05	32.44
avg	775.3	3.36	3.99	54.04	58.55	4.55	145.13	115.32	29.81

Table 5.3: Evaluation of the multi-commodity and path-based disaggregate models.

Bibliography

[1] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1993.

[2] M. Ammar, A. I. Eldosouky, and F. A. Jarad. Optimization of earthwork allocation with multiple soil types. *Alexandria Engineering Journal*, 42:111–123, 2003.

[3] W. H. Askew, S. H. Al-Jibouri, M. J. Mawdesley, and D. E. Patterson. Planning linear construction projects: automated method for the generation of earthwork activities. *Automation in construction*, 11:643–653, 2002.

[4] S. M. Easa. Earthwork allocations with nonconstant unit costs. *Journal of Construction Engineering and Management*, 113:34–50, 1987.

[5] S. M. Easa. Earthwork allocations with linear unit costs. *Journal of Construction Engineering and Management*, 114:641–655, 1988.

[6] S. M. Easa. Selection of roadway grades that minimize earthwork cost using linear programming. *Transportation Research Part A: General*, 22:121–136, 1988.

[7] R. L. Burdett, and E. Kozan. An Integrated Approach for Earthwork Allocation, Sequencing and Routing. *European Journal of Operational Research*, 238:741–759,2014.

[8] W. L. Hare, V. R. Koch, and Y. Lucet. Models and algorithms to improve earthwork operations in road design using mixed integer linear programming. *European Journal of Operational Research*, 215:470–480, 2011.

[9] A. K. W. Jayawardane and F.C. Harris. Further development of integer programming in earthwork optimization. *Journal of Construction Engineering and Management*, 116:18–34, 1990.

[10] Y. Ji, A. Borrmann, E. Rank, J. Wimmer and W. Guenthner. An Integrated 3D Simulation Framework for Earthwork Processes. *Proceedings of the 26th CIB-W78 Conference on Managing IT in Construction*, 2009.

[11] S. M. Karimi, S. J. Mousavi, A. Kaveh, and A. Afshar. Fuzzy optimization model for earthwork allocations with imprecise parameters. *Journal of construction engineering and management*, 133:181–190, 2007.

[12] M. Marzouk and O. Moselhi. Multiobjective optimization of earthmoving operations. *Journal of construction Engineering and Management*, 130:105–113, 2004.

[13] R. H. Mayer and R. M. Stark. Earthmoving logistics. *Journal of the Construction Division*, 107:297–312, 1981.

[14] A. A. Moreb. Linear programming model for finding optimal roadway grades that minimize earthwork cost. *European Journal of Operational Research*, 93:148–154, 1996.

[15] O. Moselhi and A. Alshibani. Optimization of earthmoving operations in heavy civil engineering projects. *Journal of Construction Engineering and Management*, 135:948–954, 2009.

[16] R. M. Stark and R. H. Mayer. *Quantitative construction management: Uses of linear optimization.* Wiley New York, 1983.

[17] R. M. Stark and R. L. Nicholls. *Mathematical foundations for design: Civil engineering systems.* McGraw-Hill New York, 1972.

[18] C. M. Tam, T. K. L. Tong, and B. W. L. Wong. An integrated system for earthmoving planning. *Construction Management and Economics*, 25:1127–1137, 2007.

[19] W. Xianjia, H. Yuan, and Z. Wuyue. The bilevel programming model of earthwork allocation system. In *Cutting-Edge Research Topics on Multiple Criteria Decision Making*, 35:275–281. Springer Berlin Heidelberg, 2009.

[20] H. Zhang. Multi-objective simulation-optimization for earthmoving operations. *Automation in Construction*, 18:79–86, 2008.

Chapter 6

A Decision Support System for Highway Construction and its Application to the Autostrada Pedemontana Lombarda

This chapter presents a new decision support system (DSS) to handle the activities involved in construction logistics projects. The system has been designed through a collaboration with the construction company Strabag AG. The new DSS aids managers to schedule the construction activities, determining the amounts of materials that must be moved in a time period, and the corresponding paths to obtain minimum costs. Its application to the highway construction project *Autostrada Pedemontana Lombarda* has produced significant results in terms of quickness in decision making and cost savings. Our system is based on the use of linear programming and operates in two phases: firstly the feasibility of the project is determined by an aggregate mathematical model, then a disaggregate model is executed to determine the actual flows for each material and period.

The efficiency of the proposed system is shown by means of computational results run on a set of instances drawn from the Autostrada Pedemontana Lombarda project. Graphical tools are used to obtain a clear visualization of the solutions of the models and to facilitate the decision process.

The DSS, currently in use in the company, is a powerful, flexible, and easy to replicate tool for solving real-world construction logistics problems.

Keywords: Decision support, Linear programming, Earthwork, Construction logistics.

6.1 Introduction

The construction of a highway includes various activities that have to be done in different periods and take years to be completed. *Earthwork* operations account for about 25% (see Hare et al. [5]) of the construction costs and consist mainly of activities such as digging, filling, hauling, and dumping, that depend largely on heavy equipment.

The solution of complex projects that evolve over long time horizons can obtain substantial benefits from the use of modeling techniques. The collaboration the supply chain optimization team of the construction company led to the creation of an earthwork optimization system based on the use of *linear programming* (LP) models. The use of the models produces a minimum cost earthwork plan and makes it possible to enhance the productivity of the overall construction project. The models are embedded into a *decision support system* (DSS) that aids managers to schedule the highway construction activities over a planning horizon. In particular, it helps in deciding when and how moving or storing the different materials to minimize costs. The main activities involved in the project are shown in Figure 6.1, where the arrows represent flows of materials of different kinds.

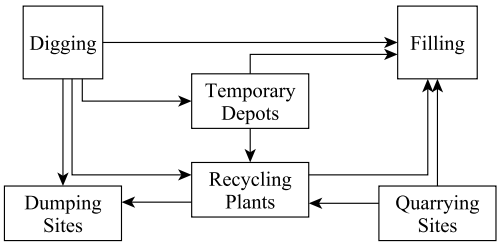


Figure 6.1: Flow chart representing the main activities involved in the project.

The construction site is discretized into segments of different length called *material locations*, where several activities must take place. *Digging* activities, that is, the excavation of earth, must be done in different material locations. A fundamental step in digging activities is represented by on-site separation of the material according to its characteristics, obtaining the so-called *rawH* (high quality), *rawM* (medium quality), *rawWaste* (waste), and *topsoil* (the outermost layer of soil). Materials such as rawH and rawM can be used directly for filling. Moreover, rawH can be recycled and the wastes from digging and recycling must be discarded on *dump sites*, while topsoil is temporary stored and reused directly to recover the soil at the end of the construction. *Filling* activities consist in bringing towards highway construction locations a certain amount of material, for example to fill earth cuts and trenches, or build bridges and artificial galleries. In the recycling

process, realized in *recycling plants*, rawH is recycled to obtain refined materials which are then mixed with *bitumen* and *cement* to obtain *concrete* and *asphalt*, materials suitable for filling. If necessary or profitable, materials can also be bought from private *quarries*. Construction materials can also be stored in *temporary depots* and used at a later stage. Quarries, dump sites, temporary depots, and recycling plants have capacity restrictions. To move the materials, different types of capacitated vehicles are used.

The DSS consists of three phases: the input, the optimization, and the output phase. The *input phase* represents the integration with the *Master Production Schedule* (MPS) where all the activities and their time windows are defined by the production department. In this phase data are collected and preprocessed. The *optimization phase* aims at checking the MPS feasibility and optimize the execution of the activities. The *output phase* receives the output of the optimization and uses it to automatically create graphs, tables, and other interactive tool that are useful for the decision maker to implement feedback actions.

In the optimization phase the highway building site is modeled as a graph composed by one part representing the building site and another part representing a network of roads external to the building site, needed to reach the dump sites and the quarries. The vertices of the network correspond to material locations, plants, dump sites, temporary depots, quarries, and junctions of the road networks. The vertices are connected by arcs that represent the private roads inside the building site and the public roads of the external network. To solve the problem we adopted a two-level approach, by using first an aggregate and then a disaggregate formulation, both based on the use of linear programming. The aggregate formulation aims at checking the feasibility of the project in the given time horizon, with the given limited resources, while minimizing an estimation of the costs. It is possible to run the aggregate formulation several times and under different network configurations to overcome possible infeasibility problems. The disaggregate formulation is executed, once a feasible solution to the aggregate formulation is on hand, to find precise minimum cost flows for each type of material and period.

Our formulations derive from those developed in Bogenberger et al. [2], but include some additional features that derive from our real-world case study and are discussed in the following. All the other components of the DSS, as well as the real-world case study, are presented in this chapter for the first time

The modeling activity follows the footsteps of a well established line of research on the so-called *earthwork* problems, that basically consist in finding the minimum cost flows of earth materials while satisfying operational constraints. The very basic earthwork optimization problem is known in the literature as the *cut and fill problem* (see, e.g., Ahuja et al. [1]) and can be solved by the use of a minimum cost flow algorithm. This simple model has

then been expanded to include constraints and complications arising in real-world construction problems and LP became in the time a standard tool for solving these models (see, e.g. Stark and Nicholls [9] and Mayer and Stark [8]). Some variations are represented by the use of non-linear costs (see, e.g., Easa [3, 4]), uncertainty in costs and facility capacities (see, e.g., Karimi et al. [6]), multi-objective frameworks and simulation (see, e.g., Marzouk and Moselhi [7] and Zhang [10]), just to cite some of the most relevant ones.

The reminder of the chapter is organized as follows. We introduce the *Autostrada Pedemontana Lombarda* project and we present the company in two dedicated sections. *The Decision Support System* section shows the DSS in detail, specifying the tools and methods involved in each step of its execution. In *Computational Evaluation of the Optimization Phase* section a computational evaluation is presented. Conclusions are drawn in the last section.

6.2 The Autostrada Pedemontana Lombarda Project

Autostrada Pedemontana Lombarda (www.pedemontana.com) is one of the largest highway construction projects currently under development in Europe. It covers the urban area of Milan, in Northern Italy (see Figure 6.2) and includes a large system of highways, access ramps, and local streets. It will be composed by 157 *km* of roads, 19% of which runs through natural tunnels and 16% through artificial tunnels. There will be 87 *km* of highway (67 *km* of highway, 20 *km* of bypass) and 70 *km* of road links to ease local vehicular load.

Its construction is part of the activities planned for the forthcoming *World Exposition Milano 2015* (Expo Milano 2015). It is an economically and socially strategic project, because the highway runs through a territory with more than 220,000 companies (40% of the regional total) and four

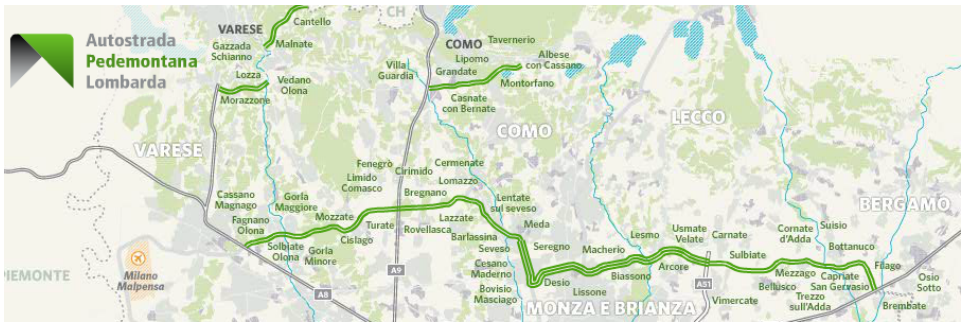


Figure 6.2: The *Autostrada Pedemontana Lombarda* project in Northern Italy.

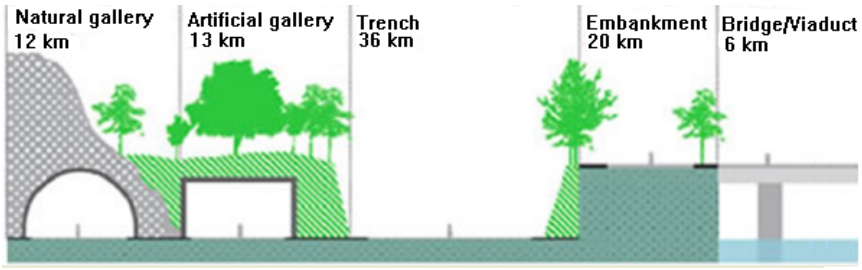


Figure 6.3: The characteristics of the main highway.

million inhabitants. The aim of this construction project is to redistribute the traffic of the existing road system, cover the lack of infrastructural offer in the northern area of Milan, link the three main airports of Lombardy and ease the traffic among the five provinces of Bergamo, Monza e Brianza, Milano, Como, and Varese.

