Study of Lane Reservation Problems in a Transportation Network

Yunfei Fang

Thèse
de doctorat
de l'UTT

Yunfei FANG

Study of Lane Reservation Problems in a Transportation Network

Spécialité :
Optimisation et Sûreté des Systèmes

2013TROY0010 **Année 2013**

utt
université de technologie
Troyes

THESE

pour l'obtention du grade de

DOCTEUR de l'UNIVERSITE DE TECHNOLOGIE DE TROYES

Spécialité : OPTIMISATION ET SURETE DES SYSTEMES

présentée et soutenue par

Yunfei FANG

le 18 juin 2013

Study of Lane Reservation Problems in a Transportation Network

JURY

M. S. COHEN	DIRECTEUR DE RECHERCHE	Président
M. J. CARLIER	PROFESSEUR DES UNIVERSITES	Rapporteur
Mme Feng CHU	PROFESSEUR DES UNIVERSITES	Directrice de thèse
M. I. KACEM	PROFESSEUR DES UNIVERSITES	Rapporteur
Mme N. LABADIE	MAITRE DE CONFERENCES - HDR	Examinateur
M. S. MAMMAR	PROFESSEUR DES UNIVERSITES	Directeur de thèse
M. M. PAPAGEORGIOU	PROFESSOR	Examinateur

Study of Lane Reservation Problems in a Transportation Network

by

Yunfei FANG

laboratoire d'Optimisation des Systèmes Industriels (LOSI)

Institut Charles Delaunay (ICD), STMR UMR 6279 CNRS

Université de Technologie de Troyes, France

Supervisors: Prof. Feng CHU and Prof. Saïd MAMMAR

June 18, 2013

Acknowledgements

It is not an easy thing to pursue a PhD degree. When I recall the past years, many people have helped me and I would like to express my appreciation to them.

Firstly, I would like to express my sincere gratitude to Prof. Jacques CARLIER, Research Director Simon COHEN, Prof. Imed KACEM, Associate Prof. Nacima LABADIE and Prof. Markos PAPAGEORGIOU for their insightful comments on my work and serving as jury members in my Ph.D defense committee.

I would like to sincerely thank my supervisors, Prof. Feng CHU and Prof. Saïd MAMMAR, for their guidance throughout my PhD study. I have been grateful to Prof. Feng CHU, who have taught me so much, both in my study and life. I have also been grateful to Prof. Saïd MAMMAR, who provide insightful comments on my work.

I would like to thank Prof. Ada CHE (Northwestern Polytechnical University (NWPU), China) for his guidance during my short stay at NWPU, as well as his insightful comments on my work. I would like to thank Prof. Mengchu ZHOU (New Jersey Institute of Technology, USA) for his insightful comments on my work.

I would like to thank Isabelle LECLERCQ, Pascale DENIS, Veronique BANSE, Sabine SEGALA for their sympathetic help for my study and life. I would like to thank all my friends and colleagues for their help and useful suggestion both in my study and life.

Finally, I would like to thank my family members, for their support my decisions and endless care. I would like to especially thank my wife for her unconditional love.

Résumé

Le concept de réservation de voie a été présenté comme une stratégie de gestion du trafic et a de nombreuses applications dans la vie réelle. Des études antérieures dans la littérature se concentrent principalement sur l'impact de la réservation de voies dans une région locale du réseau de transport. Dans cette thèse, les problèmes de réservation de voies sont étudiés dans le but de minimiser l'impact sur le trafic total par la réservation optimale des voies dans un réseau de transport. Nous nous sommes d'abord concentré sur le "lane reservation problem" (LRP) pour le transport automatisé pour les poids lourds avec temps de déplacement statique. Ce travail est généralisé au "capacitated lane reservation problem" (CLRP) pour les grands événements spéciaux. Enfin, le "lane reservation problem with time-dependent travel time" (LRP-TT), et le "lane reservation problem with time-dependent travel speed" (LRP-TS) sont étudiés. Pour chacun des problèmes étudiés, les modèles mathématiques appropriés sont formulés, leurs complexités sont démontrées. Différentes méthodes de résolution sont explorées, y compris exacte cut-and-solve méthode, cut-and-solve combinée à une méthode de coupe, et la méthode de recherche tabou. Les performances des algorithmes proposés sont évaluées sur des instances générées au aléatoirement. Les résultats numériques ont montré que les algorithmes proposés sont plus efficaces pour résoudre les problèmes étudiés que le logiciel commercial CPLEX.

Mots clé: Réservations, Transport de marchandises, Optimisation combinatoire, Problèmes de transport (programmation)

Abstract

The concept of lane reservation has been introduced as a traffic management strategy and has many applications in real life. Previous studies in the literature mainly focus on the impact of lane reservation in a local region of a transportation network. In this thesis, several lane reservation problems are studied with the objective to minimize impact on total traffic by optimally setting reserved lanes in a transportation network. We firstly focus on the lane reservation problem (LRP) for automated truck freight transportation with static link travel time. This primary work is extended to the capacitated lane reservation problem (CLRP) for large-scale special events. Finally, the lane reservation problem with time-dependent travel time (LRP-TT), and the lane reservation problem with time-dependent travel speed (LRP-TS) are studied. For each of the considered problems, appropriate mathematical models are formulated, their complexities are demonstrated. Different resolution methods are explored, including exact cut-and-solve method, cut-and-solve and cutting plane combined method, and tabu search method. The performance of the proposed algorithms is evaluated by randomly generated instances. Numerical results have shown that the proposed algorithms are more effective to solve the considered problems than the commercial package CPLEX.

Keywords: Reservation systems; Freight and freightage; Combinatorial optimization; Transportation problems (Programming)

Contents

Notation

A: set of directed arcs $(i, j), i \neq j, i, j \in N$

K: set of transportation tasks, $k \in K$

N: set of nodes

Q: set of indices of time interval

T_q: boundary of time interval, $q \in Q$

a_{ij}: traffic impact if a lane in link $(i, j) \in A$ is reserved

c_{ij}: residual capacity of a non-reserved lane in link $(i, j) \in A$

d_k: destination node of task $k \in K$

fl_k: flow of task k (number of vehicles/unit of time), $k \in K$

p_k: prescribed travel duration to complete task $k \in K$

s_k: source node of task $k \in K$

τ_{ij}: link travel time on a reserved lane in link $(i, j) \in A$

τ'_{ij}: link travel time on a non-reserved lane in link $(i, j) \in A$

τ''_{ijq}: link travel time on a non-reserved lane in link $(i, j) \in A$ when the departure time at node i is within time interval $[T_q, T_{q+1}), q \in Q$

$\tau^*_{ij}(t)$: link travel time function on a non-reserved lane in link $(i, j) \in A$ when the departure time at node i is t

List of Figures

List of Tables

Chapter 1

Introduction

This thesis investigates a new type of transportation problem: lane reservation problem (LRP). It mainly concerns designing and reconfiguring transportation networks to meet the needs for sustainable development of economy via the lane reservation strategy. The goal of the research is to develop a methodology and tool for decision-making of management of transportation. In this chapter, the background of the thesis is firstly introduced. Then the contributions and outline of this thesis are presented.

1.1 Background

With the integration process of global economic, transportation plays an important role for sustainable development of economy in many aspects, such as human daily moving, logistic transportation and so on. However, rapid urbanization and an increasing number of vehicles cause many problems, such as inefficient transportation, higher and unpredictable travel time, fuel waste, and safety/environment issues. These problems prevent achieving transportation with desirable reliability, efficiency, and safety.

One conventional solution to these problems is from the point view of improving infrastructure, including constructing new traffic networks and adding new lanes by widening the existing roads. Both of the two methods involve in construction, which requires a large amount of funding and long duration. Such solution is not always feasible nowadays. Hence, improving traffic situation and transportation efficacity via appropriate management methods and efficient utilization of the existing infrastructure become more and more important.

Recently a concept called *lane reservation strategy* has been widely applied as a traffic management method in real-life around the word. It is to reserve some lanes

in the existing transportation network for some special use. Only certain types of vehicles are allowed to use the reserved lanes. For example, exclusive bus lanes are introduced in many cities around the world and demonstrated their great success for public transportation. High-occupancy vehicles (HOV) lanes are made during certain day-time hours for vehicles with more than one occupant (sometimes more than two). These HOV lanes are introduced to traffic networks to encourage commuters to car pool such that the number of vehicles moving on the roads can be reduced. Later, high-occupancy toll (HOT) lanes are introduced to allow solo driver to pass HOV lanes by paying a premium price so that HOV lanes can be used more efficiently. Other applications, such as temporarily reserved lanes for evacuation under emergent situations and Olympic lanes for Olympic Games, can also be found in reality.

However, this lane reservation strategy reduces the number of general-purpose lanes, and maybe make the remaining general-purpose lanes congested. Traffic impact such as increase of travel time could be caused on the remaining general-purpose lanes. On the other hand, the impact of reserving a lane with busy traffic is obviously different from that with less traffic. It should be carefully considered selecting roads from the network to reserve their lanes so as to minimize the total traffic impact on the overall network. To the best of our knowledge, there are very few studies of problems on minimizing the impact of reserved lanes with mathematical models and methods. The lane reservation problem is different from some classical transportation problems, such as multi-commodity flow problem, vehicle routing problem with time windows, and facility location problem. Thus, it is meaningful to study such lane reservation problems. These problems aim to minimize the total traffic impact of the reserved lanes by optimally selecting them from the network. This thesis consists of developing new mathematical models and methods for the LRP and several its variants.

1.2 Contributions and outline

This thesis mainly investigates the lane reservation strategy in order to minimize the impact on normal traffic by reserved lanes via optimally selecting them from existing transportation networks. Four lane reservation problems are investigated successively. We firstly focus our attention on a lane reservation problem for automated trucks with static link travel time. Then we extend it to a capacitated lane reservation problem for great events, such as large-scale sport events. Finally, we address two dynamic lane reservation problems with time-dependent link travel time and time-dependent

link travel speed, respectively. The assumptions of the four problems are more and more close to reality. For each addressed problem, we formulate the mathematical model, prove the complexity of the problem, and propose solution approach.

The main contributions of this thesis are presented in detail as follows:

1) Study four lane reservation problems for different applications (automated trucks, great events) under various conditions (residual capacity, static and dynamic link travel time). These four problem are studied successively and their assumptions are more and more close to real-life.

2) For all four problems, appropriate mathematical models are formulated , their complexities are demonstrated and then exact cut-and-solve method, cut-and-solve and cutting plane combined method, and metaheuristic method are developed according to the characteristics of the problems. The performance of the proposed algorithms is evaluated by randomly generated instances. The results are compared with those obtained by a direct use of commercial software CPLEX.

This thesis is organized as follows:

Chapter 2 firstly addresses the literature review on lane reservation strategy. Some related classical transportation problems and their resolution methods are introduced. Then the principle of cut-and-solve method, cutting plane method, and tabu search which will be applied to the resolution of lane reservation problems are explained.

In chapter 3, we study a lane reservation problem, which is intended for future automated freight transportation. The background of the problem is firstly described. Then the mathematical model and complexity of the problem are presented. An optimal algorithm based on the cut-and-solve method is then proposed for the problem. New techniques of generating piercing cuts for the cut-and-solve method are developed according to the characteristics of the problem. Finally, computational results are reported via numerical experiments.

In chapter 4, we study a capacitated lane reservation problem for large-scale special events. This problem is a generalization of the lane reservation problems in [83]. The problem is formulated as an integer linear programming model and a cut-and-solve and cutting plane combined method is developed. The embedded cutting plane method in the proposed algorithm permits to accelerate the convergence of the algorithm. At last, computational results are presented to evaluate the proposed algorithm.

Chapter 5 investigated two lane reservation problems with dynamic factors: time-dependent link travel time and time-dependent link travel speed. The lane reservation problem with time-dependent link travel time is firstly formulated as a mixed integer non-linear program and later transformed into an equivalent linear one. Then an optimal algorithm based on the cut-and-solve method is proposed and computational results are presented. For the lane reservation problem with time-dependent link travel speed, a procedure is firstly proposed to calculate the travel time for each link. The obtained link travel time is a piecewise linear continuous function. Moreover, the "first-in-first-out" property is satisfied in the problem. A tabu search algorithm is developed to obtain near-optimal soltutions. Numerical experiments are conducted to evaluated the performance of the proposed algorithm.

Finally, chapter 6 concludes this thesis and discusses perspectives for future work.

Chapter 2

Literature review

In this chapter, we mainly review the applications and theoretical studies of lane reservation. Then, some related transportation problems and their mathematical formulations are presented. The characteristics of these transportation problems are described to show that the LRPs in this thesis cannot be transformed directly to any of them. Finally, optimization methods for the resolution of transportation problems are presented. Specifically, we describe in detail the methods which will be applied to the resolution of the LRPs in the following chapters.

2.1 Literature review on lane reservation

2.1.1 Applications of lane reservation

Because of the increase of travel demand and number of vehicles, traffic congestion is a common phenomenon in many cities around the world. As stating in chapter 1, the introduction of appropriate traffic management strategies becomes more important as they are economical and flexible, compared with conventional strategy such as augmentation and enhancement of transport infrastructure. In recent years, a lane reservation concept is introduced as a traffic management strategy and has many real-life applications, such as high-occupancy vehicle (HOV) lane, high-occupancy toll (HOT) lane, and exclusive bus lane (XBL), etc.

HOV is a vehicle which has multi-occupant. HOV lanes are reserved for the exclusive use of HOVs (usually referred to carpools, vanpools, buses, and other special vehicles like emergency vehicles). Usually HOV lanes are marked with special traffic signs to distinguish from general-purpose lanes. One example of HOV lane is given in Fig. 2.1. By allowing only HOVs to travel on them, HOV lanes can provide an alternative for travelers to pass congested areas during peak hours. Thus, HOV lanes

(a) Traffic sign for HOV lane.

(b) HOV lane.

Fig. 2.1: Example of high-occupancy vehicle (HOV) lane.

can offer the benefit of travel time saving to encourage and attract people to use carpooling, vanpooling, or buses.

The travel time on the HOV lanes can also be predicted. The final goal of provision of HOV lanes is to reduce traffic congestion, as well as vehicle emissions. Moreover, it was indicated in [78] that many HOV lane projects were implemented to meet one or more of the following common objectives: 1), increase the average number of persons per vehicle; 2), preserve the people-moving capacity of a freeway; 3), improve bus operations; 4), enhance mobility options for travelers.

The first freeway HOV lane was implemented in the Shirley Highway in Northern Virginia, U.S. in 1969 [51] [79]. Since then, many HOV lane projets have been implemented in the mid-to-late 1980s and 1990s. It was reported that there were approximately 2,300 miles of operational HOV lane in 28 metropolitan areas in the United States till the year 2000 [80]. Nowadays, HOV lanes have been widely applied in many cities around the world, e.g., Canada, Australia, UK, Spain, the Netherlands, and Austria in Europe [52] [65]. One successful example of HOV lane is the one which was built in the Lincoln Tunnel in New Jersey in 1970. According to the Federal Highway Administration (FHWA) [26]: "Of the HOV facilities with utilization data provided, the one with the highest number of peak hour persons in the HOV lanes is the Route 495 Lincoln Tunnel Bus Lane in New Jersey, with 23,500 vehicles in the AM peak".

Based on the past years' experiences, HOV lanes are generally effective to reduce traffic congestion by moving more people in fewer vehicles, improving people-moving capacity, and enhancing mobility options for travelers. However, not all the HOV projects achieved these objectives. Some HOV lanes are underutilized as the actual

Fig. 2.2: Example of exclusive bus lane.

traffic volume on them was significantly below the capacity [47]. To more efficiently utilize the HOV lanes, some of them are suggested to convert to HOT lanes which allow solo-occupant vehicles to access to them by paying a premium toll. The amount of toll varies with level of traffic congestion, which is intended to manage the number of vehicles on HOT lanes to maintain free-flow speeds. Besides the common benefits provided by HOV lanes, HOT lanes additionally offer a range of advantages [57]: 1), save travel time for solo-occupant vehicles; 2), generate a source of revenues; 3), improve HOV lanes' utilization; 4), provide a new travel alternative. The first operational HOT lane was implemented in SR-91 Express Lanes in California in 1995. It was said that users of the SR-91 Express Lanes saved an average of 12–13 minutes travel time [59].

Another widely used application of the lane reservation concept is the XBL, which is restricted to buses only during certain hours of the day. In some cities, XBLs are marked with special signs as shown in Fig. 2.2. Unlike HOV or HOT lanes, XBLs are open to public buses only. They provide bus priority over other vehicles to pass congestion regions to enhance bus attractiveness and promote the shift of travel mode from private vehicles to buses, which finally results in a reduction of traffic congestion [69]. In large cities with growing populations and limited space resources, one of the best feasible way to completely solve urban traffic congestion is to implement high quality public transportation systems. Obviously, XBLs can enable an on-time bus service during rush hours. Thus, many bus rapid transit (BRT) systems based on XBLs have been developed in many urban areas during

the last several decades throughout North America, Europe, Latin America, Asia, Australia, and New Zealand. BRT systems play an important role in delivering increasing ridership and saving bus travel time for the success of public transportation [22] [49]. A successful example is Curitiba BRT system implemented in Brazil since 1974. It carries about 188,000 daily passengers in the north-south corridor, 80,000 in the Boqueirao corridor, 52,000 in the east corridor, and 19,000 in the west corridor [50].

The concept of lane reservation has also studied by researchers for intelligent transportation systems. Incorporating the techniques of real-time communication, infrastructure-based cameras, and radio frequency identification readers, an intelligent lane reservation system for highways (IIRSH) is proposed in [20] [62]. The ILRSH allows drivers to reserve a slot on a high-priority lane by paying a premium price so that a time-guaranteed travel can be achieved. Ravi et al. [62] described the general architecture of the ILRSH together with its components. Later Dobre et al. [20] presented design details for the ILRSH components and evaluated their feasibility by simulation experiments. Iftode et al. [41] also proposed a lane reservation based active highways which is similar to the ILRSH.

2.1.2 Studies on lane reservation

As stated in section 2.1.1, the concept of lane reservation is introduced as a traffic management strategy and has various applications in real-life. Meanwhile, lane reservation concept has also been paid much attention by researchers and is extensively investigated in the literature.

In early days, the impact of HOV lanes was evaluated generally via empirical data, surveys, and statistical methods. Martin et al. [55] reported a two-year study evaluation on the impact of HOV lane on I-15 in Salt Lake City. The findings indicated that the HOV lane carried the same number of people as a general-purpose lane with only 44% of vehicles during the P.M. peak period. The average vehicle occupancy on the HOV lane section of I-15 increased by 17%, from from 1.1 to 1.3 people per vehicle. The HOV lane showed a respective 13% and 30% travel time saving during the A.M. and P.M. peak period. The authors concluded that the HOV lane on I-15 is a successful operation. In Oregon, [2] evaluated the I-5 before and after the introduction of an HOV lane. It concluded that HOV users saved 8–10 minutes over the entire length of the scheme. The HOV lane carried 2,600 people per hour, with 65% more people than a general-purpose lane. Evaluations of the impact of other HOV projects can be found in [28]. Although there are some successful instances of HOV lanes

that are proved to be an valuable alternative to reduce congestion, negative results were also reported since one HOV lane facility was closed due to lower utilization in New Jersey in 1998 [28]. Kwon and Varaiya [47] analyzed the California HOV system, using empirical data from the Freeway Performance Measurement System database. It was found that that HOV lanes were under-utilized and suffered from degraded operations. It was also found that HOV lanes offered small travel time saving, though it is not statistically significant. Despite these negative findings, the authors thought that the HOV facilities still can play an important role in California freeway system if it is operated efficiently.

To efficiently utilize the HOV lanes, researchers proposed HOT lanes to combine the concepts of congestion pricing and HOV lanes by offering single occupant vehicles to access to HOV lanes with paying toll. Both empirical and theoretical studies were conducted to evaluate the impact of HOT lanes. Sullivan and Burris [10] [71] conducted a benefit-cost (including travel time savings, fuel costs, emissions, capital costs and operating costs) analysis of California SR-91 and Texas QuickRide HOT lane projects. The overall benefit-cost ratios of the two projects were both between 1.5 and 1.7, with significant savings in travel time. Abdelghany et al. [3] applied a dynamic traffic assignment (DTA) simulation methodology to analyse and evaluate the network performance under various schemes for the design and operation of HOT lanes. Murray et al. [56] evaluated the impact of a hypothetical HOT lanes in a network by incorporating the DTA simulation model with an unordered multinomial logit mode-choice model. The evaluation was under a variety of scenarios, including sensitivities to departure time window, carpooling attractiveness, pricing levels, and access restrictions. It concluded that the impact and effectiveness of HOT lane depend on the complex interaction of the factors mentioned above.

To fully realize the potentials of XBL, many studies on evaluating the impact of XBL have been widely reported. Sarin et al. [64] evaluated the XBL system introduced in Delhi, India. It was reported that the system failed to save bus travel time due to the non-compliance of road users. Nevertheless, many positive results of XBL were reported in the literature. Choi and Choi [14] concluded that the XBL system was successful after conducting their study in South Korea. Bus travel time was significantly reduced, a mode shift from car to bus was estimated 12.2%. The success of XBL was attributed to various reasons, such as public acceptance, government publicity, and systematic cooperation of related authorities, etc. Wei and Chong [82] compared the performance of the first XBL implemented in Kunming, China before and after two years' operation. It was found that the average speed of

bus increased from 9.6km/h to 15.2 km/h. Due to the long duration of conducting before-and-after comparison, simulation method also has been used in recent years to evaluate the impact of XBL. Shalaby [66] used the TRANSYT-7F simulator to evaluate the performance of buses and adjacent traffic following the introduction of XBL in an urban arterial in downtown Toronto, Canada. The results indicated that the XBL was successful in improving average bus performance with an increase in bus ridership and a reduction in adjacent traffic volumes. Unlike the previous articles in which the studies were conducted under homogeneous traffic conditions, Arsan and Vedagiri [6] [7] studied the impact of provision XBLs on heterogeneous traffic flow. They applied the micro simulation model HETEROSIM and observed data to validate the developed simulation model. It was found that the maximum permissible traffic volume (except the buses) to capacity ratio that can ensure a level of service C is about 0.53 for 11.0m wide road and 0.62 for 14.5m wide road. Besides the studies on the operational performance of XBL, impacts of XBL on adjacent traffic operation was also investigated. Karim [43] evaluated the effect of XBL on travel time of other general vehicles using floating car technique, which requires a test vehicle to "float" in the traffic flow and to collect data. It was found that the mean travel time of other general vehicles was significantly increased after executing XBL during morning and evening peak hours. Yang and Wang [85] compared the XBL with the proposed new dynamic bus lane (DBL) in terms of travel time and traffic conflicts changes through the application of micro-simulation approach. Simulation results showed that both DBL and XBL have positive impact on buses and negative impact on adjacent traffic.

To summarize the literature described above, we can find that the studies on lane reservation have the following one or more common points: 1), the studies are mainly focused on the performance impact of single lane reservation project in local region of transportation networks; 2), the study methods is either based on analyzing empirical data, or conducting simulation experiments via traffic simulators. It is clear that the pervious studies provided valuable information for decision-makers when considering new lane reservations as a traffic management strategy. However, seldom studies consider optimally selecting lanes from an entire transportation network to be reserved and no mathematical methods are applied to the studies. It is interesting and necessary to fill these gaps. Thus, we investigate several optimal lane reservation problems in a transportation network in this thesis. The considered problems concern making an optimal selection of lanes from the network to be reserved so that the impact of reserved lanes is minimized. To the best of our knowledge, the only two related studies on optimal lane reservation are [83] [84]. The study on this issue

should be further conducted in depth. It is obvious that the impact is very different with different reserved lanes. The lane reservation strategy should be well studied before it is implemented. In this thesis, we propose mathematical models for the lane reservation problems, then optimal resolution methods are developed for them. Resolution methods are evaluated on randomly generated instances because there is no empirical data and parameters in the literature. The LRPs are studied as transportation problems in this thesis. In the following section, we will introduce some related transportation problems and present their resolution models. It will be seen that our optimal lane reservation problems cannot be transformed directly to any of them. It is of research interest to study the LRPs. Some related optimization methods for the resolution of the LRPs are then described.

2.2 Related Transportation problems

In this section, we will introduce some classical transportation problems, including minimum-cost multicommodity flow problem, facility location problem, and vehicle routing problem with time window, and present their mathematical formulations. The characteristics of each problem are also described to compare with those of the LRPs.

Minimum-cost multicommodity flow problem

Given a network, $G = (V, A)$ composed of a set V of nodes and a set A of directed arcs. Let K denote a set of commodities. Each commodity k corresponds to a source-destination pair (s_k, t_k) and a amount of flow b_k. For each arc $(i, j) \in A$, let u_{ij} denote its capacity and c_{ij}^k denote the cost of transporting per unit flow of commodity k on it. The minimum-cost multicommodity flow problem (MCMF) is to transport a amount of flow b_k of each commodity $k \in K$ from its source s_k to destination d_k respecting to the capacity of each arc such that the total transportation cost is minimized.

Define decision variable x_{ij}^k as the flow of commodity k moving on arc (i, j). Then the MCMF can be formulated as follows [73]:

$$\min \sum_{k \in K} \sum_{(i,j) \in A} c_{ij}^k x_{ij}^k \tag{2.1}$$

$$\text{s.t.} \sum_{i:(s_k,i) \in A} x_{s_k i}^k = b_k, \qquad \forall k \in K, \tag{2.2}$$

$$\sum_{i:(i,d_k) \in A} x_{i d_k}^k = b_k, \qquad \forall k \in K, \tag{2.3}$$

$$\sum_{i:(i,j)\in A} x_{ij}^k - \sum_{i:(j,i)\in A} x_{ji}^k = 0, \qquad \forall k \in K, \forall j \in N \setminus \{s_k, d_k\} \tag{2.4}$$

$$\sum_{k\in K} x_{ij}^k \leq u_{ij}, \qquad \forall (i,j) \in A, \tag{2.5}$$

$$x_{ij}^k \geq 0, \qquad \forall (i,j) \in A, \forall k \in K. \tag{2.6}$$

The objective function (2.1) is to minimize the total transportation cost. Constraint (2.2)– (2.4) together ensure that a amount of b_k commodity k is shipped from source s_k to destination d_k. Constraint (2.5) means that the total commodity flow moving on arc (i, j) is not violated its capacity.

