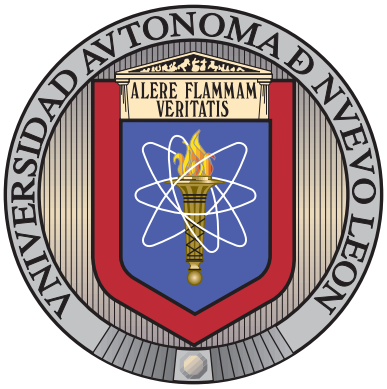
Universidad Autónoma de Nuevo León

Facultad de Ingeniería Mecánica y Eléctrica

División de Estudios de Posgrado



Models and algorithms for transit network planning

por

Omar Jorge Ibarra Rojas

en opción al grado de

Doctor en Ingeniería

con especialidad en Ingeniería de Sistemas

San Nicolás de los Garza, Nuevo León diciembre 2012

Universidad Autónoma de Nuevo León

Facultad de Ingeniería Mecánica y Eléctrica

División de Estudios de Posgrado

Models and algorithms for transit network planning

por

Omar Jorge Ibarra Rojas

en opción al grado de

Doctor en Ingeniería

con especialidad en Ingeniería de Sistemas

San Nicolás de los Garza, Nuevo León diciembre 2012

Universidad Autónoma de Nuevo León

Facultad de Ingeniería Mecánica y Eléctrica

División de Estudios de Posgrado

Los miembros del Comité de Tesis recomendamos que la Tesis "Models and algorithms for transit network planning", realizada por el alumno Omar Jorge Ibarra Rojas, con número de matrícula 1057351, sea aceptada para su defensa como opción al grado de Doctor en Ingeniería con especialidad en Ingeniería de Sistemas.

El Comité de Tesis

Dra. Yasmín Ríos Solís
Directora

Dra. Ada Álvarez Socarrás
Revisor

Dr. Fernando López Irarragorri
Revisor

Dr. Pierre Fouilhoux
Revisor

Dra. Safia Kedad-Sidhoum
Revisor

Vo. Bo.

Dr. Moisés Hinojosa Rivera
División de Estudios de Posgrado

San Nicolás de los Garza, Nuevo León, diciembre 2012

Dedicatory

To everyone.

Contents

Acknowledges

To my family and friends for their support.

To Yasmín Ríos for giving me the freedom and opportunity to travel this road with at my own pace.

To Ada, Fernando, Pierre and Safia for their important observations to make this a better project.

Abstract

The efficiency of urban transport system strongly relies on the transit network planning process that is commonly complex because of the number of related decisions. Usually this process is divided into several subproblems such as line planning, timetable generation, vehicle scheduling, and crew scheduling which are solved sequentially to obtain an entire solution. In this study we focus on timetabling problems based on the transit network of Monterrey, Mexico. The characteristics present in this network lead to the Synchronization Bus Timetabling Problem (SBT) with the objective of maximize the number of synchronization events between different lines to allow well timed passenger transfers and avoid bus bunching. We design a mixed integer linear programming formulation (MILP) for SBT considering constraints such as bounds of separation times between consecutive trips and departure time dispersion along the planning period. We prove that SBT is NP-hard. Nevertheless, the mathematical structure of SBT MILP allow to define a preprocessing stage based on constraint propagation leading to the elimination of a high percentage of decision variables and constraints.

Moreover, we define several families of valid inequalities to strength the SBT MILP by implementing the ideas of the preprocessing stage. The numerical results show that we practically solved our SBT adding our proposed valid inequalities at the MILP and using the linear solver of CPLEX 12.3, since we obtain solution with less than 3% of relative gap in seconds.

Although we solve SBT, we go a step forward to define the Multiperiod Synchronization Bus Timetabling (MSBT). This problem generates a timetable for the entire day considering synchronization of trips belonging to different planning periods and smooth

transitions between these periods. Since the implementation of valid inequalities do not obtain high quality solutions for instances of MSBT in short time, we design several Iterated Local Searches and Variable Neighborhood Searches algorithms to solve MSBT. The numerical results show that these algorithms are comparable with the exact approach for SBT. Moreover, these algorithms obtain solutions with less than 5% and 13% of mean relative gap for small and large instances of MSBT, respectively.

Chapter 1

Introduction

Summary: One of the reasons to work in this project is the need of an attractive, or at least, acceptable urban transport system. In this chapter we exhibit the social impact of an efficient transit network planning which is the context of our study. Moreover, we introduce the characteristics of the transit network we focus on. All the presented information is used to clarify the objective, justification, and contributions of our work.

"Sweet ride."

1.1 Urban transport service

A critical aspect for every city in the world is to obtain an efficient urban transport service which is a difficult task. Indeed, it relies in several facets such as planning process, equipment technology, urban design, and government policies. However, improving the urban transport system leads to important benefits such as users satisfaction and reduction of undesirable elements such as pollution, traffic congestion, and costs.

The efficiency of urban transportation is related with the operation of the system which can be improved by an accurate planning process. Indeed, the context of this study is the planning process to improve the behavior of transit networks using operation research techniques. The entire planning problem of a transit network is difficult and it is often divided in several subproblems solved in a sequential approach [Ceder, 2007, Desaulniers and Hickman, 2007].

The first subproblem of the entire transit network planning is the line planning problem that defines the routes, stops, and the frequency for each bus line in a specific territory. Then, the timetable generation determines the departure times (and arrival times) of all the trips of the lines to achieve a quality service level such as maximum synchronization to allow transfer events and avoid bus bunching events for different lines. We mostly focus on this subproblem in this study. Third subproblem is the vehicle scheduling problem that assigns vehicles to sets of trips of each bus line with the aim of minimizing vehicle costs. Finally, the crew scheduling problem defines tasks assigned to drivers subject to work regulation constraints such as limit working time, guarantee lunch time for drivers, and minimum working time with the objective of minimizing crew costs. The first two subproblems are part of the strategical planning (increasing quality service) while the last two are part of the operational planning (reducing costs).

Ceder [2007] presents a detailed scheme to remark the interaction between subproblems of transit network planning (Figure 1.1). We can notice that there are many subsequent subproblems depending on the solutions of timetabling. Therefore, efficient procedures to solve this subproblem are of great importance. Particularly, in this study we identify accurate timetabling problems and design efficient solution algorithms to obtain timetables of high quality in reasonable time.

The different characteristics in transit networks lead to a large number of different timetabling problems. In the next section we present the details of the transit network we focus on. Then, we define an accurate timetabling problem for our case study.

1.2 Problem statement

Private companies handle the transit network we base our study on. In this network, competition to cover a large demand of passengers is always present. This characteristic leads to a large network where different lines, even of the same company, share route segments causing considerable traffic congestion (left panel of Figure 1.2). Another important characteristic is that many bus lines pass across downtown defining a centralized network.

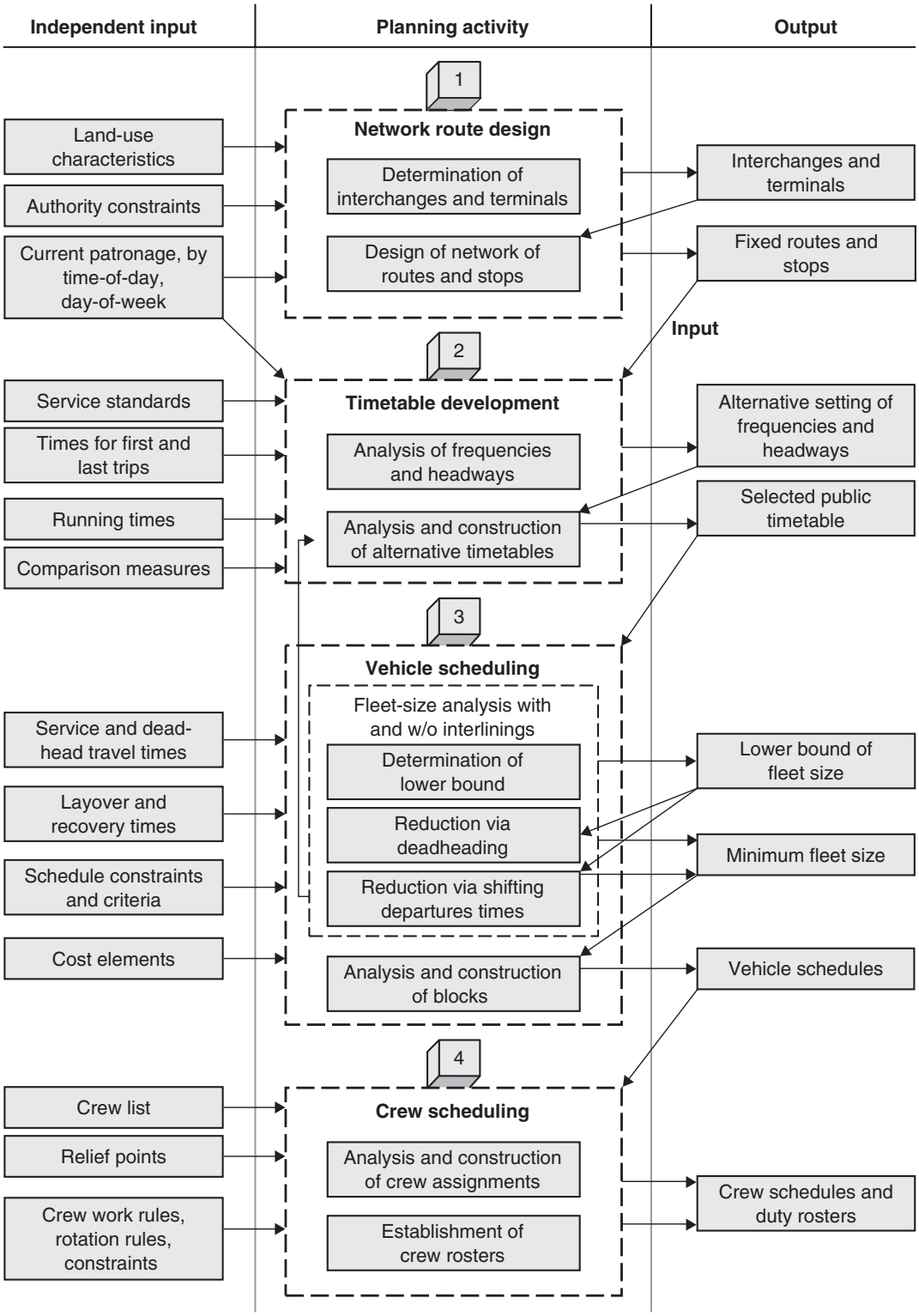


Figure 1.1: Subproblems of the transit network planning problem and their interaction [Ceder, 2007].

For example, there are more than 60 bus lines that cross by Monterrey's downtown (right panel of Figure 1.2).

Figure 1.2: The left panel shows traffic congestion caused by bus lines sharing an avenue in downtown [Centro de Desarrollo Metropolitano y Territorial, 2012]. The right panel shows the bus network of Monterrey's metropolitan area. The marked area shows the line concentration downtown [Rabut, 2010].

We are interested in the interaction between the different bus lines in the transit network to allow well timed passenger transfers and avoid bus bunching. Particularly, we define a synchronization event as the arrivals of two trips at a common stop (called node) with a specific separation time. Uncertainties present in travel times are a handicap for punctual synchronization. Therefore, we allow to the separation time to be within a specific time window to define a flexible synchronization. Although this situation models passenger transfers from one line to another, as it can be seen in the right panel of Figure 1.3, it can also model an accurate separation time between the arrivals of two trips to avoid bus bunching at specific nodes, as in the left panel of the Figure 1.3. In the case of transfer nodes we seek a small separation time while a larger separation time is needed at bus bunching nodes. Then, our Synchronization Bus Timetabling problem (SBT) is *to determine regular departure times for all trips maximizing the number of synchronizations for a given planning period.*

Timetabling problem we study is relatively new since to the best of our knowledge, synchronization between different bus lines with uneven headways in multiple nodes of the transit network has been considered only in a few studies in the 2000's. Moreover,

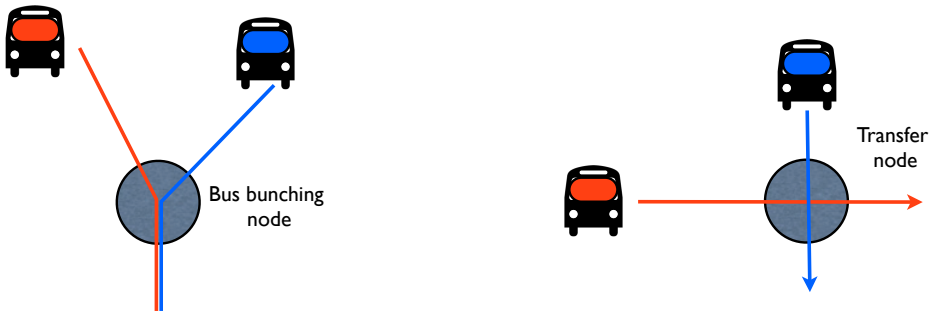


Figure 1.3: Synchronization nodes on the bus network. The left panel represents a bunching node while the right panel represents a passenger transfer node.

we redefine the synchronization events to represent two cases: passengers transfers and bus bunching between different lines. As we can see in Chapter 2 previous similar studies remark on the intractability of timetabling problems with these characteristics. Therefore, efficient methodologies obtaining high quality timetables are needed since the entire planning problem is often solved by sequential approach, i.e., subsequent subproblems like vehicle and crew scheduling depend on the quality and efficiency of the timetable generation process.

1.3 Objectives

Our main goal is to design efficient procedures based on operations research concepts to generate high quality timetables for transit networks with characteristics like the ones in Monterrey, Mexico. To achieve this main goal we need to achieve particular objectives which are the following.

- The first step is to identify issues in the timetable generation for our case of study that can be handled by operations research methodologies. Particularly, we focus on the maximization of synchronization events to allow well timed passenger transfers and avoid bus bunching between different lines. Passenger transfers is a common objective for many timetabling problems while bus bunching events between different lines is rarely taken into account. High quality timetables are needed by staff agencies to provide a more efficient urban transport system that may obtain more

passengers demand.

- Another particular objective is to define an accurate timetabling formulation to model the characteristics of our case study. Since we want to maximize passenger transfers and minimize bus bunching events it is difficult to define an objective function to model both cases. Thus, we cannot use common approaches such as the minimization of passengers waiting times because we need the separation between trips at bus bunching nodes. We cannot use headway harmonization policies to avoid bus bunching since we need the flexibility of uneven headways to get more well timed transfers. Then, auxiliary decisions to model synchronization events are needed.

- Once we define an accurate formulation, it is necessary to design efficient algorithms to solve it. As we mentioned before, the required formulation must define auxiliary decisions to model synchronization events but these events depend entirely of departure time decisions. As we show in Chapter 3, this characteristic leads to a large feasible space for departure times decisions where different combinations of these variables obtain the same value of the objective function. Therefore, exact approaches like the Branch and Bound algorithm is not capable to solve the smallest instances of our problem. Moreover, in Chapter 5 we show that there are not obvious efficient heuristic algorithms to obtain high quality feasible solutions.

- To show that our proposed formulations and solution algorithms can be used in real life instances is necessary to design an experimental stage to measure the results obtained by our solution methodologies.

As we show along this dissertation, we have a new timetabling formulation with characteristics that makes difficult to obtain even high quality bounds and feasible solution for large instances. However, the mathematical structure allow to define several procedures to design efficient exact and heuristics approaches.

1.4 Justification

The urban transport system of Monterrey, Mexico,an be enhanced with strategic planning methodologies scientifically supported. Commonly, in this kind of network, the planning process is done manually by agencies staff and focuses only on feasibility and operation costs rather than quality service, as we propose in this study. Considering all interactions in the network, transport planning becomes an almost impossible task to perform in a reasonable time even by the most experienced staff member of a transport company. Therefore, well defined methodologies to improve the quality of the service maintaining acceptable operational cost are quite useful. Moreover, since our methodology improves the service quality, the use of bus lines instead of private vehicles is promoted which represents a big impact in saving energy consumption and reducing pollution levels.

1.5 Scientific contributions

Our case of study has different characteristics than the ones studied in literature. Therefore, we generate several scientific contributions such as the following.

- Two new formulations for timetabling problems with the objective of maximizing the number of well timed transfers and minimizing the bus bunching events.
- Answer an open questions in the transit planning research area: we prove the complexity of our timetabling problems and others similar present in literature.
- Obtain exact solutions for large instances of timetabling problems is a common issue. We define several families of valid inequalities for our proposed formulation that allow to find optimal solutions for large instances of the problem.
- Several metaheuristic algorithms (based in the mathematical structure of our proposed formulations) are designed to obtain high quality solutions for timetabling problems considering multiple planning periods.
- The results of this study were presented in the following seminars and conferences.

  - Seminars of the Graduate Program of Systems Engineering, Universidad Autónoma de Nuevo León (UANL), 2010-2012.
  - Seminar of the Graduate Program of Industrial and Systems Engineering, Instituto Tecnológico y de Estudios Superiores de Monterrey (ITESM), September 2012.
  - ALIO-INFORMS Joint International Meeting. Buenos Aires, Argentina, June 2010.
  - II Encuentro Iberoamericano de Investigación Operativa y Ciencias Administrativas (IOCA), Monterrey, Mexico, July 2010.
  - Operational Research Peripatetic Postgraduate Programme (ORP^3), Cadiz, Spain, September 2011.
  - First Conference of the *Sociedad Mexicana de Investigación de Operaciones (SMIO)*, Guadalajara, Mexico, October 2012.

- We published results of this dissertation in *Ciencia UANL*, *Transportation Research Part B*, and *International Journal of Productions Economics*. Moreover, one article is under review in *Transportation Science* and another article is going to be submitted in *Computers & Operations Research*.

1.6 Dissertation structure

This dissertation includes several mathematical formulations for timetabling problems and different solution algorithms. Each chapter includes results published or submitted for publication. The content of these chapters is summarized in the following.

Chapter 2 presents the necessary elements to define a timetabling problem and details of timetabling problems present in literature. We remark on timetabling problems with characteristics present in our case study and their solution methodology.

Chapter 3 defines our timetabling problem for the transit network of Monterrey, Mexico. It is a large bus network where passenger transfers must be favored, almost

evenly spaced departures are sought, and bus bunching of different lines must be avoided. We formulate the timetabling problem of this network with the objective of maximizing the number of synchronizations to facilitate passenger transfers and avoid bus bunching along the network. We define these synchronizations as the arrivals of two trips with a separation time within a time window to make a flexible formulation. This flexibility is a critical aspect for the bus network, since travel times vary because of reasons such as driver speed, traffic congestion, and accidents. By proving that our problem is NP-hard we answer a 10-year-old open question about the complexity of similar problems present in literature. Next, we analyze the structural properties of the feasible solution space of our model. This analysis leads to a preprocessing stage that eliminates numerous decision variables and constraints. The results of this chapter are published in the journal *Transportation Research: Part B* [Ibarra-Rojas and Rios-Solis, 2012].

Chapter 4 is based on the fact that a timetabling solution that is not close to the optimum has strong repercussions in the vehicle and crew scheduling problems since a sequential resolution approaches are often required for solving the entire planning process. We focus on the Synchronization Bus Timetabling Problem (SBT). Based on the formulation of Chapter 3, we develop two classes of valid inequalities using combinatorial properties of SBT on the number of synchronizations. Additionally, we present two lifting procedures leading to new inequalities. Experimental results show that the enhanced MIP formulation yields high quality solutions using a small computational time. In particular, large instances based on real transit networks are solved within few minutes with a relative deviation from the optimal solution usually less than 1%. The results of this chapter were obtained during a research stage at the Laboratoire d' Informatique de Paris 6 (LIP6) and are going to be submitted to the journal *Computers & Operations Research.*

In **Chapter 5** we show that usual procedures for generating a bus timetable that covers a whole day are not suitable for our timetabling problem. Indeed, to compute independent timetables for each one of the planning periods of the day (e.g., morning rush hour, afternoon hours, night hours) and merge them leads to suboptimal solution since it does not consider that trips from different periods of the day could have a transfer o may bunch at some bus stop. Thus, we propose the Multiperiod Synchronization Bus

Timetabling problem that considers smooth transitions between periods and synchronization events between trips belonging to different planning periods. We design six Iterated Local Search and two Variable Neighborhood Search algorithms. The main contribution of this work is that the neighborhoods needed by the metaheuristics are based on feasible departure time windows (structure inherited from the mathematical model) that are applied by a constraint propagation methodology. We empirically obtain high quality feasible solutions for real size instances. The results of this chapter are submitted to the journal *Transportation Science.*

Chapter 6 presents the conclusions of our study and several future research areas.

In particular, the complete integration of two or more subproblems of transit network planning is an interesting research area addressed by a few studies in literature. **Appendix B** shows our preliminary results about complete integration approaches. On the one hand, we present the integration of a less flexible timetabling problem (MT) and the single type vehicle scheduling problem. Since quality service and operational cost are naturally in conflict, we propose a multiobjectvie approach to handle this integration. We define a biobjective formulation and design an ϵ-constraint method to solve considerable large instances of the problem. On the other hand, we present a multiobjective formulation for MT, vehicle scheduling and crew scheduling problems with particular characteristics of Monterrey's transit network such as working regulation constraints.

A background for this dissertation is presented in **Appendix 2.3**. It includes basic concepts, definitions, and algorithms that helps to a better understanding of this study.

Although the main topic of this dissertation are subproblems of transit network planning, we also study manufacturing planning problems. Particularly, **Appendix C** addresses a real manufacturing process of pieces that are produced with molds that are mounted on machines. The characteristics of the system include setup times between jobs, dedicated parallel machines, dedicated molds, and a different production rate for each piece-mold pair. There is a demand for each type of piece, and when the company fails to meet this demand, is often forced to buy pieces from other companies to avoid loss of customers. We describe the system with a new integer quadratically constrained

programming model. Our proposed formulation improves others in literature as we do not force a mold to be mounted on a single machine, which is a more realistic description of the production process itself. To solve the problem we decompose the formulation into two subproblems: one that solves the lot-sizing of the products and another one that verifies if there is a feasible schedule for the solution of the first subproblem. This methodology is empirically tested, demonstrating its effectiveness on real size instances. Moreover, it reveals that the counter intuitive case where a mold visits more than one machine happens more often than expected. The results presented in this appendix are published in the journals *Ciencia UANL* [Chacón-Mondragón et al., 2012] and *International Journal of Production Economics* [Ibarra-Rojas et al., 2011].

Chapter 2

State of the art

Summary: We have been working more than three years in this exciting research area and we found literature about many interesting studies considering different characteristics of urban transport systems. Based on our literature review, we present the key elements that must be considered to define a Timetabling Problem. Once this elements are understood, it is easier to characterize most of the timetabling problems present in literature.

"Thanks to Ada, Fernando, Paulina, and Yasmin for joining me in the insight into other studies."

2.1 Timetabling problems

Timetabling problems determine the specific time in which a set of events occur to achieve a specific goal. In the context of transit network planning, these events are the departure and arrival of bus lines along the network. Figure 2.1 represents an example of a timetable for bus line a. Rows represent the trips of line a while the columns show the arrival and departure times at each stop for these trips. Column "depot" exhibits that departures from the depot occur between 8:00 am and 10:00 am, i.e., there is a planning period of two hours. The number of trips considered for this planning period, called frequency, is 12. We also remark that the separation time between consecutive trips, called headway, is 10 minutes. Often, passengers have access to this kind of timetable. In other cases, this timetable can be perceived by the passengers as regular services since buses should pass at each stop every ten minutes.

line a	depot	stop 1		...	stop 20	
	departure	arrival	departure		arrival	departure
trip 1	8:00	8:07	8:08	...	9:13	9:15
trip 2	8:10	8:17	8:18	...	9:23	9:25
⋮	⋮	⋮	⋮	⋱	⋮	⋮
trip 12	10:00	10:07	10:08	...	11:13	11:15

Figure 2.1: Timetable in a planning period of two hours (8:00 am to 10:00 am) for a line a with a frequency of 12 trips, headway of 10 minutes, and 20 stops in its route.

Timetabling is a critical aspect to quality service for passengers. As it is mentioned in Ceder [2007], inadequate and/or inaccurate timetables not only confuse the passengers but also reinforce the bad image of public transit as a whole. Therefore, efficient procedures to generate timetables fitting the passengers requirements are needed.

For the point of view of operations research, the main decisions are the departure and arrival times of the bus lines along the transit network. There are as many timetabling formulations as different transit systems. However, the difference between other decisions and constraints depend on the timetable structure. In the following, we clarify the elements used to define the timetable structure.

2.2 Timetable structure

In Section 2.3, we present a detailed literature review. In this section, we point out that there is a large number of different timetabling problems. But these differences rely on the type of decisions and constraints arising due to the following elements of the timetable structure.

2.2.1 Headway policy

Based in the kind of information about the demand along the transit network, the two main approaches for timetabling problems are even headways and even loads (passenger activity based). On the one hand, timetables with even headways are regular trip services and simple to memorize. This policy assumes that the passengers demand is adjusted

to the timetable and not vice verse. It can be implemented when there is not enough information about the demand along the transit network and also if passengers do not have access to the timetable but they only known an estimate of waiting time to board into a bus [Bookbinder and Désilets, 1992, Cevallos and Zhao, 2006, Jansen and Nielsen, 2002, Ibarra-Rojas and Rios-Solis, 2012]. On the other hand, timetabling with even loads can be implemented when accurate information or efficient forecasting procedures of the demand behavior are at hand [Ceder, 2007]. Indeed, its efficiency relies on the quality of the information about the passenger demand.

The main disadvantage for timetables with even headways is the demand variability. Figure 2.2 exhibits this case for a single bus line, horizontal axis represents the departure times and vertical axis represents the accumulated demand in a specific stop. We can remark that the arrival times at each stop leads to an almost total lack of balanced loads. For example, the second trip arriving at 7:30am has a passengers load of $43 - 15 = 28$ while fifth trip arriving at 8:15 has a passengers load of $90 - 85 = 5$.

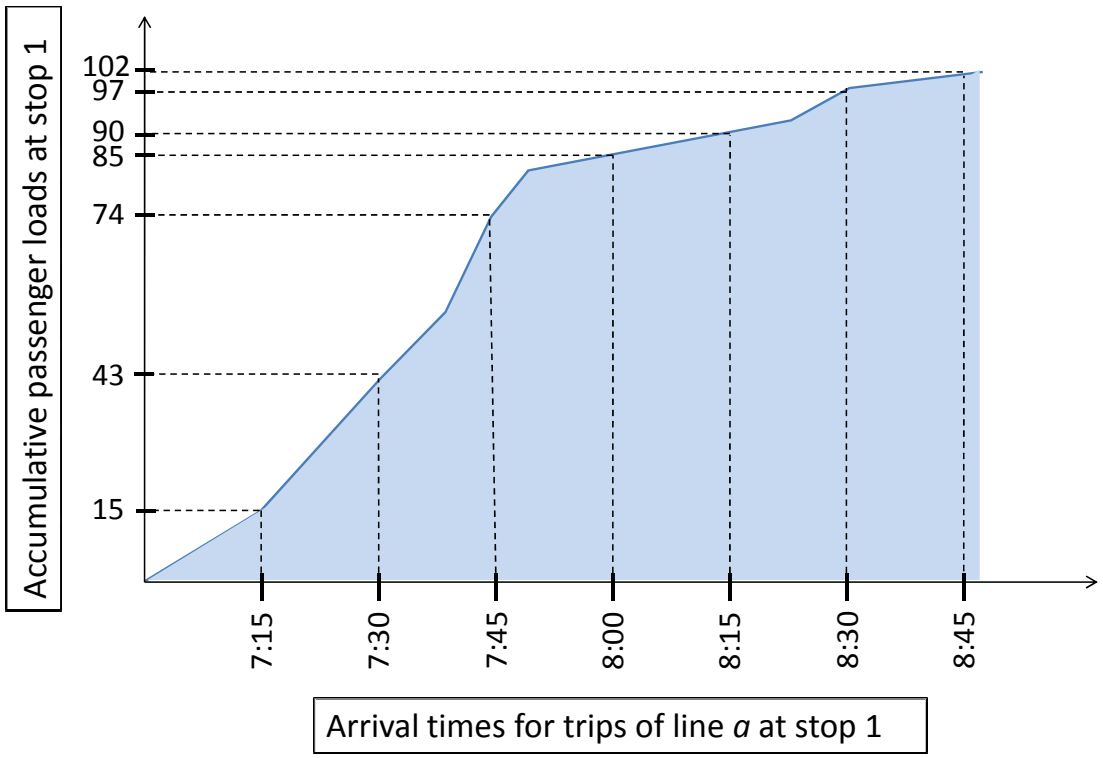


Figure 2.2: Departure times versus accumulated passenger demand at some stop for a bus line.

Although uncertainties in passenger demand and travel times are handicaps for the implementation of timetabling with even headways, an entire day could be divided in smaller planning periods with less variability in passengers demand and travel times. In the example presented in Figure 2.2, a planning period could be from 8:00 am to 9:00 am since a more accurate estimation for passenger demand and/or deterministic travel times can be computed. Indeed, only a few timetabling problems consider an entire day while a common assumption is the existence of smaller planning periods with less variability in the transit network "behavior". Then, the solution of these timetabling problems are combined to generate a timetable for the entire day.

Another element based on demand information is the number of trips for each bus line to fulfill passenger demand, called frequency. It can be considered as a parameter or as a decision variable. For example, we can define the frequency as a parameter previously computed (by software or staff analysis) to satisfy an assumed constant demand in a planning period [Ceder and Tal, 2001, Ceder et al., 2001, Adamski, 1993, Eranki, 2004]. On the other hand, if we have information about the demand behavior along the time and we have a policy of maximum load for each bus, we can define the frequency of each line as a decision variable [Ceder, 2007, Ávila and López, 2012]. We consider the first case in this study, i.e., the agencies staff determines a frequency considering a policy of regular services and constant demand in small planning periods.

2.2.2 Flexibility

Suppose that there is a passenger that uses a bus line because he thinks that it provides regular services as, for example, every 10 minutes but he really does not know the exact timetable. However, the departure time of trips in the timetable are separated by an amount of time between 8.5 and 11.5 minutes. If the variation in the regularity of the service is considerable small, is really difficult that this passenger loses his idea of regular services provided by the bus line. Then, the passenger will be using the same bus line. However, even allow small variations in the separation times of consecutive trips could be used to achieve other criteria such as reduce number of vehicles, keep a crew scheduling,

allow passenger transfers, and avoid bus bunching.

The flexibility could be defined in different ways but a common approach is to consider headway bounds instead of an unique value [Ceder and Tal, 2001, Ceder et al., 2001, Eranki, 2004, Wong et al., 2008, Wong and Leung, 2004]. For example, assuming there is an unique headway h for some bus line. If we determine a value for the departure time of the first trip denoted as X_1 the departure time of the second trip is automatically determined as X_1+h, the third trip departs at X_1+2h, and so on. On the other hand, flexibility given by headway bounds allow to decide departure times for each trips respecting the headway bounds to provide a regular service or minimum service requirement. Then, each trip p must satisfy the following constraint.

$$lower\_headway \leq X_p - X_{p-1} \leq upper\_headway \tag{2.1}$$

Above panel in Figure 2.3 shows an example of departure times considering an even headway of 10 minutes while below panel shows an example of "almost" evenly spaced departure times considering headway lower and upper bounds of 8 minutes and 12 minutes, respectively.

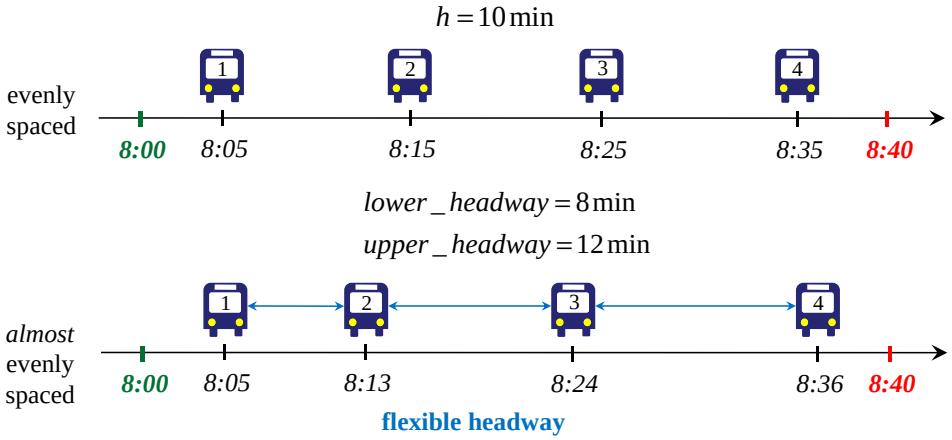


Figure 2.3: Example of departure times with an even headway of 10 minutes and headway lower and upper bounds of 8 minutes and 12 minutes, respectively.

Obviously, headway bounds should be carefully determined since the quality service depends on their values. For example, a headway lower bound of 0 minutes lack of sense

if we want to avoid bus bunching while large values for headway upper bounds may lead to long intervals within the planning period without trip departures. In our case of study, we use a relative deviation from the even headway to define the corresponding bounds.

Analogously to the flexibility, there are cases where we have more restricted timetabling problems. In the following section we clarify on how we can restrict the feasible set of timetabling problems considering other stages of transit network planning.

2.2.3 Previous planning solutions

In many cases, timetabling problems consider planning solutions previously obtained [Chakroborty et al., 1997, 1995, Guihaire and Hao, 2010a,b]. This commonly happens to reduce the impact of a completely different timetable for the transit network that may cause recalculation of solutions for other subproblems of the planning process such as vehicle and crew scheduling. For example, some timetabling formulation consider an initial timetable as a reference. Then, the deviation between the departure times obtained by solving the timetabling problem and the departure times of the initial solution must be bounded, i.e.,

$$lower\_deviation \leq X_p - \overline{X}_p \leq upper\_deviation \tag{2.2}$$

where $\overline{X}_p$ is the departure time of trip p in the initial solution. In some cases, this deviation is used as objective function [Kwan and Chang, 2008].

Another bounds and constraints could arise if we are restricted to other elements of the planning process such as maintain the fleet size, a vehicle schedule or/and a crew schedule. For example, assume that a vehicle schedule must be maintained. Then, for every two trips p and p' such that p' is just finishing trip p by the same vehicle, the difference between the departure times of trip p' and trip p, i.e., $X_{p'} - X_p$ must be greater or equal that the total travel time of trip p plus the time required for the vehicle to get ready for the next trip p'.

2.2.4 Travel times nature

As we mentioned before, the main decisions in timetabling problems are the departure and arrival times of bus lines along the transit network. Nevertheless, in some cases is not possible to slow down or speed up a bus to arrive at some stop at a desired time. For example, if a city has high levels of traffic congestion, lack of holding stops, monitoring tools, or space limitations, it is almost impossible to control the travel times of the bus lines by speed variations. Therefore, the only decision variables are the departure times from the depot for all trips while arrival times at each stop are entirely determined by the transit network behavior. In this cases, it is useful to have small planning periods where accurate estimations of travel times and passenger demand can be computed. In the case where it is possible to control the speed of a bus or stop it in a specific node, the decision variables are not only departures from the depot but also departure and arrivals at each stop of the bus line [Hall et al., 2001, Dessouky et al., 1999].

To define the timetable structure, we should determine the type of the elements such as headway policy, frequency nature, possible flexibility, consideration of previous planning solutions, and travel times nature. For example, our timetabling problem consider headway bounds to make it flexible, a given frequency to fulfill demand, and short planning periods considering deterministic travel times. Once the timetable structure is identified, an objective function should be considered. We present some common objective functions in the next section.

2.2.5 Objectives of timetabling problems

Common objectives for timetabling problems are to satisfy headway policies, satisfy passenger demand and minimize load unevenness, headway unevenness, travel times, or passengers waiting times [Guihaire and Hao, 2008b]. However, more interesting and complex objectives are related with the interaction between different lines in the transit network. As it is mentioned in Desaulniers and Hickman [2007] “perhaps the most pressing challenge in timetabling is the synchronization of vehicle timetables so that transfers within

the network are well timed. Specifically, one would like to time the arrival of a vehicle on one line with that on another line so that passengers transferring between lines could make the connection with the minimum waiting time. Much of the early work on this problem focused on methods for synchronization at a single timepoint; more recent methods have used heuristics for larger network problems. However, the combinatorial nature of the problem indicates that it is NP-hard, and the computational issues of exact solutions are still vexing."

In this study we address the two concerns mentioned in Desaulniers and Hickman [2007]: complexity of timetables with the objective of synchronize different bus lines in the network and efficient procedures to obtain optimal solution for large instances. On the one hand, Chapter 3 present our timetabling problem with the objective of maximize the synchronization between different lines to allow well timed transfers and avoid bus bunching events. The timetable structure is determined by a given frequency for each line, flexibility given by headway bounds, no consideration of planning solutions previously obtained, and deterministic travel times for small planning periods. Moreover, we prove that our problem is NP-hard along with other similar problems present in literature. On the other hand, Chapter 4 present several families of valid inequalities that strength our formulation to obtain optimal solutions for large instances with a reasonable amount of time.

We present a literature review related with characteristics of our case study. To achieve this, we complement an excellent literature review of timetabling problems for bus transit networks [Guihaire and Hao, 2008b].

2.3 Literature review

Passenger transfer is crucial in public transit systems and it is present in all of them. Zhao and Zeng [2008] present an overview of how transfers are affected by the network structure, frequency of the lines, and other elements. In several studies [Bookbinder and Désilets, 1992, Cevallos and Zhao, 2006, Chakroborty et al., 1997, 1995, Daduna and Voß,

1995, Jansen and Nielsen, 2002, Wong et al., 2008, Zhao and Zeng, 2008] the authors seek to minimize the overall transfer waiting times and consider evenly spaced departure times. However, the minimization of overall waiting time could not be a representative objective in our study since we also consider bus bunching events. Schröder and Solchenbach [2006] remark on the necessity of better measurement of the transfer function, since minimizing waiting times may lead to risky passenger transfers due common delays in bus arrivals at stops. They make a formulation to minimize the cost of different kinds of transfers using time windows. Ceder and Tal [2001] address the problem of synchronizing arrivals of different lines at some nodes to achieve more efficient passenger transfer. They define synchronization as the simultaneous arrival of two buses and consider limits for consecutive trips' separation times (referred to as headway times) and frequency (number of trips for each line) as given. The objective of the problem is to maximize the number of simultaneous arrivals. The authors note that their problem seems intractable since solving small instances lead to large computational times. They design constructive heuristic procedures to generate timetables. In Ceder et al. [2001], the mathematical formulation of the previous problem is presented. In addition, they show the heuristic implementation on a real case study, where they found feasible solutions within seconds.

One extension of Ceder et al. [2001] is presented in Eranki [2004], where synchronization is redefined as the arrival of two trips at a synchronization node with a separation time within a small time window instead of punctual synchronization (simultaneous arrivals). Eranki designs a constructive heuristic algorithm on the basis of Ceder's algorithm to solve her problem. In Chapter 3 we propose a formulation that is based on the one developed in Eranki [2004], but we enhanced several of its components to define a more realistic and accurate representation of our case study. The complexity proof for problems of Ceder and Tal [2001], Ceder et al. [2001], and Eranki [2004] is presented in Chapter 3. Liu et al. [2007] address the synchronization timetabling problem based on the formulation presented in Ceder et al. [2001]. They define an objective function for simultaneous arrivals. The function consists of a ratio of the number of lines where there are vehicles arriving simultaneously at a connection stop to the number of all lines passed at the same stop. Another approach to achieve accurate passengers transfers is Hall et al. [2001] where

a model for holding times to ensure passenger transfers is presented. We remark that we are not able to implement this last approach in our problem since our case of study have only a few potential stops where buses can wait. Nevertheless, considering bus holding times would be an interesting research area.

The bus bunching problem has been addressed by Daganzo [2009], where the author present a dynamic method based on real-time data to generate holding times for bus runs to avoid bunching. Adamski [1993] addresses the synchronization problem of lines that share common route segments. He presents a mathematical formulation to harmonize headway times of different lines to separate trips in a planning period. He considers possible departure times as given and solves the problem with an integrated tabu search and genetic approach using small computational resources. This is different from our case study since it might be difficult to harmonize headway times for bus lines of different companies. In addition, he considers evenly spaced departures, i.e., an unique headway for each line. However, in Chapter 5 we present a similar concept which is harmonize arrivals at bus bunching nodes which is caused by maximizing the number of synchronizations of our proposed timetabling problem.