To minimize the impact over the residential areas involved, nearly three-quarters of the 87 km of the highway will run in trench and artificial or natural galleries. This implies the digging of a huge amount of earth, roughly 30 millions m^3. Figure 6.3 summarizes in a simple way the characteristics of the highway. The building site extends for 12 km in natural galleries, 13 km in artificial galleries, 36 km in trenches, 20 km in embankments, and 6 km in bridges and viaducts.

The construction has been entrusted to a consortium, formed by the Italian constructors Grandi Lavori Fincosit and Impresa Costruzioni Giuseppe Maltauro, and by the Austrian company Strabag AG, which holds 60% stake of the project.

6.3 Strabag AG: a Leader in the construction industry

Strabag AG Group, with headquarter in Vienna (Austria), has its origins in ILBAU, founded in 1835 in Austria, and in the namesake STRABAG, founded in 1895 in Germany. Over the years, the company has grown into one of Europe largest construction companies, now operates in many European markets, and also on individual cases in other continents. The company offers a wide range of services that span all areas of the Construction Industry: building construction and civil engineering, infrastructure construction and tunneling, and facility management.

Zentrale Technik is an Engineer-center of excellence that develops the technical know-how of the Strabag AG Group and spreads it within the company. The *Supply Chain Optimization* (STRAsco) team of Strabag AG, composed of experts in operations research, network optimization, simula-

tion, and logistics, treats topics along the whole supply chain and delivers a well founded basis to decision makers in the field of construction logistics. STRAsco operates in the *Zentrale Technik* of Vienna since January 2013. By making use of instruments such as mathematical optimization methods and simulation, the STRAsco group manages design, planning, and ongoing optimization of logistics in major projects.

6.4 The Decision Support System

Figure 6.4 schematically depicts the operating diagram of the DSS that we developed, highlighting the tools and the methods used in each step for managing the logistics of the highway construction project. The main steps are the collection and management of input data, the optimization phase for finding the best earthwork configuration, and, finally, the processing of the output, the graphical representation of the solutions found, and the use of these results as feedbacks in an iterative framework. In the next three subsections we describe the details of these steps.

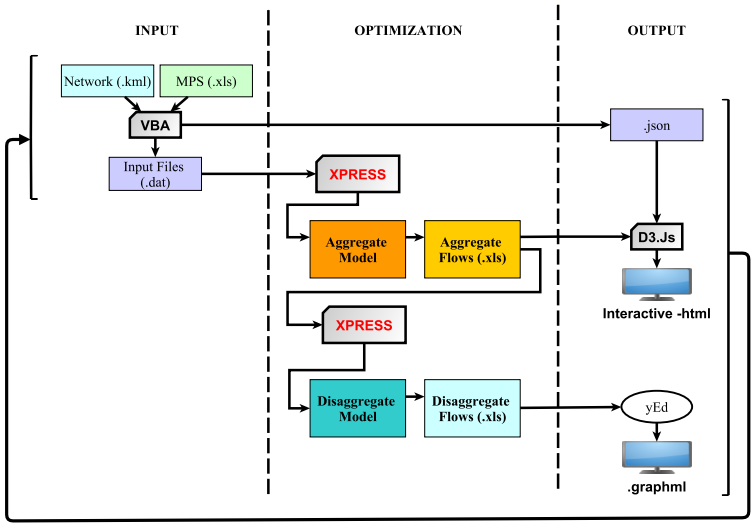


Figure 6.4: Operation Diagram of the developed Decision Support System.

Input Phase

The input is handled through a code in *Visual Basic for Applications* (VBA). In particular two input files are considered. The first one is the MPS, in the form of an excel file (.xls), that is an output of the design phase of civil engineers that contains information about the tasks to be done in different material locations and their time windows. This file also contains useful

information for the planning of activities such as the capacity of quarries, temporary storages, plants, and dump sites, the type and capacity of the vehicles involved, and the quality of materials.

The second input is a *Keyhole Markup Language* file (.kml), which collects information from Google Earth Pro. This information concerns the geographic coordinates of the characteristic vertices of the project such as quarries, dump sites, plants, material locations, access points, and temporary depots. Using Google Earth Pro, one can create and edit paths and add other useful information. For example, it is possible to specify whether a particular path is subject to capacity restrictions and to include the characteristics of the roads (public roads, road to be constructed, etc.). The information deriving from the .kml file and from the .xls files is collected by a specifically designed routine in VBA, and then translated into text files to move to the optimization phase.

In the optimization phase, we use the FICO®Xpress Optimization Suite to solve the mathematical models. All information is passed to Xpress using a few data files (.dat) that contain all the values of the parameters used to describe the model. In this way the optimization models receive information about vertices, arcs, periods, materials, vehicles, digging and filling demands, capacities of plants, quarries, and dump sites, but also information about the coordinates of the vertices thanks to the details taken from the .kml file.

Optimization Phase

The optimization phase is based on LP models and is divided in two subphases represented by an *aggregate formulation* and a *disaggregate formulation*, respectively. Both formulations are defined and extensively explained in the Annex 6.A. The aggregate formulation uses two sets of variables: the first one models the flows of materials in each period on the arcs of the network representing the building site. The second set of variables takes into account the amount of material stored in depots, acquired from quarries, and dumped away in every period. The objective function minimizes the cost of storing, dumping, acquiring, and hauling materials. A set of digging and filling activities of several materials and their time windows are requirements of the MPS: the aggregate formulation must respond to these requirements minimizing the overall costs and respecting the imposed constraints. The constraints are many: the digging and filling must be accomplished respecting the time windows. When digging in a material location in different periods, the composition of materials is always the same, thus if some rawH is needed also a percentage of the others materials has to enter the network. On the other hand, a set of alternative materials can be used when filling. A number of facilities can be used during the construction of the highway: recycling plants to transform materials, quarries to buy materials, dump

sites to discard some materials, and temporary depots to store materials. Some of these facilities are located outside of the building sites and a highly constrained external network must be considered. All facilities so as all arcs of the network and all vehicles have capacity constraints to be respected. The solution of the aggregate formulation, expressed in terms of flows, is stored in text files to be passed to the disaggregate formulation.

The disaggregate formulation is used to determine precise flows, one for each material and each period, taking the solution of the aggregate formulation as input. It considers all origins and destinations for a given material in a given period, and detects all paths connecting the origins with the destinations that respect the capacities derived from the aggregate solution. Each path is represented by a variable, whose cost is normally non-linear and depends on the length of the path itself. The disaggregate formulation determines the set of paths that allow to move the materials at minimum cost. Both formulations are an evolution of the modeling activity performed in Bogenberger et al. [2], and include some new features derived by the real-world application. The main changes affected digging and filling constraints in the aggregate formulation and inclusion of costs in the disaggregate formulation. To solve the models we used the FICO®Xpress Optimization Suite software, and in particular we used the Xpress-Mosel language. The solutions are stored as plain text files that are then processed by the output phase.

Output Phase

To make it easier to understand the results, characterized by values of millions of variables, we make use of practical graphical tools. These tools can be used to adapt diagrams and tables on customer request. More specifically, we used the D3 JavaScript library to visualize the diagrams and tables representing the solution of the aggregate flows. To do this we necessitate, besides the file representing the solution obtained by Xpress, additional input files realized in the *JavaScript Object Notation* (.json) format, which contain the location classes, material classes, vertices definitions, relations, and routes. The D3.js library is useful to manipulate data files and create an interactive representation of data. Due to the security model of standard web-browser implementations, it was necessary to implement a web server in which one can see the graphs and tables filled with the information of the aggregate model.

In Figures 6.5, 6.6, 6.7, and 6.8 some of the graphical representations that can be obtained are shown. In particular, Figure 6.5 depicts the network of the *basic instance* (a segment of Autostrada Pedemontana Lombarda that is used in the next section for the computational evaluation of our models), formed by the construction site, several access points that link the construction site to external public roads, 13 dump sites, seven temporary

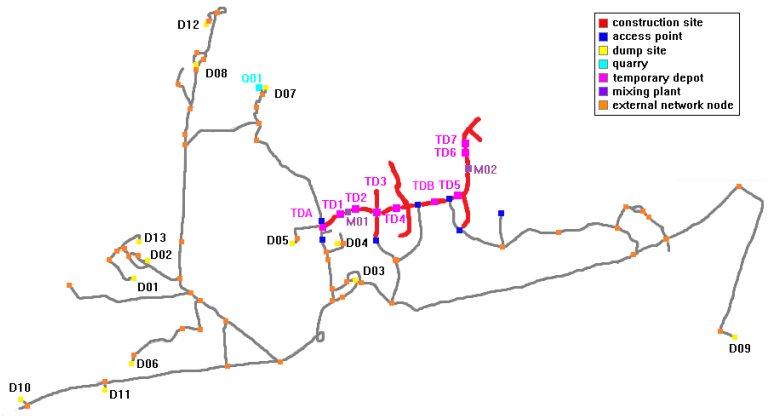


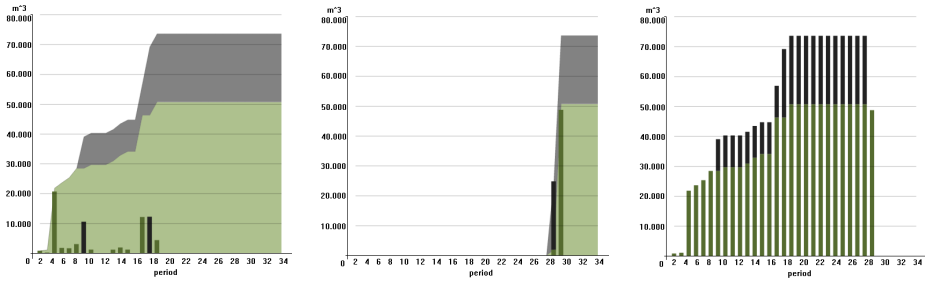
Figure 6.5: Network visualization of the basic instance , the highway section used for computational tests.

depots, a quarry, and two concrete mixing plants. Such a representation helps decision-maker to have a clear idea about the location of the points of interest and possible connections and obtain information on each point by clicking the interactive figure.

Figure 6.6 shows the inflow, storage, and outflow of material through the temporary depot TD_2 during the time horizon. In Figures 6.6, but also in Figures 6.7 and 6.8, different colors represent different materials. In Figures 6.6-(a) and 6.6-(b) the inflows and outflows are depicted. In particular, the small bars represent the amount of entering and exiting materials, respectively, for each period, while in the background the overall inflows (for Figure 6.6-(a)) and outflows (for Figure 6.6-(b)) are reported. In Figure 6.6-(c) the amount of materials stored in TD_2 is shown. One can notice that at the beginning of the construction process the inflows are more relevant because of the largest number of planned digging activities. Then the capacity of the depot is reached and no additional inflows are registered. The last periods of the time horizon are characterized by a larger number of filling activities, thus the temporary depot outflows increase.

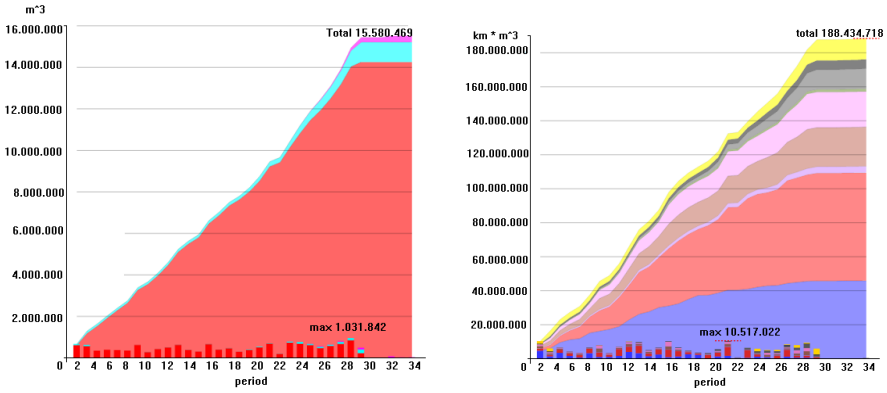
In Figure 6.7-(a) one can observe the total handled material, expressed in m^3, which refers to the total amount of material loaded on vehicles during the earthwork activities. The small bars represent the handled material for each period. In Figure 6.7-(b) the hauling capacity, expressed in $km \cdot m^3$, is depicted. Here the small bars show the hauling capacity used for every period.