Both the MCMF and the LRPs are intended to transport flow or accomplish tasks for given sour-destination pairs. However, the MCMF is a linear programming with x_{ij}^k of real values. For each commodity, the flow may be transported in several paths from its source to destination. The MCMF can be directly transformed to the minimum-cost flow problem if commodity type $|k| = 1$. In this thesis, the lane reservation decision is a yes-no decision and the LRPs are formulated as either an integer linear programming or a mixed integer linear programming.

Facility location problem

Let D be a set of potential sites of facilities, C denotes a set of customers and $c_{jk}, \forall j \in D, \forall k \in C$ is the cost of transporting per unit flow of commodity from facility j to customer k. Let f_j denote the cost of opening facilities at site j and c_{jk} denote the transportation cost from site j to customer k. The classical facility location problem (FLP) is to open plants from potential sites set D and to service each customer such that the total cost of opening facilities and transporting commodity is minimized. Each customer must be serviced exactly by one opened facility.

Define decision variables as follows:

y_j $\quad y_j = 1$, if a facility is opened at site j; and otherwise $y_j = 0$, $\forall j \in D$.
x_{jk} $\quad x_{jk} = 1$, if customer k is serviced by facility j; and otherwise $x_{jk} = 0$, $\forall j \in D, \forall k \in C$.

The FLP can be formulated as the following zero-one integer linear program [5].

$$\min \sum_{j\in D} f_j y_j + \sum_{j\in D}\sum_{k\in C} c_{jk} x_{jk} \tag{2.7}$$

$$\text{s.t.} \sum_{j\in D} x_{jk} = 1, \qquad \forall k \in C, \tag{2.8}$$

$$x_{jk} \leq y_j, \qquad \forall j \in D, \forall k \in C, \tag{2.9}$$

$$y_j \in \{0,1\}, \qquad \forall j \in D, \tag{2.10}$$

$$x_{jk} \in \{0,1\}, \qquad \forall j \in D, \forall k \in C. \tag{2.11}$$

The objective function (2.7) is to minimize the total cost, including opening facilities cost and transportation cost. Constraint (2.8) ensures that each customer is serviced by exactly one facility. Constraint (2.9) implies that any customer cannot be serviced by a facility which is not open.

Although the FLP and LRPs have two following common points: 1), facility location decision and lane reservation decision are yes-no decisions; 2), in the FLP only opened facility can serviced customers, and in the LRP only reserved lanes can be used by tasks in case of chapter 3. However in the FLP, facilities are located at nodes and each customer is serviced by only one facility. Although in variants of the FLP, a customer can be serviced by several opened facilities, there are not any constraints between these facilities. In the LRPs, lanes are reserved naturally on arcs and a task path can be composed of exclusively or partial reserved lanes that have precedent constraints. The modification of one reserved lane may affect the reservation of other lanes by travel duration constraint.

Vehicle routing problem with time windows (VRPTW)

Given a graph $G = (V, A)$ with node set $V = \{0, n+1\} \cup N$ and arc set A. Node 0 represents the depot and node $n+1$ is a copy of node 0. Each node $i \in N$ corresponds to a customers serviced by a vehicle $k \in K$, which has a capacity q. Each customer i corresponds to a demand d_i, a service time s_i, and a time window $[e_i, l_i]$ which indicates the earliest and latest starting time to service it. Each arc $(i,j) \in A$ corresponds to a travel time t_{ij} and a travel cost c_{ij}. The vehicle routing problem with time windows (VRPTW) is to decide a set of routes for a fleet of vehicles to satisfy each customer's demand such that the total travel cost is minimized. Each customer is serviced only once by exactly one vehicle within the given time window. And each vehicle must start at node 0 and terminate at node $n+1$.

Defined decision variables as follows:

w_{ki}	the time vehicle k starts to service customer i, $\forall k \in K, \forall i \in N$.
x_{kij}	$x_{kij} = 1$, if arc (i,j) is used by vehicle k; and otherwise $x_{kij} = 0$, $\forall k \in K, \forall (i,j) \in A$.

The VRPTW can be formulated as the following mixed integer program [17].

$$\min \sum_{k \in K} \sum_{(i,j) \in A} c_{ij} x_{kij} \tag{2.12}$$

$$\text{s.t.} \sum_{k\in K} \sum_{j:(i,j)\in A} x_{kij} = 1, \qquad \forall i \in N, \tag{2.13}$$

$$\sum_{j:(0,j)} x_{k0j} = 1, \qquad \forall k \in K, \tag{2.14}$$

$$\sum_{i:(i,j)\in A} x_{kij} - \sum_{i:(j,i)\in A} x_{kji} = 0, \qquad \forall k \in K, \forall j \in N, \tag{2.15}$$

$$\sum_{i:(i,n+1)} x_{ki,n+1} = 1, \qquad \forall k \in K, \tag{2.16}$$

$$x_{kij}(w_{ki} + s_i + t_{ij} - w_{kj}) \le 0, \qquad \forall k \in K, \forall (i,j) \in A, \tag{2.17}$$

$$e_i \le w_{ki} \le l_i, \qquad \forall k \in K, \forall i \in V, \tag{2.18}$$

$$\sum_{i\in N} d_i \sum_{j:j\in(i,j)} x_{kij} \le q, \qquad \forall k \in K, \tag{2.19}$$

$$x_{kij} \in \{0,1\}, \qquad \forall k \in K, \forall (i,j) \in A, \tag{2.20}$$

$$w_{ki} \ge 0, \qquad \forall k \in K, \forall i \in N. \tag{2.21}$$

The objective function (2.12) is to minimize the total travel cost. Constraint (2.13) indicates that each customer is visited exactly once, while constraints (2.14)–(2.16) ensure that each vehicle is used exactly once and flow conservation is satisfied at each customer node. Constraints (2.17) and (2.18) are used to model the time window restriction. Constraint (2.19) is the vehicle capacity restriction. Note that (2.17) is not linear, but it can be linearized as follows:

$$w_{ki} + s_i + t_{ij} - w_{kj} \le M_{ij}(1 - x_{kij}), \qquad \forall k \in K, \forall (i,j) \in A, \tag{2.22}$$

where $M_{ij} = \max\{0, l_i + s_i + t_{ij} - e_j\}$ is a constant.

In an optimal solution of the VRPTW, each arc in the network is used at most once, whereas in the LRPs one reserved lane may be used by several tasks. The time window constraint in the VRPTW is imposed to each node, whereas the travel time constraint in the LRPs is imposed to each task path. In the VRPTW, the vehicles start and terminate at the same depot, whereas there is no such restriction in the LRPs.

From the above description, it can been seen that although the LRPs have some common points with the above transportation problems, the LRPs have their own characteristics which are different from them. The LRPs cannot be transformed directly to any of them and the mathematical formulations presented previously cannot be applied to the LRPs. It is necessary to formulate new mathematical models for the LRPs and study them with new resolution method.

2.3 Optimization methods for transportation problems

For solving transportation problems, various optimization methods have been proposed in the literature, such as heuristics, metaheuristics, approximate algorithms, exact methods based on branch-and-bound and branch-and-cut, and hybrid methods, etc.

Heuristics are experience-based approaches which are usually designed to solve specific problems according to their characteristics. They can find "good" solutions fast by trading optimality for speed. Examples of heuristics for solving transportation problems can be found in [31], [63], and [70]. Though heuristics can produce good enough solutions quickly, the optimality of the solutions is not guaranteed. They may be used to generate "good" initial solutions for further improvement by other methods, such as metaheuristics.

Metaheuristics are referred to general concepts or strategies that can be used as guidelines to obtain "high" quality solutions for general optimization problems. Unlike heuristics, special mechanisms are designed to avoid getting stuck in a local optimal solution. Popular metaheuristics include ant colony optimization by Dorigo [21], tabu search by Glover [34] [36], genetic algorithm by Holland [38], and simulated annealing by Kirkpatrick et al. [46]. Metaheuristics are often combined with other methods, such as Lagrangian based methods, to evaluate its performance. Lagrangian based method can provide a lower bound for minimization problems. With the information obtained by it, a feasible solution of the original problem can be constructed thereafter. But Lagrangian based methods are effective only for certain type of problems, such as the FLP [74].

Approximate algorithms can find approximate solutions for optimization problems. Unlike heuristics or metaheuristics, the solutions found by approximate algorithms have guaranteed quality, e.g., the ratio of the approximate solution value and optimal value is bounded by a constant factor. Therefore, approximate algorithms can tell us how close approximation solutions are to the optimal solution. Usually rigor mathematical study of the problems is involved when designing approximate algorithms. Approximate algorithms for solving transportation problems can be found in [9], [15], [48], [68]. However, not all approximation algorithms are suitable for all practical applications and some approximation algorithms have impractical running times even though they are polynomial time, for example $O(n^{2000})$ [1].

Exact methods can solve the problem to optimality and find an optimal solution of the problem, but the computation time will increase exponentially with the size of NP-hard problem. Analysis of properties of the problem and appropriate use of combined or hybrid methods can help solve large scale NP-hard problems. Applications of exact methods can in seen in [8], [29], [72].

To solve the LRPs in this thesis, we consider an exact method, cut-and-solve method, which was proposed by Climer and Zhang (2006) [16]. Actually cut-and-solve method is a particular branch-and-bound method with two nodes at each level of its search path, where the two nodes corresponding to a sparse problem and a remaining problem [86]. The sparse problem is solved to optimality due to its small size. Then only the node corresponding to the remaining problem needs to be branched. Different from many branch-and-bound methods, the cut-and-solve branches the remaining problem on a set of variables rather than one variable at each time [86]. These characteristics result in its following advantages. First, there are no "wrong" subtrees in which the search of cut-and-solve may get lost. As stated previously, at each level of the search only the node corresponding to the remaining problem needs to be branched. Therefore, the search is always a direct path. Second, the memory requirement for cut-and-solve is negligible as only the current best solution and the current remaining problem need to be saved at the search path. For the branch-and-bound or branch-and-cut methods, vast memory is required to store all unexplored nodes in their search tree.

The cut-and-solve method has been successfully applied to solving some difficult classical combinatorial optimization problems. It was reported that the implementation of the cut-and-solve method outperforms state-of-the-art solvers for the asymmetric traveling salesman problem [16]. It also improved the results of some benchmark instances of facility location problems in the literature [86]. Due to these reasons, we consider the cut-and-solve method as our main solution approach. In chapter 3, we consider a LRP for automated freight transportation. We develop a cut-and-solve based algorithm with new techniques of generating piercing cuts for it. In chapter 4, we study a capacitated lane reservation problem (CLRP) in which residual capacity issue is considered. The CLRP is an extension of the LRP. It becomes difficult to solve when the problem's size increases. Thus, a cut-and-solve and cutting plane combined method is developed to accelerate the convergence of the algorithm. Chapter 5 investigates two dynamic LRPs, lane reservation problem with time-dependent travel time (LRP-TT) and lane reservation problem with time-dependent travel speed

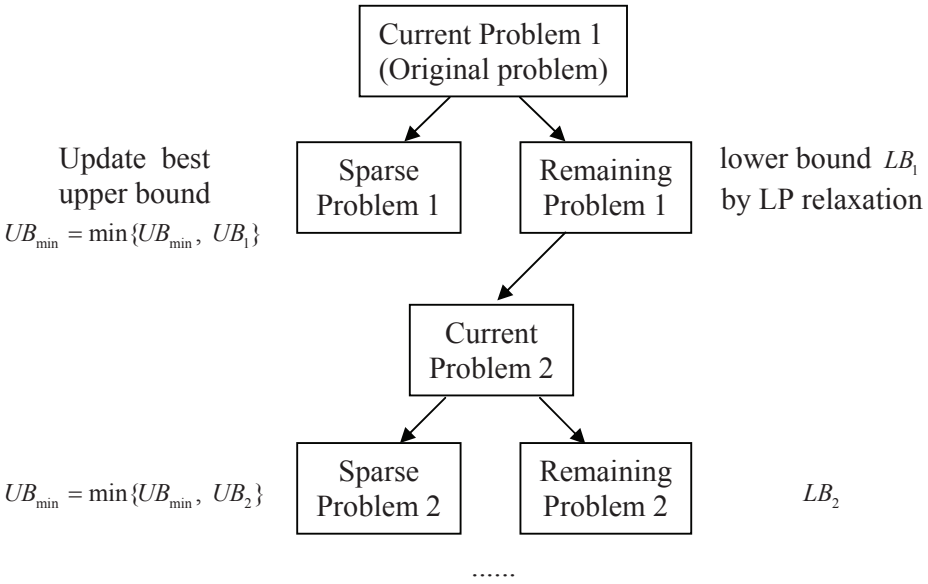


Fig. 2.3: Principle of cut-and-solve method.

(LRP-TS). These two problem are much more difficult to solve than the LRP in chapter 3 and CLRP in chapter 4 since dynamic factor is introduced to the problem. For the LRP-TT, we develop cut-and-solve based algorithm. New piercing cuts are generated according to the problem. The computational time increases rapidly with the size of the problem. Therefore, a tabu search method is applied to the LRP-TS. Some properties are explored to help solve the LRP-TS. In the following section, we will describe in detail the cut-and-solve method, cutting plane method, and tabu search, which will be applied to the resolution of the LRPs.

2.3.1 Methods for lane reservation problems

In this section, the principle of the cut-and-solve method and cutting plane method, which will be applied for the resolution of the LRPs, are described. Without lose of generality, the optimization problems are referred to minimization problems if without special mention.

2.3.1.1 Cut-and-solve method

The cut-and-solve method was firstly proposed by Climer and Zhang for solving the asymmetry traveling salesman problem [16]. It is an iterative exact method for solving combinatorial optimization problems.

The general principle of the cut-and-solve method is summarized in Fig. 2.3. It can be described as follows: 1), at each iteration of the cut-and-solve method, a

piercing cut is generated and it decomposes the *current problem* (it is initialized as the original problem for the first iteration) into a *sparse problem* and a *remaining problem*; 2), for the sparse problem, it is solved to optimality and the optimal value UB obtained. The *current best upper bound* $UB_{\min}$, is then updated if UB is less than $UB_{\min}$; 3), For the remaining problem, its linear relaxation problem is solved by relaxing the integer variables to real ones and an associated *lower bound* LB is obtained; 4), The stopping criterion: $LB \geq UB_{\min}$ is checked. If it is satisfied, $UB_{\min}$ and the solution corresponding to it are respective the optimal value and optimal solution of the original problem. The cut-and-solve method is terminated. Otherwise, the current problem is set as remaining problem and the steps described above are repeated.

Remark 1 The optimal value of the sparse problem is an upper bound of the original problem since it is a subproblem of the original problem. The best upper bound is updated in case of improvement.

Remark 2 It is difficult to solve the remaining problem optimally as its solution space is large, hence its linear relaxation problem is solved. Apparently, the optimal value of the remaining problem is greater than or equal to the lower bound obtained by the linear relaxation.

Remark 3 After the resolution of the sparse problem, its solution space is removed away. Hence, the size of the solution space considered is reduced iteratively.

Remark 4 If $LB \geq UB_{\min}$ is satisfied, then the optimal value of the remaining problem is also greater than or equal to $UB_{\min}$, which means that the remaining problem has no better feasible solutions than the solution corresponding to $UB_{\min}$. Hence $UB_{\min}$ is the optimal value of the original problem.

It can be found that the solution space of the remaining problem is reduced after each iteration. When its linear relaxation problem becomes tight enough, the lower bound is no less than the best upper bound. At this point, the stopping criterion is satisfied and the iteration is terminated. Moreover, Climer and Zhang gave the following two theorems to guaranteed the optimality and termination of the cut-and-solve method.

Theorem 1 *When the cut-and-solve algorithm terminates, the current incumbent solution must be an optimal solution.*

Theorem 2 *If the solution space for the original problem is finite, and both the algorithm for solving the relaxed remaining problem and the algorithm for selecting and solving the sparse problem are guaranteed to terminate, then the cut-and-solve algorithm is guaranteed to terminate.*

The proof of these two theorems can be found in [16]. For more details of the cut-and-solve method, readers are referred to their paper.

To well illustrate how the cut-and-solve method works, a simple example is presented as shown in Fig. 2.4. An integer linear program (ILP) is presented in Fig. 2.4(a). It represents the *current problem*. The polygon represents the feasible solution region of its linear relaxation program (LP). The dots inside or on the edge of the polygon are the feasible solution of the ILP. The optimal value of the ILP is -5, given by $(x_1, x_2) = (2, 1)$.

In Fig. 2.4(b), a *piercing cut* ($5x_1 - 12x_2 \leq -30$) is selected and separated the solution space of the current problem into two subspaces. The smaller space corresponds to a *sparse problem*. It can be obtained by simply adding the piercing cut to the original ILP, as shown in Fig. 2.4(b). The optimal value of the sparse problem (denoted by UB) can be found relatively easily since its solution space is small. It is of value 5, given by $(x_1, x_2) = (1, 3)$. It is clear that UB is an upper bound on the optimal value of original ILP. The *best upper bound* (denoted by $UB_{\min}$) is then updated as 5.

After solving the sparse problem, its solution space is removed away, the solution space of the original ILP is then reduced. The linear program shown in Fig. 2.4(c) represents a *remaining problem*. It can be obtained by adding the constraint $5x_1 - 12x_2 \geq -29$ to the original ILP. The solution space of the remaining problem is large, then its linear relaxation problem is solved by relaxing the integer variables to real ones and an associated *lower bound* of the remaining problem (denoted by LB) is obtained. It is of value -9.5, given by $(x_1, x_2) = (3.5, 1.5)$. Note that LB is less than $UB_{\min}$, then the search continues to the second iteration. Now the current problem is defined as the remaining problem.

In Fig. 2.4(d), a new *piercing cut* ($5x_1 - 4x_2 \geq 4$) is selected and it again separates the solution space of the current problem into two subspaces. The sparse problem is presented in Fig. 2.4(d). Its optimal value is -5, given by $(x_1, x_2) = (2, 1)$. Note that it is less than $UB_{\min}$, then $UB_{\min}$ is updated as -5.

In Fig. 2.4(e), the remaining problem is presented and its linear relaxation problem is solved. The lower bound of the remaining problem is of value -2.855. It is

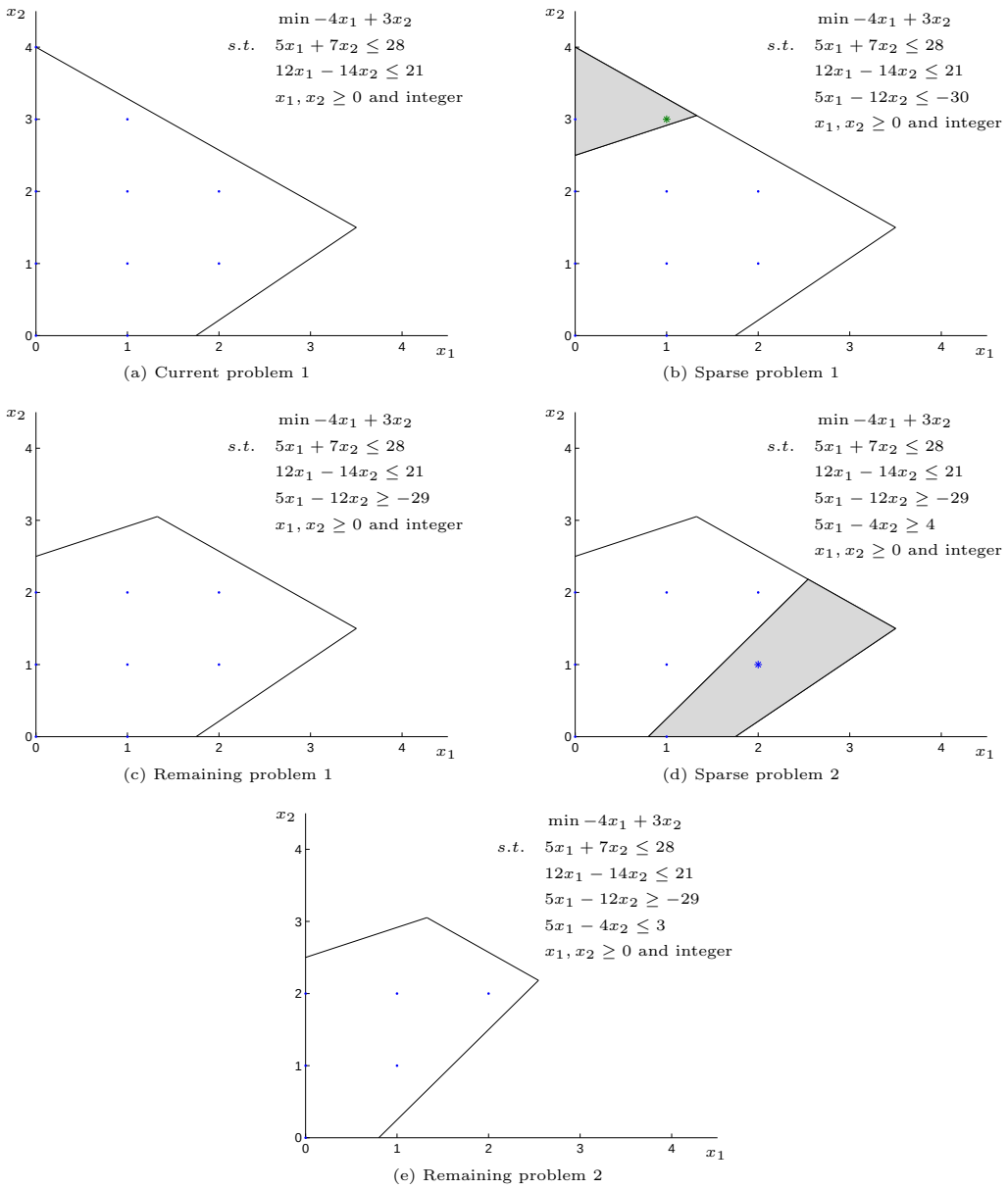


Fig. 2.4: Illustration of cut-and-solve method. (a) Current problem 1 (original integer linear program ILP). (b) Sparse problem 1 with optimal value of 5. Best upper bound is updated as 5. (C) Remaining problem 1 with lower bound of -9.5 obtained by solving its linear relaxation problem. (d) Sparse problem 2 in second iteration with optimal value of -5. Best upper bound is updated as -5. (e) Remaining problem 2 with lower bound of -2.855. It is greater than best upper bound. Then best upper bound of value -5 is the optimal value of the original ILP.

greater than $UB_{\min}$, then the optimal value of the remaining problem is also greater than $UB_{\min}$, which means that there exists no better feasible solutions of the original problem in the solution space of the remaining problem. Hence the search is terminated. Now the $UB_{\min}$ of value -5 is the optimal value of the original ILP and the optimal solution is $(x_1, x_2) = (2, 1)$.

The piercing cuts selected for the above example problem are customized for this particular instance. Actually, the efficiency of the cut-and-solve method is dependent on the selection of appropriate piercing cuts. Climer and Zhang suggested the following desirable properties of piercing cuts:

1) The subspace removed by the piercing cut from the solution space of the current problem should be adequately small, so that the sparse problem can be solved to optimality relatively easily.

2) The optimal solution of the linear relaxation problem of the remaining problem should be removed by the piercing cut so as to prevent this solution from being found in subsequent iterations.

3) The piercing cuts should attempt to capture an optimal solution of the original problem. The algorithm will not be terminated until an optimal solution of the original problem has been found in the sparse problem.

4) In order to guarantee termination, the subspace removed by each piercing cut should contain at least one feasible solution of the original problem.

Climer and Zhang defined a *variable set* composed of the decision variables whose *reduced cost* values are greater than a fixed value *alpha*. The reduced cost values can be obtained from the optimal solution of the linear relaxation problem of the remaining problem. Then the piercing cut is defined as the sum of the decision variables greater than or equal to one. Then a general cut-and-solve procedure is given as follows.

Climer and Zhang successfully applied the cut-and-solve method to solve the asymmetry traveling salesman problem, a classical optimization problem with one decision level. However, the LRPs in this thesis have at least two levels decision: the lane reservation decision and the task paths decision, which belong to different decision levels. In this case, the definition of the piercing cuts should be well selected according to the characteristics of the addressed problems. In this thesis, we make a contribution of successfully applying the cut-and-solve method to optimally solve the LRPs by developing new techniques of generating piercing cuts.

General procedure of cut-and-solve method (ILP)

1: Define current problem as original problem. Relax integrality and solve LP of current problem.
2: Let set $V = \{$ variables with reduced costs $>$ alpha $\}$.
3: Solve sparse problem exactly with constraint (sum of variables in $V = 0$). Update $UB_{\min}$ if optimal value of sparse problem is less than $UB_{\min}$.
4: Solve LP of remaining problem with constraint (sum of variables in $V \geq 1$) and obtain its lower bound LB.
5: If $(LB \geq UB_{\min})$, return $UB_{\min}$.
Otherwise, define the current problem as remaining problem and goto step 2.

2.3.1.2 Cutting plane method

In this subsection, we will introduce the cutting plane method, which will be applied to the resolution of the CLRP to tighten the lower bound of the remaining problem so that the convergence of the cut-and-solve method can be accelerated. First, the general principle of the cutting plane method is described, then the separation of *cover inequalities* (CIs) for the 0–1 knapsack polytope is explained.