Strategic planning is also based on passenger behavior (not constant demand) and real-time (travel time) data. Li et al. [2010] study the transit scheduling problem to optimize the interaction of different services at an intermodal transport network considering demand variability and therefore variable headways. They use a heuristic solution algorithm that combines the Hooke-Jeeves method and an iterative supply-demand equilibrium approach to solve their model. Dessouky et al. [1999] address the minimization of transfer delays: they make a probabilistic model based on real-time data for delays, and then determine the departure times to achieve their objective. Chung [1990] addresses a travel time forecasting process in the bus network for peak and non-peak hours. On the one hand, the author solves the simultaneous synchronization problem with the objective of minimizing overall waiting time using genetic algorithms. On the other hand, the author presents a real-time control strategy to maintain transfers against unexpected delays. Finally, Liebchen and Stiller [2009] present also a procedure to obtain timetables resistant to delays for periodic and aperiodic cases in railway systems. In our case of

study we do not consider a constant behavior of the network along the entire day. We divide each day in planning periods such as peak hour (high demand periods) and valley hour (less demand periods). In Chapter 3 we define a single period timetabling problem while Chapter 5 presents a multiperiod timetabling problem to generate a timetable for the entire day.

Although the previous studies consider a single objective, multiobjective approaches are also present in timetabling problems. For example, Kwan and Chang [2008] present a biobjetive approach to a timetabling problem considering both objectives; the cost based on the number of transfers and the cost caused by the deviation from an initial timetabling. The authors implement a NSGAII and other metaheuristic algorithms to solve their proposed problem. Another example is presented in Ávila and López [2012], where the authors formulate a problem to determine the frequency and the timetabling considering different objective function such as maximize number of synchronizations, minimize uneven loads and transfer times. Moreover they generate several metaheuristic algorithms to find efficient solutions.

Integration of subproblems of transit network planning is an important research. van den Heuvel et al. [2008] address the integration of the clock-face timetabling problem (trips depart at regular intervals, and thus at the same number of minutes past each hour) and multiple-depot multiple-vehicle-type scheduling problem. They design a tabu search to solve their proposed integration. Guihaire and Hao [2008a] also implement a similar idea. Their timetabling problem consists of minimizing the overall waiting time considering a given frequency and even headways. They design an iterated local search for solving their formulation. Fleurent and Lessard [2009] propose a measure function for transfers based on ideal waiting times. They design an optimization approach to minimize vehicle costs such as the number of vehicles and unproductive time. Fleurent et al. [2005] propose an integral formulation for timetabling and vehicle scheduling problems that considers weights on the objective function. However, these weights reflect the planner's necessity, which is a complex characteristic if two or more objectives are in conflict. Guihaire and Hao [2010a] address the integration between timetabling problem and vehicle assignment to optimize quality and service and vehicle costs forcing to respect a deviation

form an initial timetabling. The authors implement an iterated local search algorithm in instances of a real transit system in France. Later, Guihaire and Hao [2010b] present a timetabling problem to maximize the number and quality of transfers considering vehicle constraints such as maintain a initial vehicle schedule and respect a deviation of an initial timetabling. The authors implement a tabu search to solve their formulation. Although we do not design an efficient methodology to generate solutions for a complete integration of timetabling with subsequent problems such as vehicle and crew scheduling, Appendix [B] shows some mathematical formulations and preliminary results of the integration between timetabling and vehicle scheduling problem.

We complement the excellent review of timetabling problems of Guihaire and Hao [2008b] in Tables 2.1 and 2.2. Moreover, we include some studies of railway systems that can be implemented in transit networks. Papers marked with (*) are included in the review of Guihaire and Hao [2008b]. First column shows the reference of the study. Second column exhibits the objective in for the timetabling problem. Third column shows the constraints of the problem. Fourth column exhibits the solution approach. Finally, fifth column represent the instances used to the experimental stage. In this column there are several categories where "Example" represent small instances designed by the authors or partial networks in real systems; "Real" represent the implementation in real transit networks; "Test" means that the authors generate randomly or real data based instances; "Benchmark" represent a previous instance set.

Authors (year)	Objective of timetabling	Constraints of the system	Solution method	Case
Rapp and Gehner [1976]*	Minimize delays in passenger transfers	-	Interactive graphic system	Example
Koutsopoulos et al. [1985]*	Minimize waiting time, operator costs, and vehicle crowding	Fleet size and line capacity	Linear programming	Example
Klemt and Stemme [1988]*	Maximize transfer synchronization	-	Constructive heuristic	Example
Bookbinder and Désilets [1992]*	Minimize waiting times	Even headways	Simulation and mathematical prog.	Example
Daduna and Voß [1995]*	Minimize transfer times	Fulfill demand	Tabu search	Examples
Chakroborty et al. [1995, 1997]* Deb and Chakroborty [1998]*	Minimize passengers total waiting time	Fleet size, stopping time bounds, and maximum headway and transfer times	Evolutionary algorithms	Example
Chun and Chan [1999]	Minimize fleet size, headway variations, and idle time	Vehicle and crew scheduling aspects	Constraint-based search	Real
Palma and Lindsey [2001]*	Minimize total delay cost	Desired boarding times	Mathematical prog.	-
Ceder [2001]	Maximize evenness passenger loads and minimize fleet size	Fulfill demand	Heuristic	Example
Ceder et al. [2001]* Ceder and Tal [2001]*	Maximize number of simultaneous arrivals	Headway bounds	Heuristics	Example
Jansen and Nielsen [2002]*	Minimize total transfer time	Preset frequencies and even headways	Tabu search	Example
Ceder [2003]*	Demand satisfaction and number of trips	Headway bounds	Constructive heuristic	Example
Castelli et al. [2004]*	Minimize passenger transfer waiting time	Limited frequency	Lagrangian heuristic	Test
Wong and Leung [2004]* Wong et al. [2008]	Minimize waiting time	Headway and travel times bounds	Optimization-based heuristic	Real
Eranki [2004]*	Maximize number of simultaneous arrivals considering time windows	Headway bounds	Constructive heuristic	Example
Fleurent et al. [2005]*	Minimize transfer costs	Passenger waiting times bounds	Lagrangian relaxations and several heuristics	Real

Table 2.1: Literature review of bus timetabling problems. Papers marked with (*) are presented in Guihaire and Hao [2008b].

Authors (year)	Objective of timetabling	Constraints of the system	Solution method	Case
Cevallos and Zhao [2006]*	Minimize transfer waiting times	Fixed headways	Evolutionary algorithm	Example
Schröder and Solchenbach [2006]	Minimize cost of waiting times	Deviation from an initial timetabling	CPLEX's	Real
Liu et al. [2007]	Minimize transfer time subject	Headway bounds	Nesting tabu search	Example
Wong et al. [2008]	Minimize waiting time costs	Headway bounds	Lagrangian heuristic	Real
Zhao and Zeng [2008]	Minimize travel time costs	Headways, fleet size, rout length, and load factor	Simmulated annealing and tabu search	Benchmark
Kwan and Chang [2008]	Synchronization vs deviation from an initial timetable	Frequency, limited load and drivers, and bound run time	NSGA2, tabu search, and simulated annealing	Example
Guihaire and Hao [2008a, 2010a]	Optimize fleet size, quality of transfers, and headway evenness	Deviation from initial timetable	Iterated local search	Real
Shariat and Mahdi [2009]	Fulfill demand considering stochasticity	Fleet size	Heuristic and simulation	Example
Fournier [2010]	Minimize number of trips	Same quantity of forth and back trips and even headways	Greedy algorithm	Real
Guihaire and Hao [2010b]	Maximize number and quality of transfers	Vehicle and crew schedule	Tabu search	Real
Li et al. [2010]	Minimize cost of fulfilling demand considering uneven headways	Fleet size	Hooke-Jeeves Heuristic	Example
Ceder [2011]	minimize headway variation	Even loads, fleet size, and bounded deviation form an initial timetable	Constructive heuristic	Example
Ibarra-Rojas and Rios-Solis [2012]	Maximize number of simultaneous arrivals considering time windows	Headway bounds, departure dispersion along the planning period	Iterated local search	Test
Ávila and López [2012]	Maximize synchronizations, minimize unbalanced loads, transfer times, and line costs	Frequency, load limits, and headway bounds	multiobjective scatter and tabu searches	Test

Table 2.2: Literature review of bus timetabling problems. Papers marked with (*) are presented in Guihaire and Hao [2008b].

2.4 Chapter conclusions

The elements that define the structure of a timetable are headway policies, flexibility, previous planning consideration, travel time nature, and objective function. In particular, our Synchronization Bus Timetabling Problem has the objective of maximize the number of synchronization events to allow well timed passenger transfer and avoid bus bunching considering headway bounds to make it flexible, a given frequency due considering constant demand, and short planning periods considering deterministic travel times.

Most of the characteristic of our Synchronization Bus Timetabling are considered in some studies present in literature. However, there is not a single work that considers all of them. Therefore, efficient algorithms are needed.

Chapter 3

Synchronization Bus Timetabling Problem (SBT)

Summary: It was not easy to define our research topic. We start from nothing and we end with a problem suitable to the operation policies of transit network agencies of our case study. In this chapter we present the problem statement and exhibit that our problem and others similar are difficult to be solved. Moreover, we analyze the structure of the proposed mathematical formulation to understand the impact of the characteristic of our timetabling problem.

"So, it is difficult. Great!"

3.1 Problem definition

The planning process of a bus network is complicated. Therefore, it is often divided into several subproblems such as line planning, timetable generation, vehicle scheduling, and crew scheduling [Ceder, 2007]. In the timetabling problem we determine departure times for trips from different lines to achieve a specific goal, which in our case is to maximize the number of synchronizations events, to favor well timed passenger transfers and avoid bus bunching. Generate a timetable is a critical step in the planning process of a bus network since service quality and subsequent planning steps such as vehicle and crew scheduling depend on its solution.

We focus on Monterrey, Mexico, where the bus transit system is private. Moreover, there are different bus companies sharing the demand. Competition among them creates

a particularly large bus network. The left side of Figure 3.1 shows the entire bus network of Monterrey's metropolitan area.

Our case has the following characteristics:

- The bus network is private and has more than 300 bus lines (Monterrey has a population close to the 4 million).
- There is almost full coverage of the territory. The area marked with a black box on the left part of Figure 3.1 shows the line concentration downtown since there is a tendency to design lines crossing this area. Therefore, many transfers are located there.
- Many lines have "sub-lines" which share a common route segment but have variations at the beginning and at the end of the routes (see right part of Figure 3.1).
- The timetable is only for the company's administration. Passengers do not know the time at which bus arrives at each stop. They only have an estimate of their waiting time to take the next bus. Then, a regular service is required.

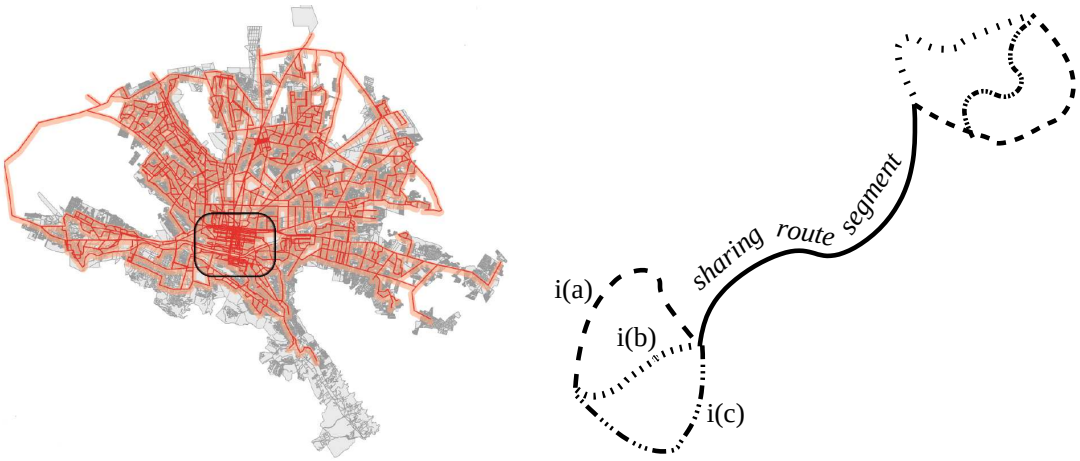


Figure 3.1: The left panel shows the bus network of Monterrey's metropolitan area. The marked area shows the line concentration downtown [Rabut, 2010]. The right panel shows a bus line i consisting of three sub-lines a, b, and c.

Considering the bus network with the above-mentioned characteristics, what can we optimize? To answer this question, we noted the need to provide high quality service and

meet other company priorities. Particularly, we found the following key components to optimize through timetable generation.

- Passenger transfers: Travel from one point to another might imply transfer between lines. We seek short waiting times at some key nodes like downtown.
- Bus bunching: Due to the presence of sub-lines, different lines commonly converge on a specific node (called a bus bunching node). We aim to avoid bus bunching between sub-lines or between different lines using trip separation.
- *Almost* evenly spaced departures: A large variation in the time between consecutive trips affects the behavior of passenger demand, even in small planning periods. Therefore, we are interested in regular services.

Passenger activity during the day also affects the bus network behavior. Peak hours imply a large amount of bus bunching of different lines on multiple nodes of the network and a large number of passenger transfers. Therefore, we categorize the days (weekend, Monday, holidays, etc.) and then, we split each day into planning periods (peak hours, morning, afternoon, etc.) to achieve more accurate deterministic travel times. However, because of the uncertainties inherent in a public transportation system such as Monterrey's, passengers prefer a flexible transfer rather than an instant transfer. For example, different travel times due to variations in driver speeds are drawbacks for achieving punctual synchronization.

In the basis of the above, the **Synchronization Bus Timetabling Problem (SBT)** can be seen as scheduling the departure times for all bus line trips in order to maximize the synchronizations between buses considering regular services in a given planning period. A formal definition of the problem is given in the following.

The type of bus network we are interested in can be represented by a set of lines denoted as I. As described in Section 1, the routes and the stops used by each bus line as well as the frequency of each line are determined before the timetable generation phase. We will call *a synchronization node*, a specific stop in the network where two lines cross

each other and where passenger transfers are needed or bus bunching can happen. We denote by B the set of all synchronization nodes. Let T be the planning horizon defined in time units. For example, T will be 2 hours in rush periods in the morning or 3 hours in valley periods in the afternoon. We will denote by f^i the frequency of line $i \in I$, that is to say the number of trips that have to be scheduled within T time units for line i. We also assume that buses of the same line will run at regular speed and denote by t^{ib}, $i \in I$ the travel time for a bus to run from an initial node, called *depot*, of line i to node $b \in B$.

A *headway* is the separation time between consecutive trips for the same line. For example, if the first (resp. second) trip of a line starts at time unit 3 (resp. 23), there is a headway of 20 time units between these two trips. Since regularly spaced departure times are required, each trip of a line i of I will start every $\frac{T}{f^i}$ time units with a flexibility of δ^i time units. Consequently, $h^i = \frac{T}{f^i} - \delta^i$ (resp. $H^i = \frac{T}{f^i} + \delta^i$) is the minimum (resp. maximum) headway for a line $i, i \in I$. Moreover, the first (resp. last) trip of line i must start before H^i (resp. after $T - H^i$). These headways guarantee that the entire planning horizon is covered by the trips which is one of the main differences between the addressed model and the ones studied in Ceder and Tal [2001], Eranki [2004]. Figure 3.2 illustrates regularly spaced trips for a given line i of frequency $f^i = 4$ with (case (a)) and without (case (b)) flexibility.

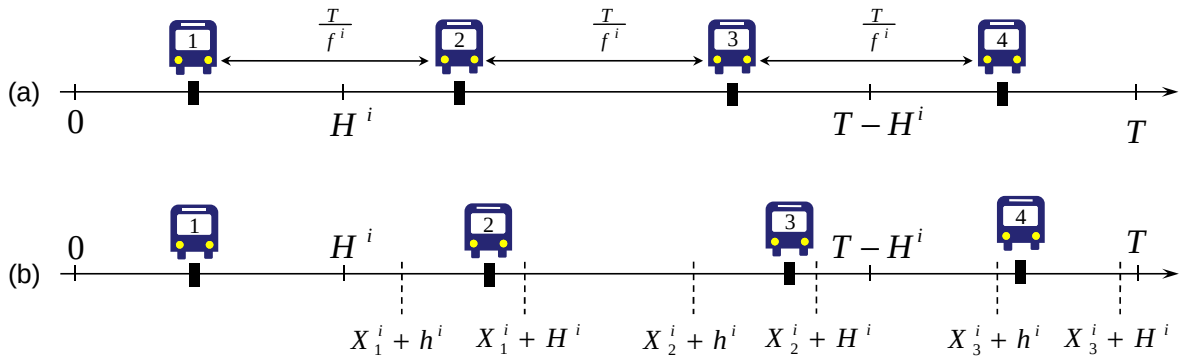


Figure 3.2: Two different timetables. Case (a) shows regular departure times while Case (b) shows departure times considering the headway flexibility given by parameter δ^i.

We recall that a synchronization at a stop between two bus trips occurs in two different cases; when the difference between their arrival times is long enough to avoid bus bunching in one hand or allow passenger transfers without long waiting times in

another hand.

A *bunching node* is a particular synchronization node in the network where two buses arriving simultaneously at this node will share a segment of their respective routes. Consequently, to avoid a bus bunching at node b, the length of the time interval between two bus arrivals at node b has to be greater than a specific value denoted w_b. Moreover, if we avoid bus bunching between $i(p)$ and $j(q)$, it is useless avoiding bus bunching between $i(p)$ and $j(q+1)$. Hence, it also important to consider a maximum separation time W^b such that $W^b - w^b < h^j$ in bus bunching nodes.

A *transfer node* is a particular synchronization node in the network where passengers may transfer from one line i to some line j. Since it is common that passengers must walk to make a transfer and uncertainties in travel times are present, flexible transfers (waiting times greater than 0 minutes) are preferred. This way, it is less probable to miss a transfer from line i to line j if there is not enough time to walk or the bus of line i (line j, resp) arrives late (early, resp). Then, in order to permit well timed passenger transfer at a transfer node b, the difference between two bus arrivals at node b has to be lower (grater, resp) than a specific value denoted W^b (w^b, resp). An important characteristic in Monterrey's transit network is that passenger transfers from i to j are needed in some node b at mornings while passenger transfers from j to i are needed in some node b', $b' \neq b$, at afternoons. Therefore, there are oriented synchronizations, i.e., $j \in J(i)$ does not implies $i \in J(j)$.

In the basis of the above, a synchronization at node b for two bus trips happens if the difference of their arrival times is in the so-called *waiting time window* $[w^b, W^b]$. Then, SBT consists in *determining the departure times* X_p^i *of each trip* $p \in \{1, ..., f^i\}$ *of every line* $i \in B$ *so that* $X_1^i \leq H^i$, $h^i \leq X_{p+1}^i - X_p^i \leq H^i$ *for every* $i \in \{1, ..., f^i - 1\}$, $T - H^i \leq X_{f^i}^i \leq T$ *and the number of synchronizations is maximized.*

Figure 3.3 illustrates the two types of synchronization nodes. Case (1) represents a bunching node b lines i and j (with $h^j = 8$ minutes) converge and then share a segment of their routes. The numbers by the side of node b represent the arrival times of a pair of trips of these lines. The minimum and maximum separation times to avoid bus bunching

between two buses at this node b are $w^b = 6$ and $W^b = 12$, respectively. Case (2) represents a node b where passengers would like to go from one trip of line i to some trip of line j considering the process of crossing a street. In this case, 5 time units is the maximum desired passenger waiting time and 2 time units is the estimated required time to cross the street. Then, the waiting time window is given by $[w^b, W^b] = [2, 7]$.

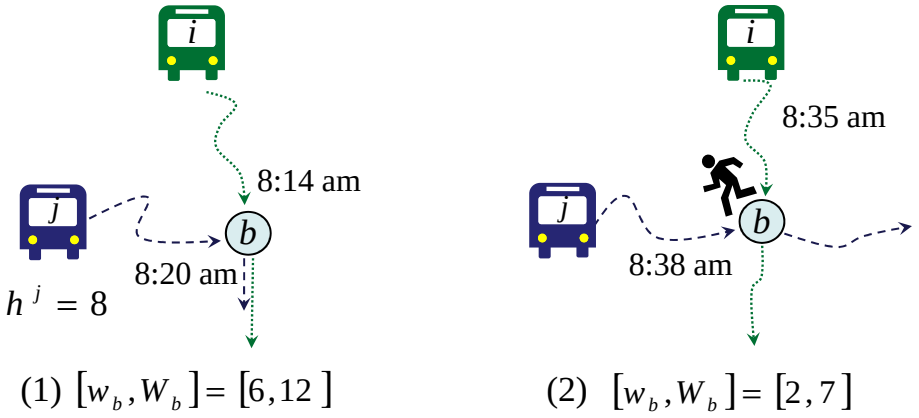


Figure 3.3: Two cases of synchronization nodes: Case (1) represents a bus bunching node where lines i and j converge and then share a segment of their routes while Case (2) represents a transfer node where passengers would like to go from trips of line i to trips of line j.

3.2 Mixed integer programming formulation for SBT

We consider in the following an instance of SBTP and we now proceed with the 0-1 MIP formulation of the problem already described in Ibarra-Rojas and Rios-Solis [2012]. For a given a line $i \in I$, we will denote by $i(p)$, $p \in \{1, \ldots, f^i\}$, the p^{th} trip of line i and we associate to $i(p)$ the real variable X_p^i that represents its departure time. For each line $i \in I$, the set $I(i)$ will contain the lines that share a synchronization node with line i. For a pair of lines $(i, j), j \in I(i)$, the set B^{ij} is the set of all the synchronization nodes shared by i and j. The binary decision variable Y_{pq}^{ijb} is equal to 1, if and only if, trip $i(p)$ arrives first at node b and if trips $i(p)$ and $j(q)$ provide a synchronization at node b. In the case that $j \in I(i)$ (and then $i \in I(j)$), we remark that if a synchronization happens at node b between trips $i(p)$ and $j(q)$, then either Y_{pq}^{ijb} or Y_{qpb}^{ji} will be set to 1. In the sequel, we

will denote by (X, Y) a solution of an SBTP instance.

Considering the previous parameters and decision variables, the MIP formulation of SBTP, denoted by SBTP MIP, is given by

$$\max \sum_{i \in I} \sum_{j \in I(i)} \sum_{b \in B^{ij}} \sum_{p=1}^{f^i} \sum_{q=1}^{f^j} Y_{pq}^{ijb}$$

$$\begin{aligned}
\text{s.t.} \quad & X_1^i \leq H^i && \forall\ i \in I && (3.1)\\
& T - H^i \leq X_{f^i}^i \leq T && \forall\ i \in I && (3.2)\\
& h^i \leq X_{p+1}^i - X_p^i \leq H^i && \forall\ i \in I,\ p = 1, \ldots, f^i - 1 && (3.3)\\
& \left(X_q^j + t^{jb}\right) - \left(X_p^i + t^{ib}\right) \geq w^b + M\left(1 - Y_{pq}^{ijb}\right) && \forall\ i \in I,\ j \in I(i),\ b \in B^{ij}, && \\
& && p = 1, \ldots, f^i,\ q = 1, \ldots, f^j && (3.4)\\
& \left(X_q^j + t^{jb}\right) - \left(X_p^i + t^{ib}\right) \leq W^b + M\left(1 - Y_{pq}^{ijb}\right) && \forall\ i \in I,\ j \in I(i),\ b \in B^{ij}, && \\
& && p = 1, \ldots, f^i,\ q = 1, \ldots, f^j && (3.5)\\
& X_p^i \in \mathbb{R}, Y_{pq}^{ijb} \in \{0, 1\} && \forall\ i \in I,\ j \in I(i),\ b \in B^{ij}, && \\
& && p = 1, \ldots, f^i,\ q = 1, \ldots, f^j && (3.6)
\end{aligned}$$

The objective function maximizes the total number of synchronizations. Constraints (3.1) and (3.2) guarantee that the entire planning horizon is covered by the trips. Constraints 3.3 impose that consecutive trips of line i must happens with a minimum (resp. maximum) headway h^i (resp. H^i). Remark that the arrival time of trip $i(p)$ at node b is $X_p^i + t^{ib}$. Hence constraints (3.4) and (3.5) activate the synchronization variables Y_{pq}^{ijb} if the difference between the arrival times of trips $i(p)$ and $j(q)$ at node b is within $\left[w^b, W^b\right]$ and trip $i(p)$ arrives first at b. where M is a large constant. Parameter M is a large number, we can bind it by the maximum difference of arrival times for every pair of lines $(i, j), j \in J(i)$ that synchronize at every node b, that is, $M = \max\left\{\max_{i,j \in J(i), b \in B^{ij}} \left\{\left(T + t^{jb}\right) - t^{ib}\right\}, \max_{i,j \in J(i), b \in B^{ij}} \left\{\left(T + t^{ib}\right) - t^{jb}\right\}\right\}$. Finally, constraints (3.6) represent the domain of the decision variables.

The previous MIP is an enhancement of the model proposed by Eranki [2004]. The first modification to adapt Eranki's model to Monterrey's case is to redefine the synchronization variables. We consider oriented synchronizations, i.e., trip $i(p)$ synchronizes with trip $j(q)$ at node b if the arrival separation $\left(X_q^j + t^{jb}\right)$ - $\left(X_p^i + t^{ib}\right)$ is between w^b and W^b. Therefore, Y_{pq}^{ijb} models transfer from line i to line j and not vice verse. Since we only seek trip separation in bus bunching nodes, we can model synchronization events in these nodes as any directed transfer. We also add constraints (3.2) to achieve departure dispersion along the planning period T. Finally, we consider the characteristics of *almost* evenly spaced headway times using the flexibility parameter δ^i for each line i, $i \in I$.

3.3 Computational complexity of SBT

Computational complexity theory states that problems can vary in the effort required to be solved them [e.g. Garey and Johnson, 1979]. An optimization problem that belongs to the NP-hard class means, in simple terms, that an efficient algorithm for the exact solution of this problem does not exist. Proving that a problem is NP-hard is important because researchers are unlikely spend time searching for an efficient algorithm but instead seek another options such as approximate solutions for large instances (as the ones we present in Section 5).

In this study, we prove that SBT belongs to the NP-hard class. Studies of Ceder et al. [2001] and Eranki [2004] remark on the intractability of closely related problems of SBT with the help of empirical results. Nevertheless, the question whether these problems were part of the NP-hard class was a 10-year-old open question. With our proof we also guarantee that problems of Ceder et al. [2001] and Eranki [2004] are also NP-hard.

To prove that an optimization problem $[P]$ is NP-hard, we have to prove that its decision version $[DP]$ is NP-complete. As it can be seen in Garey and Johnson [1979] and Papadimitriou and Steiglitz [1998], there are basic steps to prove that a decision problem $[DP]$ is NP-complete. These steps are the following.

1. Prove that problem $[DP] \in NP$, i.e., the validity of a given solution can be verified

in polynomial time.

2. Exhibit a NP-complete problem $[A]$ that can be polynomially reduced to $[DP]$ (denoted as $[A] \prec_p [DP]$).

3. Prove the solution equivalence, i.e., the answer of an instance of problem $[DP]$ is "YES" if and only if, the answer of the related instance of problem $[A]$ is "YES".

We propose a polynomial reduction from monotone NAE-3SAT, whose NP-completeness is assured by Schaefer [1978], to the decision version of SBT called dSBT. First of all, we present the decision versions of both problems.

Monotone NAE-3SAT:

INPUT: A set of r literals and a set $\{C_1, C_2, \ldots, C_c\}$ of c clauses. Each clause has three different and unnegated variables. A clause is true if the values of its literals are not all equal.

QUESTION: Is there an assignment of the literals values (false as 0 and true as 1) such that the sentence $\phi = C_1 \wedge \cdots \wedge C_c$ is true?

An example is the sentence: $\phi = (x_1 \vee x_2 \vee x_3) \wedge (x_4 \vee x_2 \vee x_3) \wedge (x_1 \vee x_4 \vee x_3)$ that has a true value when $x_1 = x_3 = 1$ and $x_2 = x_4 = 0$.

Synchronization Bus Timetabling (dSBT):

INPUT: Planning period time T, set I of bus lines, set B of stops, sets B^{ij} of common nodes for each pair of lines $(i, j) \in I \times I$, travel times t^{ib} from depot of line i to each node $b \in B^i$, frequency vector $f \in \mathbb{Z}_+^{|I|}$, minimum headway vector $h \in \mathbb{R}^{|I|}$, maximum headway vector $H \in \mathbb{R}^{|I|}$, minimum waiting time vector $w \in \mathbb{R}^{|B|}$, maximum waiting time vector $W \in \mathbb{R}^{|B|}$, and a scalar K.

QUESTION: Are there any values for departure times variables $X_p^i \in \mathbb{R}$ of each trip p of line $i \in I$ such that the first trip of each line i departs before H^i, the last trip of each line i departs after $T - H^i$, separation times between

consecutive trips of each line i are within $[h^i, H^i]$, and the sum of pairwise arrivals at synchronization nodes b with a separated time within $[w^b, W^b]$ (sum of synchronizations events) is greater or equal than K?

Next, we show the theorem of the complexity for *dSBT*.

Theorem 1. *dSBT is NP-complete.*

Proof. *Claim 1. dSBT* $\in$ *NP*

Assume there exist an algorithm that generates a solution for *dSBT*. Determining if the solution is feasible for *dSBT* implies several steps. Headway considerations represent a quantity of inequalities of the order of the number of departure time variables, i.e., $\mathcal{O}(|X|)$, where X represent the set of departure time variables. On the other hand, verify if the sum synchronization events is greater than or equal than K implies verify a number of inequalities of the order of the trip pairs, i.e., $\mathcal{O}\left(|X|^2\right)$. In the basis of the above, we need a polynomial number of steps to verify the feasibility of a solution for *dSBT*, i.e., *dSBT* $\in$ NP.

Claim 2. Monotone NAE-3SAT $\prec_p$ *dSBT*

Consider an arbitrary instance of monotone NAE-3SAT with c clauses. We build a particular instance of *dSBT*, called *dSBT**, as follows. We define a planning period of $T = 1$ minute; the number of lines as $|I| = r$ (one line corresponds to each literal of monotone NAE-3SAT); frequency $f^i = 1$ for each line i; headway bounds $h^i = 0$ and $H^i = 1$ for each line i; $w^b = W^b = 0$ for each synchronization node b. Notice that a synchronization event is indeed defined as the simultaneous arrivals of two trips at a common node. Moreover, by *dSBT** definition, there is not necessary to consider headway constraints (3.1)-(3.3). The steps to define the synchronization nodes and arrival times are the following.

We make $B^{ij} = \emptyset$ for all pair of lines $(i, j) \in I \times I$. Then, for each clause $C_l = (x_f \vee x_g \vee x_h)$, we do the following. First, we create six synchronization nodes labeled as $\{b_{l1}, b_{l2}, \ldots, b_{l6}\}$ and divide them between the sets of synchronization nodes of each

pair of lines in clause C_l as $B^{fg} = B^{fg} \bigcup \{b_{l1}, b_{l2}\}$, $B^{gh} = B^{gh} \bigcup \{b_{l3}, b_{l4}\}$, and $B^{fh} = B^{fh} \bigcup \{b_{l5}, b_{l6}\}$. Next, we define the travel times from the depots to the synchronization nodes using a reference time $o_l = 9(l-1)$. Explicitly, $t^{fb_{l1}} = o_l + 1$ and $t^{gb_{l1}} = o_l + 0$, for node b_{l1}; $t^{fb_{l2}} = o_l + 2$ and $t^{gb_{l2}} = o_l + 3$, for node b_{l2}; $t^{gb_{l3}} = o_l + 4$ and $t^{hb_{l3}} = o_l + 3$, for node b_{l3}; $t^{gb_{l4}} = o_l + 5$ and $t^{hb_{l4}} = o_l + 6$, for node b_{l4}; $t^{fb_{l5}} = o_l + 6$ and $t^{hb_{l5}} = o_l + 7$, for node b_{l5}; $t^{fb_{l6}} = o_l + 9$ and $t^{hb_{l6}} = o_l + 8$, for node b_{l8}. Finally, after create the synchronization nodes and travel times related with each clause, we define $K = 2c$. This reduction implies a number of steps bounded by the number of literals and the number of clauses of monotone NAE-3SAT and the size of instance $dSBT^*$ also depends on this numbers.

By $dSBT^*$ definition, each synchronization node b is related with two lines and the travel times from the depot of these lines to node b are defined in such a way to have a difference of 1 minute. Figure 3.4 shows the nodes and travel times related with the first clause C_1. There are three time lines and the quantities above the nodes represent the arrival times at nodes when trips depart at reference time $o_1 = 9(1-1) = 0$. Notice that the arrival times at these nodes are within a 9-min time window.

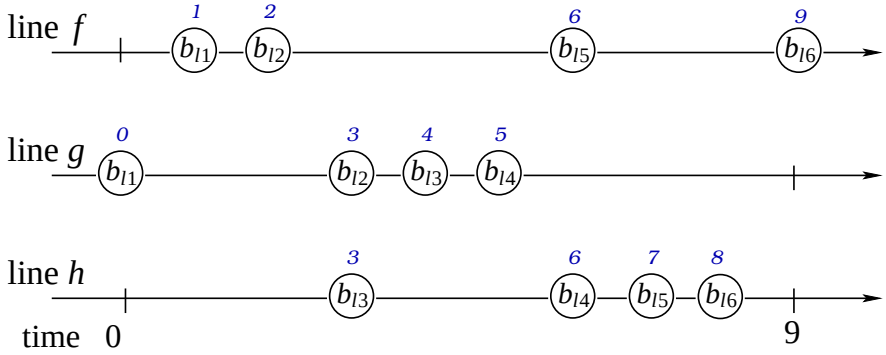


Figure 3.4: Line design for the 9-minute time interval related to the first clause C_1.

Moreover, if we assign the value of $x_i \in \{0, 1\}$ in monotone NAE-3SAT to the variable X^i in $dSBT^*$, the following must be satisfied. If x_i and x_j are not equal in monotone NAE-3SAT, $X^i \neq X^j$ and pair if lines $(i, j \in J(i))$ synchronize at one common node. Figure 3.5 shows the cases of a true clause $C_1 = (x_f \vee x_g \vee x_h)$ where the literal x_f has a different value than x_g and x_h. Dashed lines represents the synchronization (simultaneous arrivals) of two lines in $dSBT^*$. Indeed the synchronized pairs of lines are

(f, g) and (f, h) since $X^f \neq X^g$ and $X^f \neq X^h$.

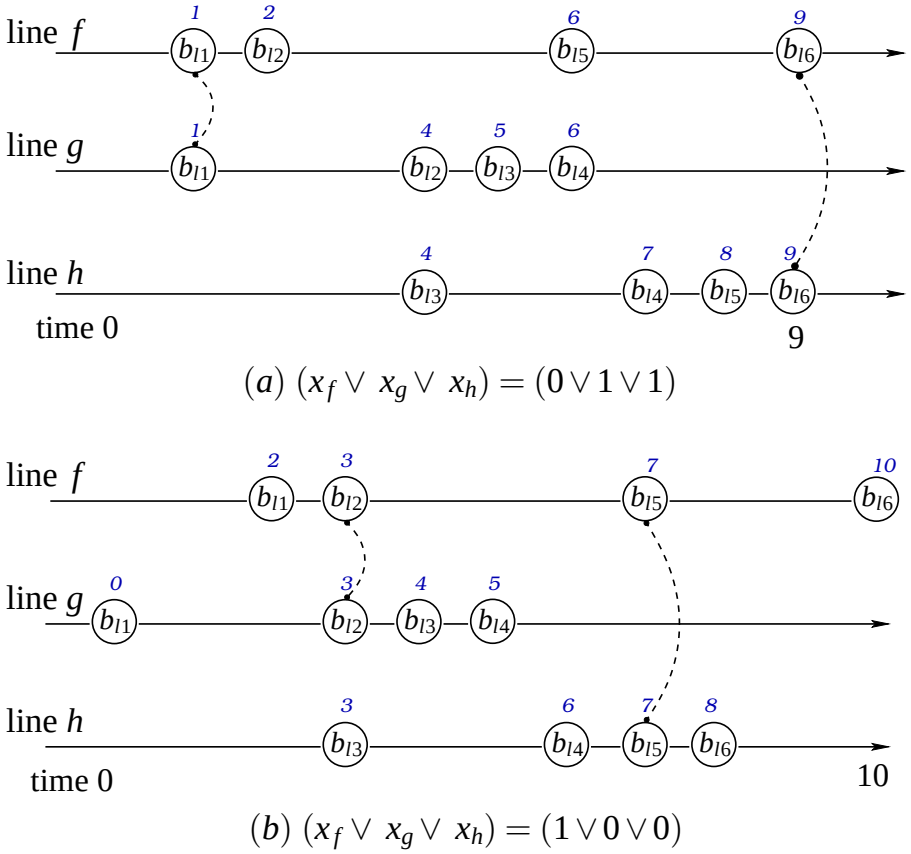


Figure 3.5: True clause $C_1 = (x_f, x_g, x_h)$ when literal x_f is different than the other two. The simultaneous arrivals at common nodes denoted with dashed lines correspond to the pairs of lines (f, g) and (f, h) since $X^f \neq X^g$ and $X^f \neq X^h$.

By definition of instance $dSBT^*$, it has the following property.

Property 1. *Every true clause C_l of monotone NAE-3SAT implies two synchronizations in instance $dSBT^*$ and every false clause $C_{l'}$ of monotone NAE-3SAT implies zero synchronizations in instance $dSBT^*$.*

Figure 3.6 shows the cases of a true clause $C_l = (x_f \vee x_g \vee x_h)$ on monotone NAE-3SAT. For the $dSBT^*$ instance, the nodes represent the bus lines and the dashed lines represent the synchronizations.

Claim 3. Solution equivalence

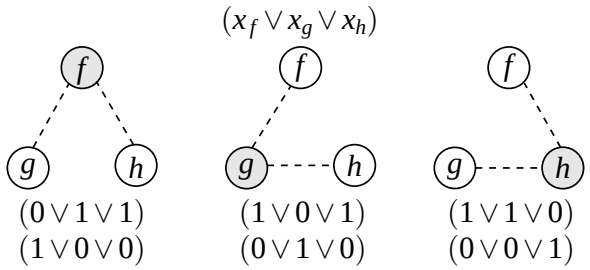


Figure 3.6: Possible values for a true clause of monotone NAE-3SAT and the related pairs of bus lines that are synchronized in instance $dSBT^*$.

($\Rightarrow$) Let $x = (x_1, \ldots, x_c)$ be a solution of the monotone NAE-3SAT problem with a "YES" answer. Making $X^i = x_i$ for each bus line $i \in I$ in $dSBT^*$ and by Property 1, each true clause implies two simultaneous arrivals. Therefore, the sum of pairwise simultaneous arrivals is $2c = K$, i.e., the answer for $dSBT^*$ is "YES."

($\Leftarrow$) Let $\mathcal{X} = (X^1, X^2, \ldots, X^r)$ be a solution of instance $dSBT^*$, with a "YES" answer. By Property 1, the sum of pairwise simultaneous arrivals is grater and equal than K.

Now, suppose the solution for the related instance of monotone NAE-3SAT has a "NO" answer, i.e., there is a false clause $C_{l'}$ in monotone NAE-3SAT. By Property 1, this clause implies 0 simultaneous arrivals in $dSBT^*$. Then, the sum of pairwise simultaneous arrivals excluding pair of lines related to clause $C_{l'}$ is greater or equal than K which is a contradiction since each clause in monotone NAE-3SAT implies at most two simultaneous arrivals in $dSBT^*$. Therefore, the solution for the instance of monotone NAE-3SAT has a "YES" answer. □

We have prove that $dSBT$ is NP-complete and if a decision problem belongs to the NP-complete class, the corresponding optimization problem belongs to the NP-hard class. Thus, the following corollary is deduced.

Corollary 1. *SBT is NP-hard.*

Notice that our complexity proof do not depend on the nature of departure time variables. Therefore, considering real or integer departure times, the computational complexity remains. The concept of the previous polynomial reduction from monotone NAE-

3SAT to *dSBT* can be used for the problems presented in Ceder et al. [2001] and Eranki [2004] since the particular instance $dSBT^*$ is precisely a subproblem of both timetabling problems.

3.4 Preprocessing stage

The solution space of SBT has an interesting structure since there is a large number of synchronization variables and many of them are zero. Moreover, there are many different values for departure times variables leading to the same value of the objective function.