Figure 6.8 shows the amount of material treated in digging and filling activities in material location, period by period, material per material. As one can see, the last periods are not interested by digging activities, which is quite normal because these periods are interested by filling activities such as paving.



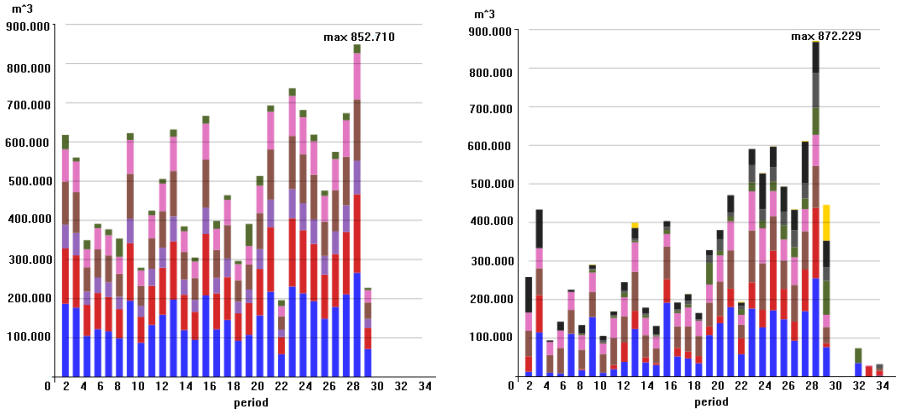
(a) Inflow material (b) Outflow material (c) Storage

Figure 6.6: Flows in temporary depot TD_2 during the construction time horizon.



(a) Total handled material (m^3). (b) Total hauling capacity ($km \cdot m^3$).

Figure 6.7: Graphical representations of the use of material during the construction time horizon.



(a) Excavated material (m^3) (b) Filled material (m^3)

Figure 6.8: Graphical representation of digging and filling activities in a vertex.

The solution of the disaggregate model gives us detailed information on the flow of a material in a certain period. It is a *point to point* solution, since by solving the model one understands exactly from which source to which destination the material travels over a certain period. For the graphical representation of the disaggregated flows we use a diagramming program, specifically the free *yEd Graph Editor* by yWorks. This editor directly reads the solution provided by the model in the form of Excel file. The result is a large diagram containing information about the flows of material that start from all sources and arrive on several destinations. Figure 6.9 is a particular of an example of the graphical output (.graphlm) of the disaggregate model in which we consider the flows coming from a certain digging activity, $DIG123$, and directed respectively to the temporary depots, $TD6$ and $TD7$, to the dump site $DS10$, and to the filling activity $FIL145$. This type of representation highlights the distance in km from sources to destinations and the amounts in m^3 of materials involved, making a distinction between the different types of materials. In particular, in $DIG123$ we represent with line m_{exc} the total amount of excavated materials such as topsoil (italic) and the others materials (plane), while in line mc_w we depict the amount of the excavated materials that are rawWaste. In $TD6$ and $TD7$ we depict the total amount of materials stored, separated in topsoil (italic) and other materials (plane). The filling activity $FIL145$ shows the total amount required in this period in line mc_r and the amount derived by the block of this subgraph in line mc_{ri}, both separated in topsoil (italic) and other materials (plane). The block representing the dump site $DS10$ shows the amount to be carried to the dump site in this period in line mc_r and the amount of materials coming from the blocks of this graph in line mc_{ri}. On the arcs are

depicted the amount of topsoil (italic) and of other materials (plane), and the distance between the two sites.

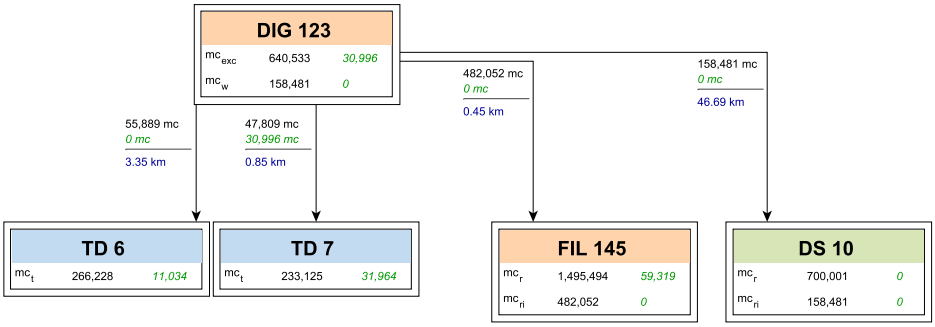


Figure 6.9: Graphical output of the disaggregate model.

6.5 Computational Evaluation of the Optimization Phase

The models, whose detailed mathematical formulation is given in Annex 6.A , were implemented using the Xpress-Mosel language available and solved with the LP solver FICO®Xpress 7.6. The PC on which the tests were performed has the following characteristics: 2.21 GHz AMD Athlon(tm) 64 Processor 3500+.

Real-world Instance

The instance used to conduct tests to validate the models corresponds to the real-world highway segment of the project Autostrada Pedemontana Lombarda between Cesano Maderno and Usmate Velate, that is what we call basic instance (see Figure 6.2). The data refer to an intermediate stage of the project and therefore they do not exactly reflect the final configuration of the considered segment. The basic instance is one of the most problematic sections of the whole construction project from an environmental and urbanistic viewpoint, due to the need of construction in proximity of urban settlements and areas of natural beauty. With a length of approximately 16.5 km and three lanes in each direction, the basic instance extends for about 6.5 km in artificial tunnels and for about 9.5 km in trenches, as well as in embankments and in viaducts. The track intersects many waterways, the most important of which is the Lambro river, crossed by a bridge. To connect the highway with the local road network, some secondary tracks will be realized in addition to the main one.

The section is discretized in segments of about 500 meters length. For each segment the amount and composition of materials to be dug has been estimated by performing a core sampling. Similarly, the amount and composition of materials to be used for filling has been determined by considering the presence of bridges, viaducts, galleries, embankments, and trenches in the plan. On the basis of the MPS we determined the time windows of the various activities of the construction process. The overall planning horizon of the basic instance is about three years, that we discretized in months.

The area interested by the basic instance includes several facilities, such as 13 dump sites, a quarry selling asphalt and cement, seven temporary depots, and two concrete mixing plants. Several access points connect the external sites, i.e., the existing public network, to the construction sites. The access points are connected to the external plants by using the available paths in the public network. The resulting network is described in Figure 6.5, that depicts the construction site (highlighted), and the paths through access points and external plants (plain).

Computational Results

The instance described in the previous subsection has been modified by perturbing some of its characteristics with the purpose of obtaining an exhaustive set of instances. We first created two instances by changing the discretization of the segments, using segments of 250 meters and of 750 meters. Xpress offers different algorithms for solving LPs, among these the Dual Simplex, the Primal Simplex, and the Barrier algorithm. To identify the best method to solve our problem we tested the three algorithms on the basic instance and on the other two perturbed instances.

Table 6.1 gives the results of the three algorithms on the instances that we created. For each instance we report the objective function value z and the computational time required to reach the optimal solution. We also provide the average of these computational times.

It should be noticed that the discretization in segments of 250 meters costs slightly more than the real-world instance. Instead, the 750 discretization cost slightly less than the basic instance. We can conclude that the used discretization of 500 meters represents an appropriate level of detail.

As regards the computational times, the results of the comparison identify the Barrier as the best algorithm when solving our set of instances. In average, it takes about two minutes to solve an instance. The other two algorithms require in average a computational time about eight times higher. The maximum values are quite small for the Barrier algorithm, i.e., about 3.5 minutes, but can be very high for the other two algorithms, reaching about 50 minutes.

The problem we assessed is very complex, thus we performed a sensitivity analysis, conducted by changing the percentages of excavated materials,

	Solution	CPU seconds		
Instance	z	Dual	Primal	Barrier
basic instance (0.500 km)	19206.1	386.8	901.3	95.8
0.250 km discretization	19298.9	923.2	2890.4	213.5
0.750 km discretization	19191.6	256.2	613.5	68.6
Avg		522.1	1468.4	126.3

Table 6.1: Results obtained by solving the aggregate model with Dual, Primal, and Barrier algorithm

to better understand the behavior of the aggregate formulation. We recall that rawH is the highest quality material obtained by digging. In the basic instance rawH amounts to about 10% of the total excavated materials. This value derives from core sampling performed at material locations (roughly, a sampling every 500 m), so it is only an estimation of the real value. Indeed this percentage is a very critical parameter, this is why we present the following analysis. We obtained three additional instances by increasing the percentage of rawH by 5% and 10%, and decreasing it by 5%, respectively.

Instance	z	% gain	CPU seconds
basic instance (10% rawH)	19206.1		95.8
rawH +5%	16104.6	16.1	109.2
rawH +10%	13485.1	29.8	120.3
rawH -5%	infeasible	-	55.1

Table 6.2: Sensitivity analysis on the quality of materials.

As one can see in Table 6.2, in more optimistic situations the solution cost decreases consistently with respect to that of the real-world instance. Instead, the more pessimistic situation gives an *infeasible* instance, because such a situation creates too much waste and the capacity of the temporary depots is not large enough to accommodate it.

6.6 Conclusions

In this chapter we described a new Decision Support System (DDS) developed to furnish Strabag AG, one of the largest European construction companies, with a tool that could assist the decision-makers in the process of construction of the Autostrada Pedemontana Lombarda. Autostrada Pedemontana Lombarda is one of the largest highway project in progress nowadays in Europe and it is part of the activities planned for the World Exposition Expo Milano 2015. Its construction requires to move roughly $5{\cdot}10^7$ m^3 of earth, for about $5{\cdot}10^8$ $km \cdot m^3$ of flows.

The newly developed DDS is based on three phases: the input, the optimization, and the output phase. The input phase collects and preprocesses the information to be passed to the optimization phase. The optimization is constituted of two sub-phases, a first one, where an aggregate model solves the earthwork optimization problem, and a second one that solves a disaggregate model which dispenses the actual flows for each material and each period. The output phase uses the data procured by the optimization phase and gives an useful, practical, and clever set of interactive outputs to improve and simplify the decision making process.

To test the efficiency of the proposed DSS in solving complex earthwork optimization problems, we produced a set of instances by perturbing some characteristics of the real-world instance corresponding to one of the nine sections of the Autostrada Pedemontana Lombarda project.

The DSS, currently in use at Strabag AG for the construction of Autostrada Pedemontana Lombarda, is an efficient, responsive, and adaptable tool suitable for solving different real-world construction logistics problems with little changes. Its application to other construction projects is under study by Stabag AG.

6.A Mathematical Models

In this Appendix we report the mathematical models used in the optimization phase.

Aggregate Formulation

Let us first define the necessary sets of vertices: $\mathcal{L}$ is the set of temporary depots, $\mathcal{D}$ of dump sites, $\mathcal{S}$ of quarries, $\mathcal{H}$ of the material locations (where digging and filling activities must be performed), $\mathcal{B}$ of vertices representing the network outside the building site, including the access points to the building site. Let $\mathcal{A}$, $\mathcal{M}$, and $\mathcal{T}$ be the sets of arcs, materials, and periods, respectively.