Given an integer linear program: $\min\{c\mathbf{x} : \mathbf{x} \in S_{ILP}\}$, where $S_{ILP} = S_{LP} \cap \mathbb{Z}^n$ and $S_{LP} = \{\mathbf{x} \in \mathbb{R}^n : A\mathbf{x} \leq b\}$. The LP corresponding to ILP can be represented as $\min\{c\mathbf{x} : \mathbf{x} \in S_{LP}\}$. As shown in Fig. 2.5, the polygon (solid line) represents the feasible solution region of LP, and the black dots represent the feasible integer solutions of ILP. The inside polygon (dashed red line) represents the *convex hull* of S_{ILP}, denoted by $\text{conv}(S_{ILP})$. It is the smallest convex set containing S_{ILP}. The optimal solution $\mathbf{x}^*$ of LP is an extreme point of S_{ILP}. If $\mathbf{x}^*$ is not an integer solution, it will be outside of $\text{conv}(S_{ILP})$. Then there exists a linear inequality that separates $\mathbf{x}^*$ from the $\text{conv}(S_{ILP})$, and this linear inequality is satisfied by all the feasible integer solutions of ILP and violated by $\mathbf{x}^*$. Such linear inequality is called as a *cutting plane* or simply a *cut* for S_{ILP}. This cut can be added as an additional linear constraint to LP, creating a modified LP. Then the non-integer solution $\mathbf{x}^*$ is no longer feasible to the modified LP. This process is repeated until an optimal solution of the modified LP is an integer solution, which is also an optimal solution of the original ILP. This procedure is called *cutting plane method*. The general procedure of the cutting plane method is given as follows.

The principle of the cutting plane method is to iteratively generate cutting planes to cut off the LP solution until it becomes an integer solution. At this point, the integer solution is an optimal solution of the original ILP. As discussed above, the cutting planes are satisfied by all the feasible integer solutions of the ILP, they are also

General procedure of cutting plane method

1: Relax the integer variables to real ones, solve the LP of ILP.
2: **while** the solution is not integer **do**
3: Generate cutting planes which are violated by the solution.
4: Add cutting planes to LP, obtain the modified LP and solve it.
5: **end while**
6: Return the integer solution.

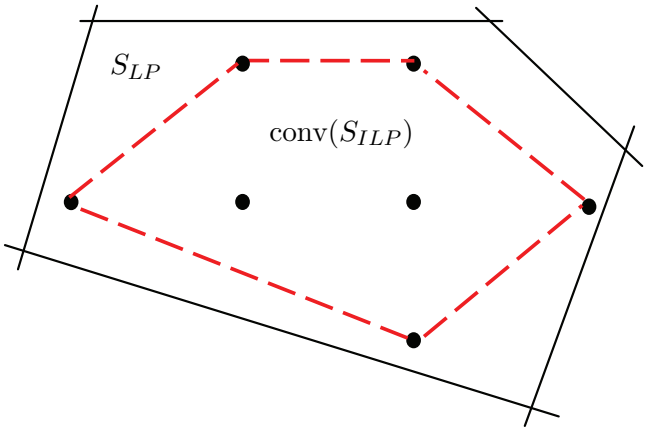


Fig. 2.5: Convex hull.

called *valid inequalities*. A valid inequalities is further called a *facet* if it is necessary to describe the convex hull of the set of feasible integer solutions of the ILP (e.g., the linear inequalities corresponding to the dashed red lines in Fig. 2.5 are facets of conv(S_{ILP})). It is obvious that if all the facets of the convex hull are known, then the optimal solution of the LP with these facets will also be optimal to the original ILP. Unfortunately, it is extremely difficult to find all facets of the convex hull as their total number is exponential. Researches therefore is devoted to finding some facets for specific problems. Given a set of linear inequality constraints and an LP solution $\mathbf{x}^*$, the *separation algorithm* is to find inequalities that are satisfied by the constraints and is violated by $\mathbf{x}^*$, or prove that no such one exists. In practice, it is not reasonable to find all the facets as their number is very large. Sometimes the separation algorithm is also NP-hard. Thus, in our application of cutting plane method to the addressed problem, the iteration will be terminated if no valid inequalities are found. Since the 0–1 Knapsack Problem (KP) is an important substructure of the CLRP, *cover inequalities* (CIs) for the 0–1 Knapsack polytope is explained as follows.

A knapsack constraint can be represented as the following form:

$$\sum_{i \in N} w_i u_i \leq b, \tag{2.23}$$

where N is a set of items, w_i is the weight of item i, b is the capacity of the knapsack. Decision variable $\mathbf{u} = \{u_1, u_2, \ldots, u_{|N|}\} \in \{0,1\}^{|N|}$ takes binary values and indicates that whether item i is selected in the knapsack ($u_i = 1$) or not ($u_i = 0$). Its *knapsack polytope* is the convex hull of feasible solutions with the following form:

$$H_t = \text{conv}\left\{ \mathbf{u} \in \{0,1\}^{|N|} \mid \sum_{i \in N} w_i u_i \leq b \right\}. \tag{2.24}$$

Set C is called as a *cover* if $\sum_{i \in C} w_i > b$, where $C \subseteq N$. For any cover C, the *cover inequality* (CI) for (2.23) is defined as follows:

$$\sum_{i \in C} u_i \leq |C| - 1. \tag{2.25}$$

It is obvious that (2.25) is satisfied by all points in H_t. Given a fractional solution $\mathbf{u}^*$, a CI is called *valid* if it is violated by $\mathbf{u}^*$ but satisfied by all the points in H_t. The *separation algorithm* is to find valid CIs, or prove that none exits. Crowder et al. [18] pointed out that the separation problem for CIs needs to solve the following 0–1 knapsack problem:

$$\theta = \min \sum_{i \in N} (1 - u_i^*) v_i \tag{2.26}$$

$$\text{s.t.} \sum_{i \in N} w_i v_i > b, \tag{2.27}$$

$$v_i \in \{0,\ 1\},\ i \in N. \tag{2.28}$$

Problem (2.26)–(2.28) can be solved by the dynamic program proposed by Kaparis and Letchford [42]. Let v^* denote its optimal solution. Define cover $C = \{i \in N \mid v_i^* = 1\}$. Since v^* is the optimal solution of problem (2.26)–(2.28), then

$$\theta = \sum_{i \in N} (1 - u_i^*) v_i^* = \sum_{i \in C} (1 - u_i^*) v_i^* + \sum_{i \in N \setminus C} (1 - u_i^*) v_i^*.$$

According to the definition of C, if $i \in C$, then $v_i^* = 1$, otherwise $v_i^* = 0$. Thus,

$$\theta = \sum_{i \in C} (1 - u_i^*) v_i^* = |C| - \sum_{i \in C} u_i^*.$$

If $\theta < 1$, i.e., $|C| - \sum_{i \in C} u_i^* < 1$, which is equivalent to

$$\sum_{i \in C} u_i^* > |C| - 1.$$

It means that the CI defined by (2.25) is violated by fractional solution $\mathbf{u}^*$, then it is a valid CI. Therefore, to find CIs for (2.23), we need solve (2.26)–(2.28). If the optimal value is greater than one, then the CI defined by (2.25) is a valid CI. The separation algorithm for CIs is presented in Fig. 2.6 [33].

2.3.1.3 Tabu search

Tabu search (TS) is a local serach-based metaheuristic which was introduced by Golver for solving combinatorial optimization problems [33]. Each solution has an associated *neighborhood*, which is a subset of feasible solutions. The TS guides the search process from the current solution to its best admissible solution in its neighborhood by an operation called *move*, even if this causes the objective function value to deteriorate. This is unlike classical descent methods in which only moves lead to improved objective function values are permitted. To avoid cycling in the search, attributes of recently visited solutions are recorded in a memory structure called *tabu list* to forbid move to them for a number of iterations (called *tabu tenure*). To lead the search to a promising region of the solution space, forbidden moves can be overridden when *aspiration criteria* are satisfied. The TS terminates when stopping criteria are satisfied. The flowchart of a standard tabu search algorithm is given in Fig. 2.7 [58]. For details of the TS, readers are referred to [34].

The feature of the TS is that a flexible memory structure and aspiration criteria are systematically used to guide its search. Moreover, due to the acceptance of deteriorated solutions in the search process, the TS can "jump" from local optimum to other region of the solution space so that the probability to find an global optimal solution is enhanced. The new solution is not randomly generated in the neighborhood of the current solution, it is the one which is better than the best current solution, or the best admissible solution which is not tabued. The best admissible solution is selected from the neighborhood of the current solution according to some pre-given rules. Compared with other metaheuristics, the TS has the following characteristics: 1) It can jump out local optimum. 2) Cycling in the search are avoid by tabu list. 3) The termination of the search of the TS is not dependent on whether finding a local optimal solution. The stopping criteria usually include: the maximum number of iterations; the consecutive number of iterations that best objective function value is not improved.

Separation algorithm for cover inequality

Given a fraction solution $\mathbf{u}^*$, for $j = 1, \ldots, |N|$ and $r = 0, \ldots, b$, define:

$$f(j,r) \coloneqq \min \left\{ \sum_{i=1}^{j} (1 - u_i^*) v_i \mid \sum_{i=1}^{j} w_i v_i = r, v_i \in \{0,\ 1\}, i = 1, \ldots, j \right\}$$

$$g(j) \coloneqq \min \left\{ \sum_{i=1}^{j} (1 - u_i^*) v_i \mid \sum_{i=1}^{j} w_i v_i \geq b + 1, v_i \in \{0,\ 1\}, i = 1, \ldots, j \right\}$$

Set $f(j,r) \coloneqq \infty$ for $j = 1, \ldots, |N|$ and $r = 0, \ldots, b$. Set $f(0,0) \coloneqq 0$.
Set $g(j) \coloneqq \infty$ for $j = 1, \ldots, |N|$.
for $j = 1$ to $|N|$ **do**
  for $r = 0$ to b **do**
    if $f(j-1,r) < f(j,r)$ **then**
      Set $f(j,r) \coloneqq f(j-1,r)$
    end if
  end for
  for $r = 0$ to $b - w_j$ **do**
    if $f(j-1,r) + (1 - u_j^*) < f(j, r + w_j)$ **then**
      Set $f(j, r + w_j) \coloneqq f(j-1,r) + (1 - u_j^*)$
    end if
  end for
  for $r = b - w_j + 1$ to b **do**
    if $f(j-1,r) + (1 - u_j^*) < g(j)$ **then**
      Set $g(j) \coloneqq f(j-1,r) + (1 - u_j^*)$
    end if
  end for
  if $g(j) < 1$ **then**
    Output the violated cover inequality.
  end if
end for

Fig. 2.6: Separation algorithm for cover inequality.

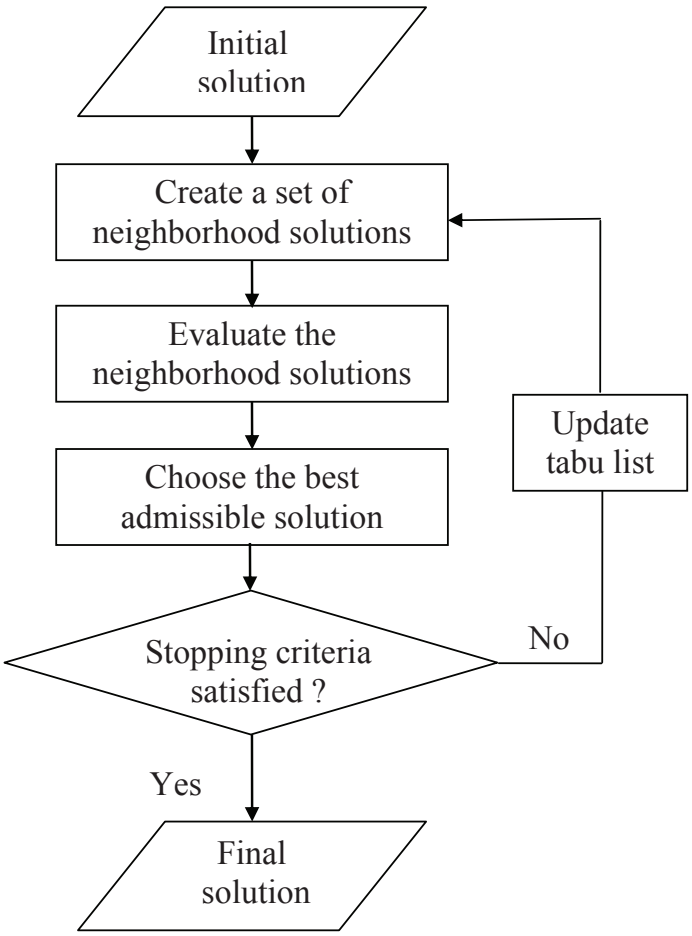


Fig. 2.7: Flowchart of a standard tabu search algorithm.

2.4 Conclusion

In this chapter, we firstly review the literature on lane reservation. The lane reservation concept has been introduced as a traffic management strategy and has many applications in real life. Many studies have been conducted based on analyzing empirical data or performing simulation experiments via traffic simulators. However, most studies focuses on the performance impact of single lane reservation project in local region of transportation networks. It is necessary to study the impact of lane reservation concept in the view of an overall network. This is the motivation of this thesis. Then, we described some related transportation problems and their characteristics to show that the LRPs in this thesis cannot be transformed to any of them. Thus, it is of research interest to study the lane reservation problems. Finally, we introduce some optimization method for solving transportation problems. Specially, we describe the methods which will be applied to our studied problems in detail.

Chapter 3

Lane reservation problem

3.1 Introduction

As stated in the previous chapter, lane reservation strategy has received increasing attention and has been applied in reality in recent years. But there are few studies concerning optimal selection of lanes to be reserved via mathematical methods. In this chapter, we investigate a lane reservation problem (LRP), which is motivated by future automated truck freight transportation. With the automated driving characteristics, the automated trucks can form as a fleet with the first one driven by human and the others automatically follow it [61], [77]. They can offer a range of advantages, such as low labor cost, high transportation efficacity, and low energetic consumption. One of the major concerns for the success of future automated truck freight transportation is the safety issue since the automated trucks themselves must be able to detect all possible dangers and respond to them promptly and correctly, as compared with manually driven trucks [67], [75]. It is thus preferable to provide an appropriate driving environment for them. Reserved truck lane is one of the smart options for reasons as follows. First, it is known that different types of vehicles have different operating characteristic, e.g., acceleration and braking capabilities. If automated trucks are allowed only on the reserved truck lanes, the deviation of the individual truck speed from the average speed is small. Thus the overall traffic flow on the reserved truck lanes is smooth, which may lead to a potential decrease of accident [76]. Second, it is also said that the reserved truck lanes simplify the driving environment for automated trucks, which can make the technical challenge for automated driving tractable while achieving acceptable safety [67]. Third, the reserved truck lanes can meet the high time-efficient transportation required for future freight transportation. If the reserved lanes are used by automated trucks, excluding a number of general road users, they can keep automated trucks from getting stuck in traffic and the travel

duration of the journey can be predictable. Hence, the reserved truck lanes can play an important role in the success of future automated truck freight transportation.

Constructing new truck lanes is not always feasible due to the large cost, long duration, limited spatial resource, and environmental issues. Hence appropriate and efficient use of the existing infrastructure becomes important. One concept is the lane reservation strategy, which is to select some existing general-purpose lanes from a transportation network and to convert them to reserved lanes, e.g., dedicated truck lanes. Since the reserved lanes can be only used by some special types of vehicles, they can provide the vehicles a relative safe and fast travel environment. However, the reserved lanes have negative impact on their adjacent general-purpose lanes that may be congested because the reserved lanes cannot be used by the general road users. Obviously, the impact of reserving a lane with busy traffic is different from that with less traffic. Thus it should be carefully considered which existing roads of the network should be selected to reserve their lanes so as to minimize the impact.

In this chapter, we investigate an LRP for automated trucks. The considered problem is to design an automated trucks transportation network by selecting existing roads from a transportation network and reserving lanes from them for the time-guaranteed transportation. The objective of the LRP is to minimize the total impact of the reserved lanes on the general-purpose lanes over the network. To the best of our knowledge, the only related research to the addressed LRP is the lane reservation problem with time-constrained transportation (LRPTCT), proposed by Wu et al. [83]. Unlike the LRPTCT which is motivated by performing time-guaranteed transportation tasks in a possibly saturated network for large-scale sport events, the LRP is motivated by future automated truck freight transportation. In the LRPTCT, the task paths are partially reserved, i.e., composed of reserved lanes and general-purpose lanes. In the LRP, the task paths is exclusively reserved, i.e., each lane in the path is a reserved lane, for the reason of safety issue. Moreover, the complexity of the LRPTCT is not demonstrated and a heuristic algorithm is proposed to obtain near-optimal solutions for the LRPTCT, whereas the LRP is proved NP-hard and a cut-and-solve based exact algorithm is proposed to obtain an optimal solution.

The remainder of this chapter is organized as follows. Section 3.2 presents the problem's formulation. Then the addressed problem is proved NP-hard. Section 3.3 describes the solution approach which is based on the cut-and-solve method. Section 3.4 reports computational results. Section 3.5 concludes this chapter.

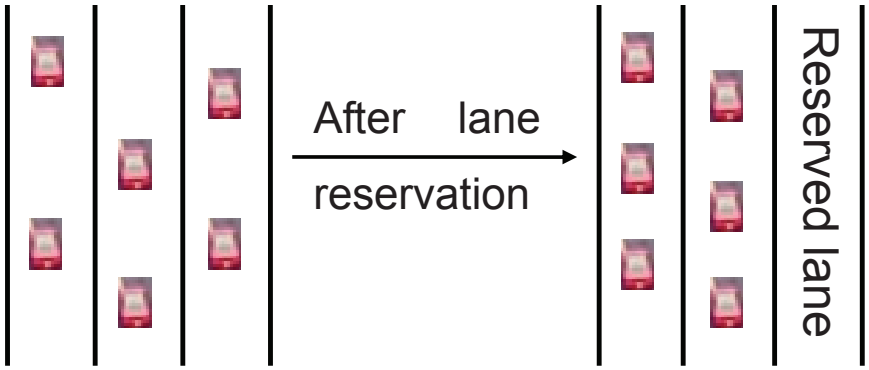


Fig. 3.1: Example of lane reservation.

3.2 Problem formulation

A transportation network can be represented by a directed graph $G = (N, A)$, where N is a set of nodes and A is a set of directed arcs. The nodes and arcs can represent road intersections and road links in a transportation network, respectively. Given a set of tasks and a set of source-destination (SD) pairs, each task corresponds to a SD pair. The considered problem aims at selecting lanes to be reserved and designing an exclusively reserved path for each task, so that each task can be completed within the prescribed travel duration. However, traffic impact such as increase of link travel time on general-purpose lanes may be caused because the reserved lanes cannot be used by the general road users. The objective of the problem is to minimize the total traffic impact of all reserved lanes on the general-purpose lanes. A simple example of lane reservation is given in Fig. 3.1.

In order to well study the problem, some assumptions are made as follows.

1) There are at least two lanes on each link allowing one lane to be reserved. This is because if there is one and only one lane on a link (i, j) and it is reserved, then general road users cannot travel from i to j directly. Such impact is too severe for the network to bear.

2) There is at most one reserved lane on each directed road link. Because the less number of lanes is reserved, the less impact of reserved lanes is. And one reserved lane can be used by multiple tasks.

3) There is one and only one designed path for each task from its source to destination. The addressed problem is aimed to design paths for future fully automated truck freight transportation. It is natural that such a fleet of automated trucks travel together in only one path.

Moreover, the lanes in the same road link are assumed to have identical parameters. To simplify the presentation, we do not distinguish "lane" and "arc" (or "link"), "general-purpose lane" and "non-reserved lane" in the remainder of this thesis.

To formulate the problem, some notations are given as follows.

Sets and parameters

A: set of directed arcs $(i,j), i \neq j, i,j \in N$
K: set of transportation tasks, $k \in K$
N: set of nodes
a_{ij}: traffic impact if a lane in link $(i,j) \in A$ is reserved
d_k: destination node of task $k \in K$
p_k: prescribed travel duration to complete task $k \in K$
s_k: source node of task $k \in K$
τ_{ij}: link travel time on a reserved lane in link $(i,j) \in A$

Decision variables

x_{kij} $x_{kij} = 1$, if a lane in link (i,j) is in the path of task k and this lane is reserved; and otherwise $x_{kij} = 0$, $\forall k \in K, (i,j) \in A$.
z_{ij} $z_{ij} = 1$, if there is a reserved lane in link (i,j); and otherwise $z_{ij} = 0$, $\forall (i,j) \in A$.

The LRP can be formulated as the following integer linear program P_l.

$$P_l: \quad \min \sum_{(i,j)\in A} a_{ij} z_{ij} \tag{3.1}$$

$$\text{s.t.} \sum_{i:(s_k,i)\in A} x_{ks_ki} = 1, \qquad \forall k \in K, \tag{3.2}$$

$$\sum_{i:(i,d_k)\in A} x_{kid_k} = 1, \qquad \forall k \in K, \tag{3.3}$$

$$\sum_{i:(j,i)\in A} x_{kji} = \sum_{i:(i,j)\in A} x_{kij}, \qquad \forall k \in K, \forall j \in N \setminus \{s_k, d_k\}, \tag{3.4}$$

$$\sum_{(i,j)\in A} \tau_{ij} x_{kij} \leq p_k, \qquad \forall k \in K, \tag{3.5}$$

$$x_{kij} \leq z_{ij}, \qquad \forall k \in K, \forall (i,j) \in A, \tag{3.6}$$

$$x_{kij} \in \{0,1\}, \qquad \forall k \in K, \forall (i,j) \in A, \tag{3.7}$$

$$z_{ij} \in \{0,1\}, \qquad \forall (i,j) \in A. \tag{3.8}$$

The objective function (3.1) is to minimize the total traffic impact of all reserved lanes on the general-purpose lanes over the network. Constraint (3.2) (resp. (3.3)) represents that there is one and only one lane departing from the source node s_k (resp. arriving at the destination node d_k) of task k. Constraint (3.4) is the flow

conservation constraint for node j in $N \setminus \{s_k, d_k\}$. It means that if task k arrives at a node j ($j \neq s_k, d_k$) via a reserved lane, it must also depart from j via a reserved lane, or it does not visit j. Thus (3.2)–(3.4) together ensure that there is one and only one path for each task from its source to destination. Constraint (3.5) ensures that the total travel duration of task k should not exceed the prescribed travel duration p_k. Constraint (3.6) means that task k cannot pass the reserved lane on (i, j) if there is no reserved lane on (i, j), i.e., if $z_{ij} = 0$, then $x_{kij} = 0$. Constraints (3.7) and (3.8) are binary constraints on the decision variables.

The complexity of the LRP is proved by the following theorem.

Theorem 3 *The LRP is NP-hard.*

Proof: When all the tasks depart from the same source node and the prescribed travel duration to complete each task is large enough (i.e., the travel duration constraint can be removed), then the LRP corresponds to the particular case, the Steiner tree problem in a directed network, which is known to be NP-hard [44]. Therefore the LRP is NP-hard. □

Note: Given a directed network $G(N, A)$, a subset $N_t \subseteq N$ of terminals and a root node $r \in N \setminus N_t$, the Steiner tree problem in a directed network is to find a set of paths from r to all terminals in N_t such that the total length of the links in these paths is as small as possible [39].

3.3 Solution approach

In this section, an optimal algorithm based on the cut-and-solve method is proposed for the LRP. Briefly speaking, the cut-and-solve method is an iterative search strategy for solving combinatorial optimization problems. Without loss of generality, we suppose the optimization problem is a minimization problem. At the n-th iteration ($n \geq 1$), a *piercing cut* (PC_n) is generated and it separates the solution space of the *current problem* (CP_n) into two subspaces (for the first iteration, the CP_1 is defined as the original problem). The small subspace corresponds to a *sparse problem* (SP_n) and the large one correspond to a *remaining problem* (RP_n). The SP_n can be solved to optimality easily because its solution space is small. Its optimal value (UB_n) is an upper bound on the original problem. Then the solution space of the SP_n is removed. The current *best upper bound* ($UB_{\min}$) is then updated as the minimum of UB_n and $UB_{\min}$. It is difficult to solve the RP_n optimally because its solution space is large. Then the linear relaxation problem (i.e., the integral decision variables are relaxed to real decision variables) of the RP_n is solved and an *associated lower bound* (LB_n) is

obtained. If LB_n is greater than or equal to $UB_{\min}$, it means that the optimal value of the RP_n is also greater than or equal to $UB_{\min}$. Then $UB_{\min}$ is the optimal value for the original problem. Otherwise, the CP_{n+1} is defined as the RP_n and a new iteration repeats. The iteration is repeated until an optimal solution of the original problem is found. For more details of the cut-and-solve method, readers are referred to Climer and Zhang's paper [16].

The PC_n is important to the efficiency of the cut-and-solve method. The reasons are explained as follows. The solution space of the SP_n should be small enough, otherwise it will be difficult to solve it in a reasonable time. On the other hand, if the solution space of the SP_n is too small, there may be no better feasible solutions in it, and $UB_{\min}$ cannot be improved rapidly. Though Climer and Zhang introduced a general procedure for the generation of the PC_n and successfully applied it to the asymmetric traveling salesman problem (ATSP). However, the addressed LRP is different from the ATSP. There are two levels of decision variables in the LRP (lane reservation variables z_{ij} and task path variables x_{kij}), whereas the decision variables belong to the same level in the ATSP. Moreover, the structure of the LRP is more complicated than that of the ATSP from the point of view of mathematical formulation. Hence how to adapt the cut-and-solve method to the LRP is a challenge. It should be carefully considered the characteristic of the problem when applying the cut-and-solve method. In order to make the solution approach more efficient, a pre-processing is performed to reduce the solution space of the original problem and a tightened model is obtained in the following subsection.