Our SBT MILP has a large feasible space and, as we show in this chapter, it is difficult to obtain high quality feasible solutions in a reasonable amount of time using the linear solver of CPLEX. Indeed in our SBT formulation, we define synchronization variables Y_{pq}^{ijb} for every trips $i(p)$, $j(q)$ with $j \in J(i)$, and node $b \in B^{ij}$. Nevertheless, it is improbable that the first trip of line i could synchronize with the last trip of line j due their departure time constraints (3.1) and (3.2). Figure 3.7 shows a typical structure of the matrix formed by synchronization variables for the pair of lines (i, j) at some node b where the element in pth row and qth column represents to variable Y_{pq}^{ijb}. Zeros in the matrix represent the variables forced to be zero. The variables between the curved lines are the only ones whose values could be 1, i.e., they might satisfy the constraints of SBT.

$$\begin{pmatrix} Y_{11}^{ijb} & Y_{12}^{ijb} & \cdots & Y_{1f^j}^{ijb} \\ Y_{21}^{ijb} & Y_{22}^{ijb} & \cdots & Y_{2f^j}^{ijb} \\ \vdots & \vdots & \ddots & \vdots \\ Y_{f^i1}^{ijb} & Y_{f^i2}^{ijb} & \cdots & Y_{f^if^j}^{ijb} \end{pmatrix} \approx$$

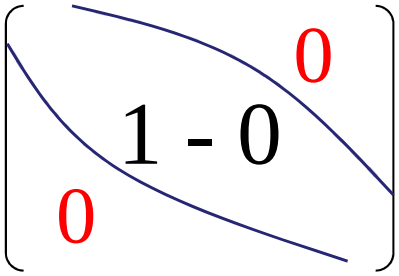


Figure 3.7: Typical structure of the matrix formed by the synchronization variables Y_{pq}^{ijb} related to the trips of the lines i and $j \in J(i)$, and synchronization node $b \in B^{ij}$.

Our preprocessing stage is based on feasible time windows for departures, arrivals, and synchronizations. To obtain these feasible time windows we implement constraint propagation which can be defined as a procedure embedding any reasoning that explicitly

forbids values or combinations of values for some variables of a problem because a given subset of its constraints cannot be satisfied otherwise [Bessiere, 2006]. As it is mentioned by Barták [2001], constraint propagation can be used to solve fully the problem but this is rarely done due to efficiency issues. It is more common to combine an efficient but incomplete consistency techniques (that eliminate many "obvious" inconsistencies) with a non deterministic search to simplify the problem and reduce the search space.

Our constraint propagation is based on headway parameters and constraints. For example, assume the value for departure time X_p^i has been set. We know that the headway times should be within $[h^i, H^i]$. Thus, we can define a feasible departure time window $D_{p'}^i$ for any other trip $i(p')$ such that $p' > p$ as $D_{p'}^i = \left[X_p^i + (p'-p)h^i, X_p^i + (p'-p)H^i\right]$. Analogously, for trips $i(p')$ such that $p' < p$ we set $D_{p'}^i = \left[X_p^i - (p-p')H^i, X_p^i - (p-p')h^i\right]$. The more values of departure times are set, the more precise the departure time windows is. If we do not have fixed values for the departure times, we can define D_p^i for trip $i(p)$ using the bounds of departure time for the first and last trips given by constraints (3.1) and (3.2), respectively. Procedure 1 shows the steps to obtain the D_p^i using these bounds.

Procedure 1: Generate D_p^i.

1. Suppose first trip departs at $X_1^i = 0$, calculate the earliest departure time for trip $i(p)$ given by $earliest_1 = (p-1)h^i$.
2. Suppose first trip departs at $X_1^i = H^i$, calculate the latest departure time for trip $i(p)$ given by $latest_1 = \min\{T, pH^i\}$.
3. Suppose last trip departs at $X_p^{f^i} = T - H^i$, calculate the earliest departure time for trip $i(p)$ given by $earliest_{f^i} = \max\{0, T - (f^i - (p-1))\, H^i\}$.
4. Suppose last trip departs at $X_p^{f^i} = T$, calculate the latest departure time for trip $i(p)$ given by $latest^{f^i} = T - (f^i - p)\, h^i$.
5. Define D_p^i as $[earliest_1, latest_1] \cap \left[earliest_{f^i}, latest_{f^i}\right]$. Therefore, the departure time window D_p^i can be defined as follows.

$$\left[\max\left\{(p-1)\, h^i, T - \left(f^i - (p-1)\right) H^i\right\}, \min\left\{pH^i, T - \left(f^i - p\right) h^i\right\}\right] \qquad (3.7)$$

Figure 3.8 shows an example of the departure time window construction for trip $i(8)$. This example has a planning period of $T = 30$ minutes, a frequency of $f^i = 10$, and headway flexibility parameter of $\delta^i = 1$ minute, i.e., $h^i = 2$ and $H^i = 4$. Figure 3.8 has four time lines. First line shows the earliest departure time $7h^i = 14$ for the eighth trip, assuming that first trip departs at $X_1^i = 0$ (step 1). Second time line shows the latest departure time $\min\{T, 8H^i\} = 30$ for the eighth trip, assuming that first trip departs at $X_1^i = H^i = 4$ (step 2). Third line shows the earliest departure time $\max\{0, 30 - 3H^i\} = 18$ for the eighth trip, assuming that the last trip departs at $X_{f^i}^i = T - H^i = 26$ (step 3). Finally, fourth line shows the latest departure time $T - 2h^i = 26$ of the eighth trip, assuming that the last trip departs at $X_{f^i}^i = T$ (step 4). Therefore, the intersection of the earliest and latest departure times $[\max\{14, 18\}, \min\{30, 26\}]$ results in the feasible departure time window D_8^i (see marked area of Figure 3.8).

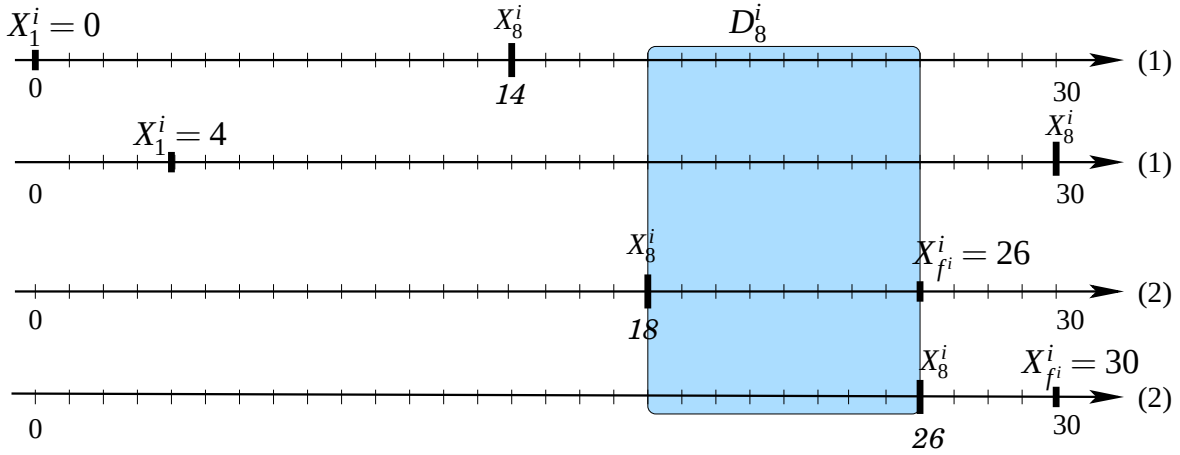


Figure 3.8: Feasible departure time window D_8^i for trip of line $i(8)$ corresponding to an instance with a planning period $T = 30$ minutes, $f^i = 10$, and $\delta_i = 1$ minute.

By definition of window D_p^i, it has an important property.

Property 2. *Time window D_p^i determines all the feasible departure time values of X_p^i.*

If we consider the departure time window D_p^i of each trip $i(p)$, we can define the arrival time window A_p^{ib} for this trip at node b. We obtain A_p^{ib} by shifting D_p^i by t^{ib} time units. It can be expressed as

$$A_p^{ib} = \left[left(D_p^i) + t^{ib}, right(D_p^i) + t^{ib}\right],$$

where $left(D_p^i)$ and $right(D_p^i)$ represent lower and upper limit of time window D_p^i, respectively. Similarly, we define a synchronization window S_p^{ib} for each trip $i(p)$ and node b as

$$S_p^{ib} = \left[left(A_p^{ib}) + w^b, right(A_p^{ib}) + W^b\right].$$

We can determine if synchronization between trips $i(p)$ and $j(q)$ is possible by looking at synchronization time window of $i(p)$ and arrival time window of $j(q)$. We clarify this idea with the following theorem.

Theorem 2. *For any trips $i(p)$, $j(q)$, and any synchronization node $b \in B^{ij}$ of SBT, $S_p^{ib} \bigcap A_q^{jb} = \emptyset$, if and only if, Y_{pq}^{ijb} is forced to be zero due feasibility, and constraints (3.4) and (3.5) related to these indices are redundant.*

Proof. ($\Rightarrow$) Let p and q be two trips of lines i and j, respectively, such that $S_p^{ib} \bigcap A_q^{jb} = \emptyset$ for a synchronization node b. Now, suppose that $Y_{pq}^{ijb} = 1$ is feasible.

By Property 2, there exist feasible values $\overline{X}_p^i \in D_p^i$ and $\overline{X}_q^j \in D_q^j$, such that $Y_{pq}^{ijb} = 1$. Then, $\left(\overline{X}_p^i + t^{ib}\right) \in A_p^{ib}$, $\left(\overline{X}_q^j + t^{jb}\right) \in A_q^{jb}$ and

$$\left[\left(\overline{X}_p^i + t^{ib}\right) + w^b, \left(\overline{X}_p^i + t^{ib}\right) + W^b\right] \subseteq S_p^{ib}.$$

We have Y_{pq}^{ijb} is feasible, therefore $w^b \leq \left(\overline{X}_q^j + t^{jb}\right) - \left(\overline{X}_p^i + t^{ib}\right) \leq W^b$. Then,

$$\left(\overline{X}_q^j + t^{jb}\right) \in \left[\left(\overline{X}_p^i + t^{ib}\right) + w^b, \left(\overline{X}_p^i + t^{ib}\right) + W^b\right] \subseteq S_p^{ib}.$$

Therefore, it follows that $\left(\overline{X}_q^j + t^{jb}\right) \in S_p^{ib} \bigcap A_q^{jb} = \emptyset$, which is a contradiction.

($\Leftarrow$) Let p and q be two trips of lines i and j, respectively, such that Y_{pq}^{ijb} must be zero for a synchronization node b. Now, suppose $S_p^{ib} \bigcap A_q^{jb} \neq \emptyset$. Then, there exists an element a such that, $a \in A_q^{jb}$ and $a \in S_p^{ib}$.

By definition of window A_q^{jb}, element a can be expressed as $a = \left(\overline{X}_q^j + t^{jb}\right)$, for some feasible $\overline{X}_q^j \in D_q^j$. By definition of window S_p^{ib}, element a can be expressed as

$a = \left(\overline{X}_p^i + t^{ib}\right) + \gamma$, where $w^b \leq \gamma \leq W^b$, for some feasible $\overline{X}_p^i \in D_p^i$. Then,

$$\left(\overline{X}_q^j + t^{jb}\right) = \left(\overline{X}_p^i + t^{ib}\right) + \gamma$$

$$\Rightarrow \left(\overline{X}_q^j + t^{jb}\right) - \left(\overline{X}_p^i + t^{ib}\right) = \gamma$$

$$\Rightarrow w^b \leq \left(\overline{X}_q^j + t^{jb}\right) - \left(\overline{X}_p^i + t^{ib}\right) \leq W^b.$$

Therefore, $Y_{pq}^{ijb} = 1$ is feasible since $w^b - M\left(1 - Y_{pq}^{ijb}\right) \leq \left(\overline{X}_q^j + t^{jb}\right) - \left(\overline{X}_p^i + t^{ib}\right) \leq W^b + M\left(1 - Y_{pq}^{ijb}\right)$, i.e., satisfies constraints (3.4) and (3.5) which is a contradiction. □

Using Theorem 2, we can remove useless variables and constraints. The preprocessing stage shown in Algorithm 1, calculates all the feasible departure, arrival, and synchronization time windows for each trip. Then, it implements Theorem 2 to identify when the synchronization of two trips is impossible, in this case, its corresponding synchronization variable and constraints of type (3.4) and (3.5) are eliminated.

Algorithm 1 : *Preprocessing*(SBT)

Input: SBT instance
Output: smaller formulation SBT'

1: **for** $(i \in I,\ j \in J(i),\ p \in \{1, \ldots, f^i\}, q \in \{1, \ldots, f^j\}$, and $b \in B^{ij})$ **do**
2: Compute A_p^{ib} and S_q^{jb}
3: **if** $\left(A_p^{ib} \bigcap S_q^{jb} = \emptyset\right)$ **then**
4: Eliminate variables Y_{pq}^{ijb} and related constraints (3.4) and (3.5)
5: **end if**
6: **end for**

After the preprocessing stage, the new SBT' formulation has considerably fewer variables and constraints than the original SBT formulation. Therefore, performance of the CPLEX's linear solver can be improved.

3.4.1 Experimental Results

We present some preliminary results in Table 3.1 which shows the implementation of a linear solver on SBT and SBT' formulations using GAMS/CPLEX configured with default options, except for execution time limited to 3 hours. We used a Sun Ray terminal

connected to a Sun Fire V440 server with 4 Ultra SPARC III processors running at 1602 GHz, and fitted with 8 GB RAM. Each test instance has a planning period of $T = 200$ minutes, 3 different lines, a frequency of $f^i = 20$ for each line i, and one synchronization node for pair of lines (1,2) and (1,3), and the same waiting time windows for both nodes. Column one shows the minimum and maximum headway times. Column two shows the waiting time window for all synchronization nodes b. Columns three and five show the relative gap (difference between the best feasible solution and the best upper bound obtained by CPLEX) for SBT and SBT', respectively. Columns four and six show the computational time needed by the linear solver of CPLEX to solve SBT and SBT', respectively. Finally, column seven shows the percentage of synchronization variables and constraints eliminated from SBT to obtain SBT'. We do not specify the computational time of the preprocessing stage since it is negligible (less than one second for all experiments in this study).

		SBT		SBT'		
$[h^i, H^i]$	$[w^b, W^b]$	gap	Time (sec)	gap	Time (sec)	% reduction
[9,11]	[3,5]	0%	694	0%	72	79%
[8,12]	[3,5]	108.4%	10800	11%	10800	69%
[5,15]	[3,5]	410.5%	10800	66%	10800	44%
[9,11]	[1,8]	35 %	10800	6.5%	10800	77%

Table 3.1: Results of solving small instances of SBT and SBT' using CPLEX's solver.

Note that even for small instances presented in Table 3.1, the formulations are sensitive to large headway flexibility parameters (rows 2 and 3). Although, the variation in waiting time windows also generates difficult instances (row 4), the formulations are not as sensitive as they are in headway time variations. This behavior is also present in larger instances of the problem. However, note that in some instances that are difficult to solve (rows 2–4), there are high levels of eliminated variables using the preprocessing stage. The new SBT' formulation has at least 44% fewer synchronization variables and constraints than the SBT formulation. This behavior is also present in larger instances. In particular, we see high percentage levels of eliminated variables for instances with small headway flexibility parameters $\left(\frac{\delta^i}{\left(\frac{T}{f^i}\right)} \leq 0.30\right)$. Therefore, the gap of the solutions obtained using SBT' formulation are improved. However, there are still large gaps inclusive in small

instances.

Although the constraint propagation is used to define our preprocessing stage, it can be also used to define solution algorithms since it give us information about the feasible space and potential synchronizations. Particularly, Section 4 present several families of valid inequalities to remove fractional solutions and obtain high quality solutions for SBT. One of these inequalities is based on Theorem 2. Moreover, Section 5 present several metaheuristic algorithms which use our constraint propagation to obtain high quality solutions for SBT and multiple period timetabling problems.

3.5 Chapter conclusions

In this chapter, we addressed Monterrey's bus network. This network is composed of several private companies and the government does not have a strong influence in the operation of the bus lines. We focus on timetables generation meant for companies, while passengers only have knowledge of an estimated frequency. We address two main issues at some stops of the network: avoid bus bunching of different lines and allow well timed passenger transfer. Bus network planners in Monterrey do not have access to real-time optimization tools. Therefore, one sought to *optimize* the performance of the whole bus network by controlling the departure time of the buses in the planning stage.

We propose a new formulation for the Synchronization Bus Timetabling Problem (SBT) that is sufficiently flexible to model Monterrey bus network. We prove that SBT is NP-hard, and this complexity proof also ensures the NP-hardness of problems presented in Ceder et al. [2001] and Eranki [2004]. Moreover, we identify that flexibility given by headway bounds is directly responsible for the intractability of the problem and also for the presence of multiple optimal solution. We find a special structure of the solution space of SBT, which gives us the possibility to identify and remove many decision variables and constraints using a preprocessing stage. Although this preprocessing stage improves the performance of CPLEXÂ´s linear solver, the intractability of small instances remains.

Obtaining a fast solution algorithm is beneficial to the planner in real-life transit net-

works since re-optimization is needed, for example, to improve the quality of a timetable considering other subproblems of the entire planning process. In this chapter we present a simple idea based in constraint propagation concept to define the preprocessing stage. However, the main idea could be used to define and explore the feasible solution space in several exact and metaheuristic solution approaches.

Chapter 4

Valid inequalities for SBT

Summary: Since our problem is relatively new, it is difficult to find accurate solution approaches. Indeed, exact solutions is a concern for most of the timetabling problems (different than the periodic case) which are often solved by heuristic algorithms. In our case, several of the ideas that we consider as good, were useless. However, the simple idea of our preprocessing stage is the the first step to develop efficient solution approaches. Pierre Fouilhoux and Safia Kedad-Sidhoum from the Laboratoire d' Informatique de Paris 6 make a vital collaboration to obtain the results presented in this chapter.

"It was like walking in a dark room. But step by step, we found some light."

4.1 Exact approaches

Some closely related problems on transport networks have been studied using integer programming approaches. We can quote the scholar bus scheduling problem introduced in Fügenschuh [2009] where starting times of schools and starting times of scholar buses must be synchronized to minimize the number of vehicles to transport all students. The authors develop different families of valid inequalities leading to a branch-and-cut algorithm. In Quadrifoglio et al. [2008], the authors optimize a weighted objective function based on vehicle resources and quality service for a transport network. They define logic constraints to reduce the feasible space which allows to reduce up to 90% of the CPU time when these cuts are added at the beginning of a branch-and-bound algorithm. In railway systems, timetabling problems have been extensively studied using integer programming

techniques. In particular, the periodic timetabling case has special structure (not present in SBT) that favors the design of valid inequalities commonly implemented in branch-and-cut algorithms [Caimi et al., 2011, Giesemann, 2002, Liebchen, 2004, 2007, Liebchen and Möhring, 2002, 2007, Schröder and Solchenbach, 2006]. Unfortunately, timetabling problems in urban transport networks do not share the structure of periodic timetabling and a diversity of solution algorithms is developed. A comprehensive review can be found in Guihaire and Hao [2008b].

An inequality is said to be *valid* for a MIP formulation if every solution of the MIP formulation satisfies this inequality. Consequently, a valid inequality can be added to a MIP in order to obtain a stronger formulation. Our main contributions are to define four families of valid inequalities that will be used for the SBT. The first two families of valid inequalities bound the number of possible synchronizations that can occur at a node for a specific trip. The two other families are obtained from the previous ones using a generic lifting procedure. Additionally, some inequalities of the SBT MIP formulation are tightened by lowering some coefficients. We solve the enhanced SBT MIP with a standard linear solver (CPLEX 12.3). Optimal solutions for most of the real size instances are found. For all the instances, solutions with less than 3% of deviation from the optimum are found in less than five minutes.

The rest of this chapter is organized as follows. We enhance the BTP MIP with our proposed valid inequalities defined in Section 4.2. A generic lifting procedure is introduced in Section 4.3 and is applied in order to produce another two families of valid inequalities. Section 4.4 presents a way to tighten some inequalities. Experimental results for instances based on a real transit network are presented in Section 4.5 where we show the impact of some combination of the valid inequalities families. Finally, chapter conclusions and future research areas are addressed in Section 4.6.

4.2 Valid Inequalities

An alternative to obtain tighter formulations for the SBT consists in adding valid inequalities to SBT MIP before its resolution by a linear programming solver. In fact, adding valid inequalities permits to cut fractional solutions of the linear relaxations of integer programs or to cut non-optimal feasible solutions Wolsey [1998], Nemhauser and Wolsey [1999]. In the following, we introduce two families of valid inequalities for the synchronization bus timetabling problem obtained from the headway parameters and the propagation of constraints (3.1), (3.2), and (3.3).

4.2.1 Synchronization inequalities

We can take advantage of the headway parameters to define a family of valid inequalities for each trip to be synchronized. Let us consider two lines i and j to be synchronized at a given node b such that the minimum headway time h^j of line j is greater than the length of the waiting time window of node b, i.e., $h^j > W^b - w^b$. If a trip $i(p)$ synchronizes with another trip $j(q)$, the synchronization of $i(p)$ with trips $j(q-1)$ or $j(q+1)$ is impossible since there would not be enough time units to ensure feasibility. Generalizing the previous idea, we obtain the following result.

Let i, j be two distinct lines with $j \in I(i)$ and $b \in B^{ij}$, the maximum number of synchronizations between one trip of line i and all the trips of line j is $1 + \left\lfloor \dfrac{W^b - w^b}{h^j} \right\rfloor$.

Proof. Let us suppose that there exists a feasible solution $(\widetilde{X}, \widetilde{Y})$ of SBT and a trip $i(p)$ such that there exist $r > 1 + \left\lfloor \frac{W^b - w^b}{h^j} \right\rfloor$ trips $q_1 < q_2 < \cdots < q_r$ of line j that may synchronize with trip $i(p)$. We will show that it is impossible to schedule these synchronized trips because of the imposed minimal headways between trips. Thus, the arrival times of these trips are within the feasible synchronization time window $\left[\widetilde{X}^i_p + t^{ib} + w^b, \widetilde{X}^i_p + t^{ib} + W^b\right]$. Therefore, the difference between the arrival times $\widetilde{X}^j_{q_r} + t^{ib}$ and $\widetilde{X}^j_{q_1} + t^{ib}$ of trips $j(q_1)$ and $j(q_r)$ must be less than $W^b - w^b$, consequently we obtain $\widetilde{X}^j_{q_r} - \widetilde{X}^j_{q_1} \leq W^b - w^b$. However, since solution $(\widetilde{X}, \widetilde{Y})$ corresponds to a regularly spaced schedule, we have that

$\widetilde{X}^j_{q_{l+1}} - \widetilde{X}^j_{q_l} \geq h^j$. We then have

$$\widetilde{X}^j_{q_r} - \widetilde{X}^j_{q_1} \geq (r-1)h^j \geq \left(1 + \left\lfloor \frac{W^b - w^b}{h^j} \right\rfloor\right) h^j > \left(1 + \left(\frac{W^b - w^b}{h^j} - 1\right)\right) h^j = W^b - w^b$$

which is a contradiction. □

Using Lemma 4.2.1, we derive the following inequalities that will be called *synchronization inequalities*.

$$\sum_{q=1}^{f^j} Y^{ijb}_{pq} \leq 1 + \left\lfloor \frac{W^b - w^b}{h^j} \right\rfloor \quad \forall i \in I, j \in I(i), b \in B^{ij}, \forall p \in \{1, \ldots, f^i\} \tag{4.1}$$

$$\sum_{p=1}^{f^i} Y^{ijb}_{pq} \leq 1 + \left\lfloor \frac{W^b - w^b}{h^i} \right\rfloor \quad \forall i \in I, j \in I(i), b \in B^{ij}, \forall q \in \{1, \ldots, f^j\} \tag{4.2}$$

Let us consider two lines i, j with $j \in I(i)$, a node $b \in B^{ij}$, and a trip $i(p)$. Given a solution $(\widetilde{X}, \widetilde{Y})$ of SBT, we can remark that $\sum_{q=1}^{f^j} \widetilde{Y}^{ijb}_{pq}$ is exactly the number of synchronization between trips of line j and trip $i(p)$ at node b. Consequently, we have the following theorem.

Theorem 3. *Synchronization inequalities (4.1) and (4.2) are valid for the SBT MIP.*

4.2.2 Headway inequalities

Using similar ideas, we can devise the following class of inequalities.

$$Y^{ijb}_{pq} + \sum_{q'=q+1}^{f^j} Y^{ijb}_{pq'} + \sum_{p'=p+1}^{f^i} Y^{ijb}_{p'q} \leq 1 + \left\lfloor \frac{W^b - w^b}{\min(h^i, h^j)} \right\rfloor \quad \begin{array}{c} \forall\ i \in I, j \in I(i), b \in B^{ij}, \\ p \in \{1, \ldots, f^i\}, q \in \{1, \ldots, f^j\} \end{array} \tag{4.3}$$

$$Y^{ijb}_{pq} + \sum_{q'=1}^{q-1} Y^{ijb}_{pq'} + \sum_{p'=1}^{p-1} Y^{ijb}_{p'q} \leq 1 + \left\lfloor \frac{W^b - w^b}{\min(h^i, h^j)} \right\rfloor \quad \begin{array}{c} \forall\ i \in I, j \in I(i), b \in B^{ij}, \\ p \in \{1, \ldots, f^i\}, q \in \{1, \ldots, f^j\}. \end{array} \tag{4.4}$$

We will denote these inequalities as *headway inequalities* as they depend on the value of the desired mininimum headway between trips. We then have the following result.

Theorem 4. *Headway inequalities (4.3) and (4.4) are valid for the SBT MIP.*

Proof. We will consider an inequality (4.3) corresponding to the two trips $i(p)$, $j(q)$ and a node $b \in B^{ij}$ (the proof for inequalities (4.4) is analogous). Let $(\widetilde{X}, \widetilde{Y})$ be a feasible solution of SBT. We consider the two quantities c and r defined as follows

$$r = \sum_{q'=q+1}^{f^j} \widetilde{Y}_{pq'}^{ijb} \text{ and } c = \sum_{p'=p+1}^{f^i} \widetilde{Y}_{p'q}^{ijb}.$$

If $r = 0$ then from inequality (4.2) we know that

$$\widetilde{Y}_{pq}^{ijb} + c \leq \sum_{p'=1}^{f^i} \widetilde{Y}_{p'q}^{ijb} \leq 1 + \left\lfloor \frac{W^b - w^b}{h^i} \right\rfloor \leq 1 + \left\lfloor \frac{W^b - w^b}{\min(h^i, h^j)} \right\rfloor.$$

Similarly, if $c = 0$, from inequality (4.1), we obtain that

$$\widetilde{Y}_{pq}^{ijb} + r \leq \sum_{q'=1}^{f^j} \widetilde{Y}_{pq'}^{ijb} \leq 1 + \left\lfloor \frac{W^b - w^b}{h^j} \right\rfloor \leq 1 + \left\lfloor \frac{W^b - w^b}{\min(h^i, h^j)} \right\rfloor.$$

Let us now suppose that both $r \geq 1$ and $c \geq 1$. Since $r \geq 1$ trip $i(p)$ synchronizes with r trips among trips $q+1, ..., f^j$ of line j. Let q' be the first of these trips that is synchronized with $i(p)$. Consequently, arrival times of trips $q', q'+1, ..., q'+(r-1)$ at node b are within the time window $\left[\widetilde{X}_p^i + t^{ib} + w^b, \widetilde{X}_p^i + t^{ib} + W^b\right]$. Since the minimal headway between two trips of line j is h^j, the arrival time of trip $j(q')$ at node b satisfies $\widetilde{X}_{q'}^j + t^{jb} \leq \widetilde{X}_p^i + t^{ib} + W^b - (r-1)h^j$. Since $\widetilde{X}_{q+1}^j \leq \widetilde{X}_{q'}^j$, we finally obtain

$$\widetilde{X}_{q+1}^j + t^{jb} \leq \widetilde{X}_p^i + t^{ib} + W^b - (r-1)h^j.$$

Similarly, since $c \geq 1$ there are c trips among $p+1, ..., f^i$ of line i that synchronizes with $j(q)$. Let p' be the last of these trips that synchronized with $j(q)$. thus we know that $\widetilde{X}_q^j + t^{jb} \geq \widetilde{X}_{p'}^i + t^{ib} + w^b$. Moreover, since trips $p' - (c-1), ..., p'$ are synchronized with $j(q)$ and because of the imposed minimum headways of line i, we know that $\widetilde{X}_q^j + t^{jb} \geq \widetilde{X}_{p'-(c-1)}^i + t^{ib} + w^b + (c-1)h^i$. Since $\widetilde{X}_{p'-(c-1)}^i \geq \widetilde{X}_{p+1}^i$, we finally obtain

$$\widetilde{X}_q^j + t^{jb} \geq \widetilde{X}_{p+1}^i + t^{ib} + w^b + (c-1)h^i.$$

Because of the minimum headway between trips of line j, we then get

$$h^j \leq (\widetilde{X}_{q+1}^j + t^{jb}) - (\widetilde{X}_q^j + t^{jb}) \leq \widetilde{X}_p^i - \widetilde{X}_{p+1}^i + W^b - w^b - (r-1)h^j - (c-1)h^i.$$

Because of the minimum headway between trips $i(p)$ and $i(p+1)$, we have $\widetilde{X}_{p+1}^i - \widetilde{X}_p^i \geq h^i$. Then, we obtain

$$h^j \leq -h^i + W^b - w^b - (r-1)h^j - (c-1)h^i,$$

which then becomes

$$rh^j + ch^i \leq W^b - w^b.$$

W.l.o.g. we can suppose that $h^j \leq h^i$. We then obtain

$$r + c \leq r + c\frac{h^i}{h^j} \leq \frac{W^b - w^b}{h^j}.$$

Thus we obtain

$$\widetilde{Y}_{pq}^{ijb} + r + c \leq 1 + \frac{W^b - w^b}{h^j},$$

which proves the validity of inequality (4.3). □

4.3 Lifting procedure

In this section, we present a generic lifting method that permits to compute new valid inequalities from the previous ones. The synchronization and headway inequalities introduced in Sections 4.2.1 and 4.2.2, respectively, use synchronization variables of a specific pair of lines (i, j) to be synchronized. Then, the main idea of our proposed lifted inequalities is to use the relation between synchronization variables belonging to different pair of

lines (i, j) and (i_0, j_0).

Let us first introduce a useful notation. Let ξ be the set of all 5-tuples (i, j, p, q, b) with $i \in I$, $j \in I(i)$, $p \in \{1, ..., f^i\}$, $q \in \{1, ..., f^j\}$ and $b \in B^{ij}$. Notice that set ξ is in a one-to-one correspondence with the set of Y's variables, that is to say, each 5-tuple $(i, j, p, q, b) \in \xi$ corresponds to a potential synchronization between trips $i(p)$ and $j(q)$ at node b so that trip $i(p)$ arrives first at node b.

Let us consider the following inequality:

$$\sum_{(i,j,p,q,b)\in E} Y_{pq}^{ijb} \leq \gamma \tag{4.5}$$

where $E \subset \xi$ and γ is an upper bound over the number of synchronizations in E. The inequalities of type (4.5) are clearly valid. We note that synchronization and headway inequalities belong to this class.

Given $(i_0, j_0, p_0, q_0, b_0) \in \xi$, we define $E_{p_0q_0}^{i_0j_0b_0}$ as the set of 5-tuples (i, j, p, q, b) of E so that trips $i(p)$ and $j(q)$ cannot be synchronized at node b if trips $i_0(p_0)$ and $j_0(q_0)$ are synchronized at node b_0 and trip $i_0(p_0)$ arrives first at node b_0, that is to say that, for a given solution $(\widetilde{X}, \widetilde{Y})$ of SBT MIP, $\widetilde{Y}_{p_0q_0}^{i_0j_0b_0} = 1$ implies $\widetilde{Y}_{pq}^{ijb} = 0$ for all $(i, j, p, q, b) \in E_{p_0q_0}^{i_0j_0b_0}$.

Then inequality (4.5) can be lifted to create the following inequality

$$\sum_{(i,j,p,q,b)\in E\cap E_{p_0q_0}^{i_0j_0b_0}} Y_{pq}^{ijb} \leq \gamma \left(1 - Y_{p_0q_0}^{i_0j_0b_0}\right) \tag{4.6}$$

which is clearly valid. In the following, we present a generic lifting procedure to obtain inequalities of type (4.6). This procedure will be applied to both synchronization and headway inequalities in order to obtain *lifted synchronization inequalities* and *lifted headway inequalities*.

In order to compute the lifted inequalities, given a 5-tuple (i, j, p, q, b), we will need to find a list of 5-tuples that can not be synchronized if (i, j, p, q, b) is synchronized. To achieve this, we will compare the *feasible departure time window* of the potentially synchronized trips, that is to say the time window during which a trip has to start.

The feasible departure time windows D_p^i shown in Section 3.4 constitutes a generic feasible departure time window. As it will turn out, when two trips synchronize, it will be possible to tighten this window. This can be done using logical inferences that are given by Theorem 2. Indeed, Theorem 2 ensures that Let $(i, j, p, q, b) \in \xi$ and D_p^i (resp. D_q^j) be a feasible departure time window of trip $i(p)$ (resp. $j(q)$). By setting $[\alpha, \beta] = \big[left(D_p^i) + t^{ib} + w^b, right(D_p^i) + t^{ib} + W^b\big] \cap \big[left(D_q^j) + t^{jb}, right(D_q^j) + t^{jb}\big]$, we obtain that (i, j, p, q, b) is synchronized with trip $i(p)$ arriving first at b if and only if $[\alpha, \beta] \neq \emptyset$. In this case, $[\alpha - t^{jb}, \beta - t^{jb}]$ is a tighter feasible departure time window for trip $j(q)$ and $D_p^i \cap [\alpha - W^b - t^{jb}, \beta - w^b - t^{jb}]$ is a tighter feasible departure time window for trip $i(p)$.

Knowing a tighter feasible departure time window D_p^i for a given trip $i(p)$, it is easy to infer tighter departure time window $D_{p'}^i$, for all trips $p' \neq p$. This can be done using the following procedure $Propagate(D_p^i)$.

Algorithm 2 $Propagate(D_p^i)$

1: **for** $p' = 1$ to $p - 1$ **do**
2: $\quad D_{p'}^i := D_{p'}^i \cap \big[left(D_p^i) + (p' - p)h^i, right(D_p^i) + (p' - p)H^i\big]$
3: **end for**
4: **for** $p' = p + 1$ to f^i **do**
5: $\quad D_{p'}^i := D_{p'}^i \cap \big[left(D_p^i) - (p - p')H^i, right(D_p^i) - (p - p')h^i\big]$
6: **end for**

Algorithm 3, called *Generic Lifting*, shows the steps to generate lifted inequalities. The general idea to obtain these inequalities is to consider a 5-tuple $(i_0, j_0, p_0, q_0, b_0) \in \xi$ and see what implications arise if the corresponding synchronization is set, that is to say when $Y_{p_0 q_0}^{i_0 j_0 b_0} = 1$. Step 2 consists in computing the initial feasible departure time windows D_p^i. Then, assuming that $(i_0, j_0, p_0, q_0, b_0)$ is synchronized at node b, we compute tighter feasible departure time windows $D_{p_0}^{i_0}$ and $D_{q_0}^{j_0}$ (Steps 3–5) using Theorem 2. Using procedure *Propagate*, we then update the departure time windows for the rest of the trips of lines i_0 and j_0 (Step 6). We then apply a specific procedure for every inequalities $\sum_{(i,j,p,q,b)\in E} Y_{pq}^{ijb} \leq \gamma$ of type (4.5): we determine a set E' of 5-tuples (i, j, p, q, b) of E that cannot be synchronized and we then can create the corresponding lifted inequalities. Unfortunately, these last Steps 7-9 must be dedicated to each of the two considered type

of inequalities in order to consider sets E' with known upper bounds. In fact, lines 7-10 have to be replaced by Algorithm 5 for synchronization inequalities and Algorithm 6 for headway inequalities.

Algorithm 3 *Generic Lifting*

1: **for** each $(i_0, j_0, p_0, q_0, b_0) \in \xi$ **do**
2: Compute generic D_p^i for every trip $i(p)$
3: $[\alpha, \beta] := \left[left(D_{p_0}^{i_0}) + t^{i_0 b_0} + w^{b_0}, right(D_{p_0}^{i_0}) + t^{i_0 b_0} + W^{b_0}\right] \cap \left[left(D_{q_0}^{j_0}) + t^{j_0 b_0}, right(D_{q_0}^{j_0}) + t^{j_0 b_0}\right]$
4: $D_{q_0}^{j_0} := [\alpha - t^{j_0 b_0}, \beta - t^{j_0 b_0}]$
5: $D_{p_0}^{i_0} := D_{p_0 b_0}^{i_0} \cap [\alpha - W^{b_0} - t^{j_0 b_0}, \beta - w^{b_0} - t^{j_0 b_0}]$
6: $Propagate(D_{p_0}^{i_0})$ and $Propagate(D_{q_0}^{j_0})$
7: **for** each inequality of type (4.5) $\sum_{(i,j,p,q,b)\in E} Y_{pq}^{ijb} \leq \gamma$ **do**
8: Find a set E' of 5-tuples (i, j, p, q, b) of E that cannot be synchronized
9: Create inequality $\sum_{(i,j,p,q,b)\in E'} Y_{pq}^{ijb} \leq \gamma \left(1 - Y_{p_0 q_0}^{i_0 j_0 b_0}\right)$
10: **end for**
11: **end for**

In order to find which trips cannot be synchronized, we introduce another procedure (Algorithm 4), called *Test synchronization*. This procedure determines if trip $i(p)$ cannot be synchronized with trip $j(q)$ due to the fact that $(i_0, j_0, p_0, q_0, b_0)$ is synchronized. Using Theorem 2, lines 2-5 tests if (i, j, p, q, b) is a potential synchronization and then, if this synchronization becomes impossible after the update of the departure time windows.

Algorithm 4 *Test_synchronization*(i, j, p, q, b)

1: $[\alpha, \beta] := \left[left(D_p^i) + t^{ib} + w^b, right(D_p^i) + t^{ib} + W^b\right] \cap \left[left(D_q^j) + t^{jb}, right(D_q^j) + t^{jb}\right]$
2: **if** $([\alpha, \beta] \neq \emptyset)$ **then**
3: $D_q^j := [\alpha - t^{jb}, \beta - t^{jb}]$
4: $D_p^i := D_p^i \cap [\alpha - W^b - t^{jb}, \beta - w^b - t^{jb}]$
5: **if** $\left[left(D_p^i) + t^{ib} + w^b, right(D_p^i) + t^{ib} + W^b\right] \cap \left[left(D_q^j) + t^{jb}, right(D_q^j) + t^{jb}\right] = \emptyset$ **then**
6: Return False
7: **end if**
8: **end if**
9: Return True

By assuming a variable $Y_{p_0 q_0}^{i_0 j_0 b_0} = 1$, the lifted inequalities find affected synchronization variables of other pair of lines $(i, j) \neq (i_0, j_0)$ to be synchronized. For creating lifted synchronization inequalities using (4.1), Algorithm 5 enumerates every synchronization

inequality of type (4.1) which can be affected by the modification of lines i_0 and j_0 (Steps 1-4). Remark that each of these inequalities is associated to a 4-tuple (i, j, p, b) (Steps 1-2). Each potential trip q is then tested by Algorithm 4 to know if (i, j, p, q, b) becomes impossible and we then create a lifted synchronization inequality from the set E' which represents the set of impossible synchronizations.

Algorithm 5 *Create lifted synchronization inequalities using* (4.1)

1: **if** $(\{i, j\} \cap \{i_0, j_0\} \neq \emptyset$ **and** $(i, j, b) \neq (i_0, j_0, b_0))$ **then**
2: **for** $(p = 1$ to $f^i)$ **do**
3: $E' := \emptyset$
4: **for** $(q = 1$ to $f^j)$ **do**
5: **if** $Test\_synchronization(i, j, p, q, b)$= False **then**
6: $E' := E' \cup \{(i, j, p, q, b)\}$
7: **end if**
8: **end for**
9: Create $\sum_{(i,j,p,q,b) \in E'} Y_{pq}^{ijb} \leq \left(1 + \left\lfloor \frac{W^b - w^b}{h^j} \right\rfloor\right) \left(1 - Y_{p_0 j q_0}^{i_0 j_0 b_0}\right)$
10: **end for**
11: **end if**

The procedure to generate lifted synchronization inequalities using (4.2), is analogous to Algorithm 5. Using similar ideas, Algorithm 6 enumerates headway inequalities (4.3) and computes for each of them a lifted headway inequality and the procedure to generate lifted headway inequalities using (4.4), is analogous to Algorithm 6.

4.4 Tightening inequalities (3.4) and (3.5)

An important aspect in integer programming is to compute tight parameters to reduce the computational time of solving the linear relaxation of integer programs. In a similar way that we use feasible departure, arrival, and synchronization time windows to define lifting inequalities, we can use them to bound big M parameters for constraints (3.4) and (3.5) of the SBT MIP.