Variables. The variables x_{ij}^{mt} represent the flow of material $m \in \mathcal{M}$ in period $t \in \mathcal{T}$ on arc $(i,j) \in \mathcal{A}$, while f_k^{mt} represent, respectively, the quantity of material m stored in a temporary depot $k \in \mathcal{L}$, carried to a dump site $k \in \mathcal{D}$, or acquired from a quarry $k \in \mathcal{S}$, at period t. Variables must be non-negative, thus:

$$x_{ij}^{mt} \geq 0 \qquad (i,j) \in \mathcal{A}, m \in \mathcal{M}, t \in \mathcal{T}, \tag{6.1}$$

$$f_k^{mt} \geq 0 \qquad k \in \mathcal{L} \cup \mathcal{D} \cup \mathcal{S}, m \in \mathcal{M}, t \in \mathcal{T}. \tag{6.2}$$

Objective Function. With each arc $(i,j) \in \mathcal{A}$ is associated an estimation of transportation cost c_{ij} proportional to the distance. The exact cost of transportation can be computed only in the disaggregate formulation, since the transportation fare depend non-linearly from the length of the traveled paths. Moreover let c_k^m denote the cost for storing, dumping, or buying a unit of material $m \in \mathcal{M}$ at location $k \in \mathcal{L} \cup \mathcal{D} \cup \mathcal{S}$. The objective function is then:

$$\min \qquad z = \sum_{(i,j)\in\mathcal{A}} \sum_{m\in\mathcal{M}} \sum_{t\in\mathcal{T}} c_{ij} x_{ij}^{mt} + \sum_{k\in\mathcal{L}\cup\mathcal{D}\cup\mathcal{S}} \sum_{m\in\mathcal{M}} \sum_{t\in\mathcal{T}} c_k^m f_k^{mt}. \tag{6.3}$$

Arc capacity constraints. Every arc $(i,j) \in \mathcal{A}$ has a maximum capacity U_{ij}^t (in cubic meters) for each period $t \in \mathcal{T}$, so:

$$\sum_{m\in\mathcal{M}} x_{ij}^{mt} \leq U_{ij}^t \qquad (i,j) \in \mathcal{A}, t \in \mathcal{T}. \tag{6.4}$$

Vehicle capacity constraints. Materials are moved by trucks belonging to a set $\mathcal{W}$ of types, each of those may transport a set $\mathcal{M}(w) \subseteq \mathcal{M}$ of materials, has a maximum transportation capacity $V(w)$, a maximum number of km that can be traveled in the time horizon $K(w)$. The number of vehicles of

type $w \in \mathcal{W}$ is denoted by $N(w)$. To each arc $(i,j) \in \mathcal{A}$ is assigned a non-negative length ℓ_{ij} in km. The overall transportation capacity is imposed by:

$$\sum_{(i,j)\in\mathcal{A}} \sum_{m\in\mathcal{M}(w)} \ell_{ij} x_{ij}^{mt} \leq N(w) \cdot V(w) \cdot K(w) \qquad t \in \mathcal{T}, w \in \mathcal{W}. \tag{6.5}$$

Filling constraints. Filling tasks represent the need of some amount of material in a location into a particular time window. If I_f is the set of filling tasks, each task $i \in I_f$ is linked to its material location $h(i) \in \mathcal{H}$, and has its own request d_i to be satisfied inside the time window, specified by its starting period $t_s(i)$ and its ending period $t_e(i)$. Outside of this time window, the flow of a task must be zero. Let us define $\mathcal{M}_f(i) \subseteq \mathcal{M}$ as the set of alternative materials that can be used to respond to the request of a filling task $i \in I_f$. We must guarantee that the request is fulfilled, and materials not included in $\mathcal{M}_f(i)$ are not used. We obtain:

$$x_{h(i),i}^{mt} = 0 \qquad i \in I_f, m \in \mathcal{M} \setminus \mathcal{M}_f(i), t \in \mathcal{T}, \tag{6.6}$$

$$x_{h(i),i}^{mt} = 0 \qquad i \in I_f, m \in \mathcal{M}, t \in \mathcal{T} : t < t_s(i) \, or \, t > t_e(i), \tag{6.7}$$

$$\sum_{m\in\mathcal{M}(i)} \sum_{t=t_s(i)}^{t_e(i)} x_{h(i),i}^{mt} = d_i \qquad i \in I_f. \tag{6.8}$$

Digging constraints. Digging tasks represent the excavation of some amount of materials in a certain location into a given time window. Let us define the set of digging tasks as I_d, where each task $i \in I_d$ has a request d_i of a composition of materials defined by the set $\mathcal{M}_d(i) \subseteq \mathcal{M}$. Each material $m \in \mathcal{M}_d(i)$ has a coefficient that determines the composition of digging task $i \in I_d$, φ_i^m, where $\sum_{m\in\mathcal{M}_d(i)} \varphi_i^m = 1$. For each task $i \in I_d$ is set a unique material location $h(i) \in \mathcal{H}$ and is defined a time window $[t_s(i), t_e(i)]$ to be respected.

We must impose zero flows for materials different from $\mathcal{M}(i)$ and outside the time windows, while the demands of the tasks must be satisfied into the time windows. We thus obtain:

$$x_{i,h(j)}^{mt} = 0 \qquad i \in I_d, m \in \mathcal{M} \setminus \mathcal{M}_d(i), t \in \mathcal{T}, \tag{6.9}$$

$$x_{i,h(j)}^{m,t} = 0 \qquad i \in I_d, m \in \mathcal{M}_d(i), t \in T : t < t_s(k) \, or \, t > t_e(k), \tag{6.10}$$

$$\sum_{m\in\mathcal{M}_d(i)} \varphi_i^m \sum_{t=t_s(i)}^{t_e(i)} x_{i,h(i)}^{m,t} = d_i \qquad i \in I_d \tag{6.11}$$

Flow conservation at material locations, and external vertices. Flow conservation among the material locations, the access points, and the external vertices is imposed by:

$$\sum_{(i,j)\in\mathcal{A}} x_{ij}^{mt} = \sum_{(j,i)\in\mathcal{A}} x_{ji}^{mt} \qquad j \in \mathcal{H}\cup\mathcal{B}, m \in \mathcal{M}, t \in \mathcal{T}. \tag{6.12}$$

Temporary depots constraints. Let us define $\mathcal{M}_s$ as the set of materials that can be stored into temporary depots and Q_k as the maximum storing capacity that must be respected at each period in $k \in \mathcal{L}$. We have:

$$\sum_{m\in\mathcal{M}_s} f_k^{mt} \leq Q_k \qquad k \in \mathcal{L}, t \in \mathcal{T}. \tag{6.13}$$

Let f_k^{m0} be the initial stock of material $m \in \mathcal{M}_s$ in depot k at the beginning of the first period, and $f_k^{m,T_{max}+1}$ the upper bound on the closing stock of m in k at the end of the last period T_{max}. We have the following material balance constraints:

$$\sum_{(i,k)\in\mathcal{A}} x_{ik}^{mt} + f_k^{m,t-1} = \sum_{(k,i)\in\mathcal{A}} x_{ki}^{mt} + f_k^{mt} \quad k \in \mathcal{L}, m \in \mathcal{M}_s, t \in \mathcal{T}, \tag{6.14}$$

$$f_k^{m,T_{max}} \leq f_k^{m,T_{max}+1} \qquad k \in \mathcal{L}, m \in \mathcal{M}_s. \tag{6.15}$$

Dump sites constraints. Let C_k be the total capacity of a dump site $k \in \mathcal{D}$ and Q_k^m the maximum capacity for each period and material. Then:

$$\sum_{(i,k)\in\mathcal{A}} \sum_{t\in\mathcal{T}} \sum_{m\in\mathcal{M}} x_{ik}^{mt} \leq C_k \qquad k \in \mathcal{D}, \tag{6.16}$$

$$\sum_{(i,k)\in\mathcal{A}} x_{ik}^{mt} \leq Q_k^m \qquad k \in \mathcal{D}, m \in \mathcal{M}, t \in \mathcal{T}. \tag{6.17}$$

Quarries constraints. As for the dump sites, each quarry $k \in \mathcal{S}$ has an overall capacity C_k and a maximum capacity Q_k^m for each material m in each period, therefore:

$$\sum_{(k,i)\in\mathcal{A}} \sum_{t\in\mathcal{T}} \sum_{m\in\mathcal{M}} x_{ki}^{mt} \leq C_k \qquad k \in \mathcal{S}, \tag{6.18}$$

$$\sum_{(k,i)\in\mathcal{A}} x_{ki}^{mt} \leq Q_k^m \qquad k \in \mathcal{S}, m \in \mathcal{M}, t \in \mathcal{T}. \tag{6.19}$$

Separation plants constraints. The recycling process is made of a first step where material $rawH$ (the highest quality material obtained by digging) is separated into a set of materials $\mathcal{M}_s$ in separations plants, and of a second part where some of these materials can be combined with *bitumen*

and *cement* to obtain *asphalt* and *concrete* in asphalt and concrete mixing plants. Let us define $\mathcal{F}_s, \mathcal{F}_a$, and $\mathcal{F}_c$ the sets of separation plants, asphalt mixing plants, and concrete mixing plants, respectively. The flows entering a separation plant $k \in \mathcal{F}_s$ for all materials $m \in \mathcal{M}$ other than $rawH$, as well as the flows leaving the vertex for all materials other than $\mathcal{M}_s$ must be zero, that is:

$$x_{ik}^{mt} = 0 \qquad (i,k) \in \mathcal{A} \colon k \in \mathcal{F}_s, m \in \mathcal{M} \setminus \{rawH\}, t \in \mathcal{T}, \tag{6.20}$$

$$x_{ki}^{mt} = 0 \qquad (k,i) \in \mathcal{A} \colon k \in \mathcal{F}_s m \in \mathcal{M} \setminus \mathcal{M}_s, t \in \mathcal{T}. \tag{6.21}$$

The accumulated material inflow should not exceed the maximum allowable operating capacity Q_k of each separation plant $k \in \mathcal{F}_s$, thus:

$$\sum_{(i,k)\in\mathcal{A}} x_{ik}^{rawH,t} \leq Q_k \qquad k \in \mathcal{F}_s, t \in \mathcal{T}. \tag{6.22}$$

Each separation plant has its own and specified recipe, that states the quantities of exiting materials obtained from a unit of $rawH$. Formally, φ_s^m gives the quantity of exiting material $m \in \mathcal{M}_s$ that can be obtained from a unit of $rawH$ at plant $k \in \mathcal{F}_s$, and note that, for feasibility, we must have $\sum_{m\in\mathcal{M}_s} \varphi_s^m = 1$. The conservation flow at each separation plant is stated by:

$$\sum_{(k,i)\in\mathcal{A}} x_{ki}^{mt} = \varphi_s^m \sum_{(i,k)\in\mathcal{A}} x_{ik}^{rawH,t} \qquad k \in \mathcal{F}_s, m \in \mathcal{M}_s, t \in \mathcal{T}. \tag{6.23}$$

Asphalt mixing plants constraints. Each asphalt mixing plant $k \in \mathcal{F}_a$ must receive as input the set of materials $\mathcal{M}_a$, and gives in output *asphalt*, thus:

$$x_{ki}^{mt} = 0 \qquad (k,i) \in \mathcal{A} \colon k \in \mathcal{F}_a, m \in \mathcal{M} \setminus \{asphalt\}, t \in \mathcal{T}, \tag{6.24}$$

$$x_{ik}^{mt} = 0 \qquad (i,k) \in \mathcal{A} \colon k \in \mathcal{F}_a, m \in \mathcal{M} \setminus \mathcal{M}_a, t \in \mathcal{T}. \tag{6.25}$$

The plant has a total production capacity Q_k for each period. Its recipe is expressed by the φ_a^m input, which gives the quantity of material $m \in \mathcal{M}_a$ necessary for producing a unit quantity of asphalt at plant $k \in \mathcal{F}_a$. For feasibility, the values of φ_a^m satisfy $\sum_{m\in\mathcal{M}_a} \varphi_a^m = 1$. We thus get:

$$\sum_{(k,i)\in\mathcal{A}} x_{ki}^{asphalt,t} \leq Q_k \qquad k \in \mathcal{F}_a, t \in \mathcal{T}, \tag{6.26}$$

$$\sum_{(i,k)\in\mathcal{A}} x_{ik}^{mt} = \varphi_a^m \sum_{(k,i)\in\mathcal{A}} x_{ki}^{asphalt,t} \qquad k \in \mathcal{F}_a, m \in \mathcal{M}_a, t \in \mathcal{T}. \tag{6.27}$$

Concrete mixing plants constraints. Similarly as for the asphalt mixing plants, each concrete mixing plant $k \in \mathcal{F}_c$ receives as input materials $\mathcal{M}_c$,

gives in output *concrete*, and has total processing capacity Q_k for each period. Its recipe is expressed by φ_c^m, that gives the quantity of material $m \in \mathcal{M}_c$ necessary for producing a unit quantity of concrete, and is such that $\sum_{m \in \mathcal{M}_c} \varphi_c^m = 1$. Hence we impose constraints similar to (6.24)–(6.27) for the concrete mixing plants. The resulting aggregate model is thus to minimize (6.3), subject to (6.1)–(6.2), (6.4)–(6.27).

The main changes with respect to the aggregate formulation in [2] derive from the use of alternative materials for filling, which affects (6.6)–(6.8), and the use of fixed compositions of materials in digging tasks, which affects (6.9)–(6.11).