3.3.1 Pre-processing

Since the paths of the tasks are exclusively reserved, the shortest travel duration for any given pair of nodes can be computed by some shortest path algorithms (e.g., [19, 27, 30, 35]). Here we choose Floyd's all pairs shortest path algorithm [27]. Let $l(i, j)$ denote the shortest travel duration from i to j. For $\forall k \in K$, set A_k is defined as follows:

$$A_k = \{\ (s_k, j) \mid \tau_{s_k j} + l(j, d_k) > p_k,\ \forall (s_k, j) \in A\ \},$$

where node s_k is the source node of task k and $\tau_{s_k j}$ is the link travel time on a reserved lane in link (s_k, j). The link (s_k, j) in set A_k implies that the sum of the travel time on a reserved lane in (s_k, j) and the shortest travel duration from j to d_k is greater than p_k. Then (s_k, j) will not be selected for the path of task k in a feasible solution, otherwise the travel duration constraint will be violated. Since s_k is the source node

of task k, set A_k implies that the paths which depart from the lane in $(s_k, j) \in A_k$ will not be considered as feasible paths for task k. Similarly, set A'_k is defined as follows:

$$A'_k = \{\ (j, d_k) \mid l(s_k, j) + \tau_{jd_k} > p_k,\ \forall (j, d_k) \in A\ \},$$

where node d_k is the destination node of task k and τ_{jd_k} is the link travel time on a reserved lane in link (j, d_k).

As explained above, the lanes in sets A_k and A'_k will not be selected for the task paths and the corresponding variables can be fixed to zero in any feasible solutions. Then a new model P'_l is defined as follows.

$$\begin{aligned}
P'_l: \quad & \min \sum_{(i,j)\in A} a_{ij} Z_{ij} \\
\text{s.t.} \quad & \text{Constraints } (3.2) - (3.8) \\
& x_{ks_kj} = 0, && \forall k \in K, (s_k, j) \in A_k, && (3.9) \\
& x_{kjd_k} = 0, && \forall k \in K, (j, d_k) \in A'_k. && (3.10)
\end{aligned}$$

Constraints (3.9) and (3.10) means that the values of some decision variables are fixed. Therefore, the solution space of the original problem is reduced. Moreover, the optimality of the original problem is not changed and then the tightened model P'_l is considered in the remainder of this chapter.

3.3.2 Cut-and-solve method

As discussed previously, the PC_n is important to the efficiency of the cut-and-solve method. In the following subsections, some new techniques of generating piercing cut are developed for the considered problem.

3.3.2.1 Definition of piercing cut, sparse problem and remaining problem

Let V_n $(n \geq 1)$ denote a subset, which will be defined later, of all the decision variables. Since all the decision variables are binary variables, the sum of the values of the variables in V_n is greater than or equal to one, or equal to zero. Climer and Zhang define the PC_n as the sum of the variables in V_n is greater than or equal to one [16]. With this PC_n, the solution space of the CP_n is separated into two subspaces. The large subspace corresponds to the RP_n (with the constraint that the sum of the values of the variables in V_n is greater than or equal to one). The other subspace corresponds to the SP_n (with the constraint that the sum of the values of the variables in V_n is equal to zero), then each variable in V_n has value of zero.

Therefore, the solution space of the SP_n is relatively small and the SP_n can be solved relatively easily.

Now the question is how to obtain V_n. Note that the cut-and-solve method will not be terminated until an optimal solution of the SP_n is proved to be the optimal solution of the original problem. Then the basic-variables (non-zero-value variables) in this optimal solution are not in V_n because the decision variables in V_n are fixed to zero in the SP_n. Based on this observation, the definition of V_n should have the following expected property: the decision variables in V_n have small possibility to be basic-variables in the optimal solution of the original problem. Climer and Zhang used a tool called *reduced cost* as a guide for selecting variables for V_n [16]. Given an integer linear program (ILP) for a minimization problem, a linear program (LP) can be obtained by relaxing the integer decision variables to real variables. An optimal solution of the LP defines a set of values referred to as *reduced costs*. Each variable has a reduced cost, which is a lower bound on the increase of the objective value if the value of this variable is increased by one unit. For example, if x has value of zero in an optimal LP solution $\mathbf{X}_1^*$ and its reduced cost is ten. Then if x is increased by one unit, i.e., has value of one, in another LP solution $\mathbf{X}_2^*$, then the objective value of $\mathbf{X}_2^*$ will increase at least ten compared with that of $\mathbf{X}_1^*$. Moreover, decision variables with large reduced cost in the optimal LP solution have small possibility to be basic-variables in the optimal solution of the original ILP. Thus V_n is defined as a set of the decision variables whose reduced costs are greater than a positive given parameter h_n.

However, the LRP is different from the ATSP studied by Climer and Zhang. The V_n should be defined according to the characteristic of the problem. In the LRP, there are two different levels of decision levels: the strategic level (associated with the lane reservation variables z_{ij}) and tactical level (associated with the task path variables x_{kij}). Variables z_{ij} is regarded as more important than x_{kij} for the reasons as follows. First, the task paths are exclusively reserved, which means that only the reserved lanes can be chosen for the task paths. Second, the reservation of one lane or not may result in totally different task paths. This is implied by (3.6). For example, for a given (i, j), if $z_{ij} = 0$ (i.e., lane (i, j) is not reserved), then $x_{kij} = 0$ for all the task k (i.e., any task cannot pass via lane (i, j)). Third, the objective of the LRP is associated with variables z_{ij} only. Based on the above reasons, only z_{ij} is considered for defining V_n in the LRP. Let $\psi(z_{ij})$ denote the reduced cost of z_{ij} in the optimal solution of the linear relaxation problem of the CP_n. Then V_n $(n \geq 1)$ is defined as

follows:

$$V_n = \{\ z_{ij} \mid \psi(z_{ij}) > h_n,\ \forall (i,j) \in A\ \}, \tag{3.11}$$

where h_n is a given positive number. The selection of the value for h_n is decided according to the distribution of the reduced cost of z_{ij} (e.g., if V_n is expected to have nb elements, the value of h_n is simply set as the nb-th largest reduced cost of z_{ij}). After V_n is obtained, the PC_n ($n \geq 1$) is defined as follows:

$$PC_n: \quad \sum_{z_{ij} \in V_n} z_{ij} \geq 1. \tag{3.12}$$

Using the PC_n, the solution space of the CP_n is separated into two subspaces and the SP_n and RP_n can be obtained by adding new constraints to the CP_n. For example, the SP_1 and RP_1 can be defined as follows:

$$\begin{aligned} SP_1: \quad & \min \sum_{(i,j)\in A} a_{ij} Z_{ij} \\ \text{s.t.} \quad & \text{Constraints } (3.2) - (3.10) \\ & \sum_{z_{ij} \in V_1} z_{ij} = 0. \end{aligned} \tag{3.13}$$

$$\begin{aligned} RP_1: \quad & \min \sum_{(i,j)\in A} a_{ij} Z_{ij} \\ \text{s.t.} \quad & \text{Constraints } (3.2) - (3.10) \\ & \sum_{z_{ij} \in V_1} z_{ij} \geq 1. \end{aligned} \tag{3.14}$$

Then the CP_2 is defined as RP_1 for the next iteration. Therefore, for $n \geq 2$, the SP_n and RP_n can be defined as follows:

$$\begin{aligned} SP_n: \quad & \min \sum_{(i,j)\in A} a_{ij} Z_{ij} \\ \text{s.t.} \quad & \text{Constraints } (3.2) - (3.10) \\ & \sum_{z_{ij} \in V_m} z_{ij} \geq 1, \qquad m = 1, 2, \ldots, n-1. \qquad (3.15) \\ & \sum_{z_{ij} \in V_n} z_{ij} = 0. \qquad (3.16) \end{aligned}$$

$$\begin{aligned} RP_n: \quad & \min \sum_{(i,j)\in A} a_{ij} Z_{ij} \\ \text{s.t.} \quad & \text{Constraints } (3.2) - (3.10), \text{ and } (3.15) \end{aligned}$$

$$\sum_{z_{ij} \in V_n} z_{ij} \geq 1. \tag{3.17}$$

It is not difficult to see that (3.16) and (3.17) are the different constraints for the SP_n and RP_n, respectively. Actually, (3.1)–(3.10) and (3.15) together represent the CP_n ($n \geq 2$), i.e., RP_{n-1} (for the first iteration, (3.1)–(3.10) represent the CP_1). The SP_n (resp. RP_n) can be obtained by adding (3.16) (resp. (3.17)) to the CP_n. After then, the SP_n and the linear relaxation problem of the RP_n can be solved by calling Cplex solver.

3.3.2.2 New techniques of generating piercing cut

Via a preliminary test, it becomes more difficult to solve the problem with increase of its size since the problem is NP-hard. To make the cu-and-solve method more efficient, some new techniques of generating piercing cut are developed in this subsection.

Since the decision variables in V_n have fixed values of zero in the SP_n, if V_n contains more decision variables, it may be possible to solve the SP_n more easily. Intuitively, we can obtain a larger sized V_n by simply choosing a smaller parameter h_n. However, this strategy is not always appropriate because sometimes a large proportion of z_{ij} have reduced cost of zero. In our preliminary test, it is found that the variables z_{s_kj} and z_{id_k} with small values in the optimal solution of the linear relaxation problem of the CP_n have small possibility to be basic-variables in the optimal solution of the original problem. Now let (x^*_{kij}, z^*_{ij}) denote the optimal solution of the linear relaxation problem of the CP_n. Then two sets U_n and U'_n are defined as follows:

$$U_n = \{\ z_{s_kj} \mid z^*_{s_kj} < \max_{j:(s_k,j)\in A} z^*_{s_kj},\ \forall k \in K, (s_k, j) \in A\ \},$$
$$U'_n = \{\ z_{id_k} \mid z^*_{id_k} < \max_{i:(i,d_k)\in A} z^*_{id_k},\ \forall k \in K, (i, d_k) \in A\ \}.$$

For example, $z_{s_kj_1}$ and $z_{s_kj_2}$ correspond to the links departing from the source node s_k and respectively have values of 0.7 and 0.3 in the optimal solution of the linear relaxation problem of the CP_n. Then $\max_{j:(s_k,j)\in A} z^*_{s_kj} = 0.7$ and $z_{s_kj_2}$ is in set U_n. Then a new definition of the variable set V_n ($n \geq 1$) is given as follows:

$$V_n = \{\ z_{ij} \mid \psi(z_{ij}) > h_n,\ \forall (i,j) \in A\ \} \cup U_n \cup U'_n. \tag{3.18}$$

compared with (3.15), V_n defined by (3.18) contains more decision variables. Since the decision variables are fixed to zero in the SP_n, it may be possible to solve the SP_n more easily.

In the following part, we make a reduction for the SP_n and RP_n to solve them more easily. First, a theorem is given as follows.

Theorem 4 *Define the SP'_n and RP'_n ($n \geq 2$) as follows:*

$$\begin{aligned} SP'_n: \quad & \min \sum_{(i,j)\in A} a_{ij} Z_{ij} \\ & s.t. \quad \textit{Constraints } (3.2) - (3.10) \textit{ and } (3.16) \\ & \sum_{z_{ij}\in(V_{n-1}\setminus V_n)} z_{ij} \geq 1. \qquad (3.19) \\ RP'_n: \quad & \min \sum_{(i,j)\in A} a_{ij} Z_{ij} \\ & s.t. \quad \textit{Constraints } (3.2) - (3.10) \textit{ and } (3.17). \end{aligned}$$

If $V_1 \supseteq \cdots \supseteq V_{n-1} \supseteq V_n$, $n \geq 2$, then
(a) *SP'_n is equal to SP_n,*
(b) *RP'_n is equal to RP_n.*

Proof: The different constraints in the SP'_n and SP_n are (3.19) and (3.15). We need to prove their equivalence. Because $V_{n-1} \supseteq V_n$, we have $V_{n-1} = (V_{n-1} \setminus V_n) \cup V_n$ and $(V_{n-1} \setminus V_n) \cap V_n = \emptyset$. Then,

$$\sum_{z_{ij}\in(V_{n-1}\setminus V_n)} z_{ij} = \sum_{z_{ij}\in(V_{n-1}\setminus V_n)} z_{ij} + \sum_{z_{ij}\in V_n} z_{ij} = \sum_{z_{ij}\in V_{n-1}} z_{ij} \geq 1.$$

The first "=" is satisfied because we have $\sum_{z_{ij}\in V_n} z_{ij} = 0$ by (3.16). The last "$\geq$" is implied by (3.15). Hence, (3.19) can be deduced from (3.15).

On the other hand, for $\forall m < n$, we have $V_m \supseteq V_{n-1} \supseteq (V_{n-1} \setminus V_n)$. Then, $V_m = (V_m \setminus (V_{n-1} \setminus V_n)) \cup (V_{n-1} \setminus V_n)$ and $(V_m \setminus (V_{n-1} \setminus V_n)) \cap (V_{n-1} \setminus V_n) = \emptyset$. Then,

$$\sum_{z_{ij}\in V_m} z_{ij} = \sum_{z_{ij}\in(V_m\setminus(V_{n-1}\setminus V_n))} z_{ij} + \sum_{z_{ij}\in(V_{n-1}\setminus V_n)} z_{ij} \geq \sum_{z_{ij}\in(V_{n-1}\setminus V_n)} z_{ij} \geq 1.$$

The last "$\geq$" is implied by (3.19). Hence, (3.15) can be deduced from (3.15). Therefore, the equivalence of (3.15) and (3.19) is proved, and SP'_n is equal to SP_n.

The constraints in the RP'_n and RP_n are the same except that there is no (3.15) in the RP'_n. Then we need to prove that (3.15) is redundant in the RP'_n. For $\forall m < n$, we have $V_m \supseteq V_n$. Then, $V_m = (V_m \setminus V_n) \cup V_n$ and $(V_m \setminus V_n) \cap V_n = \emptyset$. Then,

$$\sum_{z_{ij}\in V_m} z_{ij} = \sum_{z_{ij}\in(V_m\setminus V_n)} z_{ij} + \sum_{z_{ij}\in V_n} z_{ij} \geq \sum_{z_{ij}\in V_n} z_{ij} \geq 1$$

The last "$\geq$" is implied by (3.12). Hence, (3.15) can be deduced from (3.17), which implies that (3.15) is redundant in the RP'_n. Hence the RP'_n is equal to RP_n. The proof of the theorem is finished. □

It can be observed that the $n-1$ inequalities in (3.15) are reduced to only one inequality (3.19) in the SP'_n, and are totally removed from the RP'_n. Both the SP'_n

Algorithm LRP

1: Implement the pre-processing for original model P_l, and obtain a new model P_l'.
2: Initialize $n \coloneqq 0$ and best upper bound $UB_{\min} \coloneqq +\infty$. Set current problem $CP_1 \coloneqq P_1'$.
3: Solve the linear relaxation problem of CP_1. If the solution is integral, an global optimal solution is found, stop the algorithm.
4: **repeat**
5: Set $n \coloneqq n+1$. Use the solution and reduced cost information of linear relaxation problem of CP_n to define V_n by (3.20).
6: Define piercing cut PC_n by (3.17) and obtain SP_n' and RP_n'.
7: Solve SP_n' exactly and obtain its optimal value UB_n. Set $UB_{\min} \coloneqq UB_n$ if $UB_n < UB_{\min}$.
8: Solve the linear relaxation problem of RP_n' and obtain its lower bound LB_n. If the solution is integral, set $UB_{\min} \coloneqq LB_n$ if $LB_n < UB_{\min}$, and go to step 10, otherwise set $CP_{n+1} \coloneqq RP_n'$.
9: **until** $(LB_n \geq UB_{\min})$
10: Return $UB_{\min}$ and its corresponding solution as optimal value and optimal solution of the original problem.

Fig. 3.2: Algorithm LRP: algorithm for the LRP.

and RP_n' are reduced with less constraints. It may be possible to solve an equivalently problem more easily without modifying optimality. To satisfy the condition of the theorem, V_n is defined as follows:

$$V_n = (\{\ z_{ij} \mid \psi(z_{ij}) > h_n,\ \forall (i,j) \in A\ \} \cup U_n \cup U_n') \cap V_{n-1}, \tag{3.20}$$

where $V_0 = \{z_{ij} \mid \forall (i,j) \in A\}$. It is not difficult to see that for $n \geq 2$, $V_1 \supseteq \cdots \supseteq V_{n-1} \supseteq V_n$ is satisfied. Then in the algorithm implementation, we define V_n by (3.20), SP_1' by SP_1, and RP_1' by RP_1, respectively.

In summary, the new techniques of generating piercing cut presented in this subsection includes two parts. The first part is to enlarge set V_n by adding some variables which have small possibility to be basic-variable in an optimal solution of the original problem. The second part involves in modifying set V_n to obtained an equivalent formulation of the SP_n and RP_n. These techniques are intended to generate appropriate piercing cuts so as to accelerate the convergence of the algorithm.

The overall Algorithm LRP is presented in Fig. 3.2.

3.4 Computational results

The proposed algorithm for the LRP was coded in C++ and combined with CPLEX 12.1 solver in default mode for the resolution of the sparse problem and remaining problem. The implementation of the algorithm was carried out on a PC with a 3.0 GHz processor and 4.0 GB RAM. Sixty-two problem sets and five problem instances for each set were randomly generated to evaluate the performance of the algorithm.

The instances are generated in the following way. The graph $G(N, A)$ is generated based on the network model proposed by Waxman [81]. The nodes of the graph are randomly and uniformly distributed in a rectangle area $[0, 100] \times [0, 100]$. The existence of an arc (i, j) is dependent on a probability function $\alpha \exp(-d_{ij}/\beta D)$, where $0 < \alpha, \beta \leq 1$, d_{ij} is the Euclidean distance between nodes i and j, and D is the maximum distance between any two nodes. Parameter α is proportional to the number of arcs and a high value of β results in a high ratio of long arcs to short arcs. The default value of the average node degree $\rho = 2|A|/|N|$ is fixed to 7 by choosing appropriate values of α and β. The following parameters are generated based on the way described in Wu et al. [83]. The source-destination pairs are randomly selected from set N. The link travel time on a reserved lane τ_{ij} and on a non-reserved lane τ'_{ij} are respectively defined as $r_{ij}d_{ij}$ and d_{ij}, where r_{ij} is randomly generated from $[0.5, 0.8]$. The prescribed travel duration p_k is generated from $[dis(s_k, d_k), dis'(s_k, d_k)]$, where $dis(s_k, d_k)$ and $dis'(s_k, d_k)$ are the shortest travel duration from s_k to d_k in an exclusively reserved path and in an exclusively non-reserved path, respectively.

It is very difficult to evaluate quantitatively the impact of reserved lanes. It is found that the impact has a very close relation with the increase of link travel time on adjacent non-reserved lanes due to the disallowing use of the reserved lanes by the general road users. Consequently, we evaluate the impact using the increase of link travel time on adjacent lanes, as is the case in [83]. Then the impact is calculated as $c_{ij} = \tau'_{ij}/(m_{ij} - 1)$, where τ'_{ij} is the link travel time on a non-reserved lane on (i, j) and m_{ij} is the number of lanes in link (i, j), respectively. Although the context of the problem in [83] is not identical to that of the LRP, both problems have the same lane reservation concept, i.e., convert some existing general-purpose lanes to reserved lanes for special users only. The impact is caused due to such concept. In addition, the actually statistical result in [60] showed that the link travel time on the general-purpose lanes increases about 53% after one of three lanes is reserved in A1 highway in Paris, which is very close to the result (50%) obtained by the formula proposed in [83]. As stated above, the formula proposed by [83] is applicable to our

problem. Hence we adopt this formula to estimate the impact. In addition, we have also conducted numerical experiment for sensitivity analysis of different setting of the impact.

Because the proposed algorithm is an optimal algorithm, the performance of it was compared with the direct use of CPLEX in terms of the computational time (in CPU seconds) of finding an optimal solution. In addition, the performance of the pre-processing in subsection 3.3.1 and new techniques of generating piercing cut in subsection 3.3.2.2 was also evaluated with randomly generated instances. To simplify the presentation of computational results, let CT_0 and CT_l denote the CPU seconds required by CPLEX and the proposed Algorithm LRP presented in Fig. 3.2, respectively. Let Algorithm LRP$'$ and Algorithm LRP$''$ respectively denote the algorithm same as Algorithm LRP presented in Fig. 3.2 except without the pre-processing step and without the new techniques of generating piercing cut step, and let CT_l' and CT_l'' respectively denote the CPU seconds required by them. The computational results are reported in Tables 3.1–3.4 and Figs. 3.3–3.6.

Table 3.1 presents the computational results for the three algorithms to evaluate the performance of the pre-process and improvement of variable set steps. It can be found that CT_l is less than CT_l' and CT_l'' over sets 1–8. The mean values of CT_l', CT_l'', and CT_l are 173.23, 92.90, and 60.99 seconds, respectively. The minimal, maximal, and mean values of CT_l'/CT_l are 1.92, 7.82, and 2.84, respectively. The minimal, maximal, and mean values of CT_l''/CT_l are 1.02, 1.89, and 1.52, respectively. These results show that the computational time will increase if the pre-process or new techniques of generating piercing cut step is not implemented. This implies that the steps of pre-process or new techniques of generating piercing cut are useful in accelerating the proposed algorithm. Fig. 3.3 presents the results of corresponding computational time and ratios over sets 1–8. It can be seen from Fig. 3.3(a) that the computational time CT_l increases gradually and CT_l' increases rapidly when the size of the problem increases. Fig. 3.3(b) presents the results of CT_l'/CT_l and CT_l''/CT_l. The curve of CT_l'/CT_l is above the curve of CT_l''/CT_l which implies that the pre-processing step is more efficient in accelerating the convergence of the proposed algorithm.

Table 3.2 presents the computational results of problems with different sizes. It can be observed from Table 3.2 that CT_l is less than CT_0 over sets 9–17. The mean values of CT_l and CT_0 are 536.31 and 1381.54 seconds, respectively. The minimal, maximal, and mean values of CT_l/CT_0 are 0.38, 0.52 and 0.42, respectively. When the size of the problem increase, the value of CT_l/CT_0 varies slightly. For example, CT_l/CT_0 ranges between 0.38 to 0.41 over sets 12–17. Fig. 3.4 presents the corresponding

Table 3.1: Comparison for Algorithm LRP, LRP′, and LRP″.

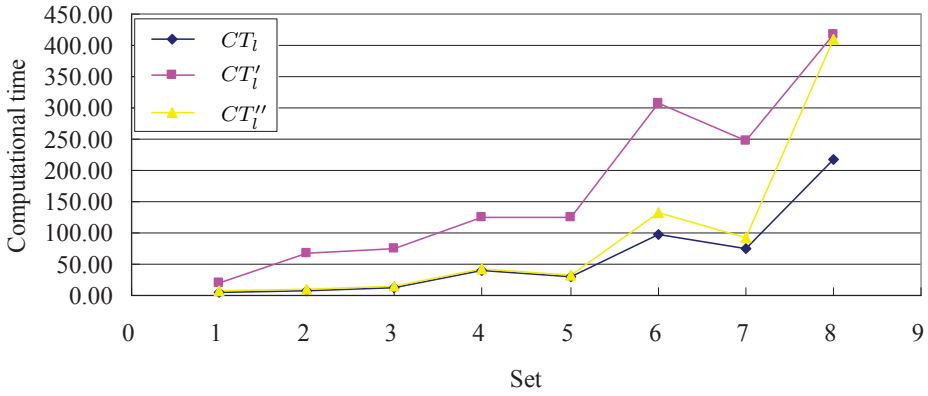
Set	$\|N\|$	$\|K\|$	CT_l'	CT_l''	CT_l	CT_l'/CT_l	CT_l''/CT_l
1	60	25	19.83	7.18	5.62	3.53	1.28
2	60	30	66.63	10.15	8.52	7.82	1.19
3	70	25	75.91	13.85	11.62	6.53	1.19
4	70	30	125.52	43.46	41.04	3.06	1.06
5	80	25	125.71	31.57	30.98	4.06	1.02
6	80	30	307.98	133.25	96.70	3.18	1.38
7	90	25	247.09	92.59	75.71	3.26	1.22
8	90	30	417.18	411.17	217.71	1.92	1.89
Average			173.23	92.90	60.99	2.84	1.52

Table 3.2: Computational results of problems with different sizes.

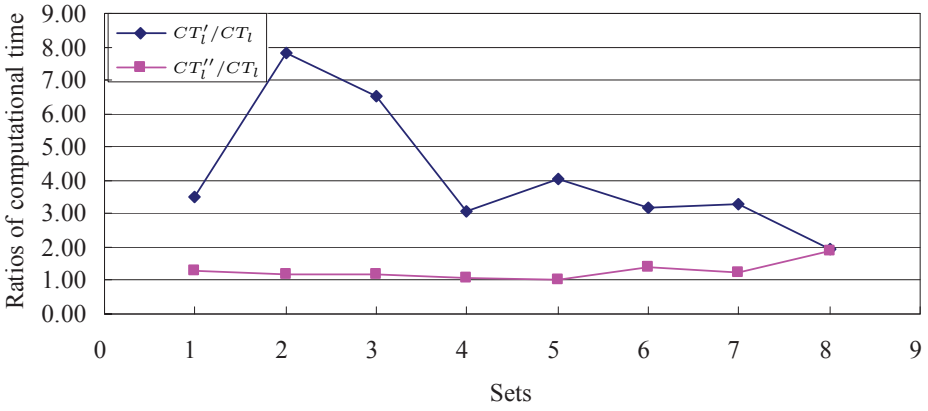
Set	$\|N\|$	$\|K\|$	CT_l	CT_0	CT_l/CT_0
9	110	10	2.37	4.57	0.52
10	110	15	11.45	45.05	0.45
11	120	15	45.80	90.49	0.51
12	120	20	166.41	417.17	0.40
13	130	20	438.96	1075.65	0.41
14	130	25	671.37	1689.28	0.40
15	140	25	765.84	2014.52	0.38
16	140	30	1172.22	3038.63	0.39
17	150	30	1543.43	4058.54	0.38
Average			536.31	1381.54	0.42

computational time over sets 9–17. It can be seen from it that CT_l increases gradually and CT_0 increases much more quickly when the size of the problem increases.