We can recall that the earliest arrival time of trip $j(q)$ at node b is $left(D_q^j) + t^{jb}$ and the latest arrival time of trip $i(p)$ at node b is $right(D_p^i) + t^{ib}$. Therefore, the minimum difference of arrival times between trips $j(q)$ and $i(p)$ at node b is $right(D_p^i) + t^{ib} -$

Algorithm 6 *Create lifted headway inequalities using* (4.3)

if $(\{i,j\} \cap \{i_0, j_0\} \neq \emptyset$ **and** $(i,j,b) \neq (i_0, j_0, b_0))$ **then**
  for $(p = 1$ to f^i **and** $q = 1$ to $f^j)$ **do**
    $E' := \emptyset$
    for $(q' \geq q$ to $f^j)$ **do**
      if $Test\_synchronization(i,j,p,q',b)$= False **then**
        $E' := E' \cup \{(i,j,p,q',b)\}$
      end if
    end for
    for $(p' > p$ to $f^i)$ **do**
      if $Test\_synchronization(i,j,p',q,b)$= False **then**
        $E' := E' \cup \{(i,j,p',q,b)\}$
      end if
    end for
    Create $\sum_{(i,j,p,q,b)\in E'} Y_{pq}^{ijb} \leq \left(1 + \left\lfloor \frac{W^b - w^b}{\min(h^i, h^j)} \right\rfloor\right)\left(1 - Y_{p_0 q_0}^{i_0 j_0 b_0}\right)$
  end for
end if

$left(D_q^j) - t^{jb}$. Consequently, given a solution $(\tilde{X}, \tilde{Y})$ of SBT MIP, we know that

$$\left(\tilde{X}_{qb}^j + t^{jb}\right) - \left(\tilde{X}_{pb}^i + t^{ib}\right) \geq right(D_p^i) + t^{ib} - left(D_q^j) - t^{jb}.$$

Similarly, we can remark that the maximum difference of arrival times between trips $j(q)$ and $i(p)$ at node b satisfies the following inequality.

$$\left(\tilde{X}_{qb}^j + t^{jb}\right) - \left(\tilde{X}_{pb}^i + t^{ib}\right) \leq right(D_q^j) + t^{jb} - left(D_p^i) - t^{jb}.$$

In the basis of the above, for the given Constraints (3.4) and (3.5) corresponding to $(i,j,p,q,b) \in \xi$, we can replace M by $m_{pq}^{ijb} = right(D_p^i) + t^{ib} - left(D_q^j) - t^{jb}$ and $M_{pq}^{ijb} = right(D_q^j) + t^{jb} - left(D_p^i) - t^{jb}$.

4.5 Experimental Results

To perform the experimental analysis, we use solver CPLEX 12.3 on a iMac OS X with an Intel Core 2 Duo 3.06 GHz processor and 4 GB RAM. We design an instances generator

based on information provided by a company of Monterrey's transit network. The next section show the different types of instances.

4.5.1 Instances

The instances size is determined by the number of lines $|I|$, the number of synchronization nodes $|B|$, and the flexibility parameter δ^i which is responsible of the size of the feasible solution space of the departure time variables. As it is mentioned in Section 3.4, the larger the flexibility δ^i is, the harder is the instance. The name of the instance types and their parameters are summarized in Table 4.1.

Instance	**T1**	**T2**	**T3**	**T4**	**T5**	**T6**	**T7**	**T8**	**T9**
$\|I\|$	15	15	40	40	100	100	200	200	200
$\|B\|$	3	3	8	8	20	20	40	40	150
$\frac{\delta^i}{\left(\frac{T}{f^i}\right)} \in$	[.10,.20]	[.25,.35]	[.10,.20]	[.25,.35]	[.10,.20]	[.25,.35]	[.10,.20]	[.25,.35]	[.25,.35]

Table 4.1: Instance types and their parameter values. For each instance type, 10 instances were generated.

All the instance types have the following common characteristics: a planning period of $T = 240$ minutes; the frequency f^i for each line i is randomly generated between [13,18]; the travel time t^{ib} from depot to synchronization node b for each line i is between [20,60]; the minimum (maximum) waiting time for each synchronization node b is within [3,5] ([9,12]); finally, the number of different pairs of lines to synchronize at each node b is between 1 and 7. We randomly generate ten instances for each one of the nine instance types (a total of 90 instances) to analyze the algorithm performance. Notice that instances of type T9 have a large number of synchronization nodes. In fact, we use these instances to reach the limits of our solution approach.

The computational effort to generate the valid inequalities for SBT MIP is negligible (less than one second) for all type of instances. Therefore, we measure the execution time of CPLEX 12.3 for solving the SBT MIP using different combinations of the valid inequalities proposed in this work.

4.5.2 Solve to optimality

To find optimal solutions for SBT, we use CPLEX's solver with default options, except for the gap (relative deviation from the best feasible solution and the best upper bound) that is set to 0% and limited to one hour of execution time. As we mentioned before, we have ten instances for each instance type. Then, for each instance type we show the mean gap (gap), mean time (time), and their standard deviation denoted as gap_dev and time_dev, respectively.

We implement all the possible combination of valid inequalities, but we present only the relevant results in Table 4.2 while details of all the experiments can be found in `http://yalma.fime.uanl.mx/~yasmin/Yasmin_Rios-Solis/Instances.html`. The labels for block of rows in Table 4.2 represent the families of valid inequalities used to strength SBT MIP before the execution of CPLEX. In the first block of rows, we can see that the original formulation of SBT presented is intractable. A remarkable difference arises when one family of valid inequalities is added to SBT. Individually, headway inequalities (4.3)-(4.4) (second block of rows) are the ones that lead to the best results. However, we can combine different families of valid inequalities to obtain better results than using a single family. In this case, combination of synchronization inequalities (4.1)-(4.2) and headway inequalities (4.3)-(4.4) (third block of rows) leads to the best results considering both gap and time for instance types T2, T6, and T9. Moreover, adding lifted headway inequalities (fourth block of rows) leads to the best results considering both gap and time for instance types T4 and T8. Lifted synchronization inequalities seem to be the weakest. In particular, the convergence of CPLEX is slower using all families of valid inequalities (fifth block of rows). For example, we do not find feasible solutions for instances of type T9 in an hour of computational time.

			T1	T2	T3	T4	T5	T6	T7	T8	T9
	none	gap(%)	47.87	154.75	68.40	174.01	60.92	181.14	153.29	288.76	557.56
		gap_dev	23.66	32.14	14.61	5.62	20.99	14.81	27.18	50.00	18.81
		time	3600	3600	3600	3600	3600	3600	3600	3600	3600
		time_dev	0.82	0.72	0.87	0.72	0.73	1.84	2.16	0.62	0.77
	(4.3)-(4.4)	gap(%)	0	0.12	0.26	0	0.38	0	0.07	0.01	42.99
		gap_dev	0	0.37	0.81	0	0.52	0	0.13	0.03	10.45
		time	10.04	362.31	419.08	15.23	2857.61	74.68	1935.55	696.22	3600
		time_dev	21.86	1138.46	1119.7	26.22	1305.87	51.128	1499.74	1109.73	2.61
	(4.1)-(4.4)	gap(%)	0	0	0.21	0	0.51	0	0.07	0.01	35.81
		gap_dev	0	0	0.66	0	0.62	0	0.15	0.02	12.81
		time	14.38	187.28	434.14	8.15	2729.5	72.25	1992.97	681.33	3600
		time_dev	27.40	582.30	1115.27	9.121	1387.95	63.075	1488.22	1055.17	2.66
(4.1)-(4.4)	(4.3)-(4.4)	gap(%)	0	0.14	0.20	0	0.45	0	0.34	0.01	37.22
		gap_dev	0	0.43	0.63	0	0.54	0	0.64	0.03	12.36
		time	16.38	363.32	460.93	7.58	2743.91	77.45	2001.99	665.40	3600
	lift	time_dev	39.87	1137.67	1114.64	6.34	1405.92	92.98	1222.56	1060.84	0
	all	gap(%)	0	0.15	0.23	0	0.67	0	0.15	0.09	-
		gap_dev	0	0.49	0.74	0	0.64	0	0.22	0.27	-
		time	28.55	369.68	487.57	32.28	3156.04	438.27	3153.94	1798.79	3600
		time_dev	57.39	1135.89	1099.53	37.44	1078.94	466.16	747.79	1263.53	0

Table 4.2: Results for the instance types T1-T9 using the linear solver of CPLEX 12.3 and different combinations of our proposed families of valid inequalities. Each value is the mean or deviation for the execution time or gap considering 10 instances of each type.

Obviously, there are limitations of our solution approach. For example, instances of type T9 cannot be solved using our proposed valid inequalities and CPLEX 12.3. However, to the best of our knowledge, an extremely large number of synchronization nodes (as in the instances T9) is not considered in real transit networks like the one in Monterrey, Mexico.

Considering all experiments, a large number of the instances were solved optimally using at least one family of valid inequalities. Moreover, non-optimal mean gaps in Table 4.2 of each specific instance type represent the existence of some particularly complex instance where the execution of CPLEX reaches the time limit. Another important result is the fast convergence of CPLEX to small gaps using our proposed valid inequalities. Particularly, most of the instances converge to gaps less than 3% in less than one minute. Therefore, we can use a small gap limit for CPLEX to obtain high quality solutions in seconds. To the best of our knowledge, these are the best results for the synchronization bus timetabling problem.

4.5.3 Solve till 3% of gap

To show the fast convergence of the CPLEX using the proposed valid inequalities, we implement inequalities (4.1)-(4.4) for instances T1-T8 using a stop criterion of 3% of relative gap. Table 4.3 show the numerical results.

	T1	T2	T3	T4	T5	T6	T7	T8
gap	1.938%	1.275%	2.557%	0.849%	2.108%	1.238%	1.738%	1.729%
gap dev	1.142	0.989	0.440	0.571	0.553	0.869	0.428	0.750
time	1.733	2.981	334.509	4.47	41.917	18.623	91.245	85.485
time dev	0.952	3.270	1040.41	3.958	31.510	9.941	75.568	64.206

Table 4.3: Results of solving instance types T1-T8 implementing inequalities (4.1)-(4.4) and CPLEX 12.3 with a stop criterion of 3% of relative gap.

Notice than our exact approach has a fast convergence to high quality solutions for instances T1-T9 of SBT. In summary, we obtain high quality solutions for most of the instances in a reasonable time (less than one hour) implementing our proposed solution approach. Moreover, we can use a specific stop criterion such as a small time or gap limit to obtain high quality solutions in short execution times. These characteristics are very important since recalculation of timetables is usually needed to obtain a solution of the entire transit network planning problem. Therefore, this study presents a tool for planners to improve the quality and efficiency of the whole transportation process.

4.6 Chapter conclusions

We define an exact solution approach for the NP-hard Synchronization Bus Timetabling problem (SBT). This problem determines regular spaced departure time for all the trips of each line to allow well timed passenger transfers and avoid bus bunching between different bus lines. The flexibility in the SBT given by headway bounds (instead of a fixed headway) allow us to define different families of valid inequalities to tighten the SBT MIP.

Our solution approach is strength the SBT MIP using our proposed valid inequalities and implement the linear solver of CPLEX 12.3. Numerical results show that high quality solutions (optimal for most cases) can be found for large instances of SBT in a short time.

Moreover, there is a fast convergence of our approach to solutions with less than 3% of relative deviation from the optimal solution in seconds.

Although, we obtain high quality solutions in a short time, there are interesting research such as determining the dimension of the valid inequalities proposed in this study. Another natural improvement for this work is to develop a polyhedra study along with a branch-and-cut approach to handle unsolvable instances by our approach. Integration of SBT with other subproblems of transit network planning such as vehicle and crew scheduling is a challenging research area. Moreover, the generalization of SBT to cover the entire day instead of short planning periods is needed to define accurate integrated approaches.

Chapter 5

Metaheuristic algorithms for Timetabling Problems

Summary: At the beginning of our work, we consider as many other studies present in literature, the timetabling problem in a short planning period. However, we are convinced that the multiperiod approach is necessary to generate an optimal timetable for the entire day. The results obtained in the previous chapter are very promising and we decide to extend the timetabling problem to the multiperiod case.

"After founding a good idea to solve our problem, we wanted to go a step forward."

5.1 Introduction

The timetabling generation problem is usually computed only once in a while (e.g., semestrally). However, this does not apply to this kind of network. Indeed, around 10% of vehicle disruptions arise per day (accidents, fines, break downs, public manifestations, etc.) and the high absenteeism of drivers requires the modification of the timetables almost every day.

In such a bus transit network, timetabling generation is part of operations and no longer part of the planning process. Therefore, it is crucial to be able to compute accurate timetables in minutes since the vehicle and crew scheduling are solved in a sequential manner and strongly depend on the timetabling solution quality. This iterative process can be executed several times until the planner is satisfied with the entire solution.

Timetables are usually built for each specific period of time during the day. But parameters such as the frequency of bus lines, travel times, and even headways vary throughout the day. Chapter 4 of Ceder [2007] presents an excellent combination of the different procedures to make smooth transitions between the different periods of the day based on partial timetables. However, these methodologies lead to suboptimal solutions for SBT since the synchronization of trips between different planning periods must be taken into account. Therefore, we propose the Multiperiod SBT problem for an entire day (MSBT) that considers smooth transitions between periods and synchronization events between trips belonging to different planning periods. We design six Iterated Local Search and two Variable Neighborhood Search algorithms based on feasible departure time windows that are applied by a constraint propagation methodology.

The rest of this chapter proceeds as follows. A discussion of solution algorithms is presented in Section 5.2. Section 5.3 presents the details of the formulation for MSBT. The main idea for generating our proposed solution algorithms is the constraint propagation of the feasible departure time windows of the trips, which is presented in Section 5.4. The different components of the metaheuristic algorithms are presented in Section 5.5 while the description of the entire algorithms is given in Section 5.6. The empirical results about the presented metaheuristics on real size instances are exhibited in Section 5.7. Finally, Section 5.8 summarizes the results and points to some lines of future research.

5.2 Solution algorithms for timetabling problems

SBT is solved in Chapter 4 using a several families of valid inequalities and the linear solver of CPLEX 12.3. With this procedure, we exactly solve large instances in minutes, considering a planning period of 240 minutes. Unfortunately, incorporating smooth transitions between periods to this strengthened MILP is, to the best of our knowledge, not possible, because of the variation of parameters for different planning periods. Moreover, assuming that the parameters of the different periods of the day are equal, as instance of MSBT can be seen as an instance of SBT with a large planning period of 20 hours. However, the enhanced MILP for MSBT does not give solutions close to the optimum in

a short time as it happens with shorter planning periods.

It is imperative to have high quality solutions for MSBT that cover 20 hours of the day, since vehicle and crew scheduling performed on partial timetables leads to suboptimal solutions. In the literature, constructive algorithms have been used to solve different SBT timetabling problems [Ceder and Tal, 2001, Ceder et al., 2001, Eranki, 2004]. However, defining efficient local search algorithms for timetabling problems is not an easy task. For example, in our case study, there is a large solution space and many different solutions yield the same value of the objective function.

A local search that takes advantage of constructive algorithms to explore the solution space is presented in Liu et al. [2007]. The main idea of the local search is to define different combinations of pairs of trips to be synchronized. Then, the departure times of the related trips are set using an algorithm proposed by Ceder and Tal [2001]. Guihaire and Hao [2008a, 2010a] define run shift and line shift operators wich are applied randomly within the domain of the related variables to design an iterated local search. Later, similar movements are implemented in Guihaire and Hao [2010b] but are limited by the vehicle schedule, i.e., it is possible to shift the departure times of a whole line or of a trip if the vehicle and driver schedules would not be modified. A similar idea is presented for clock face timetabling and the multidepot multitype vehicle scheduling problem in [van den Heuvel et al., 2008], where the authors implement shifting movements (1–5 or 10 minutes) in a local search to decrease the vehicle schedule costs. In all the previously mentioned studies, the shift operators are applied, and then, the evaluation function is considered, i.e., it is difficult to categorize attractive movements. On the other hand, Ceder [2011] address a timetabling problem to achieve even loads with minimum uneven headways at maximum load points. In this last study, the information about the even load is considered before applying the shift operators, i.e., not every possible shift is explored, since the search is limited by the information of the objective function to explore only attractive movements based on even loads.

A different kind of heuristic is presented in Wong et al. [2008], where the authors present a train timetabling problem with most of the characteristics of our case study, but

minimizing the waiting times as the objective function. The heuristic algorithm is based on iteratively solving a linear relaxation of the formulation, then, rounding fractional values of some variables. If the actual solution is not feasible, some rounded variables are released and the formulation is solved again, and so on. This kind of approach to our problem may lead to a exploration of a large number of binary variables in the objective function. Moreover, there are many different solutions leading to the same value of the objective function.

In this chapter, we propose eight metaheuristic algorithms for MSBT, which are of two types: six multistart Iterated Local Search (ILS) and two multistart Variable Neighborhood Search (VNS). One of the main contributions of this work are the neighborhood structures which the ILS and VNS algorithms rely on. Contrary to all the local searches in the existing literature, these new structures are based on feasible departure time windows that arise from the mathematical structure of MSBT MILP. Our solution algorithms are different than those presented in the literature, since they take advantage of the mathematical formulation to define and explore the feasible solution space, leading to high quality solutions for some instances of MSBT.

5.3 Mixed integer linear programming for MSBT

In this section, we propose an MILP for MSBT. This formulation is not a straightforward generalization of SBT MILP presented in Chapter 3. The main differences between SBT and MSBT are the smooth transitions and synchronization interaction between the different periods of the day. Indeed, the synchronization events depend on the values of the departure times, travel times, and waiting time parameters of each one of the trips. Figure 5.1 shows an example with two planning periods in the morning (6:00 to 8:00 and 8:00 to 11:00) and two lines sharing a synchronization node. The trips of lines 1 depart within the first planning period and the trips of line 2 depart within the second planning period. However, these departure times define two synchronizations (dashed lines) since their separation time at the common node is within the minimum and maximum waiting times. Notice that synchronizations between trips belonging to different planning periods

are considered. Therefore, we must define the waiting time parameters based on the trips to be synchronized. We show in Section 5.3.1 how to generate accurate waiting times parameters for MSBT.

Although we can obtain a timetable for an entire day by merging the solutions of SBT, the synchronization between trips of different planning periods, as in the case presented by Figure 5.1, would be ignored.

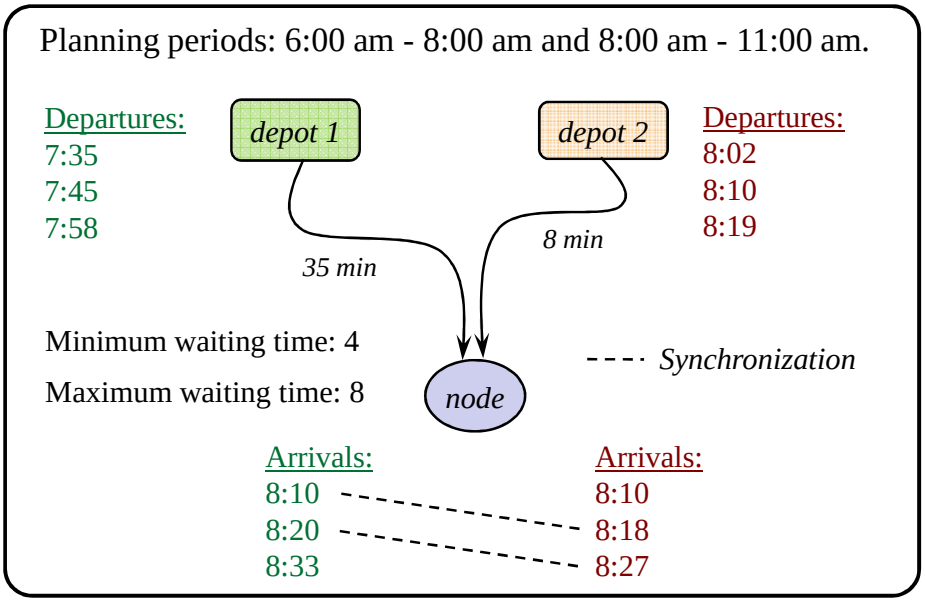


Figure 5.1: Example of a timetable for two lines with three trips each, and one synchronization node. Notice that the first trip of line 1 arrives first at the node and synchronizes with the second trip of line 2 since their separation time is 8 minutes, which is within the waiting time window [4,8].

In MSBT, we define the transit network as we did for SBT, i.e., we use the sets I, B, $J(i)$, and B^{ij} previously defined in Section 3.2. Now, let S be the set of planning periods such as morning rush hours period, afternoon period, night period, and so on. Parameter d_s with $s \in S$ represents the end of planning period s and the beginning of planning period $s+1$. Let f^i be the total number of trips of line i during the day, while f^i_s denotes the number of trips of i during period s. The travel time of trip p of line i from the initial node (depot) to node b is a parameter denoted by t^{ib}_p. Finally, the minimum and maximum waiting times to define a synchronization between trips $i(p)$ and $j(q)$ at node $b \in B^{ij}$ are w^{ijb}_{pq} and W^{ijb}_{pq}, respectively. The minimum and maximum headways for

line i at period s are defined as $h_s^i = \eta_s^i - \delta_s^i$ and $H_s^i = \eta_s^i + \delta_s^i$, where $\eta_s^i = \frac{d_s - d_{s-1}}{f_s^i}$ is the *even headway* of period s and δ_s^i is a headway flexibility parameter for period s.

The main decision variables for MSBT are precisely the ones defined for SBT, i.e., departure times variables for each trip $i(p)$, denoted by X_p^i and binary variables Y_{pq}^{ijb} to count a synchronization of trip $i(p)$ with trip $j(q)$ at node b. Then, our proposed MILP model for MSBT is the following.

$$\max \quad F_{\text{MSBT}}(Y) = \sum_{i \in I} \sum_{j \in J(i)} \sum_{b \in B^{ij}} \sum_{p=1}^{f^i} \sum_{q=1}^{f^j} Y_{pq}^{ijb} \tag{5.1}$$

$$\text{s.t.} \quad h_s^i \leq X_{p+1}^i - X_p^i \leq H_s^i \qquad \forall\, i \in I, p = \text{first}(s), \ldots, \text{last}(s) - 1, \; s \in S \tag{5.2}$$

$$d_{s-1} + \tfrac{h_s^i}{2} \leq X_{\text{first}(s)}^i \leq d_{s-1} + \tfrac{H_s^i}{2} \qquad \forall\, i \in I, s \in S \tag{5.3}$$

$$d_s - \tfrac{H_s^i}{2} \leq X_{\text{last}(s)}^i \leq d_s - \tfrac{h_s^i}{2} \qquad \forall\, i \in I, s \in S \tag{5.4}$$

$$X_q^j + t_q^{jb} - \left(X_p^i + t_p^{ib}\right) \geq w_{pq}^{ijb} - M\left(1 - Y_{pq}^{ijb}\right) \qquad \forall\, i \in I,\ j \in J(i),\ b \in B^{ij},\ p = 1, \ldots, f^i,\ q = 1, \ldots, f^j \tag{5.5}$$

$$X_q^j + t_q^{jb} - \left(X^{ip} + t_p^{ib}\right) \leq W_{pq}^{ijb} + M\left(1 - Y_{pq}^{ijb}\right) \qquad \forall\, i \in I,\ j \in J(i),\ b \in B^{ij},\ p = 1, \ldots, f^i,\ q = 1, \ldots, f^j \tag{5.6}$$

$$X_p^i \in \mathbb{R}, Y_{pq}^{ijb} \in \{0, 1\} \qquad \forall\, i \in I,\ j \in J(i),\ b \in B^{ij},\ p = 1, \ldots, f^i,\ q = 1, \ldots, f^j \tag{5.7}$$

The objective function (5.1) maximizes the number of synchronizations. The constraints (5.2) guarantee that the departure times of the trips of line i are almost evenly spaced during period s. To have smooth transitions between periods s and $s-1$, we use the constraints (5.3) and (5.4) that lead to an average headway for line i between the last trip of period $s-1$ (denoted by $\text{last}(s-1)$) and the first trip of s ($\text{first}(s)$), i.e., the separation between these trips must be within $[\frac{h_{s-1}^i + h_s^i}{2}, \frac{H_{s-1}^i + H_s^i}{2}]$. The constraints (5.5) and (5.6) allow the variable Y_{pq}^{ijb} to be one if the arrival times of the pth trip of line i and $j(q)$ at node b are within the time window $[w_{pq}^{ijb}, W_{pq}^{ijb}]$. If the variable Y_{pq}^{ijb} is equal to 0,

then these constraints are redundant, since M is a big number.

Ceder [2007] points out that the average transitions between periods can cause undesirable behavior such as uneven loads. Nevertheless, to the best of our knowledge, for our problem and the data we have, there is no possibility of implementing smooth transitions such as the one based on passenger load balance along time. As we show in Section 5.3.1, to allow synchronization events between trips belonging to different planing periods, the waiting time parameters should be carefully defined.

5.3.1 Waiting time parameters

Although most of the parameters of the MSBT MILP can be easily set by the planner, the waiting time windows must be carefully computed since the synchronization events strongly depend on them. In particular, the case of passenger transfer waiting time windows are simple: the planner sets reasonable minimum and maximum waiting time parameters such that they allow passengers to go from line i to j. Notwithstanding, the case where the planner wishes to avoid bus bunching of a pair of lines at a node considering parameter variations related with the different planning periods is not trivial. In the case of a single period timetabling, the planner could define accurate waiting time parameters since there are unique headway bounds for each bus line. But, the waiting time windows should depend on the headways to induce a correct separation of the trips. Then, in the case of MSBT, they also depend on the planning periods of the day.

To arrange synchronization at bus bunching nodes, we introduce the concept of *harmonized arrivals*. First, consider the case of a single planning period where two lines i and j are to be synchronized at some bus bunching node b such that the even headways η^i and η^j are 10 and 5 minutes, respectively (see Figure 5.2). We try to harmonize the arrivals of the different lines at the bus bunching node by defining the waiting time window $[w_{pq}^{ijb}, W_{pq}^{ijb}]$ as $\left[\frac{\max\{10,5\}}{2} - \frac{\min\{10,5\}}{2}, \frac{\max\{10,5\}}{2} + \frac{\min\{10,5\}}{2}\right]$, i.e., $[2.5, 7.5]$. The harmonized arrivals are shown in Figure 5.2. The horizontal arrow represents the time at the bus bunching node b where we want to separate arrivals between trips of lines i and j. Notice that for every trip $i(p)$, there could be two trips of line j synchronizing with it.

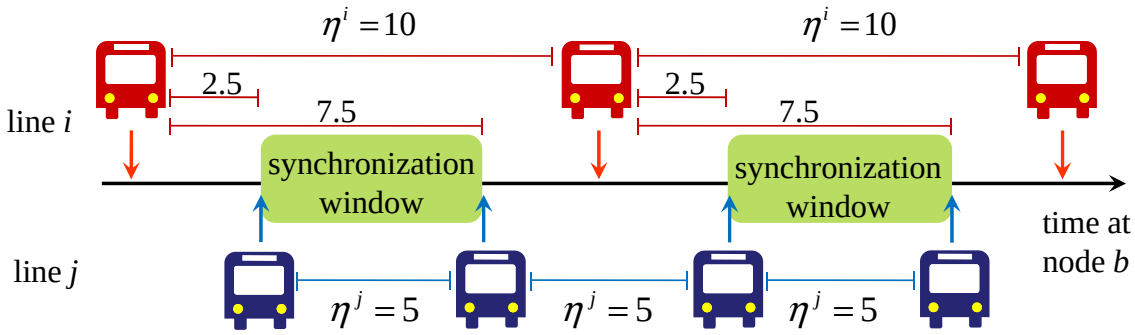


Figure 5.2: Example of the waiting time window computation such that the arrivals of the trips of lines i and j at node b are harmonized (even headways of $\eta^i = 10$ and $\eta^j = 5$).

In the previous case, it is possible that two trips of one line arrive at node b between consecutive arrivals of another line at the same node b. However, we also consider the case where only one trip of one line could arrive between consecutive arrivals of the other line. For example, in the case of $\eta^i = 10$ and $\eta^j = 6$, we define the waiting times window $[w^{ijb}_{pq}, W^{ijb}_{pq}]$ as $\left[\frac{\max\{10,6\}}{2} - \frac{\max\{10,6\}-\min\{10,6\}}{2}, \frac{\max\{10,6\}}{2} + \frac{\min\{10,6\}-\min\{10,6\}}{2}\right]$, i.e., $[3, 7]$. Figure 5.3 shows this case where we can notice that trip $i(p)$ synchronizes with trip $j(q)$ at some node b, and $j(q)$ arrives at the end of the synchronization time window of $i(p)$. Then, it is possible to synchronize $i(p+1)$ with $j(q+1)$ at the same node b, if $j(q+1)$ arrives at the beginning of the synchronization time window of $i(p+1)$.

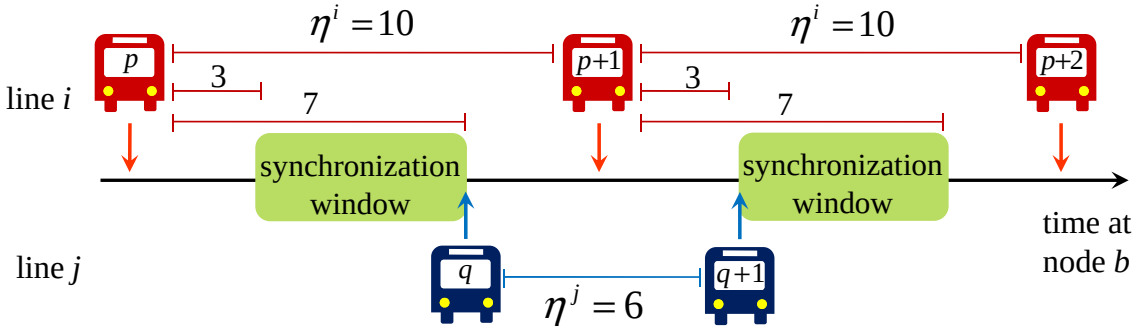


Figure 5.3: Example of the waiting time window computation such that the arrivals of the trips of lines i and j at node b are harmonized (even headways of $\eta^i = 10$ and $\eta^j = 5$).

The main idea of waiting time window generation is to try to harmonize the arrivals of the different lines at the bus bunching nodes by the maximization of the number of synchronizations.

Property 3. *Let* $\eta_{\max} = \max\{\eta^i_s, \eta^j_s\}$ *and analogously let* $\eta_{\min} = \min\{\eta^i_s, \eta^j_s\}$. *The follow-*

ing waiting time windows induce harmonized arrivals of lines i and j at node b within a single period s of the day:

$$\left[\frac{\eta_{\max}-\left(\left\lfloor\frac{\eta_{\max}}{\eta_{\min}}\right\rfloor-1\right)\eta_{\min}}{2}, \frac{\eta_{\max}+\left(\left\lfloor\frac{\eta_{\max}}{\eta_{\min}}\right\rfloor-1\right)\eta_{\min}}{2}\right] \quad if \ \left\lfloor\frac{\eta_{\max}}{\eta_{\min}}\right\rfloor > 1, \tag{5.8}$$

$$\left[\frac{\eta_{\max}}{2}-\frac{\eta_{\max}-\eta_{\min}}{\alpha}, \frac{\eta_{\max}}{2}+\frac{\eta_{\max}-\eta_{\min}}{\alpha}\right] \quad if \ \left\lfloor\frac{\eta_{\max}}{\eta_{\min}}\right\rfloor = 1. \tag{5.9}$$

This property does not need a proof since the waiting time windows are obtained by construction from the headway parameters. For the waiting time windows (5.8), more than one possible synchronization arises for each trip of the line with maximum even headway. The example shown in Figure 5.2 is of this type of time window, since $\lfloor\frac{\eta_{\max}}{\eta_{\min}}\rfloor = \lfloor\frac{10}{5}\rfloor = 2 > 1$. For waiting time windows (5.9), the headway parameters allow only one possible synchronization for each trip. To get a larger waiting time window than that illustrated in Figure 5.3, we set $\alpha = 1.5$ instead of $\alpha = 2$.

Now, let us consider the waiting time windows for trips that are from different planning periods of the day. For trips $i(p)$ and $j(q)$ such that p belongs to planning period s and q belongs to period s', the waiting time parameters can be expressed as functions of the headway values, i.e., $w^{ijb}_{pq}(\overline{\eta}^i_p, \overline{\eta}^j_q)$ and $W^{ijb}_{pq}(\overline{\eta}^i_p, \overline{\eta}^j_q)$ where $\overline{\eta}^i_p$ and $\overline{\eta}^j_q$ have the following different values depending of which trips may synchronize:

- If trip $i(p) = last(s)$, the waiting time window is defined using $\overline{\eta}^i_p = \frac{\eta^i_s+\eta^i_{s+1}}{2}$ since the separation time between $i(p)$ and the first trip of the next period $s+1$ is close the average headway related with these periods. We use a similar idea to define the parameter $\overline{\eta}^j_q$. Otherwise, $\overline{\eta}^i_p = \eta^i_s$.

- If trip $j(q) = first(s')$ or $j(q) = last(s'-1)$ and it can synchronize with $i(p)$, we consider the following three cases. Trip $i(p)$ may synchronize with trips of line j belonging to periods $s'-1$, s', or both. Then, to consider all the cases, we define the waiting time window using the headway time $\overline{\eta}^j_q = \min\left\{\eta^j_{s'-1}, \frac{\eta^j_{s'-1}+\eta^j_{s'}}{2}, \eta^j_{s'}\right\}$. Otherwise, $\overline{\eta}^j_q = \eta^j_{s'}$.

We remark that MSBT is NP-hard since SBT is a particular case of it. Indeed,

solving the MILP of MSBT by a commercial linear solver does not offer high qualit y solutions in reasonable time, as can be seen in Section 5.7. Moreover, it is not possible to add any of the families of valid inequalities proposed in Chapter 4 for the single period problem. The reason is that these valid inequalities strongly rely on the bounds of the number of synchronizations that are a function of the headway and waiting times. For MSBT, to the best of our knowledge, it is not trivial to generalize these bounds since we would need to identify which parameters are we going to use for each trip, i.e., identify which trips of a period can synchronized with trips of a different period.

5.4 Generalization of constraint propagation for MSBT

Our solution algorithms are based on feasible time windows for departures, arrivals, and synchronizations that are similar to the ones presented in Section 3.4 for SBT. These time windows allow to explore the feasible solution space, different neighborhood structures, and define efficient local search algorithms for the MSBT. To obtain these time windows, we use constraint propagation, which is a remarkable difference from the other heuristics presented in the literature, since we define a procedure to find potential synchronizations before applying an operator.

Now, we present the generalization of the constraint propagation idea presented in Section 3.4. For example, assume that the value for the departure time X_p^i belonging to a period s has been set. We know that the headways for period s are within $[h_s^i, H_s^i]$. Thus, we can define a feasible departure time window $D_{p'}^i$ for any other trip $i(p')$ in period s with $p' > p$ as $D_{p'}^i = \left[X_p^i + (p'-p)h_s^i, X_p^i + (p'-p)H_s^i\right]$. Analogously, for trips $i(p')$ in period s with $p' < p$, we set $D_{p'}^i = \left[X_p^i - (p-p')H_s^i, X_p^i - (p-p')h_s^i\right]$.

If we do not have already fixed values for the departure times, we can define D_p^i for trip $i(p)$ in period s using the bounds of trips $i(first(s))$ and $i(last(s))$. On the one hand, $\frac{h_s^i}{2}$ and $\frac{H_s^i}{2}$ are the lower and upper bounds for departure time $X_{first(s)}^i$, respectively. On the other hand, $d_s - \frac{H_s^i}{2}$ and $d_s - \frac{h_s^i}{2}$ are the lower and upper

bounds for departure time $X^i_{last(s)}$, respectively. Therefore, D^i_p based on these bounds is given by the intersection between $\left[\frac{h^i_s}{2} + (p - first(s))\, h^i_s, \frac{H^i_s}{2} + (p - first(s))H^i_s\right]$ and $\left[d_s - \frac{H^i_s}{2} - (last(s) - p)\, H^i_s, d_s - \frac{h^i_s}{2} - (last(s) - p)\, h^i_s\right]$.

The arrival time windows for each trip $i(p)$ at node b can be obtained by shifting D^i_p by t^{ib}_p time units, i.e., $A^{ib}_p = \left[left(D^i_p) + t^{ib}_p, right(D^i_p) + t^{ib}_p\right]$. Similarly, the synchronization time window for each trip $i(p)$ with trip $j(q)$ at node b is defined by $S^{ijb}_{pq} = \left[left(A^{ib}_p) + w^{ijb}_{pq}, right(A^{ib}_p) + W^{ijb}_{pq}\right]$. A useful result of these time windows is that we can detect *impossible synchronizations* with the following theorem which proof is straight forward using the idea of the proof of Theorem 2.

Theorem 5. *For any trips $i(p)$ and $j(q)$ at synchronization node $b \in B^{ij}$ of MSBT, $S^{ijb}_{pq} \bigcap A^{jb}_q = \emptyset$ if and only if Y^{ijb}_{pq} is forced to be zero due the feasibility and the constraints (5.5) and (5.6) related to these indices are redundant.*

The feasible departure, arrival and synchronization intervals obtained with the bounds of the first and last trip of each planning period are denoted by D^i_p, A^{ib}_p, and S^{ijb}_{pq}, respectively. From now on, the overline symbol " $\bar{}$ " on these intervals represents the feasible time window obtained using a partial solution. The notation " $\widehat{\ }$ " in these intervals represents the restricted feasible time windows that ensure a synchronization considering a partial solution (we introduce these intervals in the following section). By definition $\widehat{A} \subseteq \overline{A} \subseteq A$ for any feasible set A.

In the next section we show how these time windows can be used to define the operators for our proposed metaheuristic algorithms.

5.5 Components of Iterated Local Search and Variable Neighborhood Search

One of our main contributions are the eight metaheuristic algorithms for MSBT. Six of them are multistart Iterated Local Search algorithms (ILS) and two of them are multistart Variable Neighborhood Search algorithms (VNS).

On the one hand, ILS takes an initial solution Y (which can be a local optimum) and its iterations perform the following procedure. First, a perturbation is applied to Y to obtain an intermediate feasible solution Y'. Then, a local search algorithm finds a new improved solution Y'^*. If this new solution meets an acceptance criterion, the incumbent solution is now Y'^*, otherwise, the previous incumbent remains. We iterate until the algorithm reaches a stop criterion. ILS is a simple and fast tool that can be improved by increasing the quality of each one of its modules, where the interaction between intensification and diversification is critical and challenging [Lourenço et al., 2003].

On the other hand, as it is mentioned by Hansen and Mladenović [2003], the basic idea of VNS is systematic change of neighborhood within a local search. Given a list of different neighborhoods, the algorithm chooses one and moves to a random neighbor from it and improves it with a local search. If this improved point is better than the incumbent, then it becomes the new incumbent and the algorithm continues searching in this neighborhood. Otherwise, it chooses another neighborhood until an acceptance criterion is reached.

The components of ILS and VNS, such as constructive procedures, neighborhood structures, local search algorithms, and perturbation movements, are based on the constraint propagation concept presented in Section5.4. In this section we provide details of each one of the metaheuristic modules where we use a systematic constraint propagation, while in Section 5.6 we put together the modules to obtain our proposed metaheuristics for MSBT.

5.5.1 Constructive algorithms

Both ILS and VNS need constructive algorithms to build an initial solution. Algorithm 7 shows our randomized constructive algorithm for MSBT. For each line in the bus network, *Construct* randomly generates a departure time for each one of its trips within their feasible departure time window using a uniform distribution and considering the actual partial solution (step 4). The departure time windows are updated by constraint

propagation (step 6).

Algorithm 7 : *Construct*

Input: instance of MSBT
Output: solution Y for MSBT

1: **for** all lines $i \in I$ **do**
2: $\overline{D}^i_{first(s)} = D^i_{first(s)}$ and $\overline{D}^i_{last(s)} = D^i_{last(s)}$ for all period s
3: **for** $p = 1$ to f^i **do**
4: $X^i_p \longleftarrow \text{Unif}\left(\overline{D}^i_p\right)$
5: **if** $p \neq last(s)$ for all period s **then**
6: $\overline{D}^i_{p+1} \longleftarrow D^i_{p+1} \cap \left[X^i_p + h^i_s, X^i_p + H^i_s\right]$
7: **end if**
8: **end for**
9: **end for**

Construct assigns departure times to all the trips of the lines in lexicographic order, thus it ignores synchronization events. To build a solution with a larger number of synchronizations, we use the idea of *impossible synchronizations* given by Theorem 5 to establish a priority between the departure time variables.