6.A.1 Disaggregate Formulation

The disaggregate formulation aims at determining the actual paths on the network used by each material in each period by using a model one for each material/period of the time horizon. To do so, we start from the solution of the aggregate formulation, that provides all the possible origins and destinations. We enumerate the set of all possible paths $\bar{\mathcal{P}}$ and arcs $\bar{\mathcal{A}}(p)$ used by each path $p \in \bar{\mathcal{P}}$ starting from all the possible origins and heading to all the possible destinations of a given material in a given period, by applying a standard label setting algorithm. We clarify that the typical graphs of the problem have a particular structure that is very sparse and that follows the shape of the highway building site. This makes the number of possible paths quite limited so it is not prohibitive to enumerate all of them.

We use a set covering model, where, for each path, we introduce a non-negative variable y_p representing the quantity of material hauled on that path and the related cost c_p. Note that, by using this model, the cost of a path can also be given as a non-linear function, that we represent by c_p. The flows provided by the aggregate solution are denoted by $\bar{x}_{hk}$ on arc $(h,k) \in \bar{\mathcal{A}}$ (where period and material are omitted because fixed) and represent a lower bound for the material hauled on paths on each arc as imposed on constraint (6.29).

We report here a generic disaggregate model for one material and one period:

$$\min \quad \sum_{p \in \bar{\mathcal{P}}} c_p y_p \tag{6.28}$$

$$\sum_{p \in \bar{\mathcal{P}}:(h,k) \in \bar{\mathcal{A}}(p)} y_p \geq \bar{x}_{hk} \qquad (h,k) \in \bar{\mathcal{A}} \tag{6.29}$$

$$y_p \geq 0 \qquad p \in \bar{\mathcal{P}}. \tag{6.30}$$

The objective function (6.28) aims at minimizing the costs, which can be very flexible since they depend on each path. In our instances we calcu-

late the cost c_p as a function that models the transport costs given by the contracts in use at Strabag AG. These costs increase with a steep slope from 0 to 5 km, and after that, the slope has a more gradual increase.

Bibliography

[1] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network flows: theory, algorithms, and applications.* Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1993.

[2] C. Bogenberger, M. Dell'Amico, G. Fuellerer, G. Hoefinger, M. Iori, S. Novellani, and B. Panicucci, B. A Two-Phase Earthwork Optimization Model for Highway Construction. *to appear on Journal of Construction Engineering and Management*, 2014.

[3] S. M. Easa. Earthwork allocations with nonconstant unit costs. Journal of Construction Engineering and Management. 113:34-50, 1987.

[4] S. M. Easa. Earthwork allocations with linear unit costs. Journal of Construction Engineering and Management. 114:641-655, 1988.

[5] W. L. Hare, V. R. Koch, and Y. Lucet. Models and algorithms to improve earthwork operations in road design using mixed integer linear programming. *European Journal of Operational Research*, 215:470-480, 2011.

[6] S. M. Karimi, S. J. Mousavi, A. Kaveh, and A. Afshar. Fuzzy optimization model for earth-work allocations with imprecise parameters. *Journal of Construction Engineering and Management*, 133:181-190, 2007.

[7] M. Marzouk and O. Moselhi. Multiobjective optimization of earthmoving operations. *Journal of Construction Engineering and Management*, 130:105-113, 2004.

[8] R. M. Stark and R. H. Mayer. *Quantitative construction management: Uses of linear optimization.* Wiley, New York, 1983.

[9] R. M. Stark and R. L. Nicholls. *Mathematical foundations for design: Civil engineering systems.* McGraw-Hill, New York), 1972.

[10] H. Zhang. Multi-objective simulation-optimization for earthmoving operations. *Automation in Construction* 18:79-86, 2008.

Part IV

Models and Algorithms for 3D Printing Problems

Chapter 7

Optimizing the 3D Printing Process

In this chapter we treat the production technology known as 3D printing. We deal with the 3D printing process by describing it, its advantages and disadvantages and by supplying an exhaustive classification of the many features to deal with in the 3D printing process optimization. Among the many operations related to the 3D printing process, we focus on the tool path definition problem of a 3D printer, that we called 3D Routing Problem (3DRP). The 3DRP is a generalization of the rural postman problem, in the family of arc routing problems. We present an ILP formulation for the 3DRP and we developed four heuristic algorithms that can furnish with improved solutions with respect to the solutions obtained by one of the most used software for the tool path definition in 3D printing.

Keywords: 3D printing, Rural postman, Arc routing, Models, Heuristics.

7.1 Introduction

3D Printing (3DP), also known as *Additive Manufacturing*, is a production technology that can construct three-dimesional objects of multiple shapes by adding one or more materials layer by layer. Since its rise in the 80s, different techniques for 3DP have been developed, including *Stereolithography*, that makes use of liquid polymers hardened by UV beams layer by layer, *Selective Laser Sintering*, where a laser melts the powder -layer by layer- in a selective way, and *Fused Deposition Modeling* (FDM), also known as *Fused Filament Fabrication*. FDM involves extruding thermoplastic material (such as ABS plastic or polycarbonate) through heated nozzles: the filament is pulled by a feed roller, it is heated to a temperature above the material's melting point, and then it is extruded by the nozzle. Once a layer is completed the platform where the material has been deposited is lowered or the nozzles is

raised to the next layer level and a new layer is added and fused with the previous one. A supporting material can also be used in the process when it is needed for structural issues, especially in case of overhanging geometries (Campbell et al. [6], Akella [1]). From now on, when mentioning 3DP we will refer to the FDM technique.

In this work we focus on the optimization of the 3DP process, in particular we will concentrate on the optimization of the path that the nozzle has to follow to produce the three-dimensional object, in terms of distances, given several sets of constraints. Before introducing models and heuristic algorithms, we spend some words on the 3DP process.

7.1.1 The 3DP Process

The manufacturing process of a product through 3DP follows a typical scheme as depicted in Figure 7.1. The process starts with the definition of the 3D model which can be created by a computer-aided design (CAD) software or by scanning an object we want to reproduce. The second step is to reproduce the three-dimensional object in a polygonal mesh, normally made of triangles. The typical file used to report a polygonal mesh is in a .stl format. When the model is represented as a mesh, the corresponding .stl type file is passed to a slicing software that divides the mesh in layers. The following step is the most interesting to our work and it is to decide the path to be followed by the tool for reproducing the three-dimensional object. The solution obtained is expressed in a sequence of commands in a g-code format that is passed to the printer in order to produce the final 3D object (Campbell et al. [6]).

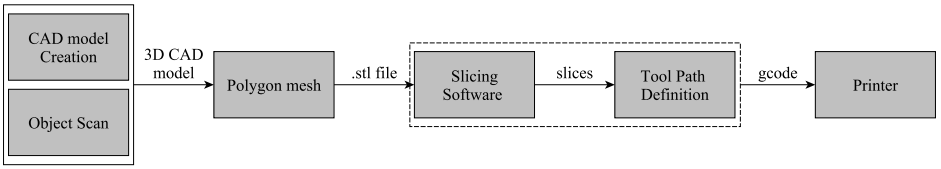


Figure 7.1: Scheme of the 3DP process.

7.1.2 Advantages and disadvantages

The 3DP processes have some advantages such as the application to rapid prototyping, the attitude towards the tool-less production of finished goods, the mass production of customized parts and low volume ones, the freedom in engineering design, the possibility of building complex geometries impossible to obtain with other techniques, and of delivering finished goods from digital data cutting also the logistic costs and minimizing carbon footprint. Some other advantages are incarnated by the limited use of chemicals

such as etching and cleaning solutions, the possibility to use recycled material and the opportunity to use the same technique with diverse materials (Akella [1]). Some costs that can decrease thanks to 3DP technology are identified in: waste generation as the one of subtractive machine is avoided and thus disposal and recycling costs; inventory costs are minimized because products can be printed on demand and there is no need of semi-finished products; labor time for designing and assembling multiple parts is avoided; less opportunity for human error since the production is driven by digital information; lastly set-up costs are minimized since many products can be produce without retooling (Bak, Beltrami and Bodin, Boding and Kursh, Campbell et al. [3, 6]).

However, 3DP is not free from limits. Indeed, 3DP does not seem appropriate for the mass production since its main disadvantage is the required amount of production time. Another issue is the quality of the material, normally used materials are polymers that do not present the same good characteristics of materials traditional of subtracting manufacturing and are typically weaker, moreover parts produced on different machines can guarantee particular properties with respect to quality and performances. Furthermore, the layer-by-layer building process makes the parts weaker in the direction of the build (Campbell et al. [6]).

7.1.3 Contribution of this Chapter

In the 3DP process there are, mainly, two aspects that can be optimized. The first one is about the quality of the printed object and it is related to the quality of the surface. We will consider some features in order to improve the quality of the surface, but to cover completely this point is not the aim of the chapter. The second aspect that can be optimized is related to the time of the print that can be minimized and that is strictly connected to the tool path. In this chapter we will account for this second aspect by focusing in minimizing the distance traveled by the nozzle during the 3DP process, and, indirectly, minimizing the time of the print.

We will present a formal description of the problem derived from 3DP, a related literature review, and a mathematical model to represent the basic problem, which is a generalization of the well known Rural Postman Problem. To solve the problem we introduce some heuristic algorithms and we produce a collection of instances derived from three-dimensional objects on which we perform computational tests. Last but not least, we list a set of features and future research direction to be followed in the field that represents the intersection between optimization and 3DP.

The chapter is organized as follows. In Section 7.2 we will define the class of the Rural Postman Problems and produce an exhaustive literature review. In Section 7.3 we define the problem we are presenting, while in Section 7.4 we give a formulation for the problem. In Section 7.5 we present

some heuristic algorithms to solve the problem and in Section 7.6 we show the computational results of the presented algorithms on a set of newly collected instances.

7.2 The Rural Postman Problems

The problem of defining the tool path in a 3DP process is a generalization of the *Rural Postman Problem* (RPP), that is a particular version of the more general class of *Arc Routing Problems* (see, e.g., Corberán and Laporte [7]). The RPP is the problem of finding the minimum cost eulerian cycle that visits a set of arcs at least once, by using also a second set of arcs, if needed, where each arc is associated with a non-negative cost. The RPP represents a generalization of the *Chinese Postman Problem* (CPP), that is the problem of finding the minimum cost path that visits at least once all the arcs. The CPP can be solved in polynomial time, while the RPP (directed or undirected) is an NP-hard problem if the set of required arcs is not strongly connected (Laporte [27]).

The RRP arises in many real-world situations such as the snow plowing (see, e.g., Haslam and Wright [24]), the garbage collection (see, e.g., Beltrami and Bodin [4]), the mail delivery (see, e.g, Levy [29]), the school bus routing (see, e.g, Angel et al. [2]), the meter reading (see, e.g., Stern and Dror [32]), and the street sweeping (see, e.g., Bodin and Kursh [5]). The interested reader can find other work related to these applications in Eiselt et al. [16]. More recent application of the RRP are referred to the control of plotting and drilling machines (see, e.g. Groetschel [22]) and to the optimization of laser-plotter beam movements (see, e.g. Ghiani and Improta [19]).

Many polyhedral studies and exact algorithms, mostly cutting plane and branch-and-cut algorithm, have been proposed in the last years for the RRP and its generalizations. See, e.g., Corberán and Sanchis [12, 13], Laprte and Ghiani [21] (where they use a formulation with just 1/0 variables), Reinelt and Theis [31], and Fernández et al. [17] (where they use a completely different approach and formulation from the previous works). According to the survey redacted by Corberán and Prins [11], the largest RPP instances that have been solved to optimality are characterized by up to 1000 vertices, 3080 edges and 204 required components and are solved in about one hour in average on a standard PC by using the branch-and-cut described in Corberán et al. [10] for the Windy General Routing Problem (which contains the RPP).

Probably, the most famous heuristic algorithm for the RPP is the one proposed by Frederikson [18], which is built on the well known heuristic for the TSP by Christofides, and, similarly, presents a worst case bound of 3/2. Hertz et al. [25] propose some improving techniques to be applied to the Frederickson algorithm and to a simply newly developed heuristic. Corberán

et al. [9] present two heuristics approaches to solve the Mixed RPP where the problem is defined on a graph with edges and arcs and required components can be both edges or arcs. Groves and Vuuren [23] present an effective local search framework for the URRP based on local searches for the TSP. They solve heuristically very large instances with up to about 5000 vertices and 30000 edges. Ghiani et al. [20] propose a constructive heuristic for the Undirected RPP (URPP) with a local post-optimization. The procedure is competitive with the Frederickson one. Holmberg [26] proposes some heuristics for the RPP built on the Frederikson heuristic by changing the order of the components. The author includes also two post-processing heuristics to improve the solution.