Table 3.3 and Fig. 3.5 present the computational results of problems with different types of impact c_{ij}. The first four types of impact are calculated as $r_e\tau_{ij}'/(m_{ij}-1), e = 1, 2, 3, 4$, where $r_1 = 1.0$ and r_2, r_3, and r_4 are randomly generated from $[0.5, 1.0]$, $[1.0, 1.5]$ and $[0.5, 1.5]$, respectively. Note that the intervals from which r_2, r_3 and r_4 are generated, they are used to generate small impact, large impact, small and large impact simultaneously, respectively. The fifth type of impact is randomly generated from $[0.5, 10]$. It is observed from Table 3.3 that the proposed algorithm is faster than CPLEX over sets 18–47. The mean values of CT_l and CT_0 are 27.73 and 89.37 seconds, respectively. The mean value of CT_l/CT_0 is 0.31. The minimal values of



(a) Computational time of CT_l, CT'_l, and CT''_l.



(b) Ratio of computational time of CT'_l/CT_l and CT''_l/CT_l.

Fig. 3.3: Comparison for Algorithm LRP, LRP′, and LRP″.

CT_l/CT_0 for each type of impact are respective 0.25, 0.26, 0.28, 0.26 and 0.26, with the maximal difference of 0.03. It can be seen from Fig. 3.5(a) that the changing trend of CT_l for each type of impact is almost the same and the differences among five curves vary slightly for the same number of nodes and tasks. In addition, it is found from Fig. 3.5(b) that the five curves of CT_l/CT_0 also vary slightly for the same combination of number of nodes and tasks. The results show that the performance of the proposed algorithm is stable for different setting of the impact.

Table 3.4 presents the computational results of problems with average node degree ρ of values 5, 7, and 12, respectively. The average node degree ρ is defined as $2|A|/|N|$, which denotes the mean number of arcs connected with a node. The larger ρ is, the denser the network is. It can be seen from Table 3.4 that CT_l is less than CT_0 over sets 49–62 and is greater than CT_0 for set 48. The computational time CT_l and CT_0 increase sharply when the average node degree increases. For example, CPLEX takes 68.50 seconds for set 52 with $\rho = 5$, but it cannot find an optimal solution within

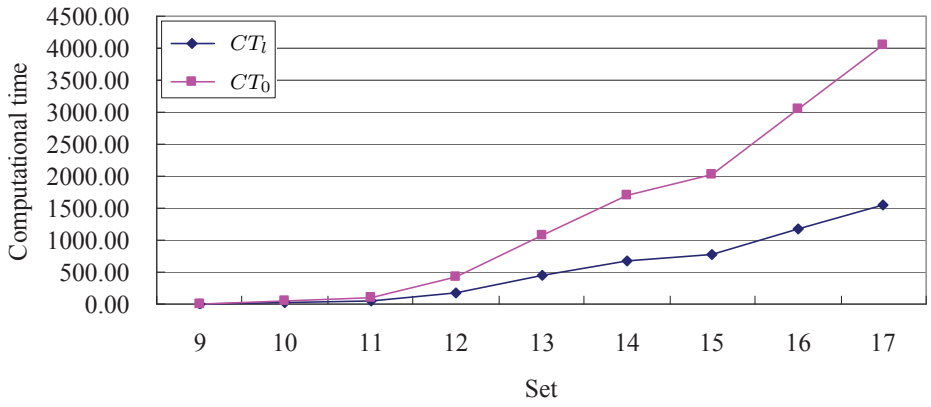
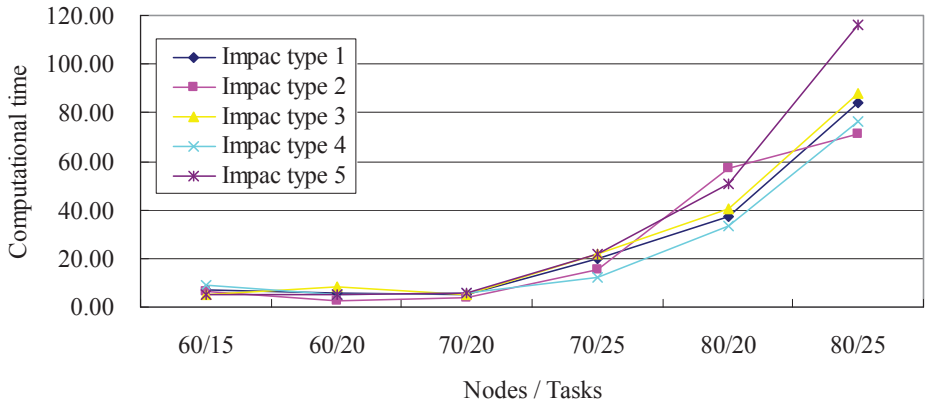
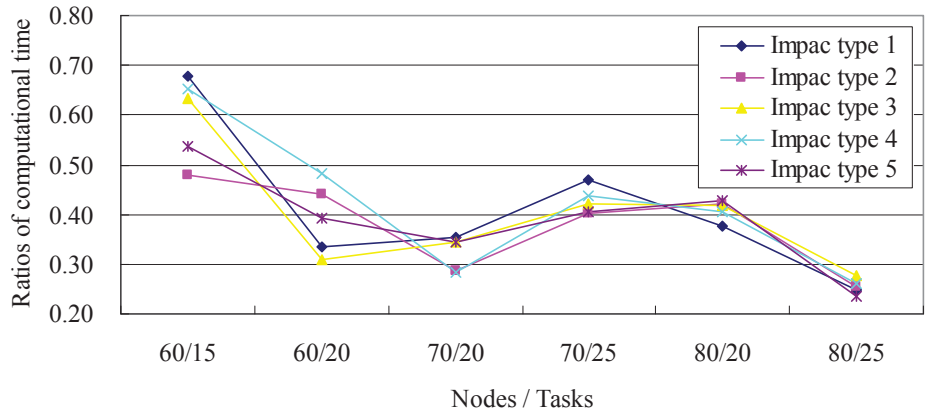


Fig. 3.4: Computational results of problems with different sizes.



(a) Computational time of T_l



(b) Ratio of computational time of CT_l/CT_0

Fig. 3.5: Computational results of problems with different types of impact.

Table 3.3: Computational results of problems with different types of impact.

Set	$\|N\|$	$\|K\|$	Impact	CT_l	CT_0	CT_l/CT_0
18	60	15	Type 1	6.90	10.17	0.68
19	60	20	Type 1	6.01	17.91	0.34
20	70	20	Type 1	5.38	15.22	0.35
21	70	25	Type 1	20.10	42.87	0.47
22	80	20	Type 1	36.97	97.86	0.38
23	80	25	Type 1	83.77	336.43	0.25
24	60	15	Type 2	6.59	13.72	0.48
25	60	20	Type 2	2.83	6.44	0.44
26	70	20	Type 2	4.09	14.20	0.29
27	70	25	Type 2	15.50	38.55	0.40
28	80	20	Type 2	56.84	134.85	0.42
29	80	25	Type 2	71.34	279.18	0.26
30	60	15	Type 3	5.17	8.18	0.63
31	60	20	Type 3	8.30	26.82	0.31
32	70	20	Type 3	5.24	15.23	0.34
33	70	25	Type 3	21.95	52.13	0.42
34	80	20	Type 3	40.47	96.42	0.42
35	80	25	Type 3	87.97	319.18	0.28
36	60	15	Type 4	9.20	14.12	0.65
37	60	20	Type 4	5.18	10.71	0.48
37	70	20	Type 4	5.49	19.48	0.28
39	70	25	Type 4	12.30	28.23	0.44
40	80	20	Type 4	33.13	82.00	0.40
41	80	25	Type 4	76.21	293.04	0.26
42	60	15	Type 5	5.33	9.93	0.65
43	60	20	Type 5	5.23	13.34	0.48
44	70	20	Type 5	5.58	16.24	0.28
45	70	25	Type 5	21.87	54.05	0.44
46	80	20	Type 5	50.88	119.36	0.40
47	80	25	Type 5	115.91	495.18	0.26
Average				27.73	89.37	0.31

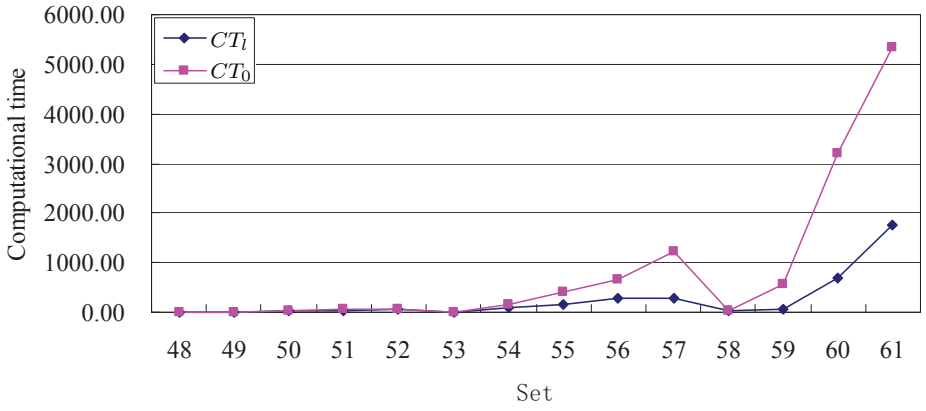
Table 3.4: Computational results of problems with different average node degree ρ.

Set	$\|N\|$	$\|K\|$	ρ	CT_l	CT_0	CT_l/CT_0
48	100	10	5	1.46	1.03	1.42
49	100	15	5	1.07	3.45	0.31
50	100	20	5	31.83	34.81	0.91
51	100	25	5	33.02	54.98	0.60
52	100	30	5	53.58	68.50	0.78
53	100	10	7	3.23	4.85	0.67
54	100	15	7	84.35	158.30	0.53
55	100	20	7	172.53	402.94	0.43
56	100	25	7	279.91	656.40	0.43
57	100	30	7	267.78	1227.46	0.22
58	100	10	12	21.03	38.71	0.54
59	100	15	12	76.00	567.30	0.13
60	100	20	12	701.71	3218.10	0.22
61	100	25	12	1751.32	5345.93	0.33
62	100	30	12	7189.59	>18000.00	<0.40
Average				711.23	>1985.52	<0.36

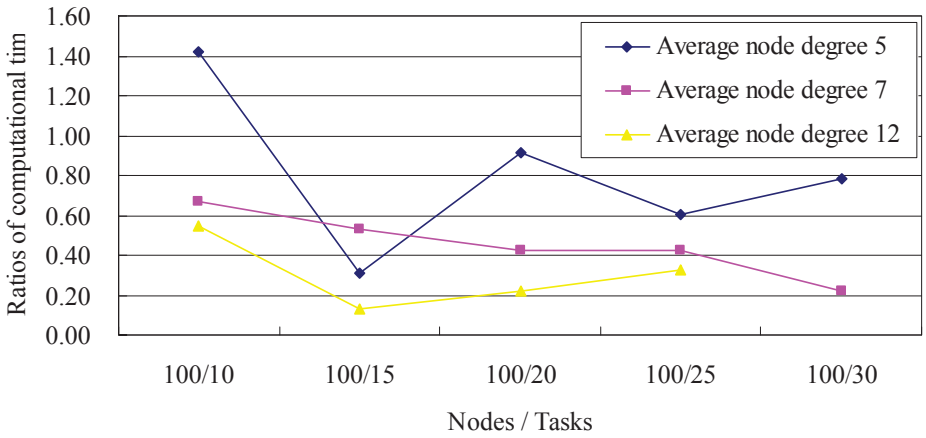
18000.00 seconds for set 62 with $\rho = 12$. However, the proposed algorithm can solve the problem set 62 and only takes 7189.59 seconds. It can be seen from Fig. 3.6(a) that CT_0 increases much sharply over sets 58–61, while CT_l increases gradually. Fig. 3.6(b) presents the results of CT_l/CT_0. Generally, the proposed algorithm is more effective than CPLEX to solve problems with large ρ than to solve problems with small ρ since the curve of CT_l/CT_0 with $\rho = 12$ is below the other two.

3.5 Conclusions

In this chapter, we have investigated a lane reservation problem. The problem is motivated by future automated truck. The problem is formulated as an integer linear program and is demonstrated NP-hard. Then, a cut-and-solve based algorithm is proposed to find an optimal solution. Some new techniques of generating piercing cuts of the cut-and-solve method are developed in this work. Numerical experiments on randomly generated instances with different parameter setting show that the proposed algorithm is more efficient than a commercial optimization solver CPLEX for



(a) Computational time of CT_l and CT_0.



(b) Ratio of computational time of CT_l/CT_0.

Fig. 3.6: Computational results of problems with different average node degree.

the problem. The corresponding work has been published in the following paper.

Y. Fang, F. Chu, S. Mammar, and A. Che. An optimal algorithm for automated truck freight transportation via lane reservation strategy. *Transportation Research Part C: Emerging Technologies*, 26:170–183, 2013.

Chapter 4

Capacitated lane reservation problem

4.1 Introduction

In this chapter, we investigate a capacitated lane reservation problem (CLRP). The problem is motivated by the transportation requirement for large-scale special events. As indicated in [83], these events have the following characteristics: 1), there are many people involving in the events; 2), many activities takes place at different geologically distributed venues. Moreover, the transportation tasks associated with the events usually have a strict travel time (e.g., transportation of perishable food, delivering athletes from their accommodations to stadium, etc.). However, it is difficult to realize such tasks within a given time due to the heavy traffic. Thus, lane reservation strategy is introduced as one solution to the time-guaranteed transportation task for these large-scale events. Nevertheless, lane reservation will affect the normal traffic. It is important to minimize the impact of reserved lanes. A lane reservation problem in time constrained transportation (LRPTCT) was firstly studied with mathematical model and method by Wu et al. [83].

The CLRP is a generation of the work of chapter 3, which was published in [24]. Because the path of a task in the CLRP is not necessary composed of exclusively reserved lanes. The CLRP is also is a generalization of the LRPTCT. The CLRP additionally considers the road capacity issue, which is ignored in the LRPTCT. In reality, traffic situation is closely related to the traffic density. In a transportation network, the capacity of a road represents the maximum flow can pass the road without any congestion. The residual capacity of a road is the difference between the capacity and the average flow of general-purpose vehicles. Specifically, the residual capacity in this chapter means the residual flow of the road that can be used by the

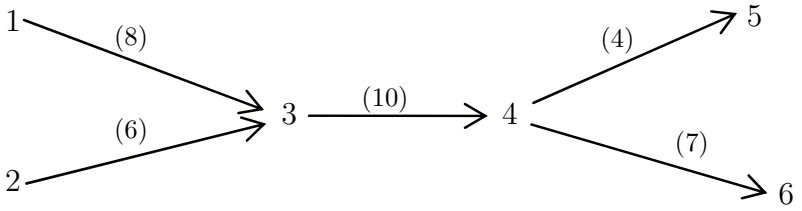


Fig. 4.1: Illustration for residual capacity. The number in the parenthesis is the residual capacity of the road. Task k_1 transports flow of 7 from node 1 to 5, and task k_2 transports flow of 4 from node 2 to 6. One lane in road $(3, 4)$ and in road $(4, 5)$ are reserved, respectively.

special tasks without causing any travel delay or congestion on this road. In this way, the tasks can be accomplished without any delay. An illustration for the residual capacity issue is given as follows. As shown in Fig. 4.1, each road has a residual capacity (the number in the parenthesis). Suppose that task k_1 transports a flow of 7 from node 1 to 5 and task k_2 transports a flow of 4 from node 2 to 6. It is easy to see that a lane in road $(4, 5)$ should be reserved as its residual capacity is less that the flow of task k_1 for it to be accomplished in time. For road $(3, 4)$, any single task k_1 or k_2 can pass via it without any delay since its residual capacity is 10, which is large enough for single task k_1 or k_2 to use. It is not necessary to reserve it in this case. However, the sum of flow transported by tasks k_1 and k_2 is 11, which is larger than the residual capacity of $(3, 4)$. And travel delay will occur on lane $(3, 4)$ if both tasks use it. In this case, we can reserve one lane of road $(3, 4)$ to let both tasks k_1 and k_2 use it. Because only the vehicles of the tasks are allowed to use the reserved lane, then we suppose the tasks can pass the reserved lane without any travel delay and any congestion.

Similar to the previous lane reservation problems, the reserved lanes can provide a relatively fast and safe travel environment for the vehicles on them. Thus the time-guaranteed transportation tasks can be ensured. On the other hand, only special types of vehicles are allowed to use the reserved lanes and other general-purpose vehicles cannot use them. As a result, there will be more vehicles on non-reserved lanes in the same roads than ever before and traffic impact on the non-reserved lanes may be caused. The CLRP is to select some roads from a transportation network and reserve lanes from them for the time-guaranteed transportation tasks considering the road's residual capacity. The objective of the CLRP is to minimize the total impact of the reserved lanes on the non-reserved lanes.

The remainder of this chapter is organized as follows. In section 4.2, the formulation of the CLRP is presented. Then its complexity is demonstrated. Then a cut-and-solve and cutting plane combined method is developed in section 4.3. The embedded cutting plane method in the proposed algorithm permits to accelerate the convergence of the algorithm. In section 4.4, computational results are reported. The last section draws some conclusions.

4.2 Problem formulation

The CLRP is described as follows. A transportation network can be represented by a directed graph $G = (N, A)$, where N is a set of nodes and A is a set of directed arcs. Given a set of tasks and corresponding source-destination (SD) pairs, the CLRP is to select some roads from a transportation network and reserve lanes from them, so that the prescribed travel duration of each ask is guaranteed in its designed path and the road's residual capacity is not violated. The reserved lanes have traffic impact on the non-reserved lanes. The objective of the CLRP is to minimize the total impact of reserved lanes on the non-reserved lanes.

The following assumptions are made so as to well study the addressed problem. 1), there are at least two lanes on each link allowing one lane to be reserved. 2), there is one and only one designed path for each task from its source to destination. 3), there is at most one reserved lane on each directed road link. 4), the reserved lanes can be shared by multiple tasks. Because the total flow of tasks (number of vehicles/unit of time) is relatively small compared to each reserved lane's capacity which can be used by tasks.

To formulate the problem, some notations are given as follows.

Sets and parameters

A: set of directed arcs $(i, j), i \neq j, i, j \in N$
K: set of transportation tasks, $k \in K$
N: set of nodes
a_{ij}: traffic impact if a lane in link $(i, j) \in A$ is reserved
c_{ij}: residual capacity of a non-reserved lane in link $(i, j) \in A$
d_k: destination node of task $k \in K$
fl_k: flow of task k (number of vehicles/unit of time), $k \in K$
p_k: prescribed travel duration to complete task $k \in K$
s_k: source node of task $k \in K$
τ_{ij}: link travel time on a reserved lane in link $(i, j) \in A$
τ'_{ij}: link travel time on a non-reserved lane in link $(i, j) \in A$

Decision variables

x_{kij} $x_{kij} = 1$, if a lane in link (i, j) is in the path of task k and this lane is reserved; and otherwise $x_{kij} = 0$, $\forall k \in K, (i, j) \in A$.

y_{kij} $y_{kij} = 1$, if a lane in link (i, j) is in the path of task k and this lane is not reserved; and otherwise $y_{kij} = 0$, $\forall k \in K, \forall (i, j) \in A$.

z_{ij} $z_{ij} = 1$, if there is a reserved lane in link (i, j); and otherwise $z_{ij} = 0$, $\forall (i, j) \in A$.

Compared with the LRP in chapter 3, τ'_{ij}, fl_k, c_{ij}, and y_{kij} are the new notations introduced for the CLRP. The CLRP can be formulated as the following integer linear program P_c.

$$P_c: \quad \min \sum_{(i,j)\in A} a_{ij} z_{ij} \tag{4.1}$$

$$\text{s.t.} \quad \sum_{i:(s_k,i)\in A} (x_{ks_ki} + y_{ks_ki}) = 1, \qquad \forall k \in K, \tag{4.2}$$

$$\sum_{i:(i,d_k)\in A} (x_{kid_k} + y_{kid_k}) = 1, \qquad \forall k \in K, \tag{4.3}$$

$$\sum_{i:(j,i)\in A} (x_{kji} + y_{kji}) = \sum_{i:(i,j)\in A} (x_{kij} + y_{kij}), \quad \forall k \in K, \forall j \in N \setminus \{s_k, d_k\}, \tag{4.4}$$

$$\sum_{(i,j)\in A} \left(\tau_{ij} x_{kij} + \tau'_{ij} y_{kij}\right) \le p_k, \qquad \forall k \in K, \tag{4.5}$$

$$x_{kij} \le z_{ij}, \qquad \forall k \in K, \forall (i, j) \in A, \tag{4.6}$$

$$y_{kij} + z_{ij} \le 1, \qquad \forall k \in K, \forall (i, j) \in A, \tag{4.7}$$

$$\sum_{k\in K} fl_k y_{kij} \le c_{ij}(1 - z_{ij}), \qquad \forall (i, j) \in A, \tag{4.8}$$

$$x_{kij}, y_{kij} \in \{0,\ 1\}, \qquad \forall k \in K, \forall (i, j) \in A, \tag{4.9}$$

$$z_{ij} \in \{0,\ 1\}, \qquad \forall (i, j) \in A. \tag{4.10}$$

The objective function (4.1) is to minimize the total traffic impact of all reserved lanes on the general-purpose lanes over the network. Constraints (4.2)–(4.4) ensure that there is one and only one path for each task from its source to destination. Constraint (4.5) ensures that the total travel duration of task k should not exceed the prescribed travel duration p_k. Constraint (4.6) means that task k cannot pass the reserved lane on (i, j) if there is no reserved lane on (i, j), i.e., if $z_{ij} = 0$, then $x_{kij} = 0$. Constraint (4.7) is tighter than constraint $x_{kij} + y_{kij} \le 1$. It means that if task k passes (i, j) (i.e., $x_{kij} + y_{kij} = 1$), either there is a reserved lane (i.e., $x_{kij} = 1$), or there is no one in (i, j) (i.e., $y_{kij} = 1$), otherwise task k does not pass (i, j) (i.e., $x_{kij} + y_{kij} = 0$). Constraint (4.8) is the residual capacity constraint, i.e., the total

flow of the tasks moving on (i, j) is no greater than its residual capacity if there is no reserved lane on it. Constraints (4.9) and (4.10) are binary constraints on the decision variables.

Theorem 5 *The CLRP is NP-hard.*

Proof: If each road's residual capacity is very small, then each lane in the task paths must be reserved. Thus the CLRP corresponds to the LRP described in chapter 3, which is proved NP-hard in section 3.2. Therefore the CLRP is NP-hard. □

Note: If the road's residual capacity is large enough, the reduced CLRP corresponds to the LRPTCT. In this sense, the CLRP is a generalization of the LRPTCT in [83].

4.3 Solution approach

The CLRP is an extension to the LRP on considering the residual capacity issue. The cut-and-solve method proposed in chapter 3 can also be adapted to the CLRP. Our previous work shown that the cut-and-solve method can find an optimal solution of the CLRP more efficiently than CLPEX solver [25]. However, it becomes difficult to solve the problem in a reasonable time when the size of the problem increases. To accelerate the convergence of the proposed algorithm, a cut-and-solve and cutting plane combined method is developed in this section. The cutting plane method is embedded in the algorithm to obtain a tight lower bound on the remaining problem. Before first, a pre-processing is performed to reduce the solution space of the original problem.

4.3.1 Pre-processing

For $\forall k \in K$, let (s_k, j) denote a link connected with the source node s_k of task k, and $l(j, d_k)$ denote the shortest travel duration from j to d_k in an exclusively reserved path, where d_k is the destination node of task k. Then $l(j, d_k)$ can be computed by Floyd's shortest path algorithm [27]. Define set A_k as follows:

$$A_k = \{\ (s_k, j) \mid \tau_{s_k j} + l(j, d_k) > p_k,\ \forall (s_k, j) \in A\ \},$$

where $\tau_{s_k j}$ is the travel time on a reserved lane in (s_k, j). The link (s_k, j) in set A_k implies that the sum of the travel time on a reserved lane in (s_k, j) and the shortest travel duration from j to d_k is greater than p_k. Then task k cannot pass (s_k, j)

respecting p_k, otherwise the travel duration constraint will be violated. Similarly, A'_k is defined as follows:

$$A'_k = \{\ (j, d_k) \mid l(s_k, j) + \tau_{jd_k} > p_k,\ \forall (j, d_k) \in A\ \},$$

where node d_k is the destination node of task k, τ_{jd_k} is the link travel time on a reserved lane in (j, d_k), and $l(s_k, j)$ is the shortest travel duration from s_k to j in an exclusively reserved path.

As explained above, task k cannot pass the links in sets A_k and A'_k and the corresponding variables can be fixed to zero in any feasible solutions. Then a new model P'_c is defined as follows:

$$\begin{aligned}
P'_c: \quad & \min \sum_{(i,j)\in A} a_{ij} z_{ij} \\
\text{s.t.} \quad & \text{Constraints } (4.2) - (4.10) \\
& x_{ks_kj} + y_{ks_kj} = 0, && \forall k \in K, (s_k, j) \in A_k, && (4.11) \\
& x_{kjd_k} + y_{kjd_k} = 0, && \forall k \in K, (j, d_k) \in A'_k. && (4.12)
\end{aligned}$$

The solution space of the original problem is reduced as the values of some decision variables are fixed to zero. Moreover, the optimality of the original problem is not missed as no feasible solutions are excluded. After the pre-processing step, a tightened model P'_c for the original problem is obtained and is considered in the remainder of the paper.

4.3.2 Cut-and-solve method

As mentioned previously, the cut-and-solve method developed in chapter 3 can be adapted to the CLRP. The piercing cuts are defined in the same way as (3.12) in section 3.3.2.1 of chapter 3. Since the mathematical model of the CLRP is different from that of the LRP, we only give the definitions of sparse problem SP_n and remaining problem RP_n for simplifying the presentation. Details of the cut-and-solve method can be found in section 3.3.2 of chapter 3.