The main idea is the following. Assume trip $i(p)$ synchronizes with trip $j(q)$ at node b (i.e., $Y^{ijb}_{pq} = 1$). We can use constraint propagation to compute the feasible departure time windows $\widehat{D}^{ijb}_{pq}$ for trip $i(p)$ and $\widehat{D}^{jib}_{qp}$ for trip $j(q)$ that would ensure $Y^{ijb}_{pq} = 1$. These departure time windows are defined as follows.

$$\widehat{D}^{jib}_{qp} = \left[left\left(\overline{S}^{ijb}_{pq} \cap \overline{A}^{jb}_q\right) - t^{jb}_q, right\left(\overline{S}^{ijb}_{pq} \cap \overline{A}^{jb}_q\right) - t^{jb}_q\right] \cap \overline{D}^i_p \tag{5.10}$$

$$\widehat{D}^{ijb}_{pq} = \left[X^j_q + t^{jb}_q - t^{ib}_p - W^{ijb}_{pq}, X^j_q + t^{jb}_q - t^{ib}_p - w^{ijb}_{pq}\right] \cap \overline{D}^i_p \tag{5.11}$$

By Theorem 5 it is possible to identify the set $E(Y^{ijb}_{pq} = 1)$ of impossible synchronizations assuming $Y^{ijb}_{pq} = 1$ in this updated feasible set. Therefore, we choose the variables inducing the fewer impossible synchronizations in a systematic way.

LConstruct, summarized by Algorithm 8, integrates these ideas. First, a list LSYNC with the possible synchronization events is defined. The first elements of LSYNC are the ones that induce the fewest impossible synchronizations while the last ones are the events that would cause many impossible synchronizations (step 1 of *LConstruct*). While

LSYNC is not empty, the synchronizations are defined iteratively as follows. The first variable of the list is selected and randomly sets the departure time of its related trips such that their synchronization is forced to happen (step 4). Next, the departure time windows and LYSTSYNC are updated. Then, the next variable in the list is selected, and so on. After explore the entire list, the remaining (and unassigned) departure times are randomly generated within their feasible departure time windows in step 9.

Algorithm 8 : *LConstruct*

Input: instance of MSBT
Output: solution Y for MSBT

1: LSYNC: increasing order list of the synchronization events Y_{pq}^{ijb} with respect to $\left|E(Y_{pq}^{ijb}=1)\right|$
2: **while** LSYNC $\neq \emptyset$ **do**
3: take first element $Y_{pq}^{ijb} \in$ LSYNC and compute $\widehat{D}_{pq}^{ijb}$ and $\widehat{D}_{qp}^{jib}$
4: $X_q^j \longleftarrow$ Unif$(\widehat{D}_{qp}^{jib})$, $X_p^i \longleftarrow$ Unif$(\widehat{D}_{pq}^{ijb})$, and $Y_{pq}^{ijb} \longleftarrow 1$
5: update feasible departure time windows for trips of lines i and j
6: LSYNC = LSYNC - $\left(\{Y_{pq}^{ijb}\} \cup E(Y_{pq}^{ijb}=1)\right)$
7: **end while**
8: **for** each trip $i(p)$ with unassigned departure time **do**
9: compute $\overline{D}_p^i$ and $X_p^i \longleftarrow$ Unif $\left(\overline{D}_p^i\right)$
10: **end for**

LConstruct is more time consuming than *Construct* but empirically its initial solution has a larger number of synchronizations. We now present the neighborhood structures that are used by the local search algorithms.

5.5.2 Trip neighborhoods

In this section we introduce three neighborhoods that rely on a trip shifting. The aim of these neighborhoods is to find solutions with an improved number of synchronization events by taking advantage of the departure, arrival, and synchronization time windows of the trips. Given a current feasible solution, the main idea is shifting the departure time of a trip within its feasible domain to force a synchronization with another trip. To achieve this, we first identify the feasible domain for the departure time of a single trip

$i(p)$ using constraint propagation, i.e.,

$$\overline{D}_p^i = \left[X_{(p-1)}^i + h_s^i, X_{(p-1)}^i + H_s^i\right] \cap \left[X_{(p+1)}^i - H_s^i, X_{(p+1)}^i - h_s^i\right] \cap D_p^i.$$

Suppose that $i(p)$ could synchronize with $j(q)$ at node b but in the actual feasible solution Y, the arrival time $X_q^j + t_q^{jb}$ of trip $j(q)$ is outside the synchronization time window $[X_p^i + t_p^{ib} + w_{pq}^{ijb}, X_p^i + t_p^{ib} + W_{pq}^{ijb}]$. Then, we shift the departure time X_p^i of $i(p)$ such that the synchronization window would also be shifted and coincide with the arrival time of $j(q)$, i.e., we move X_p^i within $\widehat{D}_{pq}^{ijb} = \left[X_q^j + t_q^{jb} - t_p^{ib} - W_{pq}^{ijb}, X_q^j + t_q^{jb} - t_p^{ib} - w_{pq}^{ijb}\right] \cap \overline{D}_p^i$. Figure 5.4 shows a feasible solution Y. The potential time window $\widehat{D}_{pq}^{ijb}$ (dashed line) illustrates where X_p^i should be to induce a synchronization with $j(q)$.

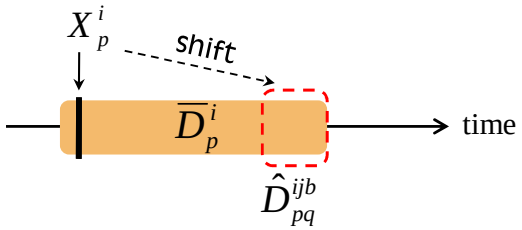


Figure 5.4: Departure time window of trip $i(p)$, and illustration of potential departure time window $\widehat{D}_{jq}^{ipb}$.

The operator $ST(Y, i(p), j(q), b)$ shifts departure time X_p^i inside $\overline{D}_{pq}^{ijb} \cap \widehat{D}_{pq}^{ijb}$ if this set is not empty. Since departure time variables are real, there is an infinite number of shifting movements that guarantee $X_p^i \in \overline{D}_{pq}^{ijb} \cap \widehat{D}_{pq}^{ijb}$. However, in Section 5.5.4 we develop a more efficient way to explore the search space. Now, we define neighborhood $N_{trip\_i}(Y)$ as the set of all solutions obtained by applying operator $ST(Y, i(p), j(q), b)$ to Y for all pairs of trips $(i(p), j(q))$ with $j \in J(i)$ and node $b \in B^{ij}$.

For neighborhood $N_{trip\_i}(Y)$, the departure time X_q^j of trip $j(q)$ is fixed and the departure time of trip $i(p)$ is shifted to achieve a synchronization. But in an analogous way, we can define shift operator $ST(Y, j(q), i(p), b)$ that moves trip $j(q)$ to achieve a synchronization with another trip $i(p)$ at node $b \in B^{ij}$. In particular, departure time X_q^j must be within $\widehat{D}_{qp}^{jib} = \left[X_p^i + t_q^{ib} + w_{pq}^{ijb} - t_q^{jb}, X_p^i + t_q^{ib} + W_{pq}^{ijb} - t_q^{jb}\right] \cap \overline{D}_q^j$ to guarantee that

$Y_{pq}^{ijb} = 1$. This idea leads to the neighborhood $N_{trip\_j}(Y)$ that contains all the solutions obtained by the implementation of operator $ST(Y, j(q), i(p), b)$ for all trips $i(p)$, $j(q)$ with $j \in J(i)$, and node b.

The previous two neighborhood structures are based on an operator that shifts the departure time of a single trip. However, we could simultaneously shift two departure times, one of line i and other of line j, to achieve a synchronization. Formally, operator $ST(Y, [i(p), j(q)], b)$ moves the trip X_q^j within $\overline{S}_{pq}^{ijb} \cap \overline{A}_q^{jb}$, and then, moves the trip X_p^i within $\left[X_q^j + t_q^{jb} - t_p^{ib} - W_{pq}^{ijb}, X_q^j + t_q^{jb} - t_p^{ib} - w_{pq}^{ijb}\right] \cap \overline{D}_p^i$. These movements guarantee that $Y_{pq}^{ijb} = 1$. Now, we define neighborhood N_{trip_ij} as the set of solutions obtained by implementing operator $ST(Y, [i(p), j(q)], b)$ for all trips $i(p)$, $j(q)$ with $j \in J(i)$, and $b \in B^{ij}$.

5.5.3 Line neighborhoods

In the previous section, we defined three neighborhood structures based on shifting operators for a single trip. A different type of neighborhood can be obtained by shifting all the trips of a specific line i, i.e., line i is forced to synchronize with line j. More precisely, let the line shifting operator $SL(Y, i, j, b)$ be the successive application of the operator $ST(Y, i(p), j(q), b)$ for the following pairs of trips: $(i(1), j(1))$, $(i(1), j(2)), \ldots,$ $(i(f^i), j(f^j - 1))$, and $(i(f^i), j(f^j))$. Thus, the neighborhood $N_{line\_i}(Y)$ contains all the solutions reached by applying the operator $SL(Y, i, j, b)$ to Y for all pair of lines $(i, j \in J(i))$.

Analogously to $SL(Y, i, j, b)$, we define operators $SL(Y, j, i, b)$ and $SL(Y, [i, j], b)$ by the successive implementation of operators $ST(Y, j(q), i(p), b)$ and $ST(Y, [i(p), j(q)], b)$, respectively. Then, the definition of neighborhoods $N_{line\_i}(Y)$ and $N_{line\_ij}(Y)$ is straight forward.

Forcing a synchronization between a pair of lines (solutions in $N_{line\_i}(Y)$, $N_{line\_j}(Y)$, and $N_{line\_ij}(Y)$) could destroy many other synchronizations related with these lines. Therefore, it could be beneficial to shift the line that has fewer synchronization nodes. This idea leads to another neighborhood, named $N_{line\_min}(Y)$, containing all the solutions reached by applying $SL(Y, i, j, b)$ if i has fewer synchronization nodes, or $SL(Y, j, i, b)$ in

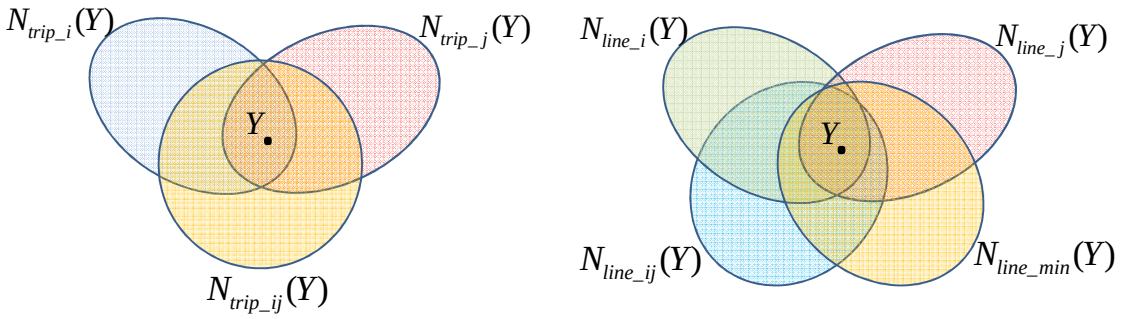


Figure 5.5: Left panel illustrates the neighborhoods $N_{trip\_i}(Y)$, $N_{trip\_j}(Y)$, $N_{trip\_ij}(Y)$, while the right one shows $N_{line\_i}(Y)$, $N_{line\_j}(Y)$, $N_{line\_ij}(Y)$, and $N_{line\_min}(Y)$.

other case, for all pair of lines $(i, j \in J(i))$.

Due to the possible synchronizations, our proposed neighborhood structures may have intersections, but none of the neighborhoods are contained one within the other, as is illustrated by Figure 5.5.

5.5.4 Local search algorithms

Designing efficient local search algorithms requires efficient procedures to explore the search space. Since departure time variables are real, there is an infinite number of shifting movements for trips or lines that guarantee synchronization events. Instead of explore an infinite number of neighbors, we define random shifting operators. For example, operator $ST(i(p), j(q), b)$ define all the shifting movements that guarantee $X_p^i \in \overline{D}_{pq}^{ijb} \cap \widehat{D}_{pq}^{ijb}$, i.e., operator $ST(i(p), j(q), b)$ synchronizes trip $i(p)$ with trip $j(q)$ at node b. Now, we define the random shift operator $randST(i(p), j(p), b)$ that makes $X_p^i = Unif\left(\overline{D}_{pq}^{ijb} \cap \widehat{D}_{pq}^{ijb}\right)$. Similarly, for each shifting operator presented in Sections 5.5.2 and 5.5.3.

The previous random shift operators allow to explore the search space in a more efficient way. Then, we define two local search algorithms for each neighborhood previously presented. For instance, consider N_{trip_i}. The local search $F_{trip\_i}(Y)$ moves to the first neighbor that surpasses in quality the actual feasible solution considering random shifting movements. $F_{trip\_i}(Y)$ is described by Algorithm 9.

The second type of local search is a greedy procedure that moves to the best neigh-

Algorithm 9 : $F_{trip\_i}(Y)$

Input: initial solution Y for MSBT
Output: improved feasible solution Y for MSBT

while there is no improvement in $F_{MSBT}(Y)$ **do**
  for each pair of lines i, and $j \in J(i)$, and node $b \in B^{ij}$ **do**
    for $p = 1$ to $p = f^i$ **do**
      for $q = 1$ to $q = f^j$ **do**
        $Y' \longleftarrow randST(Y, i(p), j(q), b)$
        $Y \longleftarrow Y'$ if $F_{\text{MSBT}}(Y') > F_{\text{MSBT}}(Y)$
      end for
    end for
  end for
end while

borhood based on random shifting movements. It is denoted by $G_{trip\_i}(Y)$ and its description is shown in Algorithm 10.

Algorithm 10 : $G_{trip\_i}(Y)$

Input: initial solution Y for MSBT
Output: improved feasible solution Y^* for MSBT

$Y^* \longleftarrow Y$
while there is no improvement in the number of synchronizations of Y **do**
  for each pair of lines i and $j \in J(i)$, and node $b \in B^{ij}$ **do**
    for $p = 1$ to $p = f^i$ **do**
      for $q = 1$ to $q = f^j$ **do**
        $Y' \longleftarrow randST(Y, i(p), j(q), b)$
        $Y^* \longleftarrow Y'$ if $F_{\text{MSBT}}(Y') > F_{\text{MSBT}}(Y^*)$
      end for
    end for
  end for
  $Y \longleftarrow Y^*$
end while

Analogously, we implement fist improvement and best improvement local searches for the rest of the neighborhoods. Since we have seven neighborhood structures, we define fourteen local search algorithms. In Section 5.6, these local searches are used as modules of the iterated local search and variable neighborhood search metaheuristics.

5.5.5 Perturbation

When a local search stops exploring the feasible solution, we propose a perturbation function that drastically modifies the synchronizations of a specific pair of lines ($i, j \in J(i)$) at some synchronization node $b \in B^{ij}$. As in any perturbation procedure, the value of the perturbed solution may decrease, however the idea is to modify the local maximum enough so as to reach a different zones of the solution space of MSBT.

The perturbation function is shown in Algorithm 11. First, there is an initialization phase that clears the departure times of all trips of lines i and j (step 1). In step 4, for each trips $i(p)$ and $j(q)$ that can synchronize, we compute their new feasible synchronization time windows $\overline{S}_{pq}^{ijb}$ using a constraint propagation based on the previously assigned departure time (denoted as $prev\_p$). In particular, this new synchronization interval is given by

$$\overline{S}_{pq}^{ijb} = \left[X_{prev\_p}^{i} + (p - prev\_p)\, h_s^i + t_p^{ib} + w_{pq}^{ijb}, X_{prev\_p}^{i} + (p - prev\_p)\, H_s^i + t_p^{ib} + W_{pq}^{ijb}\right].$$

Next, for a trip $j(q)$ that can synchronize with trip $i(p)$, we compute its new feasible arrival time window $\overline{A}_q^{jb}$ considering the previous assigned departure time (step 5):

$$\overline{A}_q^{jb} = \left[X_{prev\_q}^{j} + (q - prev\_q)\, h_s^j + t_q^{jb}, X_{prev\_q}^{j} + (q - prev\_q)\, H_s^j + t_q^{jb}\right].$$

If the arrival time window of trip $j(q)$ intersects with the synchronization time window of trip $i(p)$ (step 6), we randomly generate the departure time of trips $i(p)$ and $j(q)$ to ensure their synchronization (step 8). Finally, we calculate the departure times of the unassigned departure time trips considering their nearest (previous and posterior) assigned departure time trips (steps 13-15).

Algorithm 11 : $Perturbation(Y, i, j, b)$

Input: Feasible solution Y, pair of lines i and $j \in J(i)$, and node $b \in B^{ij}$
Output: Feasible perturbed solution Y'

clear departure times of all trips of lines i and j
for $p = 1$ to f^i **do**
  for $q = 1$ to f^j **do**
    compute $\overline{S}^{ijb}_{pq}$ considering $X^i_{prev\_p}$ for all $j(q)$ that could synchronize with $i(p)$
    compute arrival time window $\overline{A}^{jb}_q$ considering $X^j_{prev\_q}$
    if $i(p)$ can synchronize with $j(q)$ at b, i.e., $\overline{S}^{ijb}_{pq} \bigcap \overline{A}^{jb}_q \neq \emptyset$ **then**
      $X^j_q \longleftarrow \text{Unif}\left(\overline{S}^{ijb}_{pq} \bigcap \overline{A}^{jb}_q\right) - t^{jb}_q$
      $X^i_p \longleftarrow \text{Unif}\left(\left[X^j_q + t^{jb}_q - t^{ib}_p - W^{ijb}_{pq}, X^j_q + t^{jb}_q - t^{ib}_p - w^{ijb}_{pq}\right] \cap \overline{D}^i_p\right)$
      $Y^{ijb}_{pq} \longleftarrow 1$, $prev\_p = p$, and $prev\_q = q$
    end if
  end for
end for
for each unassigned departure time of trip $i(p)$ or $j(q)$ **do**
  search nearest previous and posterior assigned departure times of the trip and calculate $\overline{D}^l_p$
  $X^l_p \longleftarrow \text{Unif}\left(\overline{D}^l_p\right)$
end for

5.6 Metaheuristic algorithms

In this section we focus on the structure of two types of metaheuristics that use the local searches presented in Section 5.5 in order to solve MSBT. Six of them are multistart iterated local search algorithms (ILS) and are presented in Section 5.6.1, while the other two are multistart variable neighborhood search algorithms and are described in Section 5.6.2.

5.6.1 Multistart Iterated Local Search algorithms

Our ILS algorithms use different combinations of the constructive and local search algorithms presented in Section 5.5. Four of the ILS are single multistart iterated local searches while the other two are chained multistart iterated local searches (CILS). Table 5.1 shows the six ILS algorithms with the best performance from more than thirty algorithms using different combinations of the different components. The numbers in each

row indicate the order of the implemented component, while "-" means that the specific component is not included in that particular ILS algorithm. The block of columns "Main Iteration" shows the components implemented iteratively after the perturbation of the current solution. We implement a stop criterion of 10 iterations without improvements or a total of 50 iterations, each one of these iterations implements the perturbation procedure and the local search algorithms for all lines i, $j \in J(i)$, and synchronization node $b \in B^{ij}$. Finally, we execute the algorithm ten times to define the multistart approach.

	Construct and Improve									Main Iterations				
Algorithms	*Construct*	*LConstruct*	F_{trip_i}	F_{trip_j}	F_{line_i}	G_{line_i}	F_{line_j}	G_{line_j}	G_{line_ij}	F_{trip_i}	F_{trip_j}	F_{line_i}	F_{line_j}	F_{line_ij}
ILSa	1	-	-	-	3	-	2,4	-	-	-	-	6	5,7	-
ILSb	1	-	3	2,4	-	-	-	-	-	6	5,7	-	-	-
ILSc	-	1	-	-	-	3	-	2,4	-	-	-	6	5,7	-
ILSd	1	-	-	-	-	4	-	3,5	2	-	-	8	7,9	6

Table 5.1: Different Iterated Local Search algorithms implementing several neighborhood structures. The numbers in the table represent the order in which each component is implemented. For example, algorithm ILSa creates an initial solution with *Construct*, then implement local searches F_{line_j}, F_{line_i}, and F_{line_j}. In each iteration of ILSa the perturbation is implemented followed by local searches F_{line_j}, F_{line_i}, and F_{line_j}.

The two chained multistart sequential algorithms CILS start with one of the ILS presented in Table 5.1. Then, the obtained solution is given as an input to a second ILS, and so on. In particular, we define the algorithm CILSa as the chained implementation of ILSd, ILSa, and ILSc, while the CILSb algorithm implements ILSb, ILSd, and ILSa.

5.6.2 Multistart Variable Neighborhood Search algorithms

VNS algorithms take advantage of several neighborhood structures to diversify the search space, leading to different local optimum solutions. The two VNS procedures we present are based on Algorithm 12. The VNS take a random solution from one of the $k_{\max}$ neighborhoods (step 5). Then, this solution is improved with the best local search from the fourteen possibilities. If the number of synchronizations of the obtained solution does

not improve over the value of the current solution, neighborhood k is changed to $k+1$, otherwise, the current solution is updated (steps 6–10) and we continue exploring neighborhood k. Finally, we include a perturbation movement if there are no improvements from exploring all $k_{\max}$ neighborhoods after a fixed number of iterations (step 13).

Algorithm 12 : VNS

Input: instance of SBT
Output: feasible solution Y

1: $Y \longleftarrow LConstruct$, $k = 1$
2: **while** $F_{\text{MSBT}}(Y)$ is not improved after a fixed number of iterations **do**
3: $k = 1$
4: **while** $k \leq k_{\max}$ **do**
5: random neighbor Y' from $N_k(Y)$, $Y'^* \longleftarrow$ best local search(Y)
6: **if** $F_{\text{MSBT}}(Y'^*) < F_{\text{MSBT}}(Y)$ **then**
7: $k \longleftarrow k+1$
8: **else**
9: $Y \longleftarrow Y'^*$
10: **end if**
11: **end while**
12: **if** $F_{\text{MSBT}}(Y)$ is not improved after a fixed number of iterations **then**
13: $Y \longleftarrow perturbation(Y)$
14: $Y \longleftarrow$ best local search(Y)
15: **end if**
16: **end while**

After some preliminary experimentation, we set two different variable neighborhood search algorithms using different local search modules. In particular, VNSa does not implement the perturbation and improvement steps (steps 12-15) and uses only the first improving local search algorithms. On the other hand, VNSb does implement step 14 and uses only greedy local search algorithms. We show in the next section that there is no a clear dominance between VNSa and VNSb.

5.7 Experimental results

As it is mentioned in Chapter 3, the preprocessing stage for SBT could be used to define metaheuristic algorithms. In Section 5.4 we generalize the preprocessing stage of SBT to consider multiple planning periods. Then, we define several metaheuristic algorithms to

solve our proposed MSBT formulation since exact approaches for these cases are not at hand. Moreover, these algorithms can also be implemented for SBT with the corresponding feasible time windows defined in 3.4. We show the numerical results for SBT and MSBT in the following sections. To perform the experimental analysis, we implement the metaheuristic algorithms using C++ and an iMac OS X with an Intel Core 2 Duo 3.06 GHz processor and 4 GB RAM.

5.7.1 Results for SBT

To evaluate the behavior of the proposed metaheuristics for SBT, we use the instances types presented in Section 4.5. Moreover, ILS and VNS algorithms implements the constraint propagation procedure presented in Section 3.4.

To show the solution quality obtained by our proposed metaheuristic algorithms, we will compare the solutions with the ones obtained by CPLEX's linear solver for SBT MILP strengthened with synchronization and headway inequalities shown in Section 4.2.1 and 4.2.2, respectively. Moreover, to make an accurate comparison we implement a stop criteria of 3% of relative gap for CPLEX's solver. Tables 5.2 and 5.3 present the numerical results. The rows show the instance types and the columns show the mean gap (considering the dual bound obtained by CPLEX's linear solver) and mean time for each one of the solution algorithms. The best results for metaheuristics are highlighted with underlined text. We can notice that ILS algorithms overcome the behavior of VNS algorithms for all the instances. However, the best approach to find high quality solution in a short time is the implementation of CPLEX's linear solver in the strengthened SBT MILP.

Although, metaheuristic algorithms obtain high quality solutions comparable with the exact approach for the single period case, we compare in the next section the two approaches for MSBT.

	ILSa		ILSb		ILSc		ILSd		*CPLEX*	
	gap	time	gap	time	gap	time	gap	time	gap	time
T1	5.45	0.33	5.7	0.33	4.72	1.97	3.56	0.78	1.94	1.73
T2	4.28	0.38	2.66	0.36	4.47	2.81	4.43	0.82	1.27	2.98
T3	6.47	0.91	6.77	0.87	6.22	4.82	4.85	2.49	2.56	334.51
T4	3.12	1.1	2.36	0.94	3.01	6.49	4.57	2.9	0.85	4.47
T5	7.51	3.26	8.08	2.8	7.14	14.01	6.65	8.21	2.11	41.92
T6	4.55	3.92	3.15	2.47	4.85	18.43	7.49	7.26	1.24	18.62
T7	4.34	9.06	4.32	7.06	4.09	34.57	6.62	16.76	1.74	91.25
T8	5.54	8.83	2.85	6.83	5.78	40.06	8.61	16.23	1.73	85.48

Table 5.2: Results of the Multistart Iterated local searches and CPLEX's linear solver.

	CILSa		CILSb		VNSa		VNSb	
	gap	time	gap	time	gap	time	gap	time
T1	3.97	0.68	3.75	0.86	9.31	0.51	7.77	1.48
T2	3.53	0.81	2.94	0.49	7.25	0.68	4.89	1.99
T3	4.57	2.14	4.62	2.38	11.59	0.91	10.19	2.44
T4	2.87	3.62	2.4	2.47	9.69	1.1	8.1	2.61
T5	6.05	9.7	5.62	10.23	13.9	2.1	13.14	4.28
T6	3.89	12.74	3.9	9.06	11.66	2.68	10.18	4.98
T7	3.48	26.22	3.5	28.56	10.54	4.96	10.08	8.73
T8	4.87	33.48	4.43	30.3	12.06	6.03	11.18	9.96

Table 5.3: Results of the chained Multistart Iterated local searches and Variable Neighborhood searches algorithms.

5.7.2 Results for the Multiperiod timetabling case

For the case of multiple periods, we generalize the instances presented in Chapter 4. The instance size is determined by the number of lines $|I|$ and the number of synchronization nodes $|B|$ along the network. All the instance types have six planning periods of $T = 240$ minutes. The parameters for each one of the periods of the day are equal, to simplify the comparisons and exhibit the efficiency of the algorithms. The frequency for each line i is randomly generated between [13,18]; travel times from depot to a synchronization node are between [20,60]; the headway flexibility parameters satisfy $\frac{\delta_s^i}{\eta_s^i} \in [0.15, 0.25]$; finally, the number of different pairs of lines that synchronize at each node b is between one and seven. We randomly generate ten instances for each one of the four instance types to analyze the algorithm's performance. The name of the instance types and their parameters are summarized in Table 5.4.

Instance	**T10**	**T11**	**T12**	**T13**
$\|I\|$	10	50	100	200
$\|B\|$	1	5	10	20

Table 5.4: Instance types and parameter values.

It is worth noting that a commercial solver for solving a real sized instance of MSBT usually does not give feasible solutions in an hour due to its extremely slow convergence. Nevertheless, when the instance has the same parameters for all its periods, we can implement the linear solver of CPLEX 12.3 using the SBT MILP strengthened with synchronization and headway inequalities. We use two different stop criterion for CPLEX's solver to compare it with the metaheuristic algorithms. Notations CPLEX_1 and CPLEX_2 represent the results obtained by CPLEX's optimizer with stop criteria of 10 minutes and one hour, respectively.

Obviously, real life instances consider different planning periods but to consider these types of instances in this study is useless since we do not have a solution methodology or bounds to make a comparison and the variation in parameter values should not have a strong influence on the behavior of the metaheuristics.

Tables 5.5 and 5.6 present the results of the metaheuristic algorithms. The rows show the instance types and the columns show the mean gap (considering the dual bound obtained by $CPLEX_2$) and mean time for each one of the solution algorithms. The best results are highlighted with underlined text. Without considering the solutions obtained by CPLEX's optimizer (which in real instances we would not be able to implement), we notice that VNSb obtains the best results for instances of types T12. Nevertheless, VNSa obtains the best results for instances of types T10, T11, and T13. In overall performance, the variable neighborhood search algorithms outperform the iterated local searches. In spite of this general behavior, the ILS algorithms present the best results for some particular instances. Therefore, there is no single metaheuristic that surpasses all the others.

The solutions obtained by the metaheuristic algorithms for these academic instances improves the ones obtained by $CPLEX_1$. Moreover, there is a difference of around 5% and 11% from the result obtained by the best metaheuristic algorithm and $CPLEX_2$.

	ILSa		ILSb		ILSc		ILSd		$CPLEX_1$	
	gap	time	gap	time	gap	time	gap	time	gap	time
T10	15.73	0.10	25.43	0.13	11.10	0.12	11.48	0.15	4.64	428
T11	15.74	1.17	18.71	1.26	12.63	1.28	11.86	2.51	23.4	600
T12	20.28	3.21	21.22	3.43	14.09	3.29	14.98	6.12	65.8	600
T13	20.23	10.50	23.1	9.12	13.99	10.64	15.34	18.42	127	600

Table 5.5: Results of the Multistart Iterated local search and CPLEX's linear solver.

	CILSa		CILSb		VNSa		VNSb		$CPLEX_2$	
	gap	time	gap	time	gap	time	gap	time	gap	time
T10	8.97	0.14	10.14	0.19	4.39	2.06	5.58	4.45	0.86	2271
T11	12.00	2.44	12.11	2.15	11.59	5.61	11.86	13.45	2.15	3600
T12	14.96	5.74	14.62	6.50	12.75	12.31	12.70	19.74	2.49	3600
T13	15.14	18.27	14.59	20.24	13.50	17.38	13.77	30.25	2.26	3600

Table 5.6: Results of the chained Multistart Iterated local search and Variable Neighborhood search algorithms.

It is important to remark that our proposed metaheuristics are the only algorithms available for obtaining solutions for MSTB with less than 13.5% of relative mean gap for large instances (up to 200 lines and 20 synchronization nodes) in seconds while the implementation of the powerful valid inequalites for SBT MILP requires minutes.

5.8 Conclusions

We present the Multiperiod Synchronization Bus Timetabling Problem (MSBT) that considers variations arising during different planning periods for parameters such as travel times, frequency, and headways. MSBT offers smooth transitions between these periods and allows synchronization events between trips that belongs to different planning periods, something which is ignored by merging solutions of the single period case. The relevance of MSBT is due to the fact that posterior planning problems such as vehicle and crew scheduling need as input a complete timetable. Indeed, planning based on single period timetabling problems leads to suboptimal solutions.

MSBT is an NP-hard problem that is difficult to solve in an exact way in a limited time. We propose eight metaheuristic algorithms (six iterated local searches and two variable neighborhood searches). To the best of our knowledge, these are the only solution algorithms available for obtaining feasible high quality solutions for SBT and MSBT using a reasonable amount of time. For example, we can obtain solution with less than 3.5% of relative gap for large instances of SBT using seconds of computational time.

The quality of the solutions obtained with our metaheuristic algorithms for MSBT overcomes the ones obtained by CPLEX_1 and differ between 3% and 10% from those obtained with CPLEX_2 (which can be used only for our designed academic instances where all the periods have the same parameters). This is an interesting result, since even with implementing the powerful valid inequalities presented in chapter 4, consider multiple planning periods leads to a considerable slower convergence of the CPLEX's optimizer.

Although we obtain feasible solutions for this complex problem, there are many research areas that could lead to important and necessary results. First, integer programming techniques could be used to compute tight dual bounds. These bounds could be used to define a more accurate measure for the solutions presented in this chapter. Secondly, valid inequalities for the single period time tabling problem are not easy to generalize for MSBT. However, there is an open door for developing definitions of new parameters that would allow using these valid inequalities or considering new inequalities that would

embed the variability of the parameter values through the entire day. Finally, the generation of solutions for timetabling problem considering the entire day are important for defining an integrated approach with subsequent subproblems, such as vehicle and crew scheduling. Precisely such an integrated approach could be a very important research area where our results could be used.

Chapter 6

Conclusions and Future Research

Summary: In this chapter we state several conclusions about our results but the most important, we define research areas where our contributions could be useful.

"We leave an open door in transit network planning."

We define new and accurate timetabling problems for the Monterrey's bus network which is composed of several private companies leading to a large and centralized transit network where efficient interaction between different bus lines is needed. Indeed, our proposed Synchronization Bus Timetabling Problem has the objective of maximize the number of synchronization events between different lines to allow well timed passenger transfers and avoid bus bunching. In particular, our problem allows modeling the real transit network by considering characteristics such as: flexibility using headway bounds, a given frequency, and short planning periods. The proof that SBT is NP-Hard relies in the combinatorial nature of synchronization between different lines. This means that to choose which pairs are going to be synchronized to get an optimal solution is a difficult task while determine the departure times knowing the synchronized pair of trips is easier to handle.

Our proposed mixed integer linear programming formulation for the Synchronization Bus Timetabling Problem (SBT) has a special structure which allows removing many decision variables and constraints using a preprocessing stage based on constraint propagation. Although this preprocessing stage improves the performance of CPLEX's linear

solver, the most important characteristic is that it is a tool to define and explore the feasible space of the problem.

Constraint propagation is quite useful to define families of valid inequalities to strength the SBT MILP. Our proposed valid inequalities attack the core of the problem complexity, i.e., the synchronization events. In particular, two of the valid inequalities bound the number of synchronization events while the other two families of valid inequalities identify impossible combination of synchronizations. Then, the search space is considerable restricted and adding these inequalities to the BTP MILP leads to strengthened formulation. Then, it is possible to solve large instances of BTP with the linear solver of CPLEX 12.3 using short computational times.

As we mentioned before, one of the assumptions of SBT is the existence of short planning periods. This assumption is useful to define more accurate deterministic travel times. However, it is necessary to generate a timetable for the entire day. This could be achieved by solving several instances of SBT (one by each planning period of the day) and merging the solution with considering some criteria. This kind of approach should leads to suboptimal solution. Therefore, we introduce the Multiperiod Synchronization Bus Timetabling Problem (MSBT) that models smooth transitions between planning period periods and it allows synchronization events between trips belonging to different planning periods (which is ignored by merging solutions of the single period case). We propose eight metaheuristic algorithms (six iterated local searches and two variable neighborhood searches) to solve this NP-Hard problem. To the best of our knowledge, these are the only solution algorithms available for obtaining feasible high quality solutions MSBT in a reasonable amount of time. Moreover, we can obtain solution with less than 3.5% of relative gap for large instances of the SBT using seconds of computational time which is comparable with the implementation of valid inequalities and the CPLEX's linear solver.

Summarizing, we design two approaches to solve SBT. One is based on valid inequalities while the other is based on metaheuristic algorithms. In the case of MSBT, we have at hand only our proposed metaheuristic algorithms to obtain solutions with less than 5% and 13% of mean relative gap for small and large instances, respectively. We

consider that these are good results, since solving small instances even for SBT (single period) in a short time was not possible before this study. However, there are several research areas open to other studies such as the following.

- One of the issues for MSBT is the lack of optimal solutions for most of the instances. Then, an important research area is the design of a different solution approach such as, branch and cut approach based on the generalization of the valid inequalities for SBT presented in Chapter 4. In particular, this inequalities are focus on reduce the search space of synchronization variables and they are not accurate in MSBT due to parameter variation along the entire day. Therefore, it is necessary to identify the impact of the parameter variation in the valid inequalities. For example, the valid inequality $\sum_{q=1}^{f^j} Y_{pq}^{ijb} \leq 1 + \left\lfloor \frac{W^b - w^b}{h^j} \right\rfloor$ for trip $i(p)$ is based on a single headway parameter and an unique waiting time window while in the case of MSBT we have headway parameters for each planning period s (h_s^i) and waiting time windows depend of the trips we are trying to synchronize $\left(w_{pq}^{ijb} \text{ and } W_{pq}^{ijb}\right)$.

  In the basis of the above, a simple generalization of the previous inequality can be $\sum_{q=1}^{f^j} Y_{pq}^{ijb} \leq 1 + \left\lfloor \frac{\max_q\left\{W_{pq}^{ijb} - w_{pq}^{ijb}\right\}}{\min_s\left\{h_s^j\right\}} \right\rfloor$. However, instead of consider all the planning periods, the previous inequality should be generalized considering only the planning periods where trip $i(p)$ can synchronize. This idea could be implemented to generalize other families of valid inequalities.

- Several metaheuristics algorithms for our proposed timetabling problems are developed. All of them are based on shifting departure times of a single trip or an entire line to achieve synchronizations A characteristic of these movements is that they define neighborhoods with an infinite number of elements. Then, random shifts are implemented to explore the neighborhoods in an efficient way leading to local search algorithms.

  In particular, a trip $i(p)$ is shifted within an interval $\overline{D}_{pq}^{ijb}$ to guarantee that $Y_{pq}^{ijb} = 1$. However, this shift movement could be made based on different deterministic rules. For example, one of these rules could be solving a linear program to shift $i(p)$ inside its feasible interval to maximize the number of synchronizations between trip

$i(p)$ and all the trips of lines $J(i)$. The previous rule is really different since shifting movements are made considering all the possible synchronization variables instead of the related with only one pair of lines. Moreover, different heuristic algorithms could be defined based on movements changing the combination of active synchronization variables and then, determine the related departure times using ideas like constraint propagation. In general, a large number of different movements can be studied to define a more efficient solution algorithm.

- Another important research area is the integration of subproblems of transit network planning. There are some studies of integration approaches but only a few of them present formulations for a complete integration of two subproblems. Most of these studies are integration between vehicle and crew scheduling. Moreover, there are exact solution algorithms such as column generation and other metaheuristic to solve the proposed integrations.

  Since SBT is based on quality service, the integration of SBT with other subproblems such as vehicle and crew scheduling could be defined by a multiobjective approach. This approach allow to the planner to decide which solutions are accurate considering the agency policies instead of obtain suboptimal solutions implementing sequential approaches. Therefore, obtain efficient solution algorithms for this multiobjective problem is needed. Appendix A presents preliminary results for integrated approaches considering timetabling with synchronization events.

Appendix A

Integrated approaches

Summary: One of the hot topics in transit network planning (TNP) is the complete integration of two or more subproblems of it. In particular, there are only a few studies that address the bridge between quality service and operational cost and these studies are mostly focus on sequential approaches. Since the nature of the objectives of timetabling problems and subproblems like vehicle and crew scheduling are in conflict, we propose to study multiobjective approaches to define complete integrations of two or more subproblems of TNP.

"High quality service vs acceptable operational costs."

A.1 Introduction

The entire planning process of a bus network is divided into several subproblems such as line planning, timetable generation, vehicle scheduling, and crew scheduling. Commonly, these subproblems are solved with sequential approaches to obtain a solution for the entire planning problem. Moreover, it is difficult to obtain a common goal considering different subproblems. In particular, timetabling and vehicle scheduling are two subproblems where this issue is more obvious. The reason is that one problem is based on quality service while the other one is based on operational costs.

In general, there are two categories of integrated approaches: Partial integrations and complete integrations. On the one hand, partial integrations considers characteristics of one of more subproblems while another subproblem is optimized. For example, sequential approaches are of this type and lead to suboptimal solutions. On the other hand,

complete integrations define formulation and/or solution methodologies to determine the decisions of two of more subproblems in a simultaneous way thus, it is possible to know the optimal solution of each subproblem. In the following sections we present several formulations of the complete integration of two or more subproblems of transit network planning.

A.2 Integrating timetabling and vehicle scheduling

The vehicle scheduling problem determines the assignment between a set of trips and a set of vehicles with minimal vehicle costs. These problems are addressed in several studies and have several formulations based on the different characteristics such as types of vehicles, number of depots, compatibility of vehicles, and so on [Daduna and Paixao, 1995, Freling et al., 2001, Steinzen et al., 2010].