We report now a set of variants of the RPP that are interesting for 3DP. Dror and Langevin [15] introduced the Directed Clustered RPP and solved it by transforming the problem into a Generalized TSP. They separate the arcs in clusters that must be finished before stepping to another cluster. Letchford and Eglese [28] present the RPP with deadline classes, a RPP with time windows where the time windows have only a maximum time to be respected. They propose a formulation and a cutting plane algorithm. Corberán et al. [8] consider the mixed RPP with turn penalties. They associate a penalty to every turn and take into account also the existence of forbidden turns. The problem is to find the minimal tour avoiding forbidden tours and paying for turn penalties. The authors transform the problem into an ATSP and solve it by using exact and heuristic algorithms for the ATSP.

Some relevant applications of the RPP are listed in the following. Ghiani and Improta [19] consider the problem of drawing and decorating metal surfaces by means of a laser plotter, that works as a 2D printer, minimizing the total length of the non-drawing moves. The so-called Laser-Plotter Beam Routing Problem is modeled as a RPP with additional constraints and solved transforming it into a RPP. By using the branch-and-cut presented in Ghiani and Laporte [21] istances with up to 225 vertices can be solved. Moreira et al. [30] describe the problem that arises when manufacturing high-precision tools. They intend to determine the shortest path for the cutting of given pieces. The problem is represented as a particular RPP where non-cutting movements are allowed only after the complete cut of a piece. Hence the problem is called Dynamic RPP and solved by means of a heuristic algorithm.

7.3 Problem Description

We call *3D printing Routing Problem* (3DRP) the generalization of the URPP that is related to the tool path definition in a 3DP process. In particular, the nozzle must start from its starting position and return to the same position when the print is terminated. In addition, the edges are set

into clusters to be visited in a sequence, that are represented by the layers of the print. On each layer the set of *compulsory edges* represents the set of edges on which the nozzle is required to depose material (the first time they are used), while the set of the *optional edges* represents the other edges. Because of the printed material, no layer can be started before terminating the previous one, and the nozzle cannot move back to previous layers. Then one and only one optional edge can be used to move from one layer to the next one and no edge between two non-subsequents layers is present. Lastly, only optional edges are present between two layers. We will formalize the 3DRP in Section 7.4. The 3DRP represents the *basic problem* of optimizing the tool path of a 3DP process. In the following we are focusing on the basic problem, but some additional characteristics can arise across the 3DP process: we will list them in Section 7.7.

7.4 The 3D Printing Routing Problem

In this section we define the sets and parameters used to formulate the 3DRP. The problem is defined on an undirected graph $G = (V, E)$, where V is the set of vertices and E is the set of edges. Moreover, it is defined a subset $R \subseteq E$ of edges that must be visited at least once. In our case these edges are the compulsory edges, where the material is extruded (printed) the first time they are used. On the other hand $E \setminus R$ represents the set of optional edges.

Let us define *layer* as a set of vertices characterized by the same altitude. We call $\mathcal{L}$ the set of all layers where a layer $L_h \in \mathcal{L}$ is one layer of print, $\mathcal{L} = \cup_{h=1}^{|\mathcal{L}|} L_h$, and $L_h \cap L_k = \emptyset, h \neq k, L_h, L_k \in \mathcal{L}$. The graph G is clustered in several subgraphs induced by the layers plus the starting and ending point of print, 0, and all the edges connecting these components. Thus, $V = \mathcal{L} \cup \{0\}$.

Finally, let us define the variable x_{ij} that models the number of time the edge $(i, j) \in E \setminus R$ is traveled, and the number of times the edge $(i, j) \in R$ is traveled in addition to the first time. We define c_{ij} as the shortest distance between the vertices i and j, $(i, j) \in E$. Hence, the 3DRP aims at minimizing the path of the nozzle starting from vertex 0, traveling all the compulsory edges at least once, respecting the order of the layers, and returning to the vertex 0. In the following we present the formulation for the basic problem:

$$\min \sum_{(i,j)\in R} c_{ij}(1 + x_{ij}) + \sum_{(i,j)\in E\setminus R} c_{ij}x_{ij} \tag{7.1}$$
$$\sum_{j\in R:j>i} (1 + x_{ij}) + \sum_{j\in R:j<i} (1 + x_{ji}) +$$

$$\sum_{j\in V:j>i} x_{ij} + \sum_{j\in V:j<i} x_{ji} = 2z \qquad i \in V \quad (7.2)$$

$$\sum_{(i,j)\in R:i\in S,j\in\bar{S},j>i} (1+x_{ij}) + \sum_{(i,j)\in R:i\in S,j\in\bar{S},j<i} (1+x_{ij}) + \\ + \sum_{(i,j)\in E\setminus R:i\in S,j\in\bar{S},j>i} x_{ij} + \sum_{(i,j)\in E\setminus R:i\in S,j\in\bar{S},j>i} x_{ij} \geq 1$$

$$(S \subseteq L_h, \\ \bar{S} = L_{h+1} \cup L_h \setminus S, \\ L_h \subseteq \mathcal{L} \setminus L_{|\mathcal{L}|}) \\ \vee (S \subseteq L_{|\mathcal{L}|}, \\ \bar{S} = \{0\} \cup L_{|\mathcal{L}|} \setminus S) \quad (7.3)$$

$$\sum_{j\in L_1} x_{0j} = 1 \quad (7.4)$$

$$\sum_{j\in L_{|\mathcal{L}|}} x_{j0} = 1 \quad (7.5)$$

$$\sum_{(i,j):i\in L_h, j\in L_{h+1}} x_{ij} = 1 \qquad L_h \subseteq \mathcal{L} \setminus L_{|\mathcal{L}|} \quad (7.6)$$

$$x_{ij} \geq 0 \qquad (i,j) \in E \quad (7.7)$$

$$z \geq 0 \quad (7.8)$$

The objective function aims at minimizing the total costs, i.e. distances, counting also the compulsory edges. Constraint (7.2) states that the number of edges incident a vertex are even or zero. In (7.3) we impose the connectivity of the solution. Constraints (7.4) and (7.5) state that one and only one edge must leave and return to the depot. Constraint (7.6) imposes that one and only one edge can be used between one layer and the next one. The non-negativity of the variables is imposed in constraints (7.7) and (7.8).

7.5 Heuristic Algorithms

The 3DP process, when printing real objects, involves from thousands to millions of edges to be printed and traveled by the nozzle. Thus, real 3DRP instances result impossible to be solve exactly, because of the huge number of variables and constraints needed. For this reason we developed four heuristic algorithms to solve the 3DRP. To do so, we organized the input in couples of vertices that defines the edges to be traveled and printed on each layer (the set R), named as compulsory edges. The optional edges ($E \setminus R$) and the compulsory edges used more than once are the *non-printing edges*, and are computed and inserted just when needed during the construction of the solution. The input is obtained by analyzing the g-codes created by Cura

[33], one of the most advanced 3DP software.

7.5.1 Closest 3DP

The first algorithm we present to solve the 3DRP is called *Closest 3DP* algorithm. It starts from vertex 0 and moves from one edge to the closest unused compulsory edge (we enter in the edge by the closest vertex with respect to the current one and then we travel the edge to arrive in its second vertex, which will be the new current vertex). If the second vertex of the current edge is also the first vertex of the next edge, we link them directly, otherwise we insert a non-printing edge. When the compulsory edges of the current layer have been used at least once, the algorithm moves to the next layer by using the cheapest edge. We repeat this until the last layer is completed, then the last visited vertex is connected to vertex 0 to end the algorithm.

7.5.2 Clustered 3DP

Before explaining the second algorithm presented, we specify that each layer is divided in different subpolygons and each subpolygon can be formed, at most, by three clusters of compulsory edges: the outer part (the border of the subpolygon), the inner part (that follows the outer part internally, to strengthen the structure), and the filling part (that represents the edges used to fill the subpolygon), see e.g., Figure 7.2.

The second algorithm we present is called *Clustered 3DP* algorithm, follows a similar idea with respect to the Closest algorithm, and makes use of the different clusters. The algorithm selects a cluster on a layer (choosing among outer, inner, and filling) and travels all the edges of the cluster, before going to another cluster. The algorithm starts from vertex 0 and selects the closest vertex of the compulsory edges on layer L_1. This action not only defines the next edge, but also the first cluster among inner, outer, and filling to be completed before moving to the next one. When the current cluster is completed, the algorithm moves to the closest vertex, which defines the next edge and the next cluster. The algorithm continues until all clusters of the current layer are completed and then the closest vertex of the next layer is chosen. When the last layer is completed we travel back to

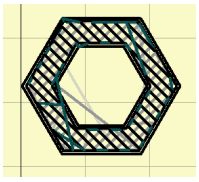


Figure 7.2: An example of polygon with outer, inner, and filling parts.

vertex 0 and the algorithm terminates.

7.5.3 Look Ahead 3DP

The third algorithm we use to solve the 3DRP is called *Look Ahead 3DP*, whose name is inspired by the fact that it makes use of heuristic information on the future possible decisions. The algorithm starts from vertex 0 and it then selects the three closest vertices among those of the compulsory edges of layer L_1. A particular cost is then computed by running the Closest 3DP algorithm from each of the three vertices till the end of the layer. The cost is computed as follows: we sum up the traveling cost of the non-printing edges needed to terminate the layer by the Closest 3DP, the vertex of the three selected ones that furnishes with the cheapest cost is used. This process is iterated until all the compulsory edges of the current layer are used at least once. We then use the cheapest edge to move to the next layer and repeat the procedure until the last layer. To conclude the algorithm we move back to the vertex 0.

7.5.4 Shortest Path Based 3DP

The forth algorithm that we are presenting is called *Shortest Path Based 3DP*. This algorithm is divided into two phases, the first one builds a supporting graph to be used in the second phase. The second phase computes the Dijkstra algorithm (see, e.g., Dijkstra [14]) on the newly built supporting graph to obtain a solution for the 3DRP. In the following we will describe the two phases more precisely.

Phase one. In this phase we define the supporting graph $G' = (V', E')$, where $V' \subseteq V$ is the set of vertices called starting points and E' the set of edges that we describe in the following. For each layer we select a random set of vertices among those of the compulsory edges (R), called *starting vertices*, from which we can start the layer. Starting from the starting vertices we terminate the layer as if using the Closest 3DP algorithm. For each starting vertex we will end up in one and only one vertex that we call *ending vertex*. The distance between a starting vertex and its ending vertex is the cost computed by the Closest 3DP algorithm. We now compute the distance between the ending vertices of one layer and the starting vertices of the next layer. At this point we obtain an edge going from each starting vertex of one layer to each starting vertex of the next layer, that define E'. We call c'_{ij} the cost of each edge of E' as the sum of the costs derived by the Closest 3DP algorithm and the distance between the corresponding ending vertex and the starting one of the next layer. The starting vertices of the first layer are linked to the vertex 0 and a cost $c'_{0j}, j \in L_1$ is associated to them. The ending vertices of the last layer are linked to the vertex 0', that has the same coordinates of vertex 0, and a cost $c'_{j0}, j \in L_{|\mathcal{L}|}$ is associated to them. The

costs $c'_{ij}, (i,j) \in V'$ give the weight of the edges of the supporting graph G'.

Phase two. In the second phase, we compute the shortest path form 0 to 0' on the supporting graph G'. This leads to a feasible solution for the 3DRP.

We tested the Shortest Path Based 3DP algorithm by selecting the number of starting vertices in two ways, in the first one we decide a fixed number between 10, 20, 30, 40, 50, 60, and 100 (or the cardinality if the layer, if exceeded), while in the second case we use the maximum value between 1 and a percentage of the vertices of each layer (1%, 2%, 5%, 10%, 15%, 20%, 25%).

7.6 Computational Results

7.6.1 Instances

We collected 30 instances on the Internet from a design community for 3DP called `thingiverse.com`. In Table 7.1 we report the name of the instance in the first column. For each instance is specified $|\mathcal{L}|$, the number of layers, $|\mathcal{P}|$, the number of polygons, and the ratio between them. We then report the total distance of the compulsory edges ($\sum_{(i,j)\in R} c_{ij}$) and the cardinality of the set of compulsory edges ($|R|$).