Then for $n \geq 1$, the SP_n and RP_n are defined as follows:

$$\begin{aligned}
SP_n: \quad & \min \sum_{(i,j)\in A} a_{ij} Z_{ij} \\
\text{s.t.} \quad & \text{Constraints } (4.2) - (4.12) \\
& \sum_{z_{ij}\in V_m} z_{ij} \geq 1, && m = 1, 2, \ldots, n-1. && (4.13)
\end{aligned}$$

$$\sum_{z_{ij} \in V_n} z_{ij} = 0. \tag{4.14}$$

$$RP_n: \quad \min \sum_{(i,j) \in A} a_{ij} Z_{ij}$$

$$\text{s.t.} \quad \text{Constraints } (4.2) - (4.12), \text{ and } (4.13)$$

$$\sum_{z_{ij} \in V_n} z_{ij} \geq 1. \tag{4.15}$$

Note that (4.15) is the piercing cut. The current problem CP_1 is defined as P'_c for the first iteration, and CP_n is defined as RP_{n-1} for $n \leq 2$. The SP_n can be optimally solved by calling a CPLEX MIP solver. Then the cutting plane method is applied to the RP_n to obtain a tight lower bound.

4.3.3 Cutting plane method to tighten remaining problem

In this subsection, a cutting plane method is applied to tighten the lower bound on the remaining problem in order to accelerate the convergence of the cut-and-solve method. Via a preliminary test, it is found that a tight lower bound results a fast convergence of the cut-and-solve method. This is because a tight lower bound can provide a helpful reference to evaluate the optimal objective value of the original problem. The iteration will be terminated until the lower bound on the remaining problem is greater than or equal to the current best upper bound. With tighter the lower bound, fewer iterations of the cut-and-solve method are required to obtain an optimal solution of the original problem. On the other hand, a tight lower bound can provide useful information to generate an appropriate piercing cut.

The principle of the cutting plane method is to iteratively generate cutting planes to add successively corresponding constraints to the current relaxation problem until the fractional solution becomes an integer solution. Moreover, it can be seen that the lower bound associated with the fractional solution is improved iteratively. Based on this principle, the cutting plane method can be applied to obtain a tight lower bound on the remaining problem for the CLRP. The finding of cutting planes is called *separation algorithm*. Specifically, given a fractional solution $\mathbf{x}^*$ and s set of constraints, the separation algorithm is to find inequalities that are violated by $\mathbf{x}^*$ and valid by feasible solutions of the original problem, or tell that none such inequalities exits. For details of the cutting plane method, readers are referred to section 2.3.1 of chapter 2. In the following, the separation algorithm for cover equalities for the studied problem is presented.

A knapsack constraint can be represented as the following form:

$$\sum_{i \in N} w_i u_i \leq b, \tag{4.16}$$

where N is a set of items, w_i is the weight of item i, b is the capacity of the knapsack. Binary variable u_i indicates that whether item i is selected in the knapsack ($u_i = 1$) or not ($u_i = 0$). Set $C \subset N$ is call *cover* for (4.16) if $\sum_{i \in C} w_i > b$. Then

$$\sum_{i \in C} u_i \leq |C| - 1 \tag{4.17}$$

is called a *cover inequality* (CI). A CI is called *valid* if it is violated by a given fractional solution but satisfied by the feasible solution of the original problem. The *separation algorithm* is to find valid CIs.

Let w_i represents τ_{ij} or τ'_{ij}, b represents p_k, and binary variable u_i represents x_{kij} or y_{kij} for our problem, the travel duration constraint (4.5): $\sum_{(i,j) \in A}(\tau_{ij} x_{kij} + \tau'_{ij} y_{kij}) \leq p_k, \forall k \in K$, can be directly written as a standard knapsack constraint form. For $\forall k \in K$, the CI for (4.5) has the following form:

$$\sum_{(i,j) \in A_x} x_{kij} + \sum_{(i,j) \in A_y} y_{kij} \leq |A_x| + |A_y| - 1, \qquad \forall k \in K, \tag{4.18}$$

where A_x and A_y are subsets of A. Then the separation algorithm proposed by Kaparis and Letchford [42] in Fig. 2.6 can be applied to find CI for (4.5).

Constrain (4.8): $\sum_{k \in K} f l_k y_{kij} \leq c_{ij}(1 - z_{ij}), \forall (i,j) \in A$, can be written as a knapsack constraint form if we move $c_{ij} z_{ij}$ to the left hand, i.e., $\sum_{k \in K} f l_k y_{kij} + c_{ij} z_{ij} \leq c_{ij}$. Let C' be a cover for (4.8), and $|C'| = m$. Suppose items $1, \ldots, m-2, m-1$ correspond to $y_{k_1 ij}, \ldots, y_{k_{m-2} ij}, y_{k_{m-1} ij}$, and item m corresponds to z_{ij}. Then the corresponding CI is $y_{k_1 ij} + \ldots + y_{k_{m-2} ij} + y_{k_{m-1} ij} + z_{ij} \leq m - 1$. Given a fractional solution (y^*, z^*), we have $y^*_{k_1 ij} + \ldots + y^*_{k_{m-2} ij} + y^*_{k_{m-1} ij} + z^*_{ij} \leq m - 2 + y^*_{k_{m-1} ij} + z^*_{ij}$. Because (4.7) implies that $y_{kij} + z_{ij} \leq 1$ is true (even for a fractional solution), then $y^*_{k-m-1 ij} + z^*_{ij} \leq 1$. Therefore, we have $y^*_{k_1 ij} + \ldots + y^*_{k_{m-2} ij} + y^*_{k_{m-1} ij} + z^*_{ij} \leq m - 1$. It means that the above CI is satisfied by the given fractional solution (y^*, z^*). It is not a valid CI. Then we define a new CI and propose separation algorithm to find valid CI that is satisfied by the feasible solutions of the original problem.

For $\forall (i,j) \in A$, the CI for (4.8) is defined as follows:

$$\sum_{k \in C} y_{kij} \leq (|C| - 1)(1 - z_{ij}), \qquad (i,j) \in A, \tag{4.19}$$

where cover C of subset of K. Given a fractional solution (y^*, z^*), the separation algorithm to find CI for (4.8) needs to solve the following problem [18] for $\forall(i,j) \in A$:

$$\theta = \min \sum_{k \in K} (1 - z^*_{ij} - y^*_{kij}) v_k \tag{4.20}$$

$$\text{s.t.} \sum_{k \in K} f l_k v_k \geq c_{ij} + 1, \tag{4.21}$$

$$v_k \in \{0,\ 1\},\ k \in K. \tag{4.22}$$

The problem (4.20)–(4.22) can be solved by the the following developed algorithm based on the dynamic program proposed by Kaparis and Letchford [42]. Let v^* denote its optimal solution. Define cover $C = \{k \in K \mid v^*_k = 1\}$. Since v^* is the optimal solution of problem (4.20)–(4.22), we have

$$\theta = \sum_{k \in K} (1 - z^*_{ij} - y^*_{kij}) v^*_k = \sum_{k \in C} (1 - z^*_{ij} - y^*_{kij}) v^*_k + \sum_{k \in K \setminus C} (1 - z^*_{ij} - y^*_{kij}) v^*_k.$$

According to the definition of C, if $k \in C$, then $v^*_k = 1$, otherwise $v^*_k = 0$. Thus,

$$\theta = \sum_{k \in C} (1 - z^*_{ij} - y^*_{kij}) = |C|(1 - z^*_{ij}) - \sum_{k \in C} y^*_{kij}.$$

If $\theta < 1 - z^*_{ij}$, i.e., $|C|(1 - z^*_{ij}) - \sum_{k \in C} y^*_{kij} < 1 - z^*_{ij}$, which is equivalent to

$$\sum_{k \in C} y^*_{kij} > (|C| - 1)(1 - z^*_{ij}).$$

It means that the CI defined by (4.19) is violated by fractional solution (y^*, z^*), then it is a valid CI.

To summarize the above description, the steps of finding valid CI for (4.8) is as follows: given a fractional solution (y^*, z^*), solve the problem (4.20)–(4.22). If the optimal value θ is less than $1 - z^*_{ij}$, then the CI defined by (4.19) is a valid CI, otherwise, it is not.

Now we focus on solving the problem (4.20)–(4.22). We adapt the dynamic program proposed by Kaparis and Letchford [42] to our problem and propose a new separation algorithm as presented in Fig. 4.2. The separation algorithm is similar to that of Kaparis and Letchford except for the definitions of $f(h, r)$ and $g(h)$, and the final output criterion. The principle of the dynamic program is to compute all of the $f(h, r)$ and $g(h)$ values, and then use the relation between them to find an optimal solution to (4.20)–(4.22). $f(h, r)$ is the recursive objective function value of a dynamic program with the total flow of selected tasks from $\{1, \ldots, h\}$ is exactly

New separation algorithm for cover inequality

Given a fractional (y^*, z^*). For $h = 1, \ldots, |K|$ and $r = 0, \ldots, c_{ij}$, define:

$$f(h,r) \coloneqq \min \left\{ \sum_{k=1}^{h} (1 - z^*_{ij} - y^*_{kij}) v_k \;\; | \;\; \sum_{k=1}^{h} fl_k v_k = r, (v_1, \ldots, v_h) \in \{0,1\}^h \right\}$$

$$g(h) \coloneqq \min \left\{ \sum_{k=1}^{h} (1 - z^*_{ij} - y^*_{kij}) v_k \;\; | \;\; \sum_{k=1}^{h} fl_k v_k \geq c_{ij} + 1, (v_1, \ldots, v_h) \in \{0,1\}^h \right\}$$

Set $f(h,r) \coloneqq \infty$ for $h = 1, \ldots, |K|$ and $r = 0, \ldots, c_{ij}$. Set $f(0,0) \coloneqq 0$.
Set $g(h) \coloneqq \infty$ for $h = 1, \ldots, |K|$.
for $h = 1$ to $|K|$ **do**
  for $r = 0$ to c_{ij} **do**
    if $f(h-1,r) < f(h,r)$ **then**
      Set $f(h,r) \coloneqq f(h-1,r)$
    end if
  end for
  for $r = 0$ to $c_{ij} - fl_h$ **do**
    if $f(h-1,r) + (1 - z^*_{ij} - y^*_{hij}) < f(h, r + fl_h)$ **then**
      Set $f(h, r + fl_h) \coloneqq f(h-1,r) + (1 - z^*_{ij} - y^*_{hij})$
    end if
  end for
  for $r = c_{ij} - fl_h + 1$ to c_{ij} **do**
    if $f(h-1,r) + (1 - z^*_{ij} - y^*_{hij}) < g(h)$ **then**
      Set $g(h) \coloneqq f(h-1,r) + (1 - z^*_{ij} - y^*_{hij})$
    end if
  end for
  if $g(h) < 1 - z^*_{ij}$ **then**
    Output the violated cover inequality.
  end if
end for

Fig. 4.2: New separation algorithm for cover inequality.

r, which is dynamically changed from 0 to the residual capacity c_{ij}. $g(h)$ is the objective value of another dynamic program with the total flow of selected tasks from $\{1, \ldots, h\}$ is larger than or equal to $c_{ij}+1$. $g(h)$ is used to compute the optimal value of problem (4.20)–(4.22). It is computed recursively in the dynamic program. When we obtain its value, the output criterion: $g(h) < 1 - z^*_{ij}$, is checked. If it is true, the corresponding CI (4.19) is a valid CI and we output it. For details of the separation algorithm for CI, please Kaparis and Letchford [42].

In the implement of the cutting plane method, we relax integer variables to real ones and solve the linear relaxation problem of the RP_n, then obtain a fractional solution (x^*, y^*, z^*) and a lower bound on the RP_n. Then the separation algorithms presented in Fig. 2.6 and Fig. 4.2 are applied to find valid CIs (4.18) and (4.19). If there exist any valid CIs, we add them to the RP_n, and again solve the relaxed problem of RP_n until no new valid CIs are found.

4.3.4 Overall algorithm

The overall algorithm for the CLRP is presented in Fig. 4.3. It is a cut-and-solve and cutting plane combined method. Steps 1 and 2 are the initialization and pre-process. Steps 3 and 4 are the cutting plane method applied to the CP_1 to obtain the initial lower bound LB_0. Steps 5–11 is the iteration for the cut-and-solve method, and steps 9 and 10 are cutting plane method applied to the RP_n to obtain lower bound LB_n. The stopping iteration of cut-and-solve method is $LB_n \geq UB_{\min}$. If it is satisfied, then $UB_{\min}$ is return as the optimal value of the original problem. Otherwise, we go back to step 5 and begin a new iteration for the cut-and-solve method.

4.4 Computational results

The proposed algorithm for the CLRP was coded in C++ and combined with CPLEX 12.1 solver in default setting mode for the resolution of the sparse problem and remaining problem. The numerical experiments were carried out on a PC with a 3.0 GHz processor and 4.0 GB RAM. Seventy-eight problem sets and five problem instances for each set were randomly generated and tested to evaluate the performance of the proposed algorithm.

The first scenario of problem instances are obtained from [25]. The overall performance of the proposed algorithm is evaluated on these instances. And the performance of the cutting plane method embedded in the proposed algorithm is also evaluated. The other scenario of problem instances are generated in a similar way

Algorithm CLRP

1: Initialize $n \coloneqq 0$ and best upper bound $UB_{\min} \coloneqq +\infty$.
2: Implement the pre-processing for original model P_c, and obtain a new model P'_c. Set current problem $CP_1 \coloneqq P'_c$.
3: Solve the linear relaxation problem of CP_1 and obtain the initial lower bound LB_0, as well as solution and reduced cost information. If the solution is integral, an optimal solution is found, stop the algorithm.
4: Apply the separation algorithm described in Fig. 2.6 and Fig. 4.2 to find possible CIs. If there exist any CIs, add them to CP_1 and go to step 3.
5: **repeat**
6: Set $n \coloneqq n + 1$.
7: Define piercing cut PC_n by (4.15) and obtain SP_n and RP_n.
8: Solve SP_n exactly and obtain its optimal value UB_n if it exists. Set $UB_{\min} \coloneqq UB_n$ if $UB_n < UB_{\min}$.
9: Solve the linear relaxation problem of RP_n and obtain its lower bound LB_n, as well as solution and reduced cost information. If the solution is integral, set $UB_{\min} \coloneqq LB_n$ if $LB_n < UB_{\min}$, and go to step 12.
10: Apply the separation algorithm described in Fig. 2.6 and Fig. 4.2 to find possible CIs. If there exist any CIs, add them to RP_n and go to step 9; otherwise set $CP_{n+1} \coloneqq RP_n$.
11: **until** $(LB_n \geq UB_{\min})$
12: Return $UB_{\min}$ and its corresponding solution as optimal value and optimal solution of the original problem.

Fig. 4.3: Algorithm CLRP: algorithm for the CLRP.

described in chapter 3. They are generated as follows. The graph $G(N, A)$ is generated based on the network model proposed by Waxman [81]. The link travel time on a reserved lane τ_{ij} and on a non-reserved lane τ'_{ij} are respectively calculated as the ratio of the length of the lane and the corresponding speed, where the speed is assumed as 60 for a reserved lane and is generated from $[30, 50]$ for a non-reserved lane. The impact of the reserved lane a_{ij} is evaluated as the increase of link travel time on adjacent lanes. It is set as $a_{ij} = r_a\tau'_{ij}$, where r_a is a given parameter. The prescribed travel duration is defined as $p_k = dis(s_k, d_k) + r_p(dis'(s_k, d_k) - dis(s_k, d_k))$, where $dis(s_k, d_k)$ and $dis'(s_k, d_k)$ are respective the shortest travel duration from s_k to d_k in an exclusively reserved path and in an exclusively non-reserved path, and r_p is a given parameter. Integers fl_k and c_{ij} are randomly and uniformly generated from $[5, 10]$ and $[20r_c, 30r_c]$, respectively. In the default case, we set $r_a \in [0.2, 0.3]$, $r_p = 0.6$, and $r_c = 1$. In addition, sensitive analysis for the performance of the proposed algorithm were conducted with different setting of r_a, r_p, and r_c.

The performance of the cut-and-solve and cutting plane combined method is compared with the direct use of CPLEX in terms of the computational time (in CPU seconds) of finding an optimal solution. In addition, the initial lower bound on the original problem is presented to show the efficiency of the cutting plane method for tightening the linear relaxation problem. To simplify the presentation of computational results, let CT_c and CT_0 respectively denote the computational time required by the proposed algorithm presented in Fig. 4.3 and using CPLEX solver directly. And CT' denote the computational time required by the cut-and-solve based algorithm which does not embed the cutting plane method in [25]. Denote by $\text{Gap}_L = (LB_0 - LB'_0)/LB'_0$ the improvement of the initial lower bound, where LB_0 and LB'_0 are the initial lower bounds obtained by the cutting plane method and linear relaxation, respectively. The computational results are reported in Tables 4.1–4.5 and Figs. 4.4–4.6.

First, we evaluate the overall performance of the proposed algorithm, as well as the cutting plane method. Table 4.1 and Fig. 4.4 present the computational results for problems with fixed number of nodes $|N| = 100$ and the number of tasks $|K|$ varies from 5 to 40. It can be seen that the computational time required by CPLEX (CT_0) is much more than that required by the algorithms presented in Fig. 4.3 (CT_c) and in [25] (CT'). For example, CT_0 is about 10690 seconds for set 8, while the other two are less than 3000 seconds. By comparing the computational time CT_c and CT', we can evaluate the performance of the cutting plane method. It can be seen that CT_c is larger than CT' over sets 1–4. As the size of the problem increases, CT'

Table 4.1: Computational results of problems with fixed 100 nodes.

Set	$\|N\|$	$\|K\|$	$\text{Gap}_L(\%)$	CT_c	CT'	CT_0
1	100	5	5.06	1.17	0.57	0.70
2	100	10	8.05	5.36	2.93	5.84
3	100	15	4.12	13.24	7.79	23.91
4	100	20	11.44	35.00	30.52	64.52
5	100	25	6.39	65.30	75.97	257.25
6	100	30	8.98	123.13	209.76	501.98
7	100	35	5.06	931.41	1497.37	5065.70
8	100	40	6.22	1137.52	2965.72	10689.96
Average			6.92	289.01	598.83	2076.23

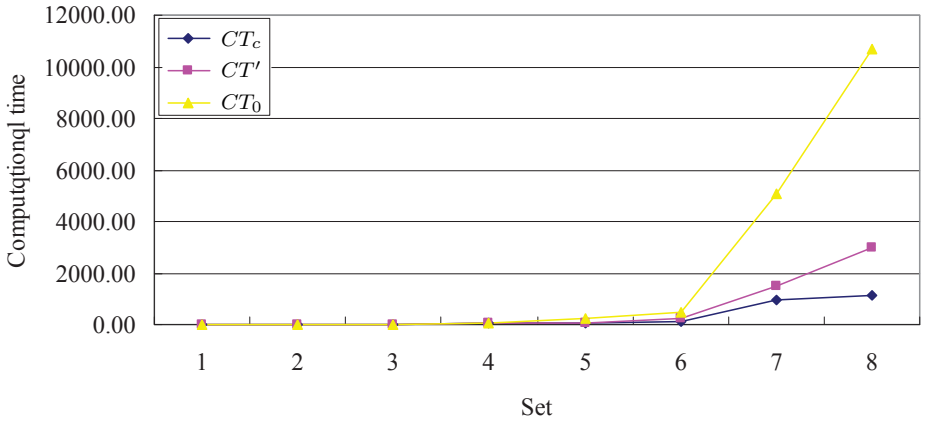


Fig. 4.4: Computational results of problems with fixed 100 nodes.

increases much more quickly than CT_c and CT_c is smaller than CT' over sets 5–8. The average computational time CT_c decreases to 48.26% (289.01/598.83) of CT'. Generally, the cut-and-solve and cutting plane combined method is effective for large sized problems, and cut-and-solve method is efficient for small sized ones. The reason can be explained as follows. When the size of the problem is small, the cut-and-solve and cutting plane combined method takes more computational time than the linear relaxation to obtain a lower bound on the remaining problem. However, as the size of the problem increases, the lower bound obtained by the cutting plane method plays a more important role in the convergence of the algorithm because the algorithm converges quickly with a tight lower bound. The gap (denote by Gap_L) between the initial lower bounds obtained by the cutting plane method and linear relaxation is also presented to show the performance of the cutting plane method. The average value of Gap_L is 6.92%.

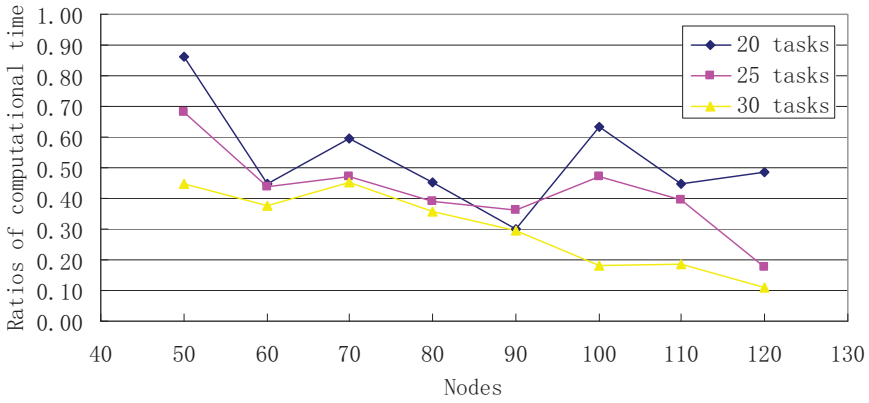


Fig. 4.5: Ratios of computational time of CT_c/CT_0 for fixed 20, 25, and 30 tasks.

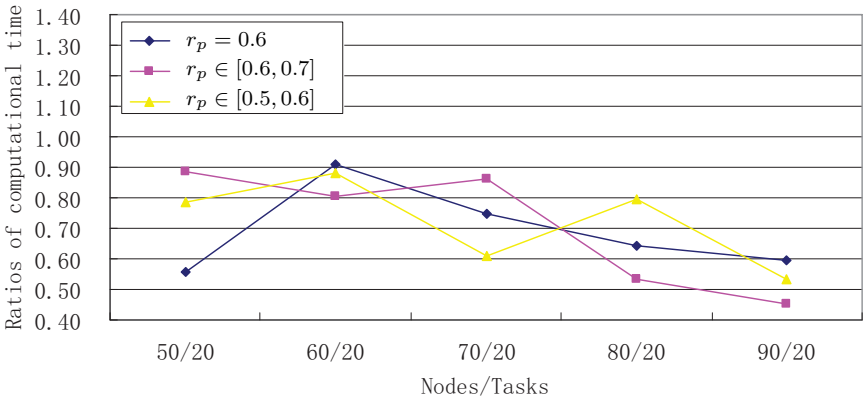


Fig. 4.6: Ratios of computational time of CT_c/CT_0 for different different prescribed travel duration $p_k = dis(s_k, d_k) + r_p(dis'(s_k, d_k) - dis(s_k, d_k))$, $dis(s_k, d_k)$ and $dis'(s_k, d_k)$ are respective the shortest travel duration from s_k to d_k in an exclusively reserved path and in an exclusively non-reserved path.

In Table 4.2, we report the results of problems with $|K| = 20$, 25, and 30 and $|N|$ increases from 50 to 120. The computational time CT_c is smaller than CT_0 for all the sets in Table 4.2. The average value of CT_c/CT_0 is 21.55% over sets 9–32. For large fixed $|K|$, the computational time by CPLEX increases quickly when $|N|$ increases. For example, CT_0 varies from 43.59 to 2801.37 seconds in case of $|K| = 30$, while it varies from 7.97 to 130.64 seconds in case of $|K| = 20$. However, the time by the proposed algorithm CT_c increases gradually when the size of the problem increases. Fig. 4.5 presents the results CT_c/CT_0 for the problems with fixed number of tasks. Generally, CT_c/CT_0 decreases when $|N|$ increases in case of $|K| = 25$ and 30. Specially, the curve for $|K| = 30$ is below the other two. It implies that the proposed algorithm is more effective for problems with large number of tasks than with small number of tasks.

To test the stability of the proposed algorithm, parameters r_c, r_a, and r_p are changed for different setting of residual capacity of lanes c_{ij}, impact of reserved lanes a_{ij}, and prescribed travel duration p_k. They are defined as follows: c_{ij} is an integer and randomly generated from $[20r_c, 30r_c]$, $a_{ij} = r_a\tau'_{ij}$ where τ'_{ij} is the travel time on a non-reserved lane in (i, j), and $p_k = dis(s_k, d_k) + r_p(dis'(s_k, d_k) - dis(s_k, d_k))$, $dis(s_k, d_k)$ and $dis'(s_k, d_k)$ are respective the shortest travel duration from s_k to d_k in an exclusively reserved path and in an exclusively non-reserved path. The results are presented in Tables 4.3–4.5. The average values of CT_c/CT_0 for all problem sets in Tables 4.3–4.5 are 42.76%, 65.66%, and 57.48%, respectively. The ranges of CT_c/CT_0 are 38.32–54.86%, 45.03–91.00%, and 36.58–82.75%, respectively. Take Table 4.5 for example, the comparisons of the corresponding CT_c/CT_0 are shown in Fig. 4.6. Generally speaking, the change trends of the curves represented three types of r_p are almost the same. This shows that the performance of the proposed algorithm does not change much with different setting of the prescribed travel duration. Similar results can also be found for cases of r_c and r_a.

4.5 Conclusions

In this chapter, we have investigated a capacitated lane reservation problem with considering residual capacity, which is a generalization of the lane reservation problem studied previously. The addressed problem is formulated as an integer linear program model and its complexity is demonstrated NP-hard. We propose a cut-and-solve and cutting plane combined method, and then develop an exact algorithm to find optimal solutions. Some strategies for reducing the solution space are developed for the combined method according to the characteristic of the addressed problem. Computational experiments on randomly generated instances demonstrate that the proposed algorithm is more efficient to solve the problem than commercial optimization solver CPLEX. The corresponding work has been published in the following paper.