The integration of vehicle scheduling with timetabling problems is an important research area and has been addressed only by few studies. For example, van den Heuvel et al. [2008] address the integration of clock-face timetabling (i.e., services depart at the same number of minutes past each hour) and multiple-depot and vehicle-type vehicle scheduling problem. They design a tabu search to solve this integration. However, the idea of the algorithm is a sequential solution methodology, since timetabling is modified and then, vehicle scheduling problem is optimized considering the actual timetable. In Guihaire and Hao [2008a], a similar sequential idea was presented. The authors address the timetabling problem with objective of minimizing the overall waiting time considering a given frequency and fixed headway times. They design an iterated local search for solving their formulation. In Fleurent and Lessard [2009], the authors propose a measure function for transfers based on ideal waiting times. They design an optimization approach to minimize other objectives such as number of vehicles and unproductive time. In Fleurent et al. [2005] and Guihaire and Hao [2010a], an integral formulation for timetabling and vehicle scheduling that considers weights on the objective function is

presented. However, these weights must reflect the planner's necessity, which is a issue for two or more objectives in conflict.

An accurate formulation to represent the planner's interests is needed. Due to the nature of the objective function of timetabling and vehicle scheduling problems, we propose a multiobjective approach to define a complete integration between these two subproblems.

In this appendix we illustrate the justification and need to implement a multiobjective approach to integrate subproblems of transit network planning. Since in many cases it is difficult to solve multiobjective formulations we introduce the integration of a simpler version of MSBT with the single type vehicle scheduling problem with homogeneous fleets.

A.2.1 Timetabling problem

To define our timetabling problem, we use exactly the same notation and decision variables than MSBT. The only difference is the redefinition of flexibility to represent the agencies that are more interested in provide regular services than achieving synchronizations. Instead of consider headway bounds as MSBT, we now consider a bounded deviation from the timetable with even headways inside planning periods and average headways between different planning periods (called smooth transitions). Figure A.1 shows a line considering a planning period s of 40 minutes, a frequency of four trips, even headway of $\eta_s^i = 10$, and relative deviation from the initial timetabling of $\delta_s^i = 10\%$. The above case represents the initial timetable. The below case shows the feasible departure time window for each one of the four trips. We can notice that a trip can depart at most 1 minute earlier or later than the corresponding departure time of the initial timetable (deviation of 10% from initial timetable).

Then, the MILP for our less flexible *Multiperiod Timetabling Problem* (MT) is given as follows.

$$\max F_{\mathrm{MT}}(Y) = \sum_{i \in I} \sum_{j \in J(i)} \sum_{b \in B^{ij}} \sum_{p=1}^{f^i} \sum_{q=1}^{f^j} Y_{pq}^{ijb}$$

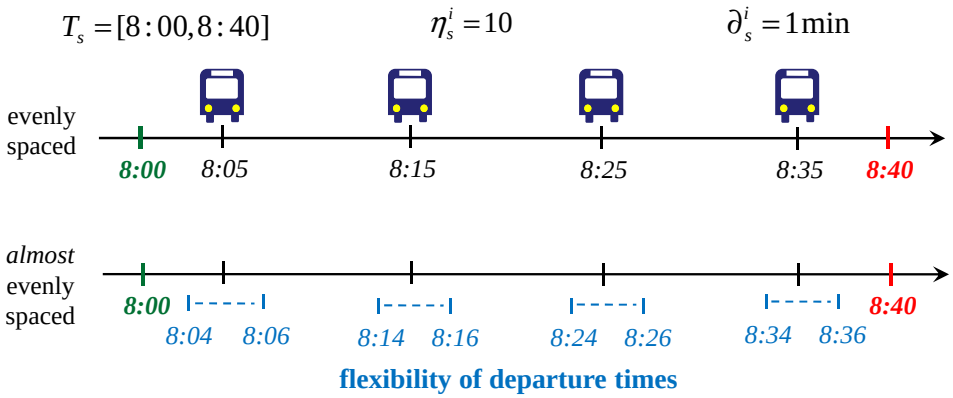


Figure A.1: Flexibility considering a bounded deviation from the timetable with even headway and smooth transitions between planning periods.

subject to

$$X_p^i \geq d_{s-1} + \frac{\eta_s^i}{2} + (p - first(s))\eta_s^i - \eta_s^i \delta_s^i \qquad \forall\ i \in I, s \in S, \quad p = \text{first}(s), \ldots, \text{last}(s) \qquad \text{(A.1)}$$

$$X_p^i \leq d_{s-1} + \frac{\eta_s^i}{2} + (p - first(s))\eta_s^i + \eta_s^i \delta_s^i \qquad \forall\ i \in I, s \in S, \quad p = \text{first}(s), \ldots, \text{last}(s) \qquad \text{(A.2)}$$

$$\left(X_q^j + t_q^{jb}\right) - \left(X_p^i + t_p^{ib}\right) \geq w_{pq}^{ijb} - M\left(1 - Y_{pq}^{ijb}\right) \qquad \forall\ i \in I,\ j \in J(i),\ b \in B^{ij}, \quad p = 1, \ldots, f^i,\ q = 1, \ldots, f^j \qquad \text{(A.3)}$$

$$\left(X_q^j + t_q^{jb}\right) - \left(X^{ip} + t_p^{ib}\right) \leq W_{pq}^{ijb} + M\left(1 - Y_{pq}^{ijb}\right) \qquad \forall\ i \in I,\ j \in J(i),\ b \in B^{ij}, \quad p = 1, \ldots, f^i,\ q = 1, \ldots, f^j \qquad \text{(A.4)}$$

Constraints (A.1) and (A.2) guarantee that each trip is inside the feasible departure time window illustrated in Figure A.1. The remaining constraints, are the synchronization constraints previously presented for SBT and MSBT. Next, we present the vehicle scheduling to define our complete integration proposal.

A.2.2 Vehicle scheduling problem

Our vehicle scheduling problem minimizes the number of vehicles to cover a set of trips considering homogeneous fleets. To define the formulation of our *Common Fleet Vehicle*

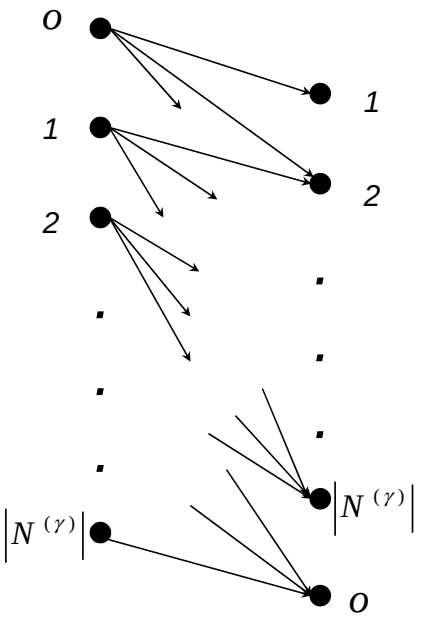


Figure A.2: Network $\left(N^{(\gamma)}, A^{(\gamma)}\right)$ for a fleet $\gamma \in \Gamma$.

Scheduling Problem (VS), we introduce the following notation. Let be Γ the set of homogeneous fleets. Set $I(\gamma)$ represent the lines that can be covered by each fleet $\gamma \in \Gamma$ where $I(\gamma) \cap I(\gamma') = \emptyset$ if $\gamma \neq \gamma'$ and $\bigcup_{\gamma \in \Gamma} I(\gamma) = I$. The parameters of VS are the fleet size $size^{\gamma}$ for each $\gamma \in \Gamma$, the total travel time r_p^i for each trip $i(p)$, and the setup time $s^{ii'}$ required for buses to be ready for making a trip of line i' just finishing a trip of line i.

The formulation of VS can be done using a bipartite network $\left(N^{(\gamma)}, A^{(\gamma)}\right)$ for each fleet γ. To achieve this, the nodes $N^{(\gamma)}$ are the trips $i(p)$ for all line $i \in I(\gamma)$ and trip $p = 1, \ldots, f^i$. Moreover, we add a node o representing the depot. There are arcs for each trip $i(p) \in N^{(\gamma)}$ departing from the depot and arriving at it, and an arc $(i(p), i'(p'))$ between two trips $i(p)$ and $i'(p')$ exists, if and only if, it is possible for a vehicle v to make trip $i(p)$ prepare the vehicle and make trip $i'(p')$. Formally, $A^{(\gamma)} = \left\{(o, i(p)), (i(p), o) : i(p) \in N^{(\gamma)}\right\} \cup \left\{(i(p), i'(p')) : i(p), i'(p') \in N^{(\gamma)}, X_{p'}^{i'} \geq X_p^i + r_p^i + s^{ii'}\right\}$. Then, the decisions of VS are determined by binary variables such as $V_{pp'}^{ii'\gamma}$ taking the value of one if a vehicle of fleet γ makes trip $i'(p')$ just finishing $i(p)$, and being zero otherwise. Figure A.2 represents a network $\left(N^{(\gamma)}, A^{(\gamma)}\right)$ for a fleet γ. Notice that a vehicle schedule can be seen as a path starting from the depot and finishing in this depot.

Considering the previous elements, the integer programming formulation for VS is

given as follows.

$$\min F_{VS}(V) = \sum_{\gamma \in \Gamma} \sum_{i(p) \in N^{(\gamma)}} V_{op}^{\ i\gamma}$$

subject to:

$$\sum_{i'(p') \in N^{(\gamma)}} V_{pp'}^{ii'\gamma} = \sum_{i'(p') \in N^{(\gamma)}} V_{p'p}^{i'i\gamma} = 1 \quad \forall \gamma \in \Gamma, i(p) \in N^{(\gamma)}, i(p) \neq o \tag{A.5}$$

$$\sum_{i(p) \in N^{(\gamma)}} V_{op}^{\ i\gamma} \leq size^{\gamma} \qquad \forall \gamma \in \Gamma \tag{A.6}$$

The objective function represents the number of vehicles required to cover all trips. Constraints (A.5) guarantee that exactly one vehicle is assigned to each trip $i(p)$ and equations (A.6) limit the number of vehicles for each fleet $\gamma \in \Gamma$. VS can be seen as several single-type single-depot vehicle scheduling problems. These problems are easy to solve [Daduna and Paixao, 1995, Freling et al., 2001] and represent simple cases of transit networks. Now, we present the integrated formulation for MT and VS.

A.2.3 Multiobjective formulation for integrated MT and VS

As we mentioned before, we need a timetable to define the network of VS. Since we want to simultaneously determine the decisions of MT and VS we can not assume an initial timetable as given. Then, to achieve the integration of these two problems we add the following decision variables.

$$Z_{pp'}^{ii'\gamma} = \begin{cases} 1 & \text{if it is possible for a vehicle of fleet } \gamma \text{ make } i'(p') \text{ just finishing } i(p), \\ 0 & \text{otherwise.} \end{cases}$$

The previous variable is useful to define potential arcs for the vehicle scheduling problem. In the basis of these variables, the integrated formulation for MT and VS, named as MTVS, is given as follows.

$$[\max F_{MT}(Y), \min F_{VS}(V)]$$

subject to: (A.1)-(A.6)

$$X_{p'}^{i'} - \left(X_p^i + r_p^i + s^{ii'}\right) \geq -M\left(1 - Z_{pp'}^{ii'\gamma}\right) \quad \forall \gamma \in \Gamma, i(p), i'(p') \in N^{(\gamma)} \tag{A.7}$$

$$V_{pp'}^{ii'\gamma} \leq Z_{pp'}^{ii'\gamma} \quad \forall \gamma \in \Gamma, i(p), i'(p') \in N^{(\gamma)} \tag{A.8}$$

Now, we have a biobjective formulation to maximize the number of synchronization events and minimize the number of vehicles. Constraints (A.7) allow to variable $Z_{pp'}^{ii'\gamma}$ take the value of 1 if it is enough time for a vehicle to make trip $i(p)$ prepare the vehicle, and make trip $i'(p')$. Constraints (A.8) allow to a vehicle making trip $i'(p')$ just finishing trip $i(p)$ only if this is possible. Notice that the decisions of timetabling problem and vehicle scheduling are determined in a simultaneous way. Analogously to MTVS, the definition of a complete integrated formulation for MSBT and VS is straight forward.

A.2.4 Solution approach

In contrast to single objective optimization, a solution to a multiobjective problem is more of a concept than a definition. Typically, there is no single global solution, and it is often necessary to determine a set of points that all fit a predetermined definition for an optimum. The predominant concept in defining an optimal point is that of Pareto optimality, which is defined as follows [Marler and Arora, 2004].

Definition 1. *Considering a multiobjective optimization problem* $\{\min[F_1(x), \ldots, F_k(x)] : x \in X\}$*, where* X *is the feasible space. A point,* $x^* \in X$*, is Pareto optimal iff there does not exist another point,* $x \in X$*, such that* $F(x) \leq F(x^*)$*, and* $F_i(x) < F_i(x^*)$ *for at least one function* $F_i(x)$*.*

This mean that a point is Pareto optimal if it improves at least one objective function without detriment to another function. Often, algorithms provide solutions that may not be Pareto optimal but may satisfy other criteria, making them significant for practical applications. For instance, weakly Pareto optimal is defined as follows.

Definition 2. *A weakly Pareto optimal is a point* $x^* \in X$ *is and only if there does not exist another point* $x \in X$ *that improves all of the objective functions simultaneously.*

Pareto optimal points are weakly Pareto optimal, but weakly Pareto optimal points are not Pareto optimal.

In multiobjective optimization the preferences of the decision maker have a great importance to define an accurate formulation and solution approach for the problem. We must consider different situations such as the following [Marler and Arora, 2004].

- There are methods with a priori articulation of preferences that allow the user to specify preferences in terms of goals or the relative importance of different objectives. Most of these methods incorporate parameters, which are coefficients, exponents, constraint limits, etc. that can either be set to reflect the decision maker preferences, or be continuously altered in an effort to represent the complete Pareto optimal set.

- In some cases, it is difficult for a decision-maker to express an explicit approximation of the preference function. Therefore, it can be effective to allow the decision maker to choose from a palette of solutions. To this end, an algorithm is used to determine a representation of the Pareto optimal set. Such methods incorporate a posteriori articulation of preferences, and they are called cafeteria or generate-first-choose-later approaches [Messac and Mattson, 2002].

- Often the decision-maker cannot concretely define what he or she prefers. Then, there are methods that do not require any articulation of preferences. Most of the methods are simplifications of the methods mentioned in the first case, typically with the exclusion of method parameters.

In our case, we implement a state of the art method that allow to obtain Pareto optimal points for MTVS without a priori preferences of the decisions maker. Our solution approach is known as the ϵ-constraint method which consist in the optimization of a single objective function $F_s(x)$ while all other objective functions are used to define additional constraints such as $F_i(x) \leq \epsilon_i$, $i = 1, 2, \ldots, k$, and $i \neq s$ [Hai, 1971]. In this case, a systematic variation of ϵ_i yields a set of Pareto optimal solutions [Hwang and Masud, 1979]. However, improper selection of $\epsilon \in \mathbb{R}^k$ can result in a formulation with no

feasible solution. A general mathematical guideline for selecting ϵ_i is provided as follows [Carmichael, 1980].

$$F_i(x_i^*) \leq \epsilon_i \leq F_i(x_s^*)$$

Where $F_i(x_i^*)$ represents the minimum value of $F_i(x)$ and $F_i(x_s^*)$ is the value of the objective i obtained by optimizing the objective s. If it exists, a solution to the ϵ-constraint formulation is weakly Pareto optimal [Miettinen, 1998]. Moreover, if a solution activates the ϵ-constraints (i.e., satisfies these constraints as equalities) is necessarily Pareto optimal [Carmichael, 1980].

In the basis of the above, our ϵ-constraint method for MTVS is given by Algorithm 13. Steps 1-6 define the extreme points of the Pareto front (optimizing an objective function subject to the optimal value of the other objective). Steps 7-12 implements the strategical variation of parameter ϵ. In particular step 10 updates the Pareto front if the point obtained by solving P_ϵ^* is a Pareto optimal point. To do this, we verify if the ϵ-constraint is active, i.e., if $F_{VS}(V) = \epsilon$.

Algorithm 13 : ϵ-constraint

Input: MTVS instance
Output: List $ListPareto$ of Pareto optimal points

1: $ListPareto = \emptyset$
2: Find $VS^* = \{\min F_{VS}(V) : (A.1) - (A.8)\}$
3: Find $MT^* = \{\max F_{MT}(Y) : (A.1) - (A.8)\}$
4: Find $P_1^* = \{\max F_{MT}(Y) : (A.1) - (A.8), F_{VS}(V) \leq VS^*\}$
5: Find $P_2^* = \{\min F_{VS}(V) : (A.1) - (A.8), F_{MT}(Y) \geq MT^*\}$
6: $ListPareto = ListPareto \cup \{(MT^*, P_2^*), (P_1^*, VS^*)\}$
7: Make $a = VS^*$ and $\epsilon = P_2^* - 1$
8: **while** $\epsilon > a$ **do**
9: Find $P_\epsilon^* = \{\max F_{MT}(Y) : (A.1) - (A.8), F_{VS}(V) \leq \epsilon\}$
10: Update $ListPareto$ considering (P_ϵ^*, ϵ)
11: $\epsilon = \epsilon - 1$
12: **end while**

Since large instances of our timetabling and vehicle scheduling problems can be solved in a reasonable amount of time, we can implement this methodology to obtain in most cases the entire set of Pareto optimal points.

A.2.5 Experimental results

To implement our ϵ-constraint method, we use different instances types. The instance size of the timetabling is determined by the number of lines $|I|$ and the number of synchronization nodes $|B|$ along the network. All the instance types have six planning periods of $T = 240$ minutes. The parameters for each one of the periods of the day are equal. The frequency for each line i is randomly generated between [13,18]; travel times from depot to a synchronization node are between [20,60]; finally, the number of different pairs of lines that synchronize at each node b is between one and seven. The name of the instance types and their parameters are summarized in Table 5.4.

Instances	**T13**	**T14**	**T15**	**T16**	**T17**	**T18**
$\|I\|$	10	50	100	200	10	50
$\|B\|$	1	5	10	20	1	5
$\delta_s^i \in$	[7.5,12.5]	[7.5,12.5]	[7.5,12.5]	[7.5,12.5]	[15,25]	[15,25]

Table A.1: Instance types and parameter values.

To define the parameters related with the vehicle scheduling problem we generate a set of fleets Γ such that the fleet size $size^\gamma$ is randomly generated within [4,6] for each fleet $\gamma \in \Gamma$. Moreover, as it happens with Monterrey's transit agencies, the lines sharing bus bunching nodes are assigned to the same fleet. Finally, turnaround times r_p^i are randomly generated between 120 and 180 and setup parameters $s^{ii'}$ are zero. We randomly generate ten instances for each one of the six instance types to analyze the algorithm's performance (we generate a total of 60 instances).

In particular, instances T13-T16 for MT are very similar to instances T9-T12 for MSBT. The only difference is that the headway amplitude factor for MT is half of the ones for MSBT. Then, by constraints (A.1) and (A.2) of MT, the headways for each line i in the same planning period s is within $[\eta_s^i - 2\eta_s^i\delta_s^i, \eta_s^i - 2\eta_s^i\delta_s^i]$ (which are the headway constraints of MSBT), i.e., every feasible solution for MT is feasible for MSBT. On the other hand, instances $T17$ and $T18$ are instances with more flexibility that may not be feasible for MSBT.

In general, Algorithm 13 is very efficient for the designed instances. Table A.2 show

the average and standard deviation of the execution times of our ϵ-constraint method for all the instances types (avg_time and dev_time, respectively).

Instances	**T13**	**T14**	**T15**	**T16**	**T17**	**T18**
avg_time (secs)	10.26	231.64	818.38	3426.01	148.76	9018.64
dev_time (secs)	4.26	78.24	424.14	2324.50	144.11	4037.17

Table A.2: Computational resources using Algorithm 13 for instances T13-T18.

Due to the less flexibility given by MT, the ideal point (solution leading to optimal value of MT and VS) is found for 95% of the instances of types T13-T16. For the remaining instances of these types, the Pareto front is defined by two or three points. However, even in this timetabling instances with less flexibility, the conflict between objectives is present. Figure A.3 shows an instance of type T16 where we can see the three points of the Pareto front. The red circular point represents the ideal solution.

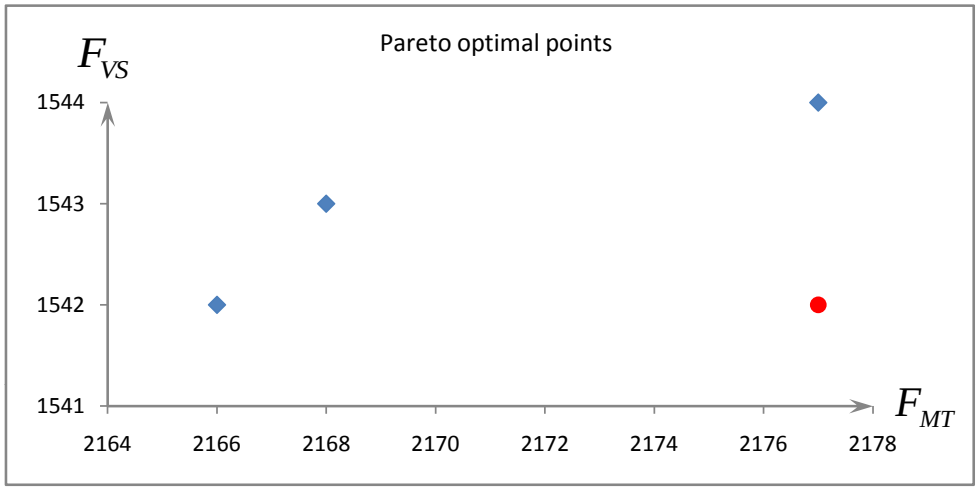


Figure A.3: Pareto front of an instance of type T16.

In the case of instances T17 and T18, ideal point is found for only 15% of the instances. In the instances of type T17, there are at most two Pareto optimal points while there are at most five Pareto optimal points for instances T18. Figure A.4 shows an instance of type T18 where we can see the five points of the Pareto front.

As we can see, the more flexible the timetable is, more Pareto optimal points can be found. This is the justification of a multiobjective approach. However, efficient solution algorithms are needed to handle larger instances. In particular, instances larger than T18

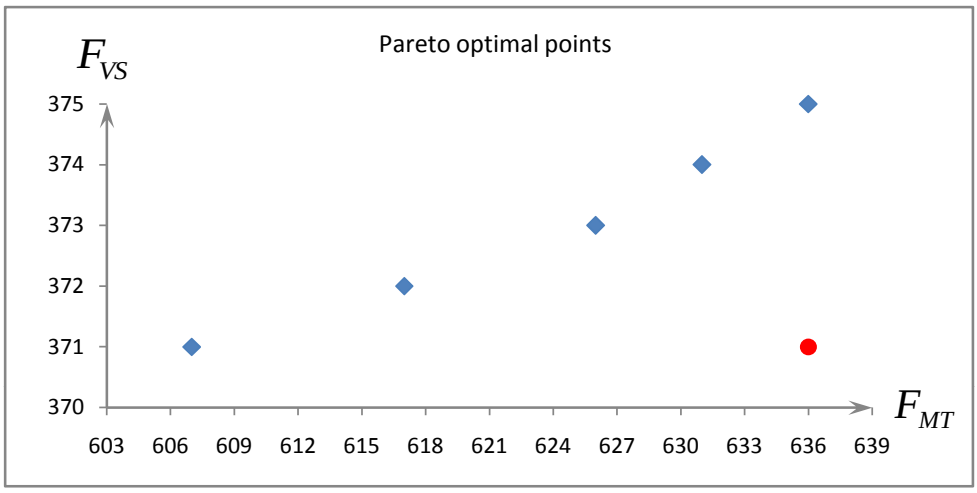


Figure A.4: Pareto front of an instance of type T18.

with the same flexibility parameters are intractable using Algorithm 13. This is also the case of the integration of MSBT and VS.

A.3 Integrating timetabling and vehicle-crew scheduling

Conflict between objectives is present in the integration of timetabling and vehicle scheduling. In the case of the integration between two subproblems such as vehicle scheduling and crew scheduling, a common objective can be seen as minimizing the total cost of the vehicles plus the cost of drivers. This characteristic leads to accurate and complete integrations of these two subproblems. For example, Friberg and Haase [1999] present an integrated formulation for vehicle and crew scheduling problem based on set partition problems and a column generation approach is developed to solve small instances of the problem (up to 20 trips). Mesquita et al. [2009] present an integration of these two subproblems with characteristics like, possibility of changing vehicles for drivers, and multidepot vehicle scheduling. The authors present an integer mathematical formulation combining multicommodity flow model with a mixed set partitioning/covering model. Solving linear relaxation along with branching procedures are proposed to solve their formulation. Similar studies based on set partition formulations, column generation ap-

proaches, and Lagrangian heuristics are presented in Freling et al. [2003] and Huisman et al. [2005].

A different kind of approach is presented in Kéri and Haase [2007] where the authors consider the possibility of modify the initial timetable to obtain better solutions for vehicle and crew scheduling problems. An approach based on Lagrangian heuristics combined with column generation is proposed to solve the problem.

The previous studies consider the trip-vehicle-driver assignment but, most of the constraints based on working regulations are missing. As it is clearly exhibit in Torrance et al. [2009], these working regulations constraints are required to obtain an accurate solution for crew scheduling problems. An example of working regulation constraint in integrated approaches can be found in Laurent and Hao [2008]. In this study, constraints for maximum spread time, maximum working time, and changeovers of vehicles (vehicle swaps) for drivers are considered to define an integration between vehicle and crew scheduling to minimize the number of buses and drivers. The authors propose a constraint based preprocessing and a GRASP algorithm to solve the proposed formulation. As it happens with this study, planning process of Monterrey's transit network must consider several working regulation constraints. In the next section we define an accurate problem definition for the case of Monterrey's transit network.

A.3.1 Vehicle and crew scheduling Problem

Monterrey's transit network is handled by several private agencies. Moreover, each agency has one or more depots to operate. Commonly, the operation planning process (i.e., assign vehicles and drivers) is done for each depot independently but the optimization can be done in a global way by minimizing the sum of the costs generated by each depot. Other characteristics of our vehicle and crew scheduling problems are the following.

- Accurate trip-vehicle-driver assignments should be done. To achieve this, we must consider that a driver can not be assigned to any vehicle since a driver is compatible with only a few vehicles (one or two in most cases).

- Each driver can be assigned to a few of the bus lines thus, although there are homogeneous fleets of vehicles, not all vehicles can be assigned to any bus line.
- By agencies policy, a working time of eight hours (regulated working time) for each driver is preferred, and less amount of time is not desired.
- By working regulations, a driver must be paid a double amount of money for every hour of work that exceeds the eight hours.
- By working regulations, each driver must have a resting time between every eight hours of work.
- A driver can change of vehicle along his day of work but the vehicle swaps must be bounded for each driver.

Figure A.5 shows a fleet γ with three vehicles and four drivers. The arrows represent the possibility of assignment between two elements. As we mentioned before, each driver is compatible with some vehicles $V(d)$. However, drivers abssentism is frequent. Then, a vehicle can be useless if its compatible drivers do not assist to work which is a major issue in the actual planning process of Monterrey's transit network. An example can be illustrated in Figure A.5 where vehicle three is useless if driver four is not present.

In the basis of the above, our *Vehicle-Crew Scheduling Problem with Working Regulations* (VCS) determines accurate trip-vehicle-driver assignments for an agency satisfying the working regulations constraints based in the previously mentioned considerations.

In the next section, we present the working regulation constraints for Monterrey's transit network. Moreover, as we did for MTVS, the integration of MT and VCS can be done using a multiobjective formulation.

A.3.2 Multiobjective formulation for integrated timetabling, vehicle, and crew scheduling

We consider homogeneous fleets but, due to drivers compatibility with vehicles and bus lines , we need to identify each vehicle. To achieve this, we introduce the following sets.

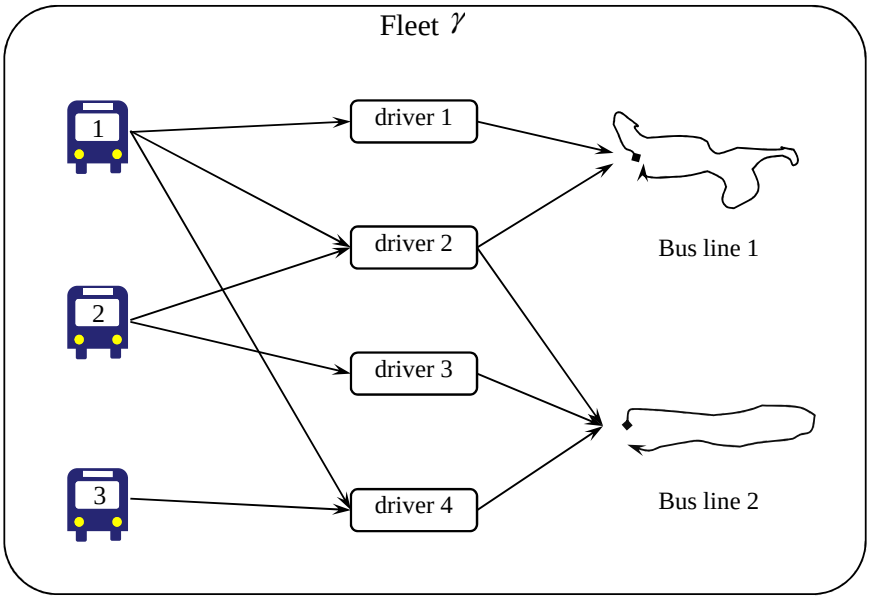


Figure A.5: Example of a fleet γ and the relations between its elements.

Let $V(\gamma)$ be the set of vehicles of fleet $\gamma \in \Gamma$. Analogously, let $D(\gamma)$ be the set of drivers of fleet γ. Compatibility between vehicles and drivers is defined by $V(d)$ representing the vehicles that can be assigned to driver d and $D(v)$ representing the drivers that can be assigned to vehicle v. Sets $V(i)$ and $D(i)$ represent the sets of vehicles and drivers compatibles with line i, respectively. Finally, the lines that can be assigned to each driver d and each vehicle v are denoted as $I(d)$ and $I(v)$, respectively.

We use the same parameters of MTVS but we also consider the following parameters: The maximum number of vehicle swaps for each driver d is given by $swap^{(d)}$; Regulated working time for driver d is represented for $work^{(d)}$; $max\_work^{(d)}$ is the maximum working time for driver d; Resting time for each driver is denoted as $rest^{(d)}$; The working time without rest for driver d is bounded by $no\_rest^{(d)}$; $c^{(d)}$ represent the cost of an hour of work of driver d. Parameter $extra\_c^{(d)}$ denotes the cost of an hour of work of driver d exceeding $work^{(d)}$; Parameter $c_p^{i(v)}$ is the cost to cover a trip $i(p)$ with vehicle v; Finally, the fixed cost of use a vehicle is given by $fix\_c^{(v)}$.

The decisions of our problems are the departure times and the sequence of trips for vehicle and drivers. We denote $i(p) \rightarrow i'(p')$ the fact of making trip $i'(p')$ just finishing trip $i'(p')$ by a specific vehicle or a specific driver. Therefore, the decision variables of our

integrated approach are the following.

- $Z_{pp'}^{ii'(v)} = \begin{cases} 1 & \text{if it is possible to make } i(p) \rightarrow i'(p') \text{ for vehicle } v, \\ 0 & \text{otherwise.} \end{cases}$
- $Z_{pp'}^{ii'(d)} = \begin{cases} 1 & \text{if it is possible to make } i(p) \rightarrow i'(p') \text{ for driver } d, \\ 0 & \text{otherwise.} \end{cases}$
- $W_{pp'}^{ii'(v)} = \begin{cases} 1 & \text{if vehicle } v \text{ makes } i(p) \rightarrow i'(p'), \\ 0 & \text{otherwise.} \end{cases}$
- $W_{pp'}^{ii'(d)} = \begin{cases} 1 & \text{if driver } d \text{ makes } i(p) \rightarrow i'(p'), \\ 0 & \text{otherwise.} \end{cases}$
- $V_{pp'}^{ii'(d)} = \begin{cases} 1 & \text{if driver } d \text{ makes } i(p) \rightarrow i'(p') \text{ changing vehicle,} \\ 0 & \text{otherwise.} \end{cases}$
- $U_{pp'}^{ii'(d)} = \begin{cases} 1 & \text{if driver } d \text{ makes } i(p) \rightarrow i'(p') \text{ with a rest between these trips,} \\ 0 & \text{otherwise.} \end{cases}$
- $WT^{(d)}$: Working time for driver d.
- $R^{(d)}$: Remaining working time of driver d to reach $work^{(d)}$.
- $E^{(d)}$: Extra working time for driver d.

In the basis of the previous decision variables, the total cost of the vehicle and crew scheduling is given by

$$F_{VCS}(W,E,R) = \min \underbrace{\sum_{\gamma\in\Gamma}\sum_{d\in D(\gamma)} c^{(d)}\left(WT^{(d)}+R^{(d)}\right) + extra\_c^{(d)}E^{(d)}}_{\text{driver costs}} +$$

$$\underbrace{\sum_{\gamma\in\Gamma}\left(\sum_{i(p)\in N^{(\gamma)}}\sum_{v\in V(i)} fix\_c_p^{i(v)} W_{op}^{\ i(v)} + \sum_{i(p)\in N^{(\gamma)}}\sum_{i'(p')\in N^{(\gamma)}}\sum_{v\in V(i)\cap V(i')} c_p^{i(v)} W_{pp'}^{ii'(v)}\right)}_{\text{vehicle costs}}$$

Therefore, our biobjective formulation MTVCS is the following.

$$[\max F_{MT}(Y), \min F_{VCS}(W, E, R)]$$

Subject to the following constraints. The first group of constraints (A.9)-(A.12) define the potential networks for vehicle and crew scheduling and are analogous to the ones defined for MTVS formulation.

$$X^{i'}_{p'} - \left(X^{i}_{p} + r^{i}_{p} + \max \left\{ s^{ii'}, rest^{(d)} \sum_{d \in D(v)} U^{ii'(d)}_{pp'} \right\} \right) \geq -M_1 \left(1 - Z^{ii'(v)}_{pp'} \right)$$

$$\forall \gamma \in \Gamma, i(p), i'(p') \in N^{(\gamma)}, v \in V(i) \cap V(i') \tag{A.9}$$

$$W^{ii'(v)}_{pp'} \leq Z^{ii'(v)}_{pp'}$$

$$\forall i(p), i'(p') \in N^{(\gamma)}, v \in V(i) \cap V(i') \tag{A.10}$$

$$X^{i'}_{p'} - \left(X^{i}_{p} + r^{i}_{p} + \max \left\{ s^{ii'}, rest^{(d)} U^{ii'(d)}_{pp'} \right\} \right) \geq -M_2 \left(1 - Z^{ii'(d)}_{pp'} \right)$$

$$\forall \gamma \in \Gamma, i(p), i'(p') \in N^{(\gamma)}, d \in D(i) \cap D(i') \tag{A.11}$$

$$W^{ii'(d)}_{pp'} \leq Z^{ii'(d)}_{pp'}$$

$$\forall \gamma \in \Gamma, i(p), i'(p') \in N^{(\gamma)}, d \in D(i) \cap D(i') \tag{A.12}$$

The next constraints (A.13) assign each trip $i(p)$ to one and only one vehicle and driver.

$$\sum_{i'(p') \in N^{(\gamma)}} \sum_{v \in V(i) \cap V(i')} W^{ii'(v)}_{pp'} = \sum_{i'(p') \in N^{(\gamma)}} \sum_{d \in D(i) \cap D(i')} W^{ii'(d)}_{pp'} = 1 \quad \forall \gamma \in \Gamma, i(p) \in N^{(\gamma)} \tag{A.13}$$

Constraints (A.14) and (A.15) guarantee that if a vehicle v or a driver d is assigned to a trip, these vehicle and driver should be assigned to other trip or return to the depot and vice versa.

$$\sum_{i'(p') \in N^{(\gamma)}} W^{ii'(v)}_{pp'} = \sum_{i'(p') \in N^{(\gamma)}} W^{i'i(v)}_{p'p} \qquad \forall \gamma \in \Gamma, i(p) \in N^{(\gamma)}, v \in V(i) \tag{A.14}$$

$$\sum_{i'(p')\in N^{(\gamma)}} W_{pp'}^{ii'(d)} = \sum_{i'(p')\in N^{(\gamma)}} W_{p'p}^{i'i(d)} \qquad \forall\gamma\in\Gamma, i(p)\in N^{(\gamma)}, d\in D(i) \tag{A.15}$$

We must guarantee that if a driver d is assigned to a trip $i(p)$, a compatible vehicle $v \in V(d)$ must be assigned to trip $i(p)$. This means that if variable $W_{pp'}^{ii'(d)}$ is one in constraint (A.16), there should be one vehicle in $V(d)$ assigned to trips $i(p)$ and $i'(p)$. In particular, if there is not an unique vehicle assigned to $i(p)$ and $i'(p')$, driver d makes trips $i(p)$ and $i'(p')$ with different vehicles, i.e., there is a vehicle swap.

$$\sum_{i''(p'')\in N^{(\gamma)})} \sum_{v\in V(d)\cap V(i)\cap V(i')} W_{p''p}^{i''i(v)} + \sum_{i''(p'')\in N^{(\gamma)}} \sum_{v\in V(d)\cap V(i)\cap V(i')} W_{p'p''}^{i'i''(v)} \geq 2W_{pp'}^{ii'(d)}$$
$$\forall\gamma\in\Gamma, i(p), i'(p')\in N^{(\gamma)}, d\in D(i)\cap D(i') \tag{A.16}$$

As we mentioned before, it is possible to identify a change of vehicles for some driver by comparing the vehicle and crew scheduling variables. In particular, the first constraints (A.17) allow to activate a vehicle swap variable related to trips $i(p)$ and $i'(p')$ for a driver d, if and only if, this driver makes $i(p) \to i'(p')$. Constraints (A.18) activate variable $V_{pp'}^{ii'(d)}$, if and only if, there is no vehicle $v \in V(d)$, making $i(p) \to i(p')$. Finally, constraints (A.19) limit the number of vehicle changes for each driver d.

$$V_{pp'}^{ii'(d)} \leq W_{pp'}^{ii'(d)} \qquad \forall\gamma\in\Gamma, i(p), i'(p')\in N^{(\gamma)}, d\in D(i)\cap D(i') \tag{A.17}$$

$$\sum_{v\in V(d)} W_{pp'}^{ii'(v)} \leq 1 - V_{pp'}^{ii'(d)} \qquad \forall\gamma\in\Gamma, i(p), i'(p')\in N^{(\gamma)}, d\in D(i)\cap D(i') \tag{A.18}$$

$$\sum_{i(p)\in N^{(\gamma)}} \sum_{i'(p')\in N^{(\gamma)}} V_{pp'}^{ii'(d)} \leq swap^{(d)} \qquad \forall\gamma\in\Gamma, d\in D(\gamma) \tag{A.19}$$

The following group of constraint control the working time for each driver. The non linear equations (A.20) define the working time for driver d. Constraints (A.21) define the variables $R^{(d)}$ and $E^{(d)}$ considering the regulated working time $work^{(d)}$, if driver d is assigned to at least one trip (i.e., is possible do not hire a driver d for some day). The

last constraints (A.22) limit the working time for each driver.

$$\sum_{i(p)\in N^{(\gamma)}} W_{po}^{i\ (d)} \left(X_p^i + r_p^i\right) - \sum_{i(p)\in N^{(\gamma)}} W_{op}^{i(d)} X_p^i = WT^{(d)} \qquad \forall \gamma \in \Gamma, d \in D(\gamma) \qquad \text{(A.20)}$$

$$WT^{(d)} + R^{(d)} - E^{(d)} \geq work^{(d)} \sum_{i(p)\in N^{(\gamma)}} W_{op}^{i(d)} \qquad \forall \gamma \in \Gamma, d \in D(\gamma) \qquad \text{(A.21)}$$

$$WT^{(d)} \leq max\_work^{(d)} \qquad \forall \gamma \in \Gamma, d \in D(\gamma) \qquad \text{(A.22)}$$

Finally, the resting times of drivers are controlled by the following constraints. Constraints (A.23) allow activate variable $U_{pp'}^{ii'(d)}$, if and only if, driver d makes $i(p) \rightarrow i'(p')$. Constraint (A.24) guarantee that it is enough free time between trips $i(p)$ and $i'(p')$ for resting time of driver d. Constraint (A.25) guarantee that a resting time is assigned to driver d every $no\_rest^{(d)}$ amount of time.

$$U_{pp'}^{ii'(d)} \leq W_{pp'}^{ii'(d)}$$
$$\forall i(p), i'(p') \in N^{(\gamma)}, d \in D(i) \cap D(i') \qquad \text{(A.23)}$$
$$X_{p'}^{i'} - \left(X_p^i + r_p^i + rest^{(d)}\right) \geq -M_4 \left(1 - U_{pp'}^{ii'(d)}\right)$$
$$\forall i(p), i'(p') \in N^{(\gamma)}, d \in D(i) \cap D(i') \qquad \text{(A.24)}$$
$$no\_rest^{(d)} \sum_{i(p)\in N^{(\gamma)}} \sum_{i'(p')\in N^{(\gamma)}} U_{pp'}^{ii'(d)} \geq WT^{(d)} - no\_rest^{(d)}$$
$$\forall \gamma \in \Gamma, d \in D(\gamma) \qquad \text{(A.25)}$$

Our formulation MTVCS consider most of the characteristics in Monterrey's transit network. Moreover, it is a similar formulation than some of the presented in the literature review. This may be an advantage since we can use ideas presented in literature to design solution methodologies. Unfortunately, exact approaches seems not accurate for solving this formulation. Then, efficient solution approaches are needed.