7.6.2 Results

The collected instances have been processed with one of the most famous software used for slicing polygonal meshes and for path definition, Cura [33]. We used the collected instances as input for Cura that takes the polygonal meshes of the instances, slice them, and defines a path of the nozzle in a code called gcode. The gcode represents, for each instance, a heuristic solution for the related 3DRP in a language that the 3D printer can understand. By processing the obtained gcode we can obtain the set of layers, the set of compulsory edges to be printed, the corresponding vertices and their coordinates. These informations define the data that we used as instances for our algorithms.

Let us call Γ the cost of the non-printing edges used in a given solution for the 3DRP. Let us call Δ_Γ the difference between the cost of the non-printing edges used by the solution defined by Cura and the cost of the non-printing edges used by the solution obtained thanks to one of our algorithms. Let $\%gap_\Gamma$ be the percentage gap between them. In the following tables presenting the results of the proposed algorithms we show the name of the instance, the corresponding Δ_Γ and $\%gap_\Gamma$, the percentage gap ($\%gap$) between the Cura solution value and the the solution value obtained by our algorithm, and the time needed to our algorithm to terminate. All tests have been performed on a on a Intel Core i3-2100 with 3.10 GHZ.

Instance	$\|\mathcal{L}\|$	$\|\mathcal{P}\|$	$\|\mathcal{P}\|/\|\mathcal{L}\|$	$\sum_{(i,j)\in R} c_{ij}$	$\|R\|$
30grams	219	222	1.01	91016.609	8693
angel	153	663	4.33	1052191.750	353671
block_4x4	37	112	3.03	32929.641	28694
brickb	332	422	1.27	2851998.700	107355
bunny	41	151	3.68	38896.449	21333
chamfer	20	147	7.35	71653.297	12700
Cityscape	342	1814	5.30	1907842.250	488409
gear_b	85	371	4.36	22001.822	19275
geoboard	82	1642	20.02	291901.188	52244
hilbert2	99	2203	22.25	55116.855	26703
Hilton_Chicago	73	73	1.00	22304.736	7179
ingranaggio	39	51	1.31	64690.348	52569
miwin1	132	396	3.00	610945.625	130267
orso_bis	52	52	1.00	73459.617	35772
pesce_bis	52	193	3.71	139978.297	65262
polysoup	3	6	2.00	3186.609	3018
portab	32	32	1.00	197571.750	47293
portaingr	51	106	2.08	106148.930	69664
RFin	188	248	1.32	54926.410	35898
RostockBottom	44	208	4.73	117799.625	18120
RostockTop	50	213	4.26	87387.406	21132
Rpenta	369	1555	4.21	1039104.563	433954
RShowerHead	274	580	2.12	513838.531	146602
tardisflat2	170	170	1.00	100367.148	24695
tetra	160	160	1.00	98250.273	20839
trail	125	425	3.40	1304639.625	268372
wbuilding	100	211	2.11	11281.513	3364
wine_fixed	39	59	1.51	93598.977	20276
wine1	39	39	1.00	147826.547	19145
WitchCastle	112	265	2.37	48852.797	56334

Table 7.1: Instances Description

In Table 7.2 we report the results of the Closest 3DP algorithm comparing them with the results obtained by Cura. One can observe that we can improve the solution value for 28 out of 30 instances improving of 5.60% the gaps in average. We can obtain an improvement of 27.79% in the bast case. If we focus on the Γ value we can see that the improvement is in average of 22.91%. Also the average solving time is very promising being lower than one second. In Table 7.3 the results for the Clustered 3DP algorithm are shown. In this case we can improve the solution value with respect to the Cura one for 27 out of 30 instances. We can obtain an improvement of 3.02% in average in solving times that are about half a second, in average. In Table 7.4 we show the results for the Look Ahead 3DP algorithm on the collected set of instances. In this case the solution value is improved for every instance with respect to the Cura solution value. In particular, we can improve it of 8.06% in average. If we just consider the non-printing edges we can notice an improvement of 36.73%. To reach these results the Look Ahead 3DP algorithm needs to run for about 39 minutes, in average, which is a relevant solving time if compared with the previously proposed algorithms. These very good solutions have a payback in solving times. In Table 7.5 and Table 7.6 the average results for the Shortest Path Based algorithms are shown. In particular, in Table 7.5 we report the results for the Shortest Path Based 3DP algorithm, where the number of starting vertices for each layer is fixed to 10, 20, 30, 50, 60, and 100. In Table 7.6 we present the results for the Shortest Path Based 3DP algorithm where the starting vertices are fixed with respect to a percentage of the vertices of each layer. The starting vertices are set to 1%, 2%, 5%, 10%, 15%, 20%, and 25% of the vertices of each layer. One can see that for both types of the Shortest Path Based 3DP algorithm the average gap between the solution value of our algorithm and the one obtained by Cura increases with the number of starting vertices explored. The gap is proportional also to the solving times. The Shortest Path Based 3DP algorithm with fixed numbers of starting vertices can reach a 6.40% gap in average in the best case, that is represented by the starting vertices set to 100. This value is obtained in about 80 seconds. One can see that this version works similarly to the version with percentage starting vertices when evaluating the average gaps, but need consistently more seconds to terminate.

The algorithm among the presented ones that shows the best results in terms of improvement of the solution value, compared to the Cura solution value, is the Look Ahead 3DP algorithm. On the other hand the Look Ahead 3DP algorithm needs higher solving times with respect to the other presented algorithms. The Shortest Path Based 3DP algorithm furnishes, in its two versions, with good results in acceptable solving times. Similar gap values can be obtained by the Closest 3DP algorithm showing very small computing times.

	Closest			
Inst.	Δ_Γ	$\%gap_\Gamma$	$\%gap$	$t(sec)$
30grams	661.348	5.66	0.64	0.001
angel	14905.000	10.12	1.24	6.672
block_4x4	6208.443	40.68	12.88	0.140
brickb	-1326.547	-1.00	-0.04	0.594
bunny	1278.109	16.88	2.75	0.079
chambler	373.906	3.17	0.45	0.094
Cityscape	29226.813	9.95	1.33	7.438
gear_b	237.855	3.86	0.84	0.032
geoboard	2005.250	3.93	0.59	0.281
hilbert2	1749.555	11.71	2.50	0.031
Hilton_Chicago	1099.865	18.09	3.88	0.001
ingranaggio	3669.916	22.22	4.52	0.422
miwin1	12697.398	13.26	1.80	1.109
orso_bis	25021.842	60.24	21.76	0.157
pesce_bis	49553.654	64.16	22.81	0.531
polysoup	1453.121	71.14	27.79	0.016
portab	4651.479	13.53	2.01	0.437
portaingr	1825.557	10.54	1.48	0.922
Rfin	970.316	10.32	1.51	0.047
RostockBottom	8677.307	31.54	5.97	0.125
RostockTop	5156.719	23.36	4.71	0.172
Rpenta	58169.266	28.52	4.68	3.125
RShowerHead	34814.063	24.85	5.32	0.515
tardisflat2	9363.699	39.63	7.55	0.031
tetra	-630.948	-4.81	-0.57	0.031
trail	118281.531	48.29	7.63	4.188
wbuilding	75.671	2.86	0.54	0.001
wine_fixed	11248.256	38.03	9.13	0.093
wine1	11980.223	40.19	6.74	0.079
WitchCastle	3353.075	26.33	5.44	0.219
Avg.	13891.725	22.91	5.60	0.919

Table 7.2: Results of the Closest algorithm on test instances.

	Clustered			
Inst.	Δ_Γ	$\%gap_\Gamma$	$\%gap$	$t(sec)$
30grams	331.556	2.84	0.32	0.000
angel	12535.188	8.51	1.05	3.563
block_4x4	3031.178	19.86	6.29	0.094
brickb	-1362.438	-1.03	-0.05	0.469
bunny	732.775	9.68	1.58	0.047
chambler	-368.381	-3.12	-0.44	0.063
Cityscape	14744.344	5.02	0.67	3.641
gear_b	443.250	7.19	1.57	0.032
geoboard	1475.285	2.89	0.43	0.235
hilbert2	1749.555	11.71	2.50	0.078
Hilton_Chicago	868.675	14.28	3.06	0.000
ingranaggio	716.305	4.34	0.88	0.328
miwin1	860.414	0.90	0.12	0.578
orso_bis	15855.703	38.17	13.79	0.109
pesce_bis	24295.828	31.46	11.19	0.359
polysoup	893.126	43.72	17.08	0.016
portab	4127.695	12.01	1.78	0.453
portaingr	1311.128	7.57	1.06	0.485
Rfin	39.749	0.42	0.06	0.032
RostockBottom	3831.223	13.92	2.64	0.141
RostockTop	2383.146	10.80	2.18	0.125
Rpenta	38163.313	18.71	3.07	2.047
RShowerHead	24379.523	17.41	3.73	0.453
tardisflat2	2159.188	9.14	1.74	0.031
tetra	-116.673	-0.89	-0.11	0.031
trail	112033.766	45.74	7.23	2.765
wbuilding	32.005	1.21	0.23	0.016
wine_fixed	3796.424	12.84	3.08	0.063
wine1	474.381	1.59	0.27	0.063
WitchCastle	2227.283	17.49	3.62	0.172
Avg.	9054.817	12.15	3.02	0.550

Table 7.3: Results of the Clustered algorithm on test instances.

	Look Ahead			
Inst.	Δ_Γ	$\%gap_\Gamma$	$\%gap$	$t(sec)$
30grams	2031.766	17.38	1.98	0.31
angel	42756.773	29.02	3.56	22664.21
block_4x4	7429.931	48.68	15.42	122.82
brickb	16783.938	12.67	0.56	765.48
bunny	2365.691	31.25	5.09	45.67
chambler	1818.251	15.39	2.18	77.05
Cityscape	89875.047	30.59	4.08	17105.26
gear_b	879.607	14.26	3.12	10.42
geoboard	7729.773	15.14	2.25	303.95
hilbert2	3211.244	21.50	4.58	13.39
Hilton_Chicago	1836.877	30.21	6.47	1.06
ingranaggio	6060.851	36.70	7.46	758.60
miwin1	30848.766	32.22	4.37	1645.93
orso_bis	30510.017	73.46	26.53	131.84
pesce_bis	54757.037	70.89	25.21	1012.54
polysoup	1594.556	78.06	30.49	21.67
portab	11027.746	32.08	4.75	1208.38
portaingr	5104.657	29.48	4.13	3032.91
Rfin	2859.243	30.41	4.45	17.16
RostockBottom	12860.236	46.74	8.85	128.41
RostockTop	8104.199	36.72	7.40	369.54
Rpenta	94428.086	46.30	7.60	5144.17
RShowerHead	56995.102	40.69	8.72	472.31
tardisflat2	11276.629	47.73	9.10	6.00
tetra	2179.606	16.62	1.96	6.94
trail	145659.258	59.47	9.40	14259.08
wbuilding	441.532	16.70	3.17	0.08
wine_fixed	14227.692	48.10	11.55	62.18
wine1	15236.461	51.12	8.58	46.33
WitchCastle	5396.093	42.37	8.76	187.03
Avg.	22876.222	36.73	8.06	2320.690

Table 7.4: Results of the Look Ahead algorithm on test instances.

Shortest Path Based	Δ_Γ	$\%gap_\Gamma$	$\%gap$	$t(sec)$
10	14284.276	23.50	5.70	8.593
20	15050.768	25.20	6.01	16.320
30	15442.301	26.04	6.15	24.157
40	15614.103	26.43	6.22	31.846
50	15752.656	26.63	6.26	39.831
60	15874.853	26.89	6.31	47.505
100	16086.931	27.38	6.40	80.940

Table 7.5: Average results of the shortest path based algorithm on the test instances.

Shortest Path Based	Δ_Γ	$\%gap_\Gamma$	$\%gap$	$t(sec)$
1%	13515.840	18.54	4.87	24.958
2%	14628.480	21.10	5.31	48.743
5%	15553.952	24.02	5.82	118.131
10%	16012.425	25.70	6.11	181.531
15%	16161.026	26.29	6.21	200.250
20%	16253.602	26.59	6.27	208.474
25%	16295.172	27.00	6.34	212.810

Table 7.6: Average results of the shortest path based algorithm on the test instances.

7.7 Conclusions and Future Research Directions

In this chapter we solved a generalization of the RPP related to the definition of the tool path in a 3DP process, called 3DRP. We provided a collection of instances for the 3DRP that are solved with four different types of heuristic algorithms we proposed and compared with one of the most famous software of path definition available, Cura [33]. We obtained very good results improving the Cura solution values from about 3% to about 8% with our algorithms in very promising solving times.