Y. Fang, F. Chu, S. Mammar, and M. Zhou. Optimal lane reservation in transportation network. *IEEE Transactions on Intelligent Transportation Systems*, 13(2): 482–491, 2012.

Table 4.2: Computational results of problems with fixed 20, 25, and 30 tasks.

Set	$\|N\|$	$\|K\|$	CT_c	CT_0	$CT_c/CT_0(\%)$
9	50	20	6.87	7.97	86.13
10	60	20	11.11	24.86	44.67
11	70	20	28.86	48.33	59.71
12	80	20	31.24	69.08	45.22
13	90	20	26.22	87.60	29.93
14	100	20	56.24	89.05	63.15
15	110	20	44.54	99.42	44.80
16	120	20	63.47	130.64	48.59
17	50	25	10.30	15.10	68.23
18	60	25	10.54	24.03	43.87
19	70	25	24.29	51.44	47.23
20	80	25	31.71	81.06	39.12
21	90	25	37.72	104.58	36.07
22	100	25	98.10	209.03	46.93
23	110	25	117.67	299.10	39.34
24	120	25	156.53	882.65	17.73
25	50	30	19.45	43.59	44.63
26	60	30	35.33	94.11	37.54
27	70	30	51.07	113.36	45.05
28	80	30	59.46	165.95	35.83
29	90	30	64.22	217.97	29.46
30	100	30	168.31	926.15	18.17
31	110	30	277.85	1493.25	18.61
32	120	30	310.44	2801.37	11.08
Average			72.56	336.65	21.55

Table 4.3: Computational results of problems with different lane's residual capacity $c_{ij} \in [20r_c, 30r_c]$.

Set	r_c	$\|N\|$	$\|K\|$	CT_c	CT_0	$CT_c/CT_0(\%)$
33	0.6	50	20	20.49	45.37	45.15
34	0.6	60	20	20.94	53.99	38.78
35	0.6	70	20	32.21	60.12	53.58
36	0.6	80	20	47.16	93.61	50.38
37	0.6	90	20	47.56	156.40	30.41
38	1.0	50	20	14.09	36.76	38.32
39	1.0	60	20	15.20	35.19	43.18
40	1.0	70	20	19.34	38.31	50.49
41	1.0	80	20	35.38	79.77	44.36
42	1.0	90	20	25.14	61.49	40.89
43	1.4	50	20	11.74	28.31	41.47
44	1.4	60	20	14.28	32.41	44.07
45	1.4	70	20	13.89	25.31	54.86
46	1.4	80	20	34.02	79.82	42.62
47	1.4	90	20	38.64	85.48	45.21
Average				26.01	60.82	42.76

Table 4.4: Computational results of problems with different impact of reserved lanes $a_{ij} = r_a \tau'_{ij}$, τ'_{ij} is the travel time on a non-reserved lane in (i, j).

Set	r_a	$\|N\|$	$\|K\|$	CT_c	CT_0	$CT_c/CT_0(\%)$
48	[0.1, 0.2]	50	20	9.11	10.31	88.39
49	[0.1, 0.2]	60	20	25.26	31.47	80.28
50	[0.1, 0.2]	70	20	32.54	37.73	86.23
51	[0.1, 0.2]	80	20	43.58	81.45	53.50
52	[0.1, 0.2]	90	20	45.32	100.64	45.03
53	[0.2, 0.3]	50	20	8.77	15.75	55.70
54	[0.2, 0.3]	60	20	41.49	45.60	91.00
55	[0.2, 0.3]	70	20	43.86	58.56	74.90
56	[0.2, 0.3]	80	20	46.42	72.33	64.18
57	[0.2, 0.3]	90	20	50.88	85.23	59.70
58	[0.3, 0.4]	50	20	9.54	12.17	78.36
59	[0.3, 0.4]	60	20	45.29	51.40	88.12
60	[0.3, 0.4]	70	20	38.40	62.92	61.03
61	[0.3, 0.4]	80	20	37.12	46.61	79.63
62	[0.3, 0.4]	90	20	43.13	80.93	53.30
Average				34.71	52.87	65.66

Table 4.5: Computational results of problems with different prescribed travel duration $p_k = dis(s_k, d_k) + r_p(dis'(s_k, d_k) - dis(s_k, d_k))$, $dis(s_k, d_k)$ and $dis'(s_k, d_k)$ are respective the shortest travel duration from s_k to d_k in an exclusively reserved path and in an exclusively non-reserved path.

Set	r_p	$\|N\|$	$\|K\|$	CT_c	CT_0	$CT_c/CT_0(\%)$
63	0.6	50	20	12.09	23.97	53.82
64	0.6	60	20	27.93	46.01	60.70
65	0.6	70	20	32.18	67.40	47.74
66	0.6	80	20	40.63	78.72	51.62
67	0.6	90	20	62.33	96.20	64.79
68	[0.6, 0.7]	50	20	22.78	27.53	82.75
69	[0.6, 0.7]	60	20	31.72	39.24	80.82
70	[0.6, 0.7]	70	20	58.45	96.15	60.79
71	[0.6, 0.7]	80	20	35.64	66.39	53.68
72	[0.6, 0.7]	90	20	59.38	99.03	59.96
73	[0.5, 0.6]	50	20	9.63	15.03	64.12
74	[0.5, 0.6]	60	20	30.78	41.51	74.16
75	[0.5, 0.6]	70	20	38.06	82.11	46.36
76	[0.5, 0.6]	80	20	22.58	61.75	36.58
77	[0.5, 0.6]	90	20	49.20	88.35	55.69
Average				35.61	61.96	57.48

Chapter 5

Lane reservation problems with dynamic link travel time

5.1 Introduction

As introduced in chapter 2, the LRPs are intended to design time-guaranteed task paths by optimally setting reserved lanes in a transportation network. However, the previously studied LRP in chapter 3 and CLRP in chapter 4 involve only static link travel time in a transportation network. In real-life, traffic situation dynamically changes due to many factors, such as peak hours, traffic flow, and weather conditions. Then link travel time dynamically change due to these factors. In the literature, dynamic link travel time has been introduced to some transportation problems, such as the VRPTW [13], [37]. It is necessary to consider dynamic link travel time in the lane reservation problems to make them closer to the realistic situations. In this chapter, we investigate two dynamic lane reservation problems: lane reservation problem with time-dependent travel time (LRP-TT) and lane reservation problem with time-dependent travel speed (LRP-TS). Compared with the problems in chapters 3 and 4, the travel time on non-reserved lanes in the LRP-TT and LRP-TS is not a constant, it can be changed with the time of day. Because of the introduction of dynamic link travel time to the LRP-TT and LRP-TS, the problems become much difficult to solve. The solution approaches in chapters 3 and 4 cannot be applied directly to them. A new cut-and-solve based method is developed for the LRP-TT. New strategies of piercing cut generation are developed for the cut-and-solve based method. Moreover, for the LRP-TS, the property "first-in-first-out" (FIFO) is satisfied due to the time-dependent travel speed model. A tabu search algorithm embedded time-dependent shortest path algorithm is developed for the LRP-TS. The remainder of this chapter is organized as follows. In sections 5.2 and 5.3, the LRP-TT

and LRP-TS are respectively addressed, including problem formulation, linearization of the nonlinear model, solution approach, and performance evaluation. The chapter is concluded in section 5.4.

5.2 Lane reservation problem with time-dependent travel time

5.2.1 Problem description and formulation

The lane reservation problems studied in the previous chapters assume static link travel time throughout the whole time horizon. In this section, we study a lane reservation problem with time-dependent travel time (LRP-TT), in which the travel time on a non-reserved lane is not a constant any more, it dynamically changes with the time of day.

The LRP-TT is described as follows. Given a network $G(N, A)$, a set of tasks and corresponding source-destination (SD) pairs, the LRP-TT is to setting reserved lanes from the network and to design paths for the time-guaranteed tasks, with the objective of minimizing the total impact of reserved lanes on the network. Unlike the previous lane reservation problems in which static link travel time in a transportation network is assumed, the travel time on a non-reserved lane $(i, j) \in A$ is assumed to be dependent on the departure time at node i and the travel time on a reserved lane is assumed a constant in the LRP-TT. Because non-reserved lanes can be used by all types of the vehicles, the traffic situation may be very different for different time of the day. For example, the traffic during morning and afternoon peak hour are much more congested than other times of the day. Therefore, the travel time on a non-reserved lane is assumed time-dependent. For reserved lanes, because they can be used by only the tasks' vehicles. Therefore, they can provide a congestion free travel environment and the travel time on a reserved lane is assumed a constant over the whole time horizon. In this sense, the LRP-TT is an extension to the previous lane reservation problems.

To well study the lane reservation problem with dynamic link travel time, the travel time on a non-reserved lane $(i, j) \in A$ is assumed to be a step function of the departure time at node i in the LRP-TT. This assumption has been widely used in many time-dependent transportation problems, such as time-dependent VRP [53], time-dependent traveling salesman problem [54], and time-dependent shortest path problem [11] [87]. The assumption of step function is an approximation of realistic

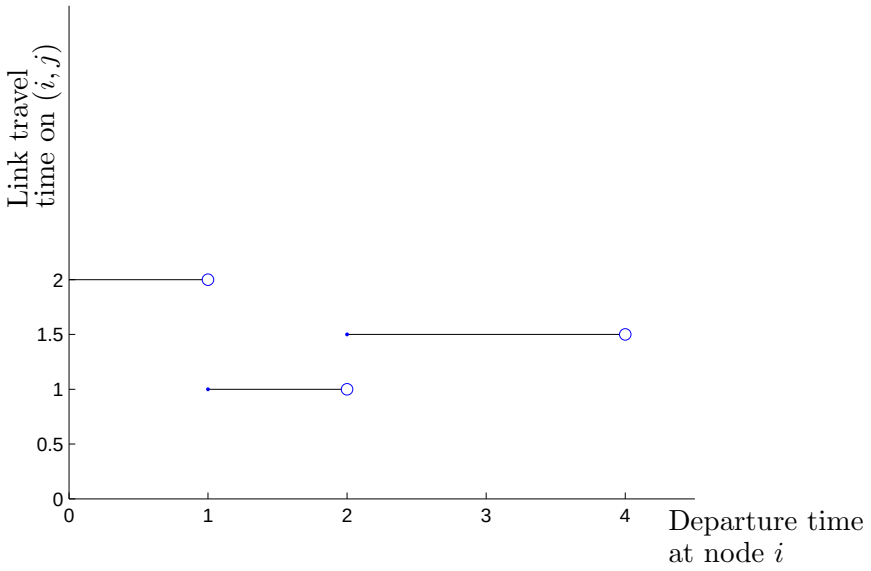


Fig. 5.1: Example of time-dependent travel time on a non-reserved lane (i, j).

continuous function due to the complexity of continuous functions that are much more difficult to deal with than step functions. We consider a tractable case of dynamic lane reservation problems with link travel time of a step function at the first time. It is described as follows. Denote the whole time horizon by $[T_{BEG},\ T_{END})$. Let $T_q \in [T_{BEG},\ T_{END}),\ q \in Q$ denote the boundary of time interval, where $Q = \{0,\ \ldots,\ nq-1\}$ and nq is a pre-given integer which represents the number of time intervals. Then the whole time horizon is divided as follows: $[T_0,\ T_1,\ \ldots,\ T_{nq-1},\ T_{nq})$, where $T_0 = T_{BEG}$, $T_{nq} = T_{END}$, and each $[T_q,\ T_{q+1}),\ q \in Q$ represents a time interval. For a non-reserved lane $(i, j) \in A$, the travel time on (i, j) is a step function of the departure time at node i. It is a constant for each time interval $[T_q, T_{q+1}), \forall q \in Q$. An example of the time-dependent travel time for a non-reserved lane (i, j) is given in Fig. 5.1. For example, if a task departs at i at a time point $t_1 = 0.5$, then $t_1 \in [0, 1)$, the travel time on (i, j) will be 2; if it departs at i at a time point $t_2 \in [1, 2)$, then travel time on (i, j) will be 1. On the other hand, only the tasks' vehicles are allowed to travel on reserved lanes and they can provide a congestion free travel environment. Thus, the travel time on a reserved lane is assumed a constant over the whole time horizon.

To formulate the problem, some notations are given as follows.

Sets and input parameters

A:	set of directed arcs $(i, j), i \neq j, i, j \in N$
K:	set of transportation tasks, $k \in K$

N: set of nodes
Q: set of indices of time interval, $Q = \{0, \ldots, nq - 1\}$
$[T_q,\ T_{q+1})$ time interval, $q \in Q$
a_{ij}: traffic impact if a lane in $(i, j) \in A$ is reserved
d_k: destination node of task $k \in K$
nq: number of time interval
p_k: prescribed travel duration to complete task $k \in K$
s_k: source node of task $k \in K$
st_k: starting time for task k at node s_k, $k \in K$
τ_{ij}: travel time on a reserved lane in $(i, j) \in A$
τ''_{ijq}: travel time on a general-purpose lane in $(i, j) \in A$ when departure time at i is within time interval $[T_q, T_{q+1}), q \in Q$

Decision variables

b_{kiq}: $b_{kiq} = 1$ if task k departs at node i in time interval $[T_q, T_{q+1})$; otherwise $b_{kiq} = 0$, $\forall k \in K, \forall i \in N, \forall q \in Q$.
t_{ki}: departure time of task k at node i; $t_{ki} = 0$ if node i is not visited by task k, $\forall k \in K, \forall i \in N$.
x_{kij}: $x_{kij} = 1$, if a lane in link (i, j) is in the path of task k and this lane is reserved; and otherwise $x_{kij} = 0$, $\forall k \in K, \forall (i, j) \in A$.
y_{kij}: $y_{kij} = 1$, if a lane in link (i, j) is in the path of task k and this lane is not reserved; and otherwise $y_{kij} = 0$, $\forall k \in K, \forall (i, j) \in A$.
z_{ij}: $z_{ij} = 1$, if there is a reserved lane in link (i, j); and otherwise $z_{ij} = 0$, $\forall (i, j) \in A$

The LRP-TT can be formulated as the following programming model P_t:

$$P_t: \min \sum_{(i,j)\in A} a_{ij} z_{ij} \tag{5.1}$$

$$\text{s.t.} \sum_{i:(s_k,i)\in A} (x_{ks_ki} + y_{ks_ki}) = 1, \qquad \forall k \in K, \tag{5.2}$$

$$\sum_{i:(i,d_k)\in A} (x_{kid_k} + y_{kid_k}) = 1, \qquad \forall k \in K, \tag{5.3}$$

$$\sum_{i:(i,j)\in A} (x_{kij} + y_{kij}) = \sum_{i:(j,i)\in A} (x_{kji} + y_{kji}), \qquad \forall k \in K, \forall j \in N \backslash \{s_k, d_k\}, \tag{5.4}$$

$$x_{kij} \le z_{ij}, \qquad \forall k \in K, \forall (i,\ j) \in A, \tag{5.5}$$

$$y_{kij} + z_{ij} \le 1, \qquad \forall k \in K, \forall (i,\ j) \in A, \tag{5.6}$$

$$\sum_{q\in Q} b_{kiq} = 1, \qquad \forall k \in K, \forall i \in N \backslash \{d_k\}, \tag{5.7}$$

$$t_{ki} \ge \sum_{q\in Q} b_{kiq} T_q, \qquad \forall k \in K, \forall i \in N \backslash \{d_k\}, \tag{5.8}$$

$$t_{ki} < \sum_{q \in Q} b_{kiq} T_{q+1}, \qquad \forall k \in K, \forall i \in N \backslash \{d_k\}, \tag{5.9}$$

$$t_{ks_k} = st_k, \qquad \forall k \in K, \tag{5.10}$$

$$t_{kj} = \sum_{i:(i,j) \in A} ((t_{ki} + \tau_{ij}) x_{kij} + (t_{ki} + \sum_{q \in Q} b_{kiq} \tau''_{ijq}) y_{kij}), \quad \forall k \in K, \forall j \in N \backslash \{s_k\}, \tag{5.11}$$

$$t_{kd_k} - t_{ks_k} \leq p_k, \qquad \forall k \in K, \tag{5.12}$$

$$t_{ki} \geq 0, \qquad \forall k \in K, \forall i \in N, \tag{5.13}$$

$$b_{kiq} \in \{0,\ 1\}, \qquad \forall k \in K, \forall i \in N, \forall q \in Q, \tag{5.14}$$

$$x_{kij},\ y_{kij} \in \{0,\ 1\}, \qquad \forall k \in K, \forall (i,\ j) \in A, \tag{5.15}$$

$$z_{ij} \in \{0,\ 1\}, \qquad \forall (i,\ j) \in A. \tag{5.16}$$

The objective function (5.1) is to minimize the traffic impact of all reserved lanes. Constraints (5.2)–(5.4) together ensure that there is exact one path for each task k from s_k to d_k. Constraint (5.5) means that task k cannot pass the reserved lane in (i, j) if there is no reserved lane in it. Constraint (5.6) is tighter than constraint $x_{kij} + y_{kij} \leq 1$. It means that if task k passes (i, j) (i.e., $x_{kij} + y_{kij} = 1$), either it passes a reserved lane (i, j) (i.e., $x_{kij} = 1$) or it passes a non-reserved lane (i, j) (i.e., $y_{kij} = 1$), otherwise task k dose not pass (i, j) (i.e., $x_{kij} + y_{kij} = 0$). Constraint (5.7)–(5.9) indicates that if $b_{kiq} = 1$, then the departure time t_{ki} of task k at node i is in the time interval $[T_q, T_{q+1})$. Constraint (5.10) indicates that task k departs at node s_k at time st_k. Constraint (5.11) indicates the time of task k at node j. If task k visits j via a reserved lane (i, j), the travel time on (i, j) is τ_{ij} and the time at j is $t_{ki} + \tau_{ij}$; if it visits j via a non-reserved lane (i, j), the travel time on (i, j) is dependent on the time interval in which the departure time at node i is. It can be represent as $\sum_{q \in Q} b_{kiq} \tau''_{ijq}$, where $b_{kiq} = 1$ if the departure time at i is in time interval $[T_q, T_{q+1})$ and τ''_{ijq} is the corresponding travel time on (i, j). Then the time at j is $t_{ki} + \sum_{q \in Q} b_{kiq} \tau''_{ijq}$. Constraint (5.12) means that the travel time of task k from s_k to d_k should not exceed the prescribed travel duration p_k. Constraints (5.13)–(5.16) are for the decision variables. Note that (5.11) is not linear, and it will be reformulated as linear inequalities in the following section.

The complexity of the LRP-TT is given by the following theorem.

Theorem 6 *The LRP-TT is NP-hard.*

Proof: For each $(i, j) \in A$ and time interval $[T_q, T_{q+1}), q \in Q$, if the corresponding travel time τ''_{ijq} is very large (e.g., greater than $\max_{k \in K}\{p_k\}$), then each lane in the

task paths must be reserved to ensure the time-guaranteed tasks, otherwise the tasks cannot be completed within the prescribed travel time $p_k, k \in K$. Thus, the LRP-TT is reduced to the LRP in chapter 3, which has been proved NP-hard. Therefore, the LRP-TT is NP-hard. □

5.2.2 Model linearization

Model P_t is not a linear programming since (5.11) is nonlinear. It difficult to solve nonlinear programming model. In this section, P_t is transformed into an equivalent linear model by reformulating (5.11) according to the following three cases.

1) Node j is not visited by task k, i.e., $\sum_{i:(i,j)\in A} x_{kij} = 0$ and $\sum_{i:(i,j)\in A} y_{kij} = 0$. Then (5.11) can be reformulated as follows:

$$t_{kj} \leq M \sum_{i:(i,j)\in A} (x_{kij} + y_{kij}), \qquad \forall k \in K, \forall j \in N\backslash\{s_k\}, \tag{5.17}$$

where M is a large constant.

2) Node j is visited by task k via a reserved lane in (i,j), i.e., $\sum_{i:(i,j)\in A} x_{kij} = 1$ and $\sum_{i:(i,j)\in A} y_{kij} = 0$. Then (5.11) can be reformulated as follows:

$$\sum_{i:(i,j)\in A} (t_{kj} - t_{ki} - \tau_{ij}) x_{kij} = 0, \qquad \forall k \in K, \forall j \in N\backslash\{s_k\}.$$

Since $\sum_{i:(i,j)\in A} x_{kij} = 1$, then each item in the left hand of the above formula is zero, we have:

$$(t_{kj} - t_{ki} - \tau_{ij}) x_{kij} = 0, \qquad \forall k \in K, \forall j \in N\backslash\{s_k\}, (i,j) \in A.$$

Then it can be further reformulated as follows:

$$\begin{array}{l} t_{kj} - t_{ki} - \tau_{ij} \leq M(1 - x_{kij}), \\ t_{kj} - t_{ki} - \tau_{ij} \geq M(x_{kij} - 1), \end{array} \qquad \forall k \in K, \forall j \in N\backslash\{s_k\}, (i,j) \in A. \tag{5.18}$$

3) Node j is visited by task k via a non-reserved lane in (i,j), i.e., $\sum_{i:(i,j)\in A} x_{kij} = 0$ and $\sum_{i:(i,j)\in A} y_{kij} = 1$. Similar to case 2), (5.11) can be reformulated as follows:

$$\begin{array}{l} t_{kj} - t_{ki} - \sum\limits_{q\in Q} b_{kiq}\tau''_{ijq} \leq M(1 - y_{kij}), \\ t_{kj} - t_{ki} - \sum\limits_{q\in Q} b_{kiq}\tau''_{ijq} \geq M(y_{kij} - 1), \end{array} \quad \forall k \in K, \forall j \in N\backslash\{s_k\}, (i,j) \in A. \tag{5.19}$$

Since each of (5.17)–(5.19) is linear, an equivalent linear model P'_t can be obtained by replacing (5.11) with (5.17)–(5.19). It is given as follows:

$$\begin{array}{ll} P'_t: & \min \sum\limits_{(i,j)\in A} a_{ij} z_{ij} \\ \text{s.t.} & \text{constraints } (5.2) - (5.10) \text{ and } (5.12) - (5.19). \end{array}$$

5.2.3 Solution approach

In this section, in first a pre-processing is firstly performed to tight the solution space of the problem P'_t. Then a cut-and-solve based optimal algorithm is developed for it. Some new strategies are proposed for generating appropriate piercing cuts in the cut-and-solve method.

5.2.3.1 Pre-processing

For $\forall j \in N$, let $l(s_k, j)$ denote the shortest travel duration from s_k to j in an exclusively reserved path and $l(j, d_k)$ denote the shortest travel duration from j to d_k in an exclusively reserved path, where s_k and d_k are the source and destination of task k, respectively. Then $l(s_k, j)$ and $l(j, d_k)$ can be computed by Dijkstra shortest path algorithm. For $\forall k \in K$, define set N_k as follows:

$$N_k = \{j \mid l(s_k, j) + l(j, d_k) > p_k, j \in N\}.$$

It is not difficult to see that the nodes in N_k will not be visited by task k otherwise the travel time constraint will be violated. Then the corresponding decision variables can be fixed to zero and represented by the following equalities:

$$\begin{aligned} \sum_{i:(i,j)\in A} (x_{kij} + y_{kij}) &= 0, \\ \sum_{i:(j,i)\in A} (x_{kji} + y_{kji}) &= 0, \qquad \forall k \in K, \forall j \in N_k. \\ t_{kj} &= 0, \end{aligned} \tag{5.20}$$

For $\forall k \in K, \forall j \in N \backslash \{s_k\}$, if node j is not visited by task k, then t_{kj} is zero by its definition; if j is visited by task k, the earliest arrived time at j for task k is $st_k + l(s_k, j)$, where st_k is the departure time of task k at node s_k and $l(s_k, j)$ is the shortest travel duration from s_k to j. These two cases can be represented by the following inequalities:

$$t_{kj} \geq (st_k + l(s_k, j)) \sum_{i:(i,j)\in A} (x_{kij} + y_{kij}), \qquad \forall k \in K, \forall j \in N \backslash \{s_k\}. \tag{5.21}$$

Then (5.20) and (5.21) can be added as new constraints to tighten P'_t. The new model P''_t is given as follows:

$$\begin{aligned} P''_t: \quad & \min \sum_{(i,j)\in A} a_{ij} z_{ij} \\ \text{s.t.} \quad & \text{constraints } (5.2) - (5.10) \text{ and } (5.12) - (5.21). \end{aligned}$$

5.2.3.2 Cut-and-solve method

The principle of the cut-and-solve method has been described in chapter 2. To apply the cut-and-solve method to the LRP-TT, we need to define the piercing cut (PC_n), current problem (CP_n), sparse problem (SP_n) and remaining problem (RP_n).

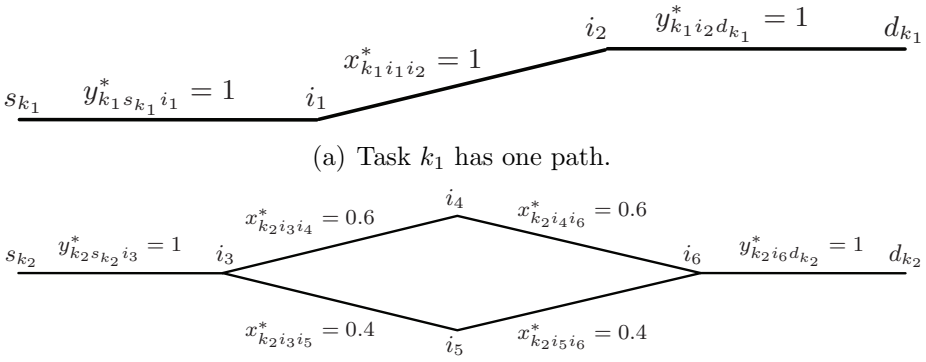
In chapters 3 and 4, PC_n is defined on a set which includes decision variables with large reduced cost. The variables' reduced cost is obtained by applying a linear relaxation strategy and solve the resulting problem. However, the linear relaxation strategy is not suitable for the LRP-TT because preliminary experiment results shown that the lower bound of the LRP-TT obtained by the linear relaxation strategy is very "bad", which is not good for the convergence of the cut-and-solve method. For the LRP-TT, we apply a partial integral relaxation strategy instead of the linear relaxation strategy. It means that only integer decision variables x_{kij} and z_{ij} are relaxed to continuous while b_{kiq} and y_{kij} are kept non-relaxed. The partial integral relaxation strategy is based on the following observations.