A.4 Conclusions

The formulations presented in this chapter are based on Monterrey's transit network and are complete integrations of subproblems of transit network planning. Using these formulation, we are able to know the optimal solution of each subproblem since decision are determined simultaneously and not in a sequential way.

In particular, Multiperiod Timetabling Problem (MT) and Vehicle Scheduling Problem (VS) are examples of two problems where our multiobjective formulation can be used. Using the Pareto optimal solutions, we are able to represent the cost of a vehicle in terms of number of synchronizations (quality service) and give this information to the decision maker. The characteristics of MT and VS, allow to solve problems of 50 lines and 5 depots (reasonable sizes for agencies in real transit networks) with an ϵ-constraint method. In the other hand, the integration of MSBT and VS is much more difficult and the ϵ-constraint method is not accurate.

We propose an integrated formulation for timetabling, vehicle scheduling, and crew scheduling based on Monterrey's transit network. Our problem definition consider working regulation constraints which are really important to obtain a representative solution for our case study. Due to the complexity of the subproblems, efficient solution algorithms are needed.

Appendix B

Background

The context of this dissertation is the definition of problems and efficient solution algorithms for transit network planning based on operations research concepts. In particular, we use several basic concepts related with integer programming and heuristic algorithms that can be easily found in several books such as Wolsey [1998], Nemhauser and Wolsey [1999], and Talbi [2009]. However we recall the concepts used along this dissertation.

B.1 Mathematical formulations

As it is mentioned in Williams [1999], the essential feature of a mathematical formulation in operational research is that it involves a set of mathematical relationships (such as equations, inequalities, logical dependencies, etc.) which correspond to some more down-to-earth relationships in the real world such as technological relationships, physical laws, marketing constraints, etc. Some of the motives for building such models are the following.

- The actual exercise of building a model often reveals relationships which were not apparent to many people. As a result a greater understanding is achieved of the object being modeled.
- Having built a model it is usually possible to analyze it mathematically to help suggest courses which might not otherwise be apparent.
- Experimentation is possible with a model whereas it is often not possible or desirable to experiment with the object being modeled. It would clearly be politically difficult,

as well as undesirable, to experiment with unconventional economic measures in a country if there was a high probability of disastrous failure. The pursuit of such courageous experiments would be more (though not perhaps totally) acceptable on a mathematical model.

Now, we present the definition of the different types of formulations used along this dissertation.

Definition 3. *A Mixed Integer Linear Program (MIP) is given by vectors* $c \in \mathbb{R}^n$, $b \in \mathbb{R}^m$, *a matrix* $A \in \mathbb{R}^{m \times n}$ *and a number* $p \in \{0, ..., n\}$. *The goal of the problem is to find a vector* $x \in \mathbb{R}^n$ *solving the optimization problem* $\left\{\max c^T x : Ax \leq b, x \in \mathbb{Z}_+^p \times \mathbb{R}_+^{n-p}\right\}$.

If $p = n$, then all variables are required to be integral. In this case, we speak of an Integer Linear Program (IP) denoted as $\left\{\max c^T x : Ax \leq b, x \in \mathbb{Z}_+^n\right\}$.

If in an Integer Linear Program all variables are restricted to values from the set $B = \{0, 1\}$, we have a $0 - 1$ Binary Linear Integer Program denoted as $\{\max c^T x : Ax \leq b, x \in B^n\}$

The use of discrete decision variables usually leads to problems where it is difficult to design solution algorithms to obtain optimal solutions. However, as we see in the next section, there are well defined algorithms to solve these types of formulations.

B.2 Branch-and-Bound algorithm

There are generic solution algorithms such as the Branch and Bound (B&B) to solve the formulation considering integer decision variables. For example, assume we have an integer program $z^* = \left\{\max c^T x : x \in X\right\}$ where $X = \left\{x : Ax \leq b, x \in \mathbb{Z}_+^n\right\}$. The main ideas of the Branch and Bound to solve a problem are:

- If we remove a integrality constraint $x_j \in \mathbb{Z}_+$, the modified problem is a linear program that can be easily solved and gives us a lower bound on z^*.
- If we fix x_j to a feasible integer, the modified problem gives an upper bound on z^*.

Branch and Bound uses these ideas in a divide and conquer strategy. The strategy consist in solve linear relaxations of the problem and fix integer variables to update lower bounds, feasible solutions, and upper bounds. The algorithm stops until the best lower bound obtained by linear relaxations is equal to the best upper bound related to feasible solutions, i.e., we have the optimal solution. An iteration k of B&B does the following:

- Select a problem P_k,
- Solve the linear relaxation of a problem P_k (relaxing the integrality constraint for non-fixed integer variables).
- Let x^* be the optimal solution obtained, and z_{P_k} the optimal objective function value.
- If $z_{P_k} \geq U$(best upper bound) we delete the problem (as z_{P_k} is a lower bound on all integer solutions in this branch of the tree).
- If $z_{P_k} \geq U$ pick some variable x_j whose optimal solution is fractional, and construct two problems by adding the constraints $x_j \leq \lfloor x_j^* \rfloor$ and $x_j \geq \lceil x_j^* \rceil$ in each problem.

Detailed examples of the implementation of B&B can be found in Wolsey [1998]. We most remark that B&B is a generic solution algorithm but, it is not capable to solve all kinds of problem. In particular, computational complexity theory states that problems can vary in the effort required to be solved them [e.g. Garey and Johnson, 1979, Papadimitriou and Steiglitz, 1998]. An optimization problem that belongs to the NP-hard class means, in simple terms, that an efficient algorithm for the exact solution of this problem does not exist. However, there are mathematical tools to help finding exact solutions for NP-hard problems implementing B&B algorithm or others similar to it. We present one of these tools in the next section.

B.3 Valid inequalities

Consider an integer program $P = \{\max cx : x \in X\}$ where $X = \left\{x : Ax \leq b, x \in \mathbb{Z}_+^n\right\}$. As it is shown in [Nemhauser and Wolsey, 1999], solving P is equivalent to solve $P_c =$

$\{\max cx : x \in conv(X)\}$, where the called convex hull $conv(X)$ is the smallest closed convex set that contains X at it is given as follows.

$$conv(X) = \left\{ x \in \mathbb{R}^n : x = \sum_{i=1}^{k} \lambda_i x^i \text{ where } k \geq 1, \lambda \in \mathbb{R}^k_+ \text{ and } x^1, \ldots, x^k \in X \right\}$$

Notice that $conv(X)$ do not has integrality constraints thus, P_c is a linear program. However, for NP-hard problems, there is almost no hope of finding a good description of $conv(X)$. Therefore, a common goal is approximate $conv(X)$ considering a given instance of the problem P. The fundamental concept to achieve this is that of a valid inequality.

Definition 4. *An inequality $\pi x \leq \pi_o$ is a valid inequality for $X \subseteq \mathbb{R}^n$, if $\pi x \leq \pi_o$ for all $x \in X$.*

By its definition, a valid inequality removes fractional solutions reducing the search space of the linear relaxation. This characteristic is really useful if we implement a state of the art algorithm such as the B&B. As it can be seen in Wolsey [1998], there are different general types of valid inequalities for integer and mixed integer programs. However, to define tailor-made valid inequalities for NP-hard problems that leads to high quality results is not an easy task. Indeed, if tailor-made valid inequalities are available for an optimization problem, different solution approaches based on them can be implemented [Nemhauser and Wolsey, 1999]. For example, an approach is to define a strengthen formulation with valid inequalities and solve it using an algorithm as a B&B.

B.4 Metaheuristics

In this section we present concepts related with metaheuristic algorithms. These concepts are extracted from an excellent review of metaheuristics presented in Talbi [2009].

The word heuristic has its origin in the old Greek word *heuriskein*, which means the art of discovering new strategies (rules) to solve problems. The term *metaheuristic* was introduced in Glover [1986]. Metaheuristic search methods can be defined as upper

level general methodologies (templates) that can be used as guiding strategies in designing underlying heuristics to solve specific optimization problems. Then, metaheuristics represent a family of approximate optimization that provide"acceptable" solutions in a reasonable time for solving hard and complex problems. Unlike exact optimization algorithms, metaheuristics do not guarantee the optimality of the obtained solutions.

In designing a metaheuristic, two contradictory criteria must be taken into account: exploration of the search space (diversification) and exploitation of the best solutions found (intensification). Promising regions are determined by the obtained "good" solutions. In intensification, the promising regions are explored more thoroughly in the hope to find better solutions. In diversification, nonexplored regions must be visited to be sure that all regions of the search space are evenly explored and that the search is not confined to only a reduced number of regions.

There is a large variety of metaheuristic algorithms. To classify them, criteria such as the following can be used.

- Nature inspired versus nonnature inspired: Many metaheuristics are inspired by natural processes: evolutionary algorithms and artificial immune systems from biology; ants, bees colonies, and particle swarm optimization from swarm intelligence into different species (social sciences); and simulated annealing from physics.

- Memory usage versus memoryless methods: Some metaheuristic algorithms are memoryless; that is, no information extracted dynamically is used during the search. Some representatives of this class are local search, GRASP, and simulated annealing. While other metaheuristics use a memory that contains some information extracted online during the search. For instance, short-term and long-term memories in tabu search.

- Deterministic versus stochastic: A deterministic metaheuristic solves an optimization problem by making deterministic decisions (e.g., local search, tabu search). In stochastic metaheuristics, some random rules are applied during the search (e.g., simulated annealing, evolutionary algorithms). In deterministic algorithms, using

the same initial solution will lead to the same final solution, whereas in stochastic metaheuristics, different final solutions may be obtained from the same initial solution. This characteristic must be taken into account in the performance evaluation of metaheuristic algorithms.

- Population-based search versus single-solution based search: Single-solution based algorithms (e.g., local search, simulated annealing) manipulate and transform a single solution during the search while in population-based algorithms (e.g., particle swarm, evolutionary algorithms) a whole population of solutions is evolved. These two families have complementary characteristics: single-solution based metaheuristics are exploitation oriented; they have the power to intensify the search in local regions. Population-based metaheuristics are exploration oriented; they allow a better diversification in the whole search space. In the next chapters of this book, we have mainly used this classification. In fact, the algorithms belonging to each family of metaheuristics share many search mechanisms.

- Iterative versus greedy: In iterative algorithms, we start with a complete solution (or population of solutions) and transform it at each iteration using some search operators. Greedy algorithms start from an empty solution, and at each step a decision variable of the problem is assigned until a complete solution is obtained. Most of the metaheuristics are iterative algorithms.

To define a metaheuristic algorithm, several components such as constructive algorithms, neighborhood, local search, and perturbation are needed. We present the definition of the previous components since they are used along this dissertation.

Constructive algorithms start from scratch (empty solution) and construct a solution by assigning values to one or several decision variables at a time, until a complete solution is generated.

A partial solution s may be seen as a subset of elements E_s from the set of all elements E. The set defining the initial solution is empty. At each step, a a well defined procedure is used to select the new element to be included in the solution. For example,

it is said that we have greedy constructive algorithm if our element selection procedure take the best element from the list considering the value of the objective function. We remark that, once an element is selected to be part of the solution, it is never replaced by another element, i.e., there is no backtracking of the already taken decisions. If we have a initial solution a natural question is, how to improve it? To answer this question we present the following definitions.

Definition 5. *A neighborhood function N is a mapping $N : S \in 2^S$ that assigns to each solution s of S a set of solutions $N(s) \subseteq S$.*

A solution s' in the neighborhood of s ($s' \in N(S)$) is called a neighbor of s. A neighbor is generated by the application of a move operator m that performs a small perturbation to the solution s. Once the concept of neighborhood has been defined, the local optimality property of a solution may be given.

Definition 6. *Relatively to a given neighbor $N(s)$, a solution $s^* \in S$ is a local optimum if it has a better quality than all its neighbors; that is, $f(s^*) \leq f(s')$ for all $s' \in N(s)$.*

For the same optimization problem, a local optimum for a neighborhood N_1 may not be a local optimum for a different neighborhood N_2. Then, it is important to define procedure to explore these local optima such as the following.

Local search is likely the oldest and simplest metaheuristic method. We present a template for it in Algorithm 14. It starts at a given initial solution. At each iteration, the heuristic replaces the current solution by a neighbor that improves the objective function. The search stops when all candidate neighbors are worse than the current solution, meaning a local optimum is reached. For large neighborhoods, the candidate solutions may be a subset of the neighborhood. The main objective of this restricted neighborhood strategy is to speed up the search.

Designing a local search algorithm has to address the selection strategy of the neighbor that will determine the next current solution. To achieve this, the following strategies can be applied in the selection of a better neighbor.

Algorithm 14 : Template of a local search algorithm

Input: initial solution s_0
Output: feasible solution s

```
1: s = s_0
2: while not Termination_Criterion do
3:    Generate N(s) (Candidate neighbors)
4:    if there is no better neighbor then
5:       stop
6:    else
7:       s = s' (select better neighbor s' ∈ N(s))
8:    end if
9: end while
```

- **Best improvement (steepest descent)**: In this strategy, the best neighbor (i.e., neighbor that improves the most the cost function) is selected. The neighborhood is evaluated in a fully deterministic manner. Hence, the exploration of the neighborhood is exhaustive, and all possible moves are tried for a solution to select the best neighboring solution. This type of exploration may be time-consuming for large neighborhoods.

- **First improvement**: This strategy consists in choosing the first improving neighbor that is better than the current solution. Then, an improving neighbor is immediately selected to replace the current solution. This strategy involves a partial evaluation of the neighborhood. In the worst case (i.e., when no improvement is found), a complete evaluation of the neighborhood is performed.

- **Random selection**: In this strategy, a random selection is applied to those neighbors improving the current solution. A compromise in terms of quality of solutions and search time may consist in using the first improvement strategy when the initial solution is randomly generated and the best improvement strategy when the initial solution is generated using a greedy procedure. In practice, on many applications, it has been observed that the first improving strategy leads to the same quality of solutions as the best improving strategy while using a smaller computational time. Moreover, the probability of premature convergence to a local optimum is less important in the first improvement strategy.

One of the main disadvantages of LS is that it converges toward local optima. Moreover, the algorithm can be very sensitive to the initial solution. Moreover, there is no means to estimate the relative error from the global optimum and the number of iterations performed may not be known in advance. Local search works well if there are not too many local optima in the search space or the quality of the different local optima is more or less similar. If the objective function is highly multimodal, which is the case for the majority of optimization problems, local search is usually not an effective method to use. However, many alternatives algorithms have been proposed to avoid becoming stuck at local optima. Four different families of approaches that can be used to avoid local optima are presented in Figure B.1:

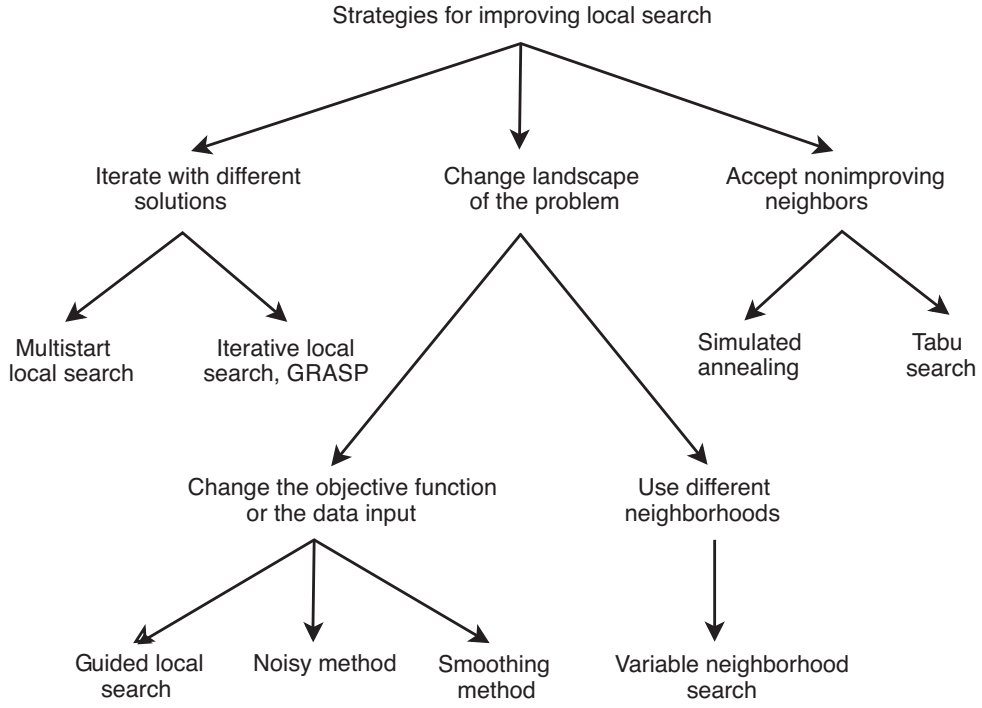


Figure B.1: Family of algorithms for improving local search and escaping from local optima [Talbi, 2009].

To finish this chapter, we must remark that some of the previous metaheuristics implement **perturbation moves** to escape from local optima. The perturbation operator may be seen as a large random move of the current solution. The perturbation method should keep some part of the solution and perturb strongly another part of the solution to move hopefully to another basin of attraction.

Appendix C

Piece-mold-machine manufacturing planning

Summary: The main topic of this dissertation is the transit network planning. However, I found some time to get involved in other research areas. One of these studies is manufacturing planning which is the research area of my master thesis. In particular, Mario Saucedo made a vital participation to obtain the results presented in this chapter. I also thank to Yasmin Rios-Solis and Roger Rios-Mercado for collaborating in this study.

"We remembered an old friend during our trip along transit network planning."

C.1 Introduction

A number of manufacturing companies require auxiliary equipment, such as molds, for their production processes. We base our study in a plastic injection system where molds are employed for shaping plastics into useful objects. These molds vary in shape, size, and mechanical properties as they are used for the production of different types of pieces. Generally, molds are extremely heavy and need to be moved with cranes, resulting in a large amount of time used for their installation, preparation, and removal (two hours on average for the plastic injection molds). The problem arises when different molds are able to process the same type of piece and when each mold can be installed on different machines, which represent assignment decisions. If we add the fact that for every piece-mold pair there is a different production rate, then the problem is evident: how to make an

accurately schedule of the process to reduce wasted time due to installation and removal of molds. Moreover, the objective of minimizing the cost of the unfulfilled demand is interesting for the companies, since they are commonly forced to buy pieces from a third party to fulfill the demand, in an effort to retain customers. We name this problem as the Piece-Mold-Machine Problem (PMMP).

The characteristics of the PMMP system are the following (see Figure C.1).

- Dedicated molds: Each piece of type i has a set of molds $J(i)$ that can be used to make it. In Figure C.1 piece 1 can be produced with either mold 1 or mold 2.
- Different production rates: Each piece-mold compatible combination has its specific production rate due to technical differences of the molds.
- Dedicated machines: Due to technical differences, each mold j can be assigned to a certain number of machines $M(j)$. In Figure C.1, mold 2 can only be placed on machine 1.
- The demand of the plastic pieces is seldom fulfilled totally by the company. Thus, the production capacity is expected to be fully used.

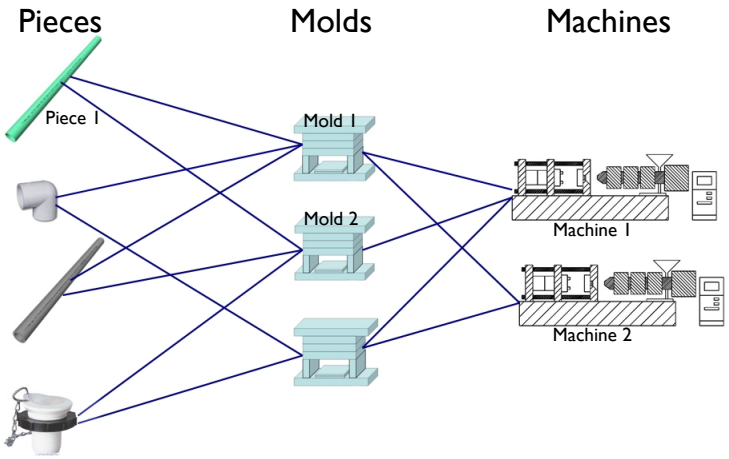


Figure C.1: Illustration of a PMMP diagram.

Therefore, it is necessary to make an accurate lot-sizing and scheduling of the pieces considering real system constraints. The system we base our study on has constraints such

as scarce auxiliary equipment (molds), time-machine bounds, and presence of setup times. When the production of a specific type of piece on some machine is finished, there are two possible situations: to process the next type of piece with the same mold or to switch the mold. In the former case, a preparation time (named here as *piece-setup time*) for the next type of piece must be considered. For instance, this piece-setup time could represent a color or plastic change. In the latter case, if another mold is used, its installation and removal time incur in a *mold-setup time*. Indeed, setup times are dependent of both the last processed event and the event to be next. In general, mold-setup times are larger than piece-setup times.

In Figure C.2 a feasible solution of the PMMP system we are focusing on is represented. On machine 2 there is a mold-setup time because mold 1 is placed after mold 4. Notice that mold 1 is also used on machine 1, so we must consider that a mold cannot be used on more than one machine at the same time. On machine 3 there are piece-setup times between the types of pieces 7 and 4 since mold 3 is used to produce pieces 7, 4, 5, and 9.

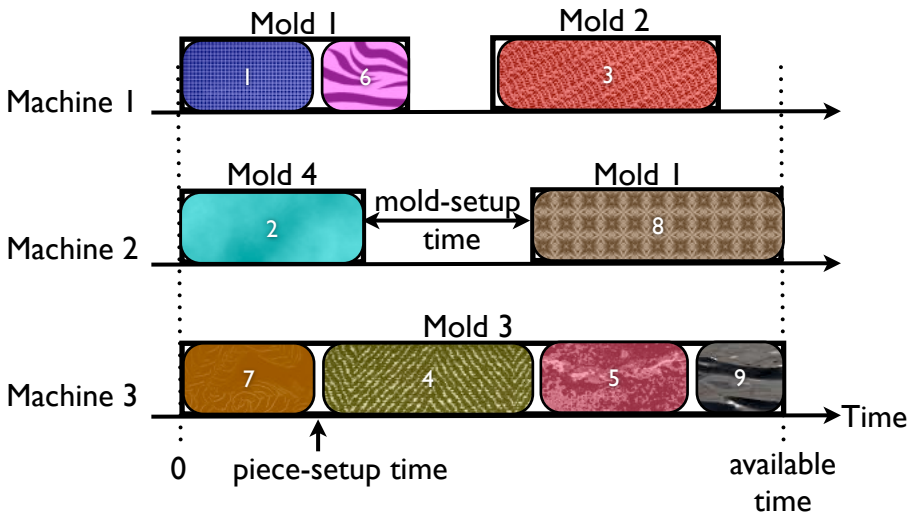


Figure C.2: PMMP production planning Gantt chart.

To consider the auxiliary equipment requirements in parallel machines one needs to avoid the overlapping of processes that use the same mold, since most of the times this equipment is scarce. Usually, scheduling formulations make assumptions about the aux-

iliary equipment such as unique machine assignments. However, this implies sub-optimal solutions since it does not consider the possibility of processing pieces with a specific mold in different machines, which is a feasible solution and often occurs in real-life systems. Therefore, our main contribution is a new approach that gives us the quantities of pieces to produce and the piece-mold-machine assignments such that there is no overlapping of scarce equipment along the planning period.

The new model representing PMMP is a quadratically constrained integer linear program (QCILP) which is difficult to solve. Therefore, we propose a decomposition approach based on lot-sizing and scheduling mixed integer linear programming (MILP) formulations that gives high quality solutions in a reasonable time. A QCIQP is an optimization problem in which both the objective function and the constraints are quadratic functions (Boyd and Vandenberghe [2004]). PMMP is a special case of QCIQP since the quadratic coefficients of its objective function are equal to 0, i.e., its objective function is linear but some of its restrictions are quadratic.

The rest of this appenidx is organized as follows. First, in Section C.2 is presented some related literature from the scheduling and lot-sizing fields. The mathematical model of PMMP is described in Section C.3. Since its computational time is large, in Section C.4 we propose a decomposition approach by decomposing its QCILP formulation into two MILP subproblems. The first one is a formulation with the objective of minimizing the weighted unfulfilled demand that sets the lot-sizing of each piece and the piece-mold-machine assignments. The second one verifies if the solution from the first stage has a feasible mold to machine schedule within the planning period. In Section C.5, we provide empirical evidence of the efficiency of our approach on real-word instances. We conclude with Section C.6 with some final remarks and directions for future research.

C.2 Literature review

There are several features present in our problem corresponding to scheduling and lot-sizing problems. Scheduling problems often consider sequence dependent setup times

between the processing of different types of jobs. However, most of these works are focused on the single machine case (Eren [2007]; Eren and Güner [2006]; Sourd [2006]; Stecco et al. [2008]). Studies that deal with the parallel machine case and sequence dependent setup times, for example Weng et al. [2001], do not consider the use of auxiliary equipment such as molds. Other formulations that present related characteristics to our problem are based on job grouping or scheduling families of jobs (Chen and Powell [2003]; Haase and Kimms [2000]; Li et al. [2005]; Shim and Kim [2008]; Vilím [2006]). However, these types of problems, once again, do not consider additional equipment (e.g. molds).

Chen and Wu [2006] study a scheduling problem on parallel machines with dedicated requirements for the use of molds. Setup times occur when there is a job requiring a different mold than the previous one. Although the authors consider the requirement of molds, there is no possibility of producing a piece with a set of molds, i.e., the piece-mold assignment is not needed as it is in our problem. There are other formulations of scheduling problems that consider sequence dependent setup times and mold requirements, such as Lin et al. [2002], where pieces have to be assigned to specific molds with different production rates. It is noteworthy that a mold is assigned to one machine only. Boctor et al. [2009] address a scheduling problem with the multiple-mold requirement, where processing a specific job requires two specific mold types and a setup implies stopping the entire production, since molds are unique and share common tools. This study has, as far as we know, the most similar characteristics of mold handling when compared to our problem, since the authors ensure molds that use common tools do not overlap for different machines and each instance of time. The authors design heuristic procedures to solve it.

The PMMP has some features present in lot-sizing models such as machine-time capacity, equipment assignment, and sequence dependent setup times (an excellent overview on lot-sizing problems can be found in Karimi et al. [2003]). Chen et al. [2006] address the problem of scheduling, where setups appear when different batch types (or job families) are contiguously processed. Even so, auxiliary equipment is not considered. Dastidar and Nagi [2005] consider lot-sizing and scheduling characteristics such as quantity production calculation, mold-machine assignment, and sequence-dependent setup times. Neverthe-

less, the authors suppose that a mold can be assigned to only one machine in each planning period. Ibarra-Rojas et al. [2010] make the same simplification. This allows them to derive an integer linear program by eliminating the sequence-dependent setup times since they make the assumption of producing the pieces assigned to the same mold one after the other. They determine the piece-mold assignment and the lot-size for each piece-mold pair. They show that the problem is NP-hard and propose an iterated local search algorithm for finding high quality solutions.

C.3 Quadratically constrained linear program model for PMMP

In this section we present a model that integrates the lot-sizing and the scheduling characteristics of PMMP. We assume that a mold is installed at most once on each machine, but notice that it can visit many machines. Indeed, if we have a mold that is installed twice in the same machine then a solution that reduces the mold-setup times is the one that merges these two occurrences into one. On the left hand side of Figure C.3, on machine 1 we have that mold 1 is installed twice. On the right hand side we have another solution that has merged these two occurrences. Notice that mold 1 on machine 2 had to be rescheduled to avoid mold overlapping. This grouping idea is consistent with the way that companies make their production and has been reported to yield positive results when implemented in solution algorithms (e.g., Chen and Wu [2006]). In fact, the company in which this study is based on, uses this assumption in practice.

For the formulation, let I be the set of pieces the company can produce, J the set of available molds for the production of the pieces, and M the set of available machines. We define $J(i)$ as the set of molds that can produce piece i, $M(j)$ as the set of machines compatible with mold j, $I(j)$ as the set of pieces that can be produced using mold j, and $Jm(k)$ as the set of molds that can be placed on machine k. The problem parameters required for our mathematical model are now described, and are summarized in Figure C.5 of Section C.4.

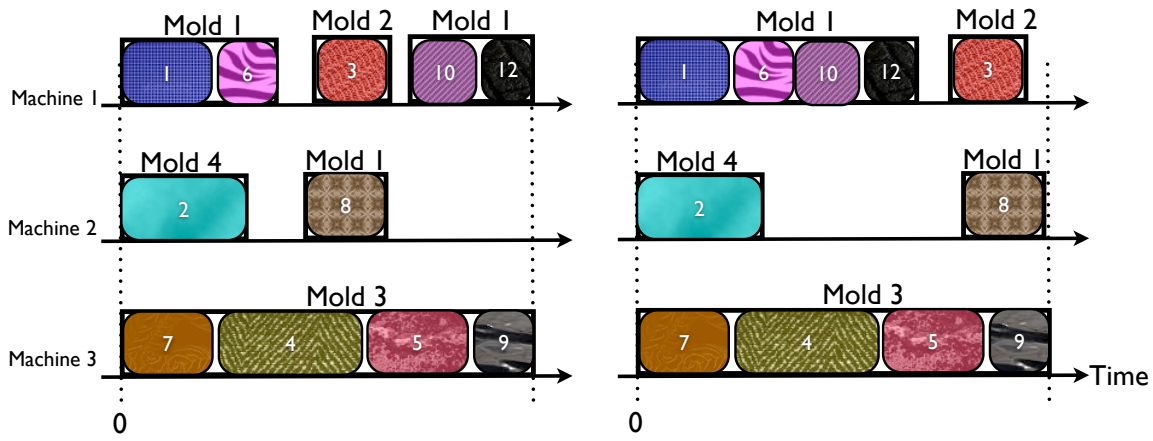


Figure C.3: Illustration of the assumption that a mold is installed at most once on each machine.

Demand of piece i is denoted as d_i. The parameter denoted by st_{ij} corresponds to the piece-setup time of i if produced with mold j, which is independent of the job sequence. Parameters it_{jk} and dt_{jk} represent the installation and removal time of mold j on machine k, respectively. There are molds that can produce the same piece type but at a different rate (more or less cavities) due to technological reasons. Hence the inverse production speed of piece i when using mold j is represented by v_{ij}. Each machine k has an available time for production: tm_k. These times can vary from one machine to another due to the company's preventive maintenance plans.

It is necessary to include a set of binary variables B_{ijk}, taking the value of 1 when there is at least one piece of type i produced with mold j in machine k. Similarly, we make use of another set of binary variables N_{jk}, which take the value of 1 if mold j is installed on machine k. Variables X_{ijk} represent the amount of pieces i to be produced with mold j on machine k.

A common problem in industry is minimizing the cost of the unfulfilled demand. As a consequence, we implicitly seek to maximize the cost of the weighted fulfilled demand. The weight w_i of each piece can be the cost of buying it from another company. Therefore, the objective function of the Piece-Mold-Machine Problem is then

$$\max \sum_{i \in I} w_i \sum_{j \in J(i)} \sum_{k \in M(J)} X_{ijk}. \tag{C.1}$$

System constraints for the lot-sizing of the pieces and for the piece-mold-machine assignments are described by the following expressions.

$$\sum_{j \in J(i)} \sum_{k \in M(j)} X_{ijk} \leq d_i \qquad i \in I \qquad \text{(C.2)}$$

$$X_{ijk} \leq d_i B_{ijk} \qquad i \in I, j \in J(i), \; k \in M(j) \qquad \text{(C.3)}$$

$$\sum_{j \in Jm(k)} [N_{jk}(it_{jk} + dt_{jk}) + \sum_{i \in I(j)} (X_{ijk} v_{ij} + B_{ijk} st_{ij})] \leq tm_k \qquad k \in M \qquad \text{(C.4)}$$

$$\sum_{i \in I(j)} B_{ijk} \leq |I(j)| N_{jk} \qquad j \in J, k \in M(j) \qquad \text{(C.5)}$$

$$X_{ijk} \in \mathbb{Z}^+, B_{ijk} \in \{0,1\}, N_{jk} \in \{0,1\} \qquad j \in J, i \in I(j), k \in M(j)$$

Constraints (C.2) limit the quantity of produced pieces of type i to the demanded amount, as demand must not be exceeded. Constraints (C.3) avoid producing pieces of type i with mold j mounted on machine k if the pieces of type i are not assigned to mold j or if mold j is not assigned to machine k. Constraints (C.4) limit the available time of each machine by taking into account the mold-setup times, the production speed of the pieces, and the pieces-setup times. Constraints (C.5) assign mold j to machine k if there is at least one piece produced with this mold-machine pair, regardless of the type of piece. Finally, last constraints define the domain of the decision variables. Notice that sets $I(j)$, $M(j)$, $J(i)$, and $Jm(k)$ represent the equipment compatibility.

It is important to recognize as an infeasible case the situation where a mold is assigned to different machines in such a way that this mold processes pieces simultaneously on these machines, i.e., the situation when we have overlapping molds. One may think that this situation is not very common, since the setup times tend to be minimized, but in Section C.5 we empirically prove that it happens more often than imagined. However, by adding to the model the following set of constraints

$$\sum_{k \in M(j)} [N_{jk}(it_{jk} + dt_{jk}) + \sum_{i \in I(j)} (X_{ijk} v_{ij} + B_{ijk} st_{ij})] \leq \max_k \{tm_k\} \quad \forall j \in J, \qquad \text{(C.6)}$$

we considerably decrease the amount of overlapping molds as we are limiting the time used by mold j in all machines to the maximum available machine time. Nevertheless, the possibility of overlapping molds still exists. To ensure that we have a feasible solution we must add new sets of variables and restrictions that guarantee a feasible scheduling of the molds on the machines. First, one needs dependent variables T_{jk} to track the total time each mold j is used on machine k:

$$T_{jk} = N_{jk}(it_{jk} + dt_{jk}) + \sum_{i \in I(j)} (X_{ijk} v_{ij} + B_{ijk} st_{ij}), \quad \forall j \in J, k \in M(j). \tag{C.7}$$

Then, we introduce the set of precedence binary variables $F_{j'jk}$, taking the value of 1 when mold j is installed after mold j' in machine k. In counterpart, the set of binary variables $G_{jk'k}$ take the value of 1 if mold j is installed on machine k' before it is installed on machine k. Moreover, we make use of binary variables Y_{jlk}, acquiring the value of 1 if mold j is the l-th event in machine k, i.e., each event of machine j is the installation of a new mold. Finally, variables Z_{jqk} take a value of 1 if machine k is the q-th machine visited by mold j. Event index l belongs to the set $L^k = \{1, \cdots, \sum_j N_{jk}\}$ for all k. Similarly, $q \in Q^j = \{1, \cdots, \sum_k N_{jk}\}$ for all j. The set of constraints that make a feasible scheduling of the molds on the machines are as follows.

$$\sum_{j' \neq j} T_{j'k} F_{j'jk} + T_{jk} \leq tm_k \qquad j \in J, k \in M(j) \tag{C.8}$$

$$\sum_{j' \neq j} T_{j'k'} F_{j'jk'} + T_{jk'} \leq \sum_{j' \neq j} T_{j'k} F_{j'jk} + \mathcal{M}(1 - G_{jk'k}) \quad j \in J, k, k' \in M \tag{C.9}$$

$$\sum_{l \in L^k} Y_{jlk} = N_{jk} \qquad k \in M, j \in Jm(k) \tag{C.10}$$

$$\sum_{j \in Jm(k)} Y_{jlk} = 1 \qquad k \in M, l \in L^k \tag{C.11}$$

$$\sum_{l \in L^k} (lY_{jlk}) - N_{jk} = \sum_{j' \neq j} F_{j'jk} \qquad k \in M, j \in Jm(k) \tag{C.12}$$

$$\sum_{q \in Q^j} Z_{jqk} = N_{jk} \qquad j \in J, k \in M(j) \tag{C.13}$$

$$\sum_{k \in M(j)} Z_{jqk} = 1 \qquad j \in J, q \in Q^j \tag{C.14}$$

$$\sum_{q \in Q^j} (qZ_{jqk}) - N_{jk} = \sum_{k' \neq k} G_{jk'k} \qquad j \in J, k \in M(j) \qquad \text{(C.15)}$$

$$F_{j'jk} \leq N_{jk} N_{j'k} \qquad j \in J, k \in M(j), j' \neq j \qquad \text{(C.16)}$$

$$G_{jk'k} \leq N_{jk} N_{jk'} \qquad j \in J, k \in M(j), k' \neq k \qquad \text{(C.17)}$$

$$G_{jk'k} + G_{jkk'} \leq 1 \qquad j \in J, k, k' \in M \qquad \text{(C.18)}$$

$$F_{j'jk} + F_{jj'k} \leq 1 \qquad j, j' \in J, k \in M \qquad \text{(C.19)}$$

$$F_{j'jk},\ G_{jk'k},\ Y_{ljk},\ Z_{qjk} \in \{0,1\} \qquad j, j' \in J, k, k' \in M, l \in L^k, q \in Q^j$$

Constraints (C.8) bound the completion time of mold j on machine k to the available machine time. Constraints (C.9), illustrated by Figure C.4, imply that the time when mold j can be installed on machine k must be larger than its completion times on other machines k' where it was placed before. Thus, mold j must be placed on machine k after all its previous uses on different machines have been entirely finished. Value $\mathcal{M}$ corresponds to a very large number. Notice that (C.8) and (C.9) are non linear.

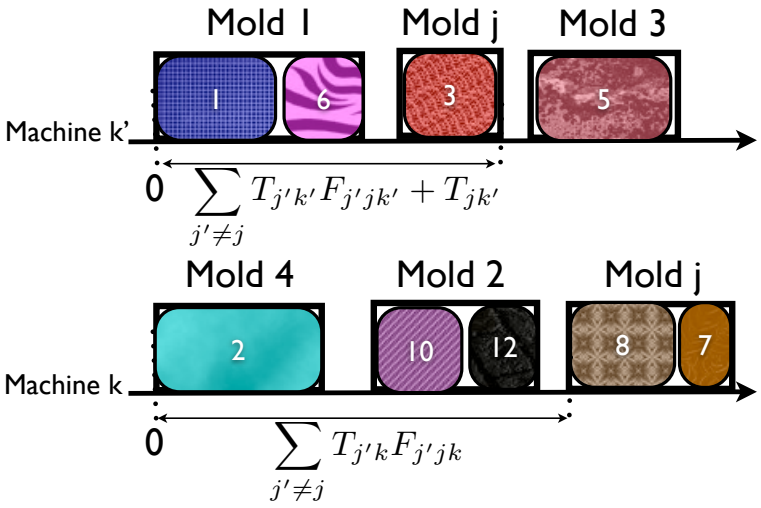


Figure C.4: Illustration of constraints (C.9) when $G_{jk'k} = 1$.

Assignment constrains (C.10) express that if mold j must be placed on machine k ($N_{jk} = 1$) then, this event must be one of the l events taking place on machine k, i.e., these equations assign one of the possible events to the task conformed by the use of mold j in machine k. Constraints (C.11) say that each event in L^k of each machine k

corresponds to only one mold-(machine k) assignment. Expressions (C.12) define that the number of events that take place before mold j is installed on machine k must be equal to the number of molds j' installed before j on machine k.

Constraints (C.13)-(C.15) are analogous to (C.10)-(C.12) but now from the point of view of the machines that every mold visits. For instance, constraints (C.13) assign the q-th machine where mold j is installed. Constraints (C.14) are the assignments of the q visits of mold j. Constraints (C.15) sets the $q-1$ machines visited by mold j before being installed on machine k. Moreover, constraints (C.16) and (C.17), that can be easily linearized, set the relationship between variables $F_{j'jk}$ and $G_{jk'k}$ with N_{jk} and $N_{j'k}$. Constraints (C.18) and (C.19) avoid inconsistencies on the precedence variables. Finally, last constraints define the domain of the decision variables.