The verision of the 3DRP we solved is what we called the basic problem of the 3DRP, but many other secondary features can be considered, starting from features to account for an improvement of the quality of the surface. In the following we report an overview of these features divided by topic.

Quality constraints. The 3DP process can produce some blemish on the external surface of the product we intend to print. Indeed, when traveling the optional edges after ejecting material, the nozzle can eject some leftovers and ruin the quality of the external surface. To avoid it we could consider additional constraints.

1. *Second level of clusterization: Subpolygons.* Several separated polygons, called *subpolygons*, can be present on the same layer. Subpolygons produces a second level of clusterization. To define a tool path that follows the second level of clusterization is not strictly required as for the first level of clusterization (layers), but it can be considered in order to improve the quality of the external surface. Indeed, moving from one polygon to the other it can lead to some blemish on the external surface, because of the characteristics of the materials (normally heated plastic). Then, we can impose to complete a polygon before stepping to the next one, by taking one and only one optional arc between a subpolygon to another, on the same layer.

2. *Corner.* To preserve the quality of the surface from the blemishes, we can consider not only to leave the polygon just once, but also to impose the nozzle to pass only through a vertex of the Outer part of the same subpolygon, called *corner*, in order to avoid the intersection of the tool path and the surface when this has already been built. Doing this we will leave the majority of the blemishes on a corner and make it easier to remove them after the print.

3. *Temperature.* The 3DP process includes the melting of the material before extruding it from the nozzle. After being melt, the material conserves a high temperature for a certain amount of time. This means that we cannot extrude some new material on the top of the previous layer until the temperature of the material of the former layer is not

under a certain value, otherwise some flaw or bending can be produced. Considering that layers are overlapped, some edges of a layer cannot be completed before the underlying edge (of the previous layer) had the time to cool down. In order to cool down the temperature of some edges, the speed of the fan can also be considered as a variable.

4. *First vertex of a layer.* When moving from a non-printing edge to a printing one a small excess of material can be extruded. This can lead to an accumulation of unwilled material on the surface, especially when the first vertex of a layer has the similar x-y coordinates with respect to the first vertices of the previous layers. To avoid this, we can impose that the first vertex of a layer must have different coordinates if compared to the first vertex of the previous layer. Indeed, the same line of reasoning can be used when starting overlappingsubpolygons on different layers, and not just the first vertex.

Costs. In the basic version of the 3DRP, we consider the cost of the edges as the geometric distance between two vertices and the time to travel the edge as directly linear to the distance. Considering a higher level of detail, we can observe that the time of traveling an edge is not necessarily linear and that it can depend on an acceleration at first, a constant speed in the middle, and a deceleration of the nozzle at the end when traveling the edge. Moreover, the deceleration can depend on the angle between the current edge and the following one visited and the acceleration can depend on the angle between the current edge and the previous one. This can lead to different options. Taking into account the acceleration and deceleration needed to obtain the maximum possible speed on a given edge, we can recompute the cost of each edge starting from the distance. If we considering the angles between the previous and following edges, then the problem must be considered as a directed one. Other possibilities can include non-linear distances between the vertices or considering the speed of the nozzle as a variable.

Other Features.

1. *Layers with different thickness.* Sometimes, to deal with structural problems or just to avoid to obtain a texture too subtle inside the printed object, we can print some parts of the same layer with a different thickness. Two different thicknesses must be related (one must be a multiple of the other).

2. *Supporting material.* Some 3D objects can have undercuts, thus a supporting material is needed when printing. In order to minimize the supporting material, the positioning of the object can be a variable to account for.

3. *Filling.* In this work we consider the filling part of a certain polygon as an input, but to decide the filling, in terms of distances between the edges of the filling, their direction, and even a particular pattern can change the optimization of the print or be part of it.

4. *Colors.* Some printing objects can have different set of materials and/or colors to be used. Indeed we can see the supporting material part of this case. When printing an object made of different colors or materials we can decide to print first the polygons of one material and then the others by deciding a sequence. Otherwise, we can decide to mix the use of colors to make the printing time shorter. Then the number of nozzles becomes relevant: if we have more materials than nozzle we must consider the time spent in changing material and going back to the vertex 0 to clean the nozzle, otherwise we must consider the distance between the nozzle of one color and the others.

All these features have a strong impact on the speed of the print and on the quality of the surface, both very important characteristics in 3DP. We are convinced that they will be the focus of many future research works.

Bibliography

[1] S. Akella. Hot off the press: The technology behind 3d printing. *Professor Scarlatos HON 301*, 2012.

[2] R. D. Angel, W. L. Caudle, R. Noonan, and A. Whinston. Computer-assisted school bus scheduling. *Management Science*, 18:B279–B288, 1972.

[3] D. Bak. Rapid prototyping or rapid production? 3d printing processes move industry towards the latter. *Assembly Automation*, 23:340–345, 2003.

[4] E. J. Beltrami and L. D. Bodin. Networks and vehicle routing for municipal waste collection. *Networks*, 4:65–94, 1974.

[5] L. D. Bodin and S. J. Kursh. A computer-assisted system for the routing and scheduling of street sweepers. *Operations Research*, 26:525–537, 1978.

[6] T. Campbell, C. Williams, O. Ivanova, and B. Garrett. Could 3d printing change the world? *Technologies, Potential, and Implications of Additive Manufacturing. Washington, DC: Atlantic Council*, 2011.

[7] Á. Corberán and G. Laporte. *Arc Routing: Problems, Methods, and Applications*. SIAM, 2014.

[8] Á. Corberán, R. Martí, E. Martínez, and D. Soler. The rural postman problem on mixed graphs with turn penalties. *Computers & Operations Research*, 29:887–903, 2002.

[9] Á. Corberán, R. Martí, and A. Romero. Heuristics for the mixed rural postman problem. *Computers & Operations Research*, 27:183–203, 2000.

[10] Á. Corberán, I. Plana, and J. M. Sanchis. A branch & cut algorithm for the windy general routing problem and special cases. *Networks*, 49:245–257, 2007.

[11] Á. Corberán and C. Prins. Recent results on arc routing problems: An annotated bibliography. *Networks*, 56:50–69, 2010.

[12] Á. Corberán and J. M. Sanchis. A polyhedral approach to the rural postman problem. *European Journal of Operational Research*, 79:95–114, 1994.

[13] Á. Corberán and J. M. Sanchis. The general routing problem polyhedron: facets from the rpp and gtsp polyhedra. *European Journal of Operational Research*, 108:538–550, 1998.

[14] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, 1:269–271, 1959.

[15] M. Dror and A. Langevin. A generalized traveling salesman problem approach to the directed clustered rural postman problem. *Transportation Science*, 31:187–192, 1997.

[16] H. A. Eiselt, M. Gendreau, and G. Laporte. Arc routing problems, part ii: The rural postman problem. *Operations Research*, 43:399–414, 1995.

[17] E. Fernández, O. Meza, R. Garfinkel, and M. Ortega. On the undirected rural postman problem: Tight bounds based on a new formulation. *Operations Research*, 51:281–291, 2003.

[18] G. N. Frederickson. Approximation algorithms for some postman problems. *Journal of the ACM (JACM)*, 26:538–554, 1979.

[19] G. Ghiani and G. Improta. The laser-plotter beam routing problem. *Journal of the Operational Research Society*, 52:945–951, 2001.

[20] G. Ghiani, D. Laganà, and R. Musmanno. A constructive heuristic for the undirected rural postman problem. *Computers & operations research*, 33:3450–3457, 2006.

[21] G. Ghiani and G. Laporte. A branch-and-cut algorithm for the undirected rural postman problem. *Mathematical Programming*, 87:467–481, 2000.

[22] M. Grötschel, M. Jünger, and G. Reinelt. Optimal control of plotting and drilling machines: a case study. *Mathematical Methods of Operations Research*, 35:61–84, 1991.

[23] G. W. Groves and J. H. Van Vuuren. Efficient heuristics for the rural postman problem. *ORiON: The Journal of ORSSA*, 21:33–51, 2005.

[24] E. Haslam and J. R. Wright. Application of routing technologies to rural snow and ice control. *Transportation Research Record*, 1304, 1991.

[25] A. Hertz, G. Laporte, and P. N. Hugo. Improvement procedures for the undirected rural postman problem. *INFORMS Journal on Computing*, 11:53–62, 1999.

[26] K. Holmberg. Heuristics for the rural postman problem. *Computers & Operations Research*, 37:981–990, 2010.

[27] G. Laporte. Modeling and solving several classes of arc routing problems as traveling salesman problems. *Computers & operations research*, 24:1057–1061, 1997.

[28] A. N. Letchford and R. W. Eglese. The rural postman problem with deadline classes. *European Journal of Operational Research*, 105:390–400, 1998.

[29] L. S. Levy. *The walking line of travel problem: An application of arc routing and partitioning.* PhD thesis, University of Maryland, College Park, 1987.

[30] L. M. Moreira, J. F. Oliveira, A. M. Gomes, and J. S. Ferreira. Heuristics for a dynamic rural postman problem. *Computers & operations research*, 34:3281–3294, 2007.

[31] G. Reinelt and D. O. Theis. A note on the undirected rural postman problem polytope. *Mathematical programming*, 106:447–452, 2006.

[32] H. I. Stern and M. Dror. Routing electric meter readers. *Computers & Operations Research*, 6:209–223, 1979.

[33] Ultimaker. "Cura Software". `https://ultimaker.com/en/products/software`.

Conclusions

In this PhD thesis we tackled severl real-world logistics and routing problems. In Part II we considered a tactical problem that arises in bike sharing systems, the Bike sharing Rebalancing Problem (BRP). The BRP is the problem of driving a fleet of capacitated vehicles to redistribute bicycles in bike sharing systems at minimum costs. In Chapter 2 we defined the BRP and modeled it proposing four formulations and several inequalities to be embedded in branch-and-cut algorithms. We evaluated the formulations on a test bed of 60 newly collected real-world instances with up to 116 vertices. The best formulation could efficiently solve to optimality all instances with up to 50 vertices, and obtain small gaps for the larger instances in about 20 minutes. In Chapter 3 we proposed a destroy and repair algorithm to solve the BRP, including a new constructive heuristic and a set of local searches made more efficient by dedicated speed-up techniques. We collected new real-world instances with up to more than 500 vertices. With our metaheuristic algorithm we could obtain all the optimal solutions obtained thanks to the previously presented branch-and-cut in very short times and reach very good solutions for the large instances. In the same chapter we also solve the generalization of the BRP where a maximum duration constraint is imposed to each route, the so-called one-commodity Pickup and Delivery Vehicle Routing Problem (1-PDVRP). We adapted both the branch-and-cut, including new inequalities, and the metaheuristic algorithms developed for the BRP to solve the 1-PDVRP. To test our algorithms for the 1-PDVRP we used literature instances. For the smaller instances an optimal value has been found for the first time with our branch-and-cut and the adaptation of the destroy and repair algorithm to solve the 1-PDVRP could improve the solution values of about 12% in average with respect to literature results. In Chapter 4 we defined the BRP with stochastic demand. We proposed some stochastic programming formulations and solved them by means of L-Shaped and branch-and-cut algorithms. We compared the different formulations on real-world newly collected instances. All instances with up to 30 vertices have been solved to optimality in short computing times. We also developed two heuristics algorithms to solve the SBRP. Those algorithms include a novel consideration on positive and negative correlations between the vertices. In Part III we solved earthwork optimization problems

considering some new features for the first time. In Chapter 5 we propose a novel two-phase approach to solve the earthwork optimization problem that arises in highway construction. In particular we developed this new method and the Decision Support System (DSS) presented in Chapter {ch:II2 thanks to the collaboration with one of the major construction companies in Europe (Strabag AG) for the construction of the Autostrada Pedemontana Lombarda, a highway in Northern Italy. The DSS includes the optimization part and the graphical component to ease the decision making. The DSS is currently in use at Strabag AG, being an easy-to-use, efficient tool suitable for solving several real-world construction logistics problems. In Part IV we summarized the 3D printing process, its advantages and disadvantage. We focused on the problems that arise in the 3D printing process, giving a list of possible optimization issues and constraints. We then concentrated in solving the generalization of the arc routing problem related to the definition of the tool path in 3D printing, that we called the 3D printing Routing Problem (3DRP). We proposed an ILP model and four heuristic algorithms. We tested the proposed algorithms on a set of 30 instances and compared the results with respect to those obtained by one of the most famous available software for the tool path definition: Cura. We could improve the solution values, in average, between about 3% to about 8%, in average, within good computing times.