Summarizing, the QCILP for PMMP is as follows:

$$
\begin{aligned}
\max \quad & \sum_{i\in I} w_i \sum_{j\in J(i)} \sum_{k\in M(j)} X_{ijk} \\
\text{subject to} \quad & (C.2)-(C.19) \\
& X_{ijk} \in \mathbb{Z}^+, B_{ijk} \in \{0,1\}, N_{jk} \in \{0,1\} \quad j \in J, i \in I(j), k \in M(j) \\
& F_{j'jk},\ G_{jk'k},\ Y_{ljk},\ Z_{qjk} \in \{0,1\} \qquad j, j' \in J, k, k' \in M, l \in L^k, q \in Q^j
\end{aligned}
$$

As mentioned in Section C.2, in Ibarra-Rojas et al. [2010] the authors propose a similar problem to PMMP but suppose that a mold can be assigned to only one machine. Their mathematical model substantially differs from ours since they do not have to guarantee a feasible schedule. Nevertheless, the same reduction they use to prove the NP-hardness of their problem can be adapted for PMMP.

Solving PMMP with the quadratic solver of CPLEX 11.2 did not yield feasible solutions in less than one hour. Indeed, constraints (C.8) and (C.9) are not convex, therefore the CPLEX's solver tries to convexify them which is probably the reason for the inefficiency of this approach. The number of original variables is already considerable, therefore

we do not try to linearize the quadratic constraints. Instead, we propose to decompose PMMP into two MILPs and solve them sequentially, as it is shown in Section C.4.

C.4 A decomposition approach for PMMP

Given the complexity of PMMP, a decomposition approach is proposed. This method attempts to exploit the structure of two related subproblems. In the first level, the Lot-Sizing and Assignment subproblem (LSA) obtains a lot-sizing of the pieces together with a mold-machine assignment. Then, in a second level, a feasibility subproblem called Mold Overlapping Detection (MOD) is solved to verify if there is a scheduling of the molds on the machines for the solution obtained by LSA.

LSA is a MILP given by:

$$\begin{aligned} \max \quad & \sum_{i \in I} w_i \sum_{j \in J(i)} \sum_{k \in M(j)} X_{ijk} \\ \text{subject to} \quad & (C.2) - (C.6) \\ & X_{ijk} \in \mathbb{Z}^+, S_{ijk} \in \{0,1\}, N_{jk} \in \{0,1\} \quad j \in J, i \in I(j), k \in M(j) \end{aligned}$$

As mentioned before, LSA gives as a solution the number of pieces to produce of each type, the molds in which the pieces are processed, and the machines where the molds are placed. Nevertheless, this solution may be infeasible, since a single mold can be installed in two machines at the same time. Although the presence of constraints (C.6) considerably reduce the amount of overlapping molds, in Section C.5 we provide evidence of the mold overlapping situation. Note LSA formulation is a relaxation of PMMP.

There is a simple way to verify if a solution given by LSA is feasible by looking at the values of the variables N_{jk} (mold j is installed or not on machine k). There is no risk of having overlapping molds if each of the molds is assigned to at most one machine, that

is,

$$\sum_{k \in M(j)} N_{jk} \leq 1, j \in J. \tag{C.20}$$

This is a sufficient condition that guarantees that the solution given by LSA is optimal to PMMP. If conditions (C.20) are not satisfied, we have at least one mold assigned to two or more machines implying there could be mold overlapping in the schedule that must be detected. To this end, we solve MOD that is an integer linear constraint satisfaction problem where N_{jk}, S_{ijk}, and T_{jk} correspond now to input parameters.

MOD: Are there 0-1 values for variables $F_{j'jk}$, $G_{jk'k}$, Y_{ljk}, Z_{qjk} $(j, j' \in J, k, k' \in M, l \in L^k, q \in Q^j)$ such that constraints (C.7)-(C.19) are satisfied?

Figure C.5 compiles the notation we have used so far. It also schemes the input/output relations between LSA and MOD.

The decomposition method, named as DPMMP, is presented in Algorithm 15. First

Algorithm 15 DPMMP: Decomposition algorithm for PMMP.

Input: Instance of PMMP

Output: X_{ijk}, Y_{jlk} {Quantity-assignment variables and scheduling variables}

```
1: FeasibleFlag ← ∅
2: while (FeasibleFlag = ∅) do
3:     Solve LSA to obtain the values of the variables X̄_ijk, N̄_jk, and B̄_ijk
4:     if (N̄_jk ≤ 1, ∀j ∈ J, k ∈ M(j)) then
5:         FeasibleFlag ← 1 {The LSA solution is mold overlapping free}
6:     else
7:         Solve MOD with values of N̄_jk, B̄_ijk and X̄_ijk as input
8:         if (MOD is feasible) then
9:             FeasibleFlag ← 1 {The LSA solution is mold overlapping free}
10:        else
11:            Add to LSA cut (C.21) associated to the actual values of X̄_ijk
12:        end if
13:    end if
14: end while
```

LSA is solved. If sufficient conditions (C.20) are verified, then the LSA solution is overlapping free, hence optimal. Otherwise, MOD is solved with the values of variables $\bar{X}_{ijk}$,

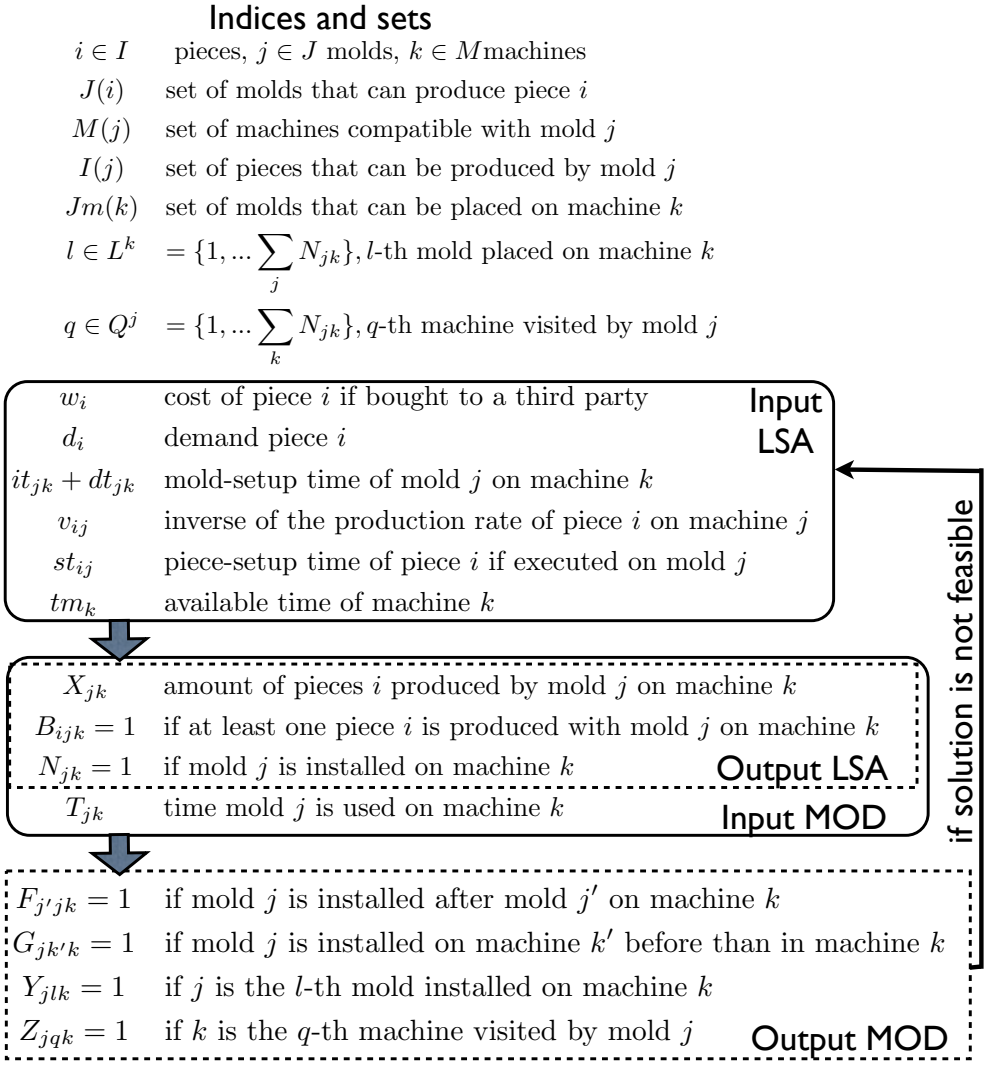


Figure C.5: Input and output relations between LSA and MOD.

$\bar{N}_{jk}$, and $\bar{B}_{ijk}$ as parameters. If there is a feasible mold to machine schedule, then the LSA solution is optimal for PMMP. If this is not met, then one must solve again the LSA model but this time the previously obtained solution $\bar{X}_{ijk}$ must be avoided (as described in the next subsection).

LSA and MOD can be solved by the CPLEX's linear solver. Note that although Step 11 may yield an exponential number of iterations, in practice this seldom happens. In Section C.5, experimental evidence shows that DPMMP procedure needs at most two iterations.

C.4.0 Cutting-off a Previous LSA Infeasible Solution for MOD

As mentioned before, when the problem MOD turns out to be infeasible it means that the last LSA solution must be avoided from the solution set of LSA. To this end, we propose the following cut.

Let $\bar{x} = (\bar{x}_1, \ldots, \bar{x}_n)$ be the values of the solution to be cutoff. Let $\mathcal{I} = \{i|\bar{x}_i = 1\}$ and $\bar{\mathcal{I}} = \{i|\bar{x}_i = 0\}$. Then the following constraint cuts-off this particular solution without cutting off any other feasible solution for LSA:

$$\sum_{i\in\mathcal{I}} x_i + \sum_{i\in\bar{\mathcal{I}}}(1 - x_i) \leq n - 1. \tag{C.21}$$

Indeed, the only manner this inequality could be equal to n is by taking the values of $\bar{x}$. This cut is useful for binary variables but it is well known that any integer can be expressed in a binary form.

C.5 Experimental results

In this section we test the efficiency of algorithm DPMMP to solve PMMP. To this end, we use the same instance generator proposed by Ibarra-Rojas et al. [2010] since, except for the weight of pieces, it is based on information provided by a real manufacturer of plastic products. Parameters are integers uniformly generated within the following ranges: $d_i \in [1000, 18000]$, $v_{ij} \in [1/120, 1/1000]$, $w_i \in [1, 10]$. The piece-setup time is set to $st_{ij} = 0$, for all i and j. Indeed, these values do not affect the behavior of the algorithm, provided that $st_{ij} \ll it_{jk} + dt_{jk}$. The available time of all the machines is set to $t_k = 24$ hours.

On the one hand, the difficulty of the instances resides on the mold-setup times (in hours). Hence, we generate two set of instances. Mold-setup times of instance set E take integer values uniformly generated as follows: $it_{jk} + dt_{jk} \in [.75, 1.15]$. More

difficult instances H are obtained by uniformly generating the installation setup time and the removal setup time independently from [.75, 1.15], i.e., $it_{jk} \in [.75, 1.15]$, and $dt_{jk} \in [.75, 1.15]$.

On the other hand, we consider (as in Ibarra-Rojas et al. [2010]) two density levels (percentage of compatibilities) for the piece-mold compatibility, r_{ij}, and the mold-machine compatibility, m_{jk}. In general, a higher density makes a more difficult instance since there are more choices (therefore more variables). We use densities of 5% and 15% for the piece-mold compatibility together with a density of 60% for the mold-machine compatibility. Preliminary tests showed that the difficulty of an instance does not reside in the mold-machine density and 60% is close to the reality. Instances with 5% for the piece-mold compatibility correspond to a small enterprise or to old molds and machines. Realistic values are around 15%.

Note that we cannot compare our results to the ones of Chen and Powell [2003], Tsai and Tseng [2007], or Ibarra-Rojas et al. [2010] because they assume that a mold can be only placed on at most one machine, so in essence we address a different problem.

For solving the instances we used GAMS/CPLEX 11.2 as the MILP optimizer (we kept the default settings except for the optimality tolerance that we set to 1×10^{-8}) on a Sun Fire V440 with 4 UltraSparc III processors at 1.062 GHz and 8 GB of RAM. We set a time limit of one hour for each iteration of LSA, and another hour limit for each iteration of MOD.

Table C.1 presents the experimental results obtained by solving the instance set E with DPMMP, while Table C.2 is for the instance set H. Each row of these tables is the average of 10 executions of different instances. The variance between each class of instances is not significant, so the average is a good indicator of the results. The tables present only the first iteration of the algorithm since only two out of 100 needed a second iteration (these two instances will be discussed later). Column "Instances" refers to the instance size: number of pieces, number of molds, and number of machines, $(|I|, |J|, |M|)$. Column "Densities" is about the densities of the piece-mold and the mold-machine compatibilities denoted as r_{ij} and m_{jk}, respectively. Column "LSA GAP %" is

the relative gap obtained when solving LSA with CPLEX's solver. Column "LSA Time" is the average time in seconds that CPLEX took to solve LSA, limited to one hour. Column "Molds ($\sum_k N_{jk}$)" indicates how many molds are assigned to different machines in the most extreme single instance: a value of 5→2 indicates that there is an instance for which 5 molds visit, each of them, 2 machines ("-" indicates that all molds visit only one machine). Column labeled as "MOD %" shows the percentage of instances that needed MOD to check for a feasible scheduling. The "Overlap free %" column represents the percentage of instances that have a feasible scheduling for the LSA solution. Finally, the last column is the average time in seconds needed by CPLEX to solve MOD.

Instances ($\|I\|,\|J\|,\|M\|$)	Densities (r_{ij}, m_{jk})	LSA GAP %	LSA Time	Molds ($\sum_k N_{jk}$)	MOD %	Overlap free %	MOD Time
(50,30,5)	(15,60)	0.00	103.4	1→2	10	100	0.4
(120,80,20)	(5,60)	0.76	3600.0	5→2	100	100	14.3
(120,80,20)	(15,60)	1.00	3600.0	5→2	80	100	14.3
(200,120,25)	(5,60)	1.09	3600.0	7→2	100	100	39.9
(200,120,25)	(15,60)	2.49	3600.0	13→2 and 1→3	100	80	41.0

Table C.1: Results obtained by solving PMMP with algorithm DPMMP on the instance set E (shorter mold-setup times).

Instances ($\|I\|,\|J\|,\|M\|$)	Densities (r_{ij}, m_{jk})	LSA GAP %	LSA Time	Molds ($\sum_k N_{jk}$)	MOD %	Overlap free %	MOD Time
(50,30,5)	(15,60)	0.00	71.4	–	0	–	–
(120,80,20)	(5,60)	2.73	3600.0	3→2	80	100	14.2
(120,80,20)	(15,60)	4.17	3600.0	3→2	80	100	14.5
(200,120,25)	(5,60)	3.83	3600.0	6→2	100	100	39.9
(200,120,25)	(15,60)	6.20	3600.0	4→2	80	100	39.6

Table C.2: Results obtained by solving PMMP with algorithm DPMMP on the instance set H (longer mold-setup times).

The real size of the instances we observed in a plastic manufacturing company were (120, 80, 20) with a density of (5, 60), from the instance set E. Indeed, the maximum size of the instances presented in Tables C.1 and C.2 is, to the best of our knowledge, the largest in literature.

The density of the piece-mold compatibilities is a factor that makes an instance harder than another one, regardless of the mold-setup times. Indeed, when the piece-

mold compatibility is only 5% there are less options, therefore less variables.

Tables C.1 and C.2 show that the solutions given by LSA are close to the optimum since the gap is never above 7% (or is never above 2.5% under the real system characteristics denoted in set E), which is for the industry more than acceptable. Notice that the gaps for the harder instance set H are larger since longer mold-setup times make the assignments of the molds to the machines more difficult.

Except for the smallest instances, the computational times for the PMMP model reach the time limit for LSA. Nevertheless, letting LSA run for more than one hour did not improve the gap much. Figure C.6 represents the relative gap of LSA for a single instance of set E along the time. Notice that the quality of the solution is already acceptable after one hour: Although the gap keeps decreasing beyond this limit, the drop in the gap gets shorter as the computational time increases.

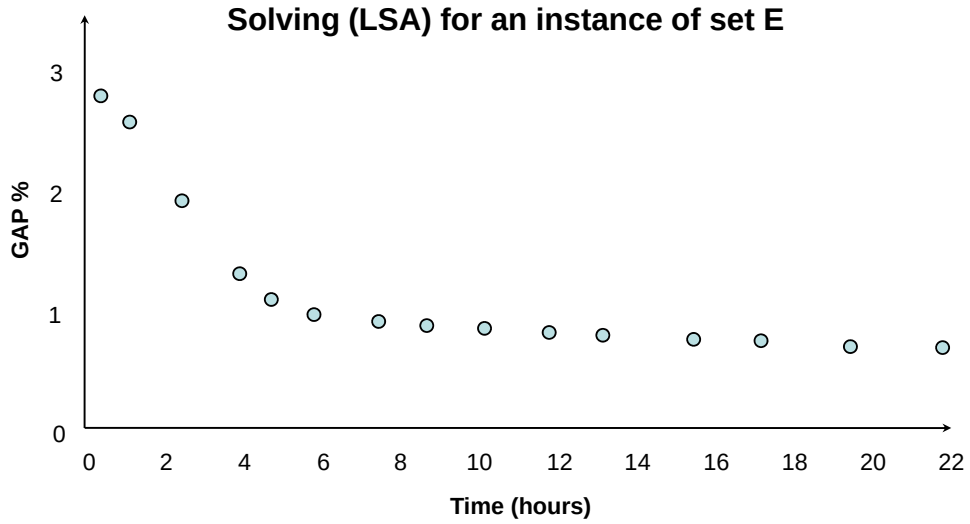


Figure C.6: Relative gap of LSA for a single instance of size (200,120,80) with density (15,60) of set E along the time.

We noted that the harder instances H required fewer times the use of MOD. Indeed, since the mold-setup times are larger, in the optimal solution there is less tendency to use a mold on several machines. Nevertheless, notice the case (200,120,25) from instance set H where six molds visit, each one of them, two machines.

The counterintuitive case of molds assigned to several machines is in fact real and happens frequently, making it necessary, in most of the cases, to use MOD. Moreover, MOD finds most of the times a feasible mold to machine scheduling in less than one minute which is surprising for a scheduling problem that has a very large value $\mathcal{M}$ in constraint (C.9). Actually, the number of molds that may overlap in several machines is small compared to the ones that are fixed to only one machine, making easy for MOD to find a feasible scheduling. Notice that the computational time of MOD is equivalent for both H and E.

In Table C.1 we can see more counterintuitive cases. For example, the last line instances present an instance where 13 molds are used twice on different machines and one mold is used on three different machines. Nevertheless, it is in this set where we find the two instances out of 100 for which MOD could not find a feasible mold to machine scheduling for the solution given by LSA. These instances needed a second iteration of the algorithm. We summarize the results obtained:

- The execution times of the second iteration where similar to the ones of the first iteration.
- After the second iteration, the decrement of the initial solution value is around 0.25%.
- Although the more convenient way of cutting off the infeasible solution of LSA is by including the cut (C.21) (this restriction only forbids the infeasible solution), we obtained the same results (same solution and same objective value) by forbidding the previous value of the objective function $\bar{z}$ i.e., adding to LSA $z \leq \bar{z} - 1$. Contrary to cut (C.21), this last option does not guarantee that we are leaving out an interesting feasible solution.
- Experimentally, we did not find two different solutions with the same objective value which validates the use of the simpler cut.

From this experimental section we conclude that our decomposition approach is efficient since for the company the solutions are close enough to the optimum. In practice,

the production planning must be done only once a week therefore the computational time of our algorithm is reasonable. As mentioned before, the optimality gap in this study is set close to 0. For a practical use, a company could set the optimality gap to 1% to reduce the execution time of LSA.

C.6 Conclusions

We present in this study a real manufacturing process of pieces that are produced with molds which are mounted on machines. Setup times between jobs, dedicated parallel machines, dedicated molds and different production rates for each piece-mold pair are some of the main characteristics of the problem. Moreover, we do not make the common assumption of forcing a mold to be placed on a single machine. As a consequence, we deal with a more realistic description of the production process itself. The objective function is about maximizing the weighted cost of the produced pieces.

We first propose a new integer quadratically constrained linear programming that represents the PMMP problem. Since this model is difficult to tackle by itself, we present a decomposition approach based on two new MILPs: the first one determines the lot-size of each piece and the piece-mold-machine assignments, and the second one verifies that there is a feasible mold to machine scheduling along the planning period.

Experimental results show that indeed, most of the solutions yield to configurations where the molds visit more than one machine. Some instances have molds that visit up to three machines. Our exact-based methodology gives, in a reasonable amount of time, solutions of high quality for real size instances.

Bibliography

On a Bicriterion Formulation of the Problems of Integrated System Identification and System Optimization. *Systems, Man and Cybernetics, IEEE Transactions on*, 1(3): 296–297, July 1971.

A. Adamski. Public transport lines synchronization problem. In *Scientific Bulletin of AGH*, volume 64, pages 221–234. AGH, 1993.

P. Ávila and F. López. Un enfoque integrado multicriterio para la planificación de las frecuencias de paso y las tablas de tiempo de una empresa de transporte urbano. Master's thesis, Universidad Autónoma de Nuevo León, Mexico, 2012.

R. Barták. Theory and practice of constraint propagation. In *In Proceedings of the 3rd Workshop on Constraint Programming in Decision and Control (CPDC 2001)*, pages 7–14, 2001.

C. Bessiere. Constraint propagation. Technical Report LIRMM 06020, University of Montpellier, 2006.

F. Boctor, J. Renaud, and J. E. Rapp. Sequencing and scheduling multi-mold injection molding machines. *Bibliographie du Québec*, 2009-015:18p, 2009.

J. H. Bookbinder and A. Désilets. Transfer optimization in a transit network. *Transportation Science*, 26(2):106–118, 1992.

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

G. Caimi, M. Fuchsberger, M. Laumanns, and K. Schüpbach. Periodic railway timetabling with event flexibility. *Networks*, 57(1):3–18, 2011.

D. G. Carmichael. Computation of pareto optima in structural design. *International Journal for Numerical Methods in Engineering*, 15(6):925–929, 1980.

L. Castelli, R. Pesenti, and W. Ukovich. Scheduling multimodal transportation systems. *European Journal of Operational Research*, 155(3):603–615, 2004.

A. Ceder. Efficient timetabling and vehicle scheduling for public transport. In Stefan Voßand Joachim R. Daduna, editors, *Computer-Aided Scheduling of Public Transport*, volume 505 of *Lecture Notes in Economics and Mathematical Systems*, pages 37–52. Springer Berlin Heidelberg, 2001.

A. Ceder. Designing public transport network and routes. In W. Lam and M. Bell, editors, *Advanced Modeling for Transit Operations and Service Planning*, chapter 3, pages 59–91. Pergamon Imprint, Elsevier Science Ltd Pub, 2003.

A. Ceder. *Public Transit Planning and Operation: Theory, Modeling and Practice*. Elsevier, Butterworth-Heinemann, 2007.

A. Ceder. Optimal multi-vehicle type transit timetabling and vehicle scheduling. *Procedia-Social and Behavioral Sciences*, 20:19–30, 2011.

A. Ceder and O. Tal. Designing synchronization into bus timetables. *Transportation Research Record: Journal of the Transportation Research Board*, 1760(1):28–33, 01 2001.

A. Ceder, B. Golany, and O. Tal. Creating bus timetables with maximal synchronization. *Transportation Research Part A: Policy and Practice*, 35(10):913–928, 2001.

Centro de Desarrollo Metropolitano y Territorial. Fotogalería. `http://cedem.mty.itesm.mx/5.htm`, 2012. Accessed: 19/10/2012.

F. Cevallos and F. Zhao. Minimizing Transfer Times in Public Transit Network with Genetic Algorithm. *Transportation Research Record*, 1971:74–79, 2006.

O. L. Chacón-Mondragón, O. J. Ibarra-Rojas, Y. A. Ríos-Solís, and R. Z. Ríos-Mercado. Programación pieza-molde-máquina en planeación de la producción. *Ciencia UANL*, 15(58):59–65, 2012.

P. Chakroborty, K. Deb, and P. S. Subrahmanyam. Optimal scheduling of urban transit systems using genetic algorithms. *Journal of Transportation Engineering*, 121(6):544–553, 1995.

P. Chakroborty, K. Deb, and H. Porwal. A genetic algorithm based procedure for optimal transit system scheduling. In *Proceedings of Fifth International Conference on Computers in Urban Planning and Urban Management*, pages 330–341, Mumbai, India, 1997.

B. Chen, Y. Ye, and J. Zhang. Lot-sizing scheduling with batch setup times. *Journal of Scheduling*, 9(3):299–310, 2006.

J. F. Chen and T. H. Wu. Total tardiness minimization on unrelated parallel machine scheduling with auxiliary equipment constraints. *Omega*, 34(1):81–89, 2006.

Z. L. Chen and W. B. Powell. Exact algorithms for scheduling multiple families of jobs on parallel machines. *Naval Research Logistics*, 50(7):823–840, 2003.

H. W. Chun and S. H. Chuen Chan. The design of a multi-tired bus timetabling system. In *IEA/AIE '99 Proceedings of the 12th international conference on Industrial and engineering applications of artificial intelligence and expert systems: multiple approaches to intelligent systems*, pages 771–779, 1999.

E. Chung. *Transfer coordination model and real-time strategy for inter-modal transit services*. PhD thesis, Graduate Department of Civil Engineering, University of Toronto, 1990.

J. R. Daduna and J. M. Paixao. Vehicle scheduling for public mass transit, an overview. In *Computer-Aided Transit Schedulig*, volume 430, pages 76–90, 1995.

J. R. Daduna and S. Voß. Practical experiences in schedule synchronization. In J.R. Daduna, I. Branco, and J.M.P. Paixao, editors, *Computer-Aided Scheduling of Public Transport*, volume 430 of *Lecture Notes in Economics and Mathematical Systems*, pages 39–55. Springer, Berlin, 1995.

C. F. Daganzo. A headway-based approach to eliminate bus bunching: Systematic analysis and comparisons. *Transportation Research Part B: Methodological*, 43(10):913–921, 2009.

S. G. Dastidar and R. Nagi. Scheduling injection molding operations with multiple resource constraints and sequence dependent setup times and costs. *Computers & Operations Research*, 32(11):2987–3005, 2005.

K. Deb and P. Chakroborty. Time scheduling of transit systems with transfer considerations using genetic algorithms. *Evolutionary Computation*, 6(1):1–24, 1998.

G. Desaulniers and M. Hickman. Public transit. *Transportation, Handbooks in Operations Research and Management Science*, pages 69–127, 2007.

M. Dessouky, R. Hall, A. Nowroozi, and K. Mourikas. Bus dispatching at timed transfer transit stations using bus tracking technology. *Transportation Research Part C: Emerging Technologies*, 7(4):187–208, 1999.

A. Eranki. A model to create bus timetables to attain maximum synchronization considering waiting times at transfer stops. Master's thesis, Department of Industrial and Management Systems Engineering, University of South Florida, 2004.

T. Eren. A multicriteria scheduling with sequence-dependent setup times. *Applied Mathematical Sciences*, 1(58):2883–2894, 2007.

T. Eren and E. Güner. A bicriteria scheduling with sequence-dependent setup times. *Applied Mathematics and Computation*, 179(1):378–385, 2006.

C. Fleurent and R. Lessard. Integrated timetabling and vehicle scheduling in practice. Technical report, GIRO Inc, Montreal, Canada, 2009.

C. Fleurent, R. Lessard, and L. Séguin. Transit Timetable Synchronization: Evaluation and Optimization. In *9 thInternational Conference on Computer Aided Scheduling in Public Transport (CASPT)*, pages 9–11, San Diego, California, August 2005.

S. Fournier. Dedicated heuristic for a back-and-forth single-line bus trip timetabling problem. In *Proceedings of the XLII Simpósio brasileiro de pesquisa operacional*, 2010.

R. Freling, A. Wagelmans, and J. Paixao. Models and algorithms for single-depot vehicle scheduling. *Transportation Science*, 35(2):165–180, 2001.

R. Freling, D. Huisman, and A. P. M. Wagelmans. Models and algorithms for integration of vehicle and crew scheduling. *Journal of Scheduling*, 6(1):63–85, 2003.

C. Friberg and K. Haase. An exact algorithm for the vehicle and crew scheduling problem. In *Computer-aided transit scheduling. Volume 171 of Lectures Notes in Economics and Mathematical Systems*. Springer, 1999.

A. Fügenschuh. Solving a school bus scheduling problem with integer programming. *European Journal of Operational Research*, 193:867–884, 2009.

M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1 edition, 1979.

C. Giesemann. Periodic timetable generation. Published in a Seminar of the University of Constance, 2002.

F. Glover. Future paths for integer programming and links to artificial intelligence. *Comput. Oper. Res.*, 13(5):533–549, 1986.

V. Guihaire and J. K. Hao. Transit network re-timetabling and vehicle scheduling. In Hoai An Le Thi, Pascal Bouvry, and Tao Pham Dinh, editors, *Modelling, Computation and Optimization in Information Systems and Management Sciences*, volume 14 of *Communications in Computer and Information Science*, pages 135–144. Springer Berlin Heidelberg, 2008a.

V. Guihaire and J. K. Hao. Transit network design and scheduling: a global review. *Transportation Research Part A: Policy and Practice*, 42:1251–1273, 2008b.

V. Guihaire and J. K. Hao. Transit network timetabling and vehicle assignment for regulating authorities. *Computers and Industrial Engineering*, 59(1):16–23, 2010a.

V. Guihaire and J. K. Hao. Improving timetable quality in scheduled transit networks. In *Proceedings of the 23rd international conference on Industrial engineering and other applications of applied intelligent systems, IEA/AIE '10*, pages 21–30, 2010b.

K. Haase and A. Kimms. Lot sizing and scheduling with sequence-dependent setup costs and times and efficient rescheduling opportunities. *International Journal of Production Economics*, 66(2):159–169, 2000.

R. Hall, M. Dessouky, and Q. Lu. Optimal holding times at transfer stations. *Computers & Industrial Engineering*, 40(4):379–397, 2001.

P. Hansen and N. Mladenović. Variable neighborhood search. *Handbook of metaheuristics*, pages 145–184, 2003.

D. Huisman, R. Freling, and A. P. M. Wagelmans. Multiple-depot integrated vehicle and crew scheduling. *Transportation Science*, 39(4):491–502, 2005.

C. L. Hwang and A. S. M. Masud. *Multiple objective decision making, methods and applications: a state-of-the-art survey*. Lecture notes in economics and mathematical systems. Springer-Verlag, 1979.

O. J. Ibarra-Rojas and Y. A. Rios-Solis. Synchronization of bus timetabling. *Transportation Research Part B: Methodological*, 46(5):599–614, 2012.

O. J. Ibarra-Rojas, Y. A. Rios-Solis, and O. L. Chacon-Mondragón. Piece-mold-machine manufacturing planning. In Barin Nag, editor, *Intelligent Systems in Operations: Models, Methods, and Applications in the Supply Chain*, pages 105–117. IGI Global, 2010.

O. J. Ibarra-Rojas, R. Z. Ríos-Mercado, Y. A. Rios-Solis, and M. A. Saucedo-Espinosa. A decomposition approach for the piecemoldmachine manufacturing problem. *International Journal of Production Economics*, 134(1):255 – 261, 2011.

L. N. Jansen and O. A. Nielsen. Minimizing passenger transfer times in public transport timetables. In *7th Conference of the Hong Kong Society for Transportation Studies, Transportation in the information age*, pages 229–239, Hong Kong, 2002.

B. Karimi, S. M. T. Fatemi, and J. M. Wilson. The capacitated lot sizing problem: a review of models and algorithms. *Omega*, 31(5):365–378, 2003.

A. Kéri and K. Haase. Vehicle and crew scheduling with flexible timetable. In K. H. Waldmann and U. M. Stocker, editors, *Operations Research Proceedings 2006*, volume

2006 of *Operations Research Proceedings*, pages 339–342. Springer Berlin Heidelberg, 2007.

W. D. Klemt and W. Stemme. Schedule synchronization for public transit networks. In *Computer-Aided Transit Scheduling: Proceedings of the 4th International Workshop on Computer-Aided Scheduling of Public Transport*, pages 327–335, Hamburg, Germany, 1988. Springer Verlag.

H. N. Koutsopoulos, A. Odoni, and N. H. M. Wilson. Determination of headways as a function of time varying characteristics on a transit network. In J.M. Rousseau, editor, *Computer Scheduling of Public Transport 2*, pages 391–414. North-Holland, Amsterdam, 1985.

C. M. Kwan and C. S. Chang. Timetable synchronization of mass rapid transit system using multiobjective evolutionary approach. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 38(5):636–648, 2008.

B. Laurent and J. K. Hao. Simultaneous vehicle and crew scheduling for extra urban transports. In *Proceedings of the 21st international conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems: New Frontiers in Applied Artificial Intelligence*, IEA/AIE '08, pages 466–475, Berlin, Heidelberg, 2008. Springer-Verlag.

S. Li, G. Li, and S. Zhang. Minimizing makespan with release times on identical parallel batching machines. *Discrete Applied Mathematics*, 148(1):127–134, 2005.

Z. C. Li, W. Lam, S. Wong, and A. Sumalee. An activity-based approach for scheduling multimodal transit services. *Transportation*, 37(5):751–774, 2010.

C. Liebchen. Symmetry for periodic railway timetables. *Electronic Notes in Theoretical Computer Science*, 92:34–51, 2004.

C. Liebchen. Periodic timetable optimization in public transport. In Karl-Heinz Waldmann and Ulrike M. Stocker, editors, *Operations Research Proceedings 2006*, volume 2006 of *Operations Research Proceedings*, pages 29–36. Springer Berlin Heidelberg, 2007.

C. Liebchen and R. Möhring. A case study in periodic timetabling. *Electronic Notes in Theoretical Computer Science*, 66(6):18–31, 2002.

C. Liebchen and R. Möhring. The modeling power of the periodic event scheduling problem: Railway timetables and beyond. In Frank Geraets, Leo Kroon, Anita Schoebel, Dorothea Wagner, and Christos Zaroliagis, editors, *Algorithmic Methods for Railway Optimization*, volume 4359 of *Lecture Notes in Computer Science*, pages 3–40. Springer Berlin / Heidelberg, 2007.

C. Liebchen and S. Stiller. Delay resistant timetabling. *Public transport*, 1(1):55–72, 2009.

C. K. Y. Lin, C. L. Wong, and Y. C. Yeung. Heuristic approaches for a scheduling problem in the plastic molding department of an audio company. *Journal of Heuristics*, 8(5): 515–540, 2002.

Z. Liu, J. Shen, H. Wang, and W. Yang. Regional bus timetabling model with synchronization. *Journal of Transportation Systems Engineering and Information Technology*, 7(2):109–112, 2007.

H. Lourenço, O. Martin, and T. Stützle. Iterated local search. *Handbook of metaheuristics*, pages 320–353, 2003.

R. T. Marler and J. S. Arora. Survey of multi-objective optimization methods for engineering. *Structural and Multidisciplinary Optimization*, 26(6):369–395, 2004.

M. Mesquita, A. Paias, and A. Respício. Branching approaches for integrated vehicle and crew scheduling. *Journal of Public Transport*, 1(1):21–37, 2009.

A. Messac and C. Mattson. Generating well-distributed sets of pareto points for engineering design using physical programming. *Optimization and Engineering*, 3:431–450, 2002.

K. Miettinen. *Nonlinear Multiobjective Optimization*. International Series in Operations Research & Management. Springer, 1 edition, 1998.

G. L. Nemhauser and L. A. Wolsey. *Integer and combinatorial optimization*. Wiley Publishing, 1999.

A. De Palma and P. Lindsey. Optimal timetables for public transportation. *Transportation Research Part B*, 35(8):789–813, 2001.

C. H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Dover Publications, 1998.

L. Quadrifoglio, M. M. Dessouky, and F. Ordóñez. Mobility allowance shuttle transit (MAST) services: MIP formulation and strengthening with logic constraints. *European Journal of Operational Research*, 185:481–494, 2008.

P. Rabut. Système de transport urbain de l'aire métropolitaine de monterrey (mexique) analyse de l'offre et de la demande. Master's thesis, Université de Franche-Comté, France, 2010.

M. H. Rapp and C. D. Gehner. Transfer optimization in an interactive graphic system for transit planning. *Transportation Research Record*, 619:27–33, 1976.

T. Schaefer. The complexity of satisfiability problems. In *Proceedings of the tenth annual ACM symposium on Theory of computing*, pages 216–226, San Diego, California, US, 1978.

M. Schröder and I. Solchenbach. Optimization of transfer quality in regional public transit. Technical Report 84, Berichte des Fraunhofer Instituts for Techno-und Wirtschafts, 2006.

A. Shariat and S. M. Mahdi. Creating bus timetables under stochastic demand. *International Journal of Industrial Engineering & Production Research*, 20(3):83–91, 2009.

S. O. Shim and Y. D. Kim. A branch and bound algorithm for an identical parallel machine scheduling problem with a job splitting property. *Computers & Operations Research*, 35(3):863–875, 2008.

F. Sourd. Dynasearch for the earliness-tardiness scheduling problem with release dates and setup constraints. *Operations Research Letters*, 34(5):591–598, 2006.

G. Stecco, J. F. Cordeau, and E. Moretti. A branch-and-cut algorithm for a production scheduling problem with sequence-dependent and time-dependent setup times. *Computers & Operations Research*, 35(8):2635–2655, 2008.

I. Steinzen, V. Gintner, L. Suhl, and N. Kliewer. A time-space network approach for the integrated vehicle- and crew-scheduling problem with multiple depots. *Transportation Science*, 44(3):367–382, 2010.

E. G. Talbi. *Metaheuristics: From Design to Implementation.* Wiley Publishing, 2009.

K. Torrance, A. R. Haire, and R. B. Machemehl. Vehicle and driver scheduling for public transit. Technical Report SWUTC/09/476660-00063-1, Center for Transportation Research, 2009.

C. Tsai and C. Tseng. Unrelated parallel machines scheduling with constrained resources and sequence–dependent setup time. In A. B. Eltawil M. H. Elwany, editor, *Proceedings of the 37th International Conference on Computers and Industrial Engineering*, Alexandria, Egypt, 2007.

A. van den Heuvel, J. van den Akker, and M. van Kooten. Integrating timetabling and vehicle scheduling in public bus transportation. Technical Report UU-CS-2008-003, Department of Information and Computing Sciences, Utrecht University, Utrecht, The Netherlands, 2008.

P. Vilím. Batch processing with sequence dependent setup times. In Pascal Van Hentenryck, editor, *Principles and Practice of Constraint Programming - CP 2002*, volume 2470 of *Lecture Notes in Computer Science*, pages 153–167. Springer Berlin, 2006.

M. X. Weng, J. Lu, and H. Ren. Unrelated parallel machine scheduling with setup consideration and a total weighted completion time objective. *International Journal of Production Economics*, 70(3):215–226, 2001.

H. P. Williams. *Model Building in Mathematical Programming.* Wiley Publishing, 4 edition, 1999.

L. A. Wolsey. *Integer Programming.* Wiley Publishing, 1998.

R. C. W. Wong and J. M. Y. Leung. Timetable synchronization for mass transit railway. In *Proceedings of the 9th International Conference on Computer-Aided Scheduling of Public Transport (CASPT)*, San Diego, California, 2004.

R. C. W. Wong, T. W. Y. Yuen, K. W. Fung, and J. M. Y. Leung. Optimizing timetable synchronization for rail mass transit. *Transportation Science*, 42(1):57–69, 2008.

F. Zhao and X. Zeng. Optimization of transit route network, vehicle headways, and timetables for large-scale transit networks. *European Journal of Operational Research*, 186(2):841–855, 2008.

List of Figures

List of Tables

Autobiography

Omar Jorge Ibarra Rojas

Candidato para el grado de Doctor en Ingeniería
con especialidad en Ingeniería de Sistemas

Universidad Autónoma de Nuevo León

Facultad de Ingeniería Mecánica y Eléctrica

tesis:

Models and algorithms for transit network planning

I was born in December 14th of 1983 in Nuevo Leon, Mexico. I am the second son of Mr. Rodolfo Hugo Ibarra Zarazua and Ms. Ma. Santos Rojas Barron. I get my Bachelor Sc degree in mathematics at the Universidad Autonoma de Nuevo Leon (UANL) in July of 2005. I finished my master studies in 2009 at the Graduate Program in Systems Engineering in the UANL with the support of the National Council of Science and Technology (CONACYT). Later, in 2009, I began the study of this dissertation based on problems of transit network planning for Latin American cases under the guidance of Dra. Yasmín Ríos Solís . My research area is the definition of optimization problems in manufacturing and transit network planning along with the design of efficient solution techniques for these problems.