1) Preliminary experiment results shown that if we apply the linear relaxation strategy (i.e., relax all the integer variables to continuous), most of the variables z_{ij} have values of zero. And the lower bound (LB_n) of the RP_n is very "bad", which leads to a very slow convergence of the cut-and-solve method since the method will not be terminated until LB_n is greater than or equal to the current best upper bound ($UB_{\min}$). To accelerate the termination of the cut-and-solve method, LB_n must be "good" enough. Therefore, we applied a partial integral relaxation strategy in order to obtain a "good" LB_n.

2) The variables relaxed to the task paths are x_{kij} and y_{kij}. If y_{kij} are relaxed, experiment results shown that they have fractional values and most of x_{kij} have values of zero, which means that very few reserved lanes are passed by the tasks. The LB_n is very "bad". Therefore, y_{kij} should not be relaxed. b_{kiq} are not relaxed for the same reason.

On other hand, variables' reduced cost cannot be obtained by partial integral relaxation strategy. Thus, the previous methods of generating PC_n in chapters 3 and 4 is not suitable for the LRP-TT. New techniques for generating PC_n will be explained as follows.

In our implementation of the algorithm, we apply the partial integral relaxation strategy to the CP_n and solve the resulting problem. Denote the solution by $sol_n^* = (t_{kj}^*, b_{kiq}^*, x_{kij}^*, y_{kij}^*, z_{ij}^*)$. Since x_{kij}^* may be fractional value, there may be more than



(a) Task k_1 has one path.

(b) Task k_2 has multi-path.

Fig. 5.2: Example of task path.

one path for some tasks from their sources to destinations. Let K_n denote the set of tasks that have more than one path in sol^*_n:

$$K_n = \{k \in K \mid \text{there are more than one path for task } k \text{ in solution } sol^*_n\}. \quad (5.22)$$

For each $k \in K_n$, let i_k denote the first node in its paths where multi-path appears. Then set V_n composed of *critical links* (i_k, j_k) is defined as follows:

$$V_n = \{\ (i_k, j_k) \mid (i_k, j_k) = \arg \max_{(i_k,j)\in A} \{x^*_{ki_kj}\},\ \forall k \in K_n\ \}. \quad (5.23)$$

To explain the above definition, an illustration is presented in Fig. 5.2. In Fig. 5.2, task k_1 has only one path while task k_2 has two paths from their sources to destinations, respectively. Hence $k_1 \notin K_n$ and $k_2 \in K_n$. In addition, for task k_2, the first node where multi-path appears is node i_3. The links connected with i_3 are (i_3, i_4) and (i_3, i_5) and $x^*_{k_2 i_3 i_4}(= 0.6)$ is larger than $x^*_{k_2 i_3 i_5}(= 0.4)$. Hence the critical link for task k_2 is (i_3, i_4). According to the definition of V_n, (i_k, j_k) is the link with largest value among all the links outgoing from i_k. Preliminary tests shown that the critical links have large possibility to be selected (i.e., $x_{ki_kj_k} + y_{ki_kj_k} = 1$) for the task paths in the optimal solution of the original problem. If all the critical links are selected, then $\sum_{(i_k,j_k)\in V_n} x_{ki_kj_k} + y_{ki_kj_k} = |K_n|$. However, experiments results shown that this is not always true. Therefore, we suppose that at least h_n ($h_n < |K_n|$) critical links are selected. In our implementation, h_n is set as $|K_n| - 1$ or $|K_n| - 2$. Then according to the principle of the cut-and-solve method, the PC_n ($n \geq 1$) is defined as follows:

$$PC_n : \quad \sum_{(i_k j_k)\in V_n} (x_{ki_kj_k} + y_{ki_kj_k}) \leq h_n - 1, \quad (5.24)$$

For $n \geq 1$, the SP_n and RP_n are defined as follows:

$$
\begin{aligned}
SP_n: \quad & \min \sum_{(i,j)\in A} a_{ij} Z_{ij} \\
\text{s.t.} \quad & \text{constraints } (5.2)-(5.10) \text{ and } (5.12)-(5.21) \\
& \sum_{(i_k j_k)\in V_m} (x_{k i_k j_k} + y_{k i_k j_k}) \leq h_n - 1, \qquad m = 1, 2, \ldots, n-1. \qquad (5.25) \\
& \sum_{(i_k j_k)\in V_n} (x_{k i_k j_k} + y_{k i_k j_k}) \geq h_n. \qquad (5.26) \\
RP_n: \quad & \min \sum_{(i,j)\in A} a_{ij} Z_{ij} \\
\text{s.t.} \quad & \text{constraints } (5.2)-(5.10), (5.12)-(5.21), (5.24) \text{ and } (5.25)
\end{aligned}
$$

Since $x_{k i_k j_k} + y_{k i_k j_k} \leq 1$, (5.26) implies that at least h_n items of $x_{k i_k j_k} + y_{k i_k j_k}$ have non-zero values, i.e., at least h_n critical links will be selected for the task paths. The special case $h_n = |K_n|$ implies that $x_{k i_k j_k} + y_{k i_k j_k} = 1, \forall k \in K_n$. Then the SP_n can be solved to optimality relatively easily. The CP_1 is defined as P_t'' and CP_n $(n \geq 2)$ is defined as RP_{n-1}.

5.2.3.3 Overall algorithm

The overall algorithm is described in Fig. 5.3.

5.2.4 Computational results

In this section, the performance of the proposed algorithm was evaluated on randomly generated problem instances. The proposed algorithm was coded in Visual C++ embedded with CPLEX 12.1 MIP solver used to solve the sparse problem and relaxed remaining problem. All the experiments were carried out on a PC with 3.0 GHz CPU and 4.0 GB RAM. Each problem set includes five instances.

The graph $G(N, A)$ is generated based on the network model proposed by Waxman [81]. The time horizon is set as $[0, 12]$ and is divided into $|Q|$ time intervals. Obviously, large $|Q|$ made the problem difficult to solve. In [53], a time-dependent VRPTW was studied and the number of time interval $|Q|$ is set as 2 or 3. In our experiments, $|Q|$ is set as 4 or 6. For each $(i, j) \in A$ and $q \in Q$, the travel time on a reserved lane τ_{ij} and on a non-reserved lane τ''_{ijq} are set as the ratio of the length of (i, j) and a corresponding speed, where the speed is set as 60 for a reserved lane and is uniformly generated in $[30, 50]$ for a non-reserved lane. The average travel time on a non-reserved lane over the whole horizon is then computed as $\overline{\tau}''_{ij} = \sum_{q\in Q} \tau''_{ijq}/|Q|$.

Algorithm LRP-TT

1: Initialize $n \coloneqq 0$ and best upper bound $UB_{\min} \coloneqq +\infty$.
2: Implement pre-processing for original nonlinear model P_t, and obtain a new linear model P'_t. Set current problem $CP_1 \coloneqq P'_t$.
3: Apply the partial integral relaxation strategy to CP_1, i.e., relax x_{kij} and z_{ij} to continuous, and solve the resulting relaxed problem.
4: **repeat**
5: Set $n \coloneqq n + 1$. Use the solution information of relaxed problem of CP_n to define critical link set V_n by (5.23).
6: Define piercing cut PC_n by (5.24) and obtain SP_n and RP_n.
7: Solve SP_n exactly and obtain its optimal value UB_n if it exists. Set $UB_{\min} \coloneqq UB_n$ if $UB_n < UB_{\min}$.
8: Apply the partial integral relaxation strategy to RP_n. Solve the resulting relaxed problem and obtain its solution information and lower bound LB_n of RP_n. If the solution is integral, set $UB_{\min} \coloneqq LB_n$ if $LB_n < UB_{\min}$, and go to step 10; otherwise set $CP_{n+1} \coloneqq RP_n$.
9: **until** $(LB_n \geq UB_{\min})$
10: Return $UB_{\min}$ and its corresponding solution as optimal value and optimal solution of the original problem.

Fig. 5.3: Algorithm LRP-TT: algorithm for the LRP-TT.

The impact of reserved lane is set as $a_{ij} = r_a \overline{\tau}''_{ij}$, where r_a is a given parameter. The prescribed travel duration is set as $p_k = dis(s_k, d_k) + r_p(dis'(s_k, d_k) - dis(s_k, d_k))$, where $dis(s_k, d_k)$ (resp. $dis'(s_k, d_k)$) is the shortest travel duration from s_k to d_k in the case of the travel time on (i, j) is τ_{ij} (resp. $\overline{\tau}''_{ij}$), where r_p is a given parameter. In the default case, $r_a \in [0.2, 0.3]$, $r_p = 0.7$. In addition, experiments of performance sensitive analysis for these parameters have also been conducted.

The performance of the proposed algorithm is compared with the direct use of CPLEX to model P'_t in terms of the computational time (in CPU seconds) of finding an optimal solution. Let $|N|$, $|K|$, and $|Q|$ denote the number of nodes, tasks, and time intervals; and CT_t, CT_0 denote the average computational time required by the proposed algorithm and CPLEX, respectively. Each size of problem has five instances. With these notations, the computational results are presented as follows.

Table 5.1 and Fig. 5.4 present the computational results of different sized problems with $|N|$ varies from 80 to 95 and $|K|$ varies from 10 to 30. It can be observed from Table 5.1 that CT_t is less than CT_0 over sets 1–7 and CT_t/CT_0 varies from 26.59% to 87.67%. CPLEX cannot solve problem set 8 exactly within 3600.00 seconds whereas

Table 5.1: Computational results of problems with different sizes.

Set	$\|N\|$	$\|K\|$	$\|Q\|$	CT_t	CT_0	$CT_t/CT_0(\%)$
1	80	10	4	12.80	49.31	25.96
2	80	15	4	41.67	86.93	47.93
3	85	15	4	103.63	150.16	69.01
4	85	20	4	237.23	323.22	73.40
5	90	20	4	392.38	516.28	76.00
6	90	25	4	955.30	1132.64	84.34
7	95	25	4	1278.54	1458.40	87.67
8	95	30	4	1710.38	–	–

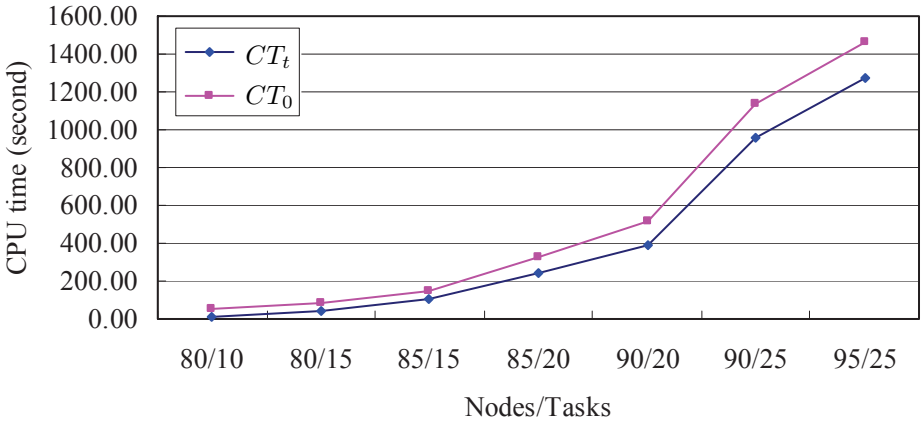


Fig. 5.4: Computational results of problems with different sizes.

the proposed algorithm takes 1710 seconds. The proposed algorithm can solve the problems more quickly than CPLEX. However, it also can be found that CT_t/CT_0 increases with the size of the problem, which means that the proposed algorithm becomes less efficient to solve large sized problems. It can be seen from Fig. 5.4 that both the computational time by CPLEX and the proposed algorithm increases quickly with the size of problem. The time required by the proposed algorithm is slightly less than that by CPLEX.

Table 5.2 presents the results of problems with 4 and 6 time intervals. The computational time CT_t is less than CT_0 for each set in Table 5.2. The average time CT_t are 52.76 seconds for $|Q| = 4$ and 98.58 seconds for $|Q| = 6$, whereas average time CT_0 are 77.30 seconds for $|Q| = 4$ and 150.51 seconds for $|Q| = 6$. The difference between the average values of CT_t/CT_0 for $|Q| = 4$ (68.25%) and for $|Q| = 6$ (65.50%) is small. Fig. 5.5 presents the computational time CT_t and CT_0 for $|Q| = 4$ and 6. It can be found that both CT_t and CT_0 increase quickly when $|Q|$ increases from 4

to 6. This is easy to understand since larger number of time intervals results in more decision variables and more constraints in the model, which makes the model difficult to solve.

Table 5.3 presents the results of problems with different impact a_{ij}. Parameter r_a is changed to generate different values of a_{ij}. It can be found from Table 5.3 and that CT_t is less than CT_0 for problem sets 24–43 except for set 23. The average values of CT_t/CT_0 are 68.03% (for $r_a \in [0.1, 0.2]$), 72.11% (for $r_a \in [0.2, 0.3]$), and 74.44% (for $r_a \in [0.3, 0.4]$), respectively. The differences among the above three values are small, which shown that the performance of the proposed algorithm is similar for different impact. From Fig. 5.6 it can be seen that the change trend of CT_t for the three types of r_a is similar.

Table 5.4 present the results of problems with different prescribed travel duration p_k. Parameter r_p is changed to generate different values of p_k. It can be found from Table 5.4 that CT_t is less than CT_0 for all the problem sets 45–64 except for set 44. The average values of CT_t/CT_0 are 70.83%, 61.27%, and 67.19% for $r_p = 0.65$, 0.70, and 0.75, respectively. It can be seen from Fig. 5.7 that the change trend of CT_t for the three types of r_p is similar. Both CT_t and CT_0 increase quickly when problem size increases from 70 nodes and 15 tasks to 80 nodes and 20 tasks.

The numerical experiment results shown that the proposed algorithm can optimally solve all the tested problem sets. The proposed algorithm can find an optimal solution of the problem faster than CPLEX for 62 out of 64 problem sets. However, due to the introduction of the dynamic link travel time to the LRP-TT, the computational time increases sharply with the size of the problem.

The corresponding work in this subsection has been accepted to be published in the following paper.

Y. Fang, F. Chu, S. Mammar, and A. Che. A cut-and-solve based algorithm for optimal lane reservation with dynamic link travel times. *International Journal of Production Research* (accepted).

Table 5.2: Computational results of problems with different number of time intervals $|Q|$.

Set	$\|N\|$	$\|K\|$	$\|Q\|$	CT_t	CT_0	$CT_t/CT_0(\%)$
9	65	5	4	1.32	4.61	28.70
10	65	10	4	10.70	20.91	51.15
11	70	10	4	32.61	33.36	97.78
12	70	15	4	44.93	65.69	68.40
13	75	15	4	81.62	86.56	94.29
14	75	20	4	108.46	129.39	83.82
15	80	20	4	89.67	200.56	44.71
Average				52.76	77.30	68.25
16	65	5	6	4.78	6.33	75.57
17	65	10	6	14.52	23.04	63.00
18	70	10	6	13.13	52.85	24.85
19	70	15	6	65.54	92.05	70.86
20	75	15	6	85.03	161.91	52.52
21	75	20	6	82.75	242.17	34.17
22	80	20	6	424.30	474.75	89.37
Average				98.58	150.51	65.50

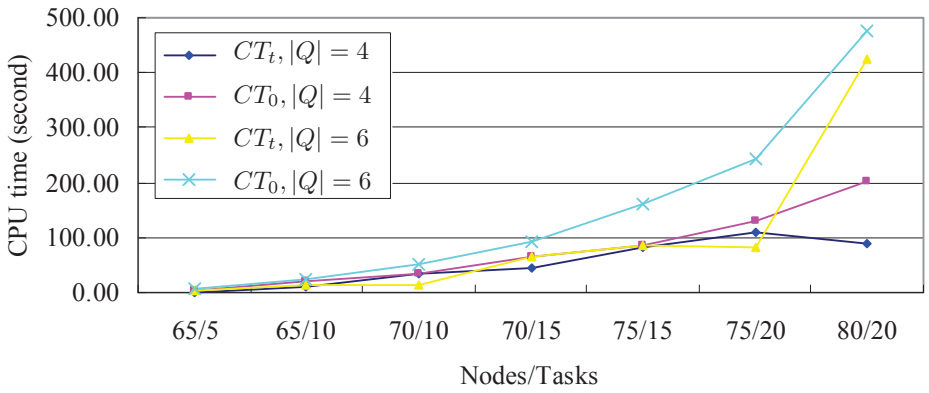


Fig. 5.5: Computational results of problems with different number of time intervals.

Table 5.3: Computational results of problems with different impact of reserved lanes $a_{ij} = r_a \overline{\tau}''_{ij}$, $\overline{\tau}''_{ij}$ is the average travel time on a non-reserved lane in (i, j) through the whole time horizon.

Set	$\|N\|$	$\|K\|$	$\|Q\|$	r_a	CT_t	CT_0	$CT_t/CT_0(\%)$
23	50	5	4	[0.1, 0.2]	0.95	0.92	102.71
24	50	10	4	[0.1, 0.2]	2.03	2.62	77.27
25	60	10	4	[0.1, 0.2]	4.76	5.65	84.28
26	60	15	4	[0.1, 0.2]	17.42	31.18	55.85
27	70	15	4	[0.1, 0.2]	15.60	22.24	70.17
28	70	20	4	[0.1, 0.2]	51.96	98.09	52.97
29	80	20	4	[0.1, 0.2]	96.67	117.68	82.14
Average					27.06	39.77	68.03
30	50	5	4	[0.2, 0.3]	0.81	1.00	81.50
31	50	10	4	[0.2, 0.3]	2.54	3.02	83.99
32	60	10	4	[0.2, 0.3]	5.68	7.28	78.03
33	60	15	4	[0.2, 0.3]	26.58	33.34	79.74
34	70	15	4	[0.2, 0.3]	32.04	53.59	59.78
35	70	20	4	[0.2, 0.3]	73.19	100.37	72.92
36	80	20	4	[0.2, 0.3]	91.01	122.93	74.03
Average					33.12	45.93	72.11
37	50	5	4	[0.3, 0.4]	0.79	0.94	84.05
38	50	10	4	[0.3, 0.4]	2.93	3.89	75.34
39	60	10	4	[0.3, 0.4]	7.67	8.07	94.95
40	60	15	4	[0.3, 0.4]	16.95	31.95	53.06
41	70	15	4	[0.3, 0.4]	32.73	60.12	54.45
42	70	20	4	[0.3, 0.4]	72.12	90.33	79.83
43	80	20	4	[0.3, 0.4]	98.20	115.52	85.01
Average					33.06	44.40	74.44

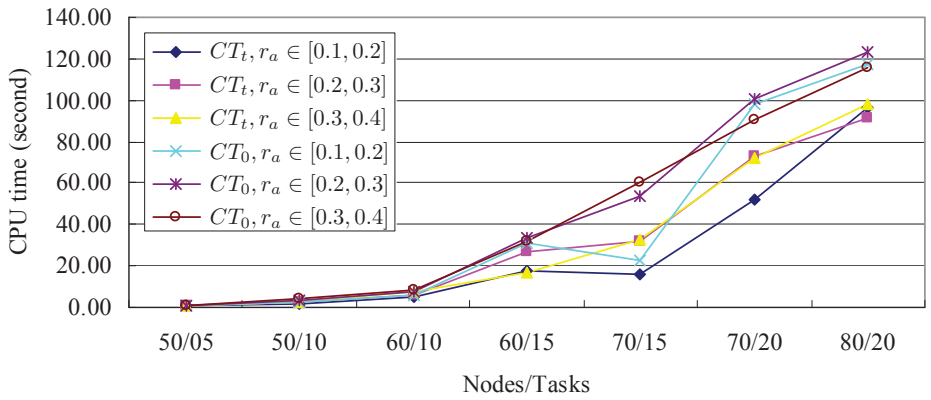


Fig. 5.6: Computational results of problems with different impact.

Table 5.4: Computational results of problems with different prescribed travel duration $p_k = dis(s_k, d_k) + r_p(dis'(s_k, d_k) - dis(s_k, d_k))$, $dis(s_k, d_k)$ (resp. $dis'(s_k, d_k)$) is the shortest travel duration from s_k to d_k in the case of the travel time on (i, j) is τ_{ij} (resp. $\overline{\tau}''_{ij}$).

Set	$\|N\|$	$\|K\|$	$\|Q\|$	r_p	CT_t	CT_0	$CT_t/CT_0(\%)$
44	50	5	4	0.65	0.26	0.22	120.01
45	50	10	4	0.65	2.22	3.61	61.61
46	60	10	4	0.65	7.62	10.76	70.79
47	60	15	4	0.65	9.77	16.16	60.48
48	70	15	4	0.65	18.83	35.16	53.56
49	70	20	4	0.65	44.52	53.81	82.73
50	80	20	4	0.65	68.64	94.71	72.48
Average					21.70	30.63	70.83
51	50	5	4	0.70	0.18	0.19	98.17
52	50	10	4	0.70	1.67	2.37	70.58
53	60	10	4	0.70	7.00	9.93	70.46
54	60	15	4	0.70	12.79	26.28	48.65
55	70	15	4	0.70	16.33	67.13	24.32
56	70	20	4	0.70	55.49	94.59	58.67
57	80	20	4	0.70	126.73	158.91	79.75
Average					31.46	51.34	61.27
58	50	5	4	0.75	0.26	0.33	77.57
59	50	10	4	0.75	2.48	3.14	78.93
60	60	10	4	0.75	6.27	8.65	72.56
61	60	15	4	0.75	20.47	27.96	73.22
62	70	15	4	0.75	25.66	45.60	56.27
63	70	20	4	0.75	62.88	97.00	64.83
64	80	20	4	0.75	70.58	98.00	72.02
Average					26.94	40.10	67.19

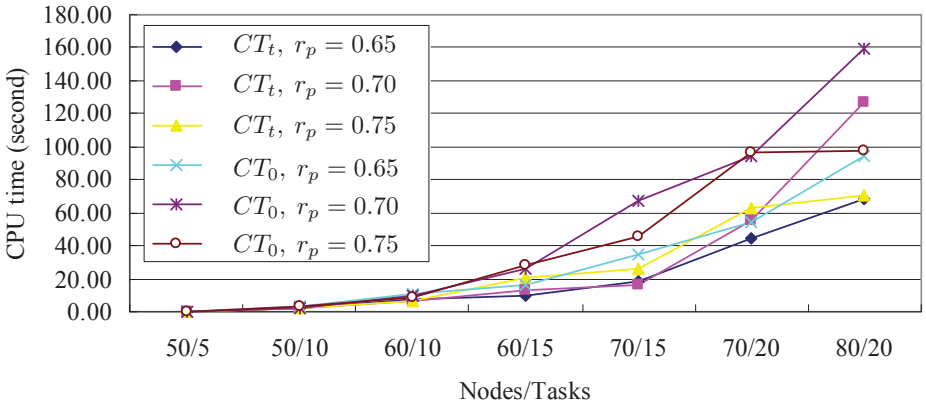


Fig. 5.7: Computational results of problems with different prescribed travel duration.

5.3 Lane reservation problem with time-dependent travel speed

5.3.1 Time-dependent travel speed and travel time on non-reserved lanes

In this section, a dynamic lane reservation problem with time-dependent travel speed (LRP-TS) is investigated. The LRP-TS in this section and the LRR-TT in section 5.2 have some common points. In both problems, the travel time on a reserved lane is a constant, while the travel time on a non-reserved lane is dynamically changed. Both problems have the same objective of minimizing the total impact of reserved lanes and a mixed integer nonlinear programming model can be formulated for each of them. However, the two problems have differences. In the LRP-TT, the travel time on a non-reserved lane $(i, j) \in A$ is only dependent on the departure time at node i [23]. For a given non-reserved lane (i, j) and given departure time at node i, the travel time on (i, j) is a constant. But the "first-in-first-out" (FIFO) property, which is common sense as pointed by Ichoua et al. [40], is not satisfied in the LRP-TT. The FIFO property [40] guarantees that if a vehicle leaves a node i for a node j at a given time, any identical vehicle leaving node i for node j at a later time will arrive later at node j. In the LRP-TS, due to the assumption of the time-dependent travel speed, the FIFO property is satisfied. In addition, the travel time on each non-reserved lane $(i, j) \in A$, which can be calculated based on the time-dependent travel speed, is a piecewise linear continuous function of the departure time at node i. The time-dependent travel speed was firstly introduced to the time-dependent VRP [40]. Details of the time-dependent travel speed are explained as follows.

Similar to section 5.2, the time horizon can be divided as $[T_0,\ T_1,\ \ldots,\ T_{nq-1},\ T_{nq})$, where nq is the number of time intervals, and for each $q \in Q = \{0,\ \ldots,\ nq-1\}$, $[T_q,\ T_{q+1})$ represents a time interval. T_0 and T_{nq} represent the beginning time and ending time of the time horizon, and all the tasks should be started and completed within this time horizon. For each non-reserved lane (i, j), the travel speed on (i, j) at a time t is dependent on the time interval in which t is. The travel speed is assumed a constant for each time interval $[T_q, T_{q+1}), q \in Q$, but it can be changed if the travel duration includes at least two consecutive time intervals. Therefore, the travel time on a non-reserved lane (i, j) is depend on the departure time at node i and travel speeds in related time intervals. An example is given in Fig. 5.8. Suppose that the whole time horizon is divided as $[T_0, T_1, T_2, T_3, T_4) = [0, 1, 2, 3, 4)$. The length of (i, j) is denoted by $D_{ij} = 2$ and the travel speeds on (i, j) are $v_{ij0} = 1, v_{ij1} = 2, v_{ij2} = 1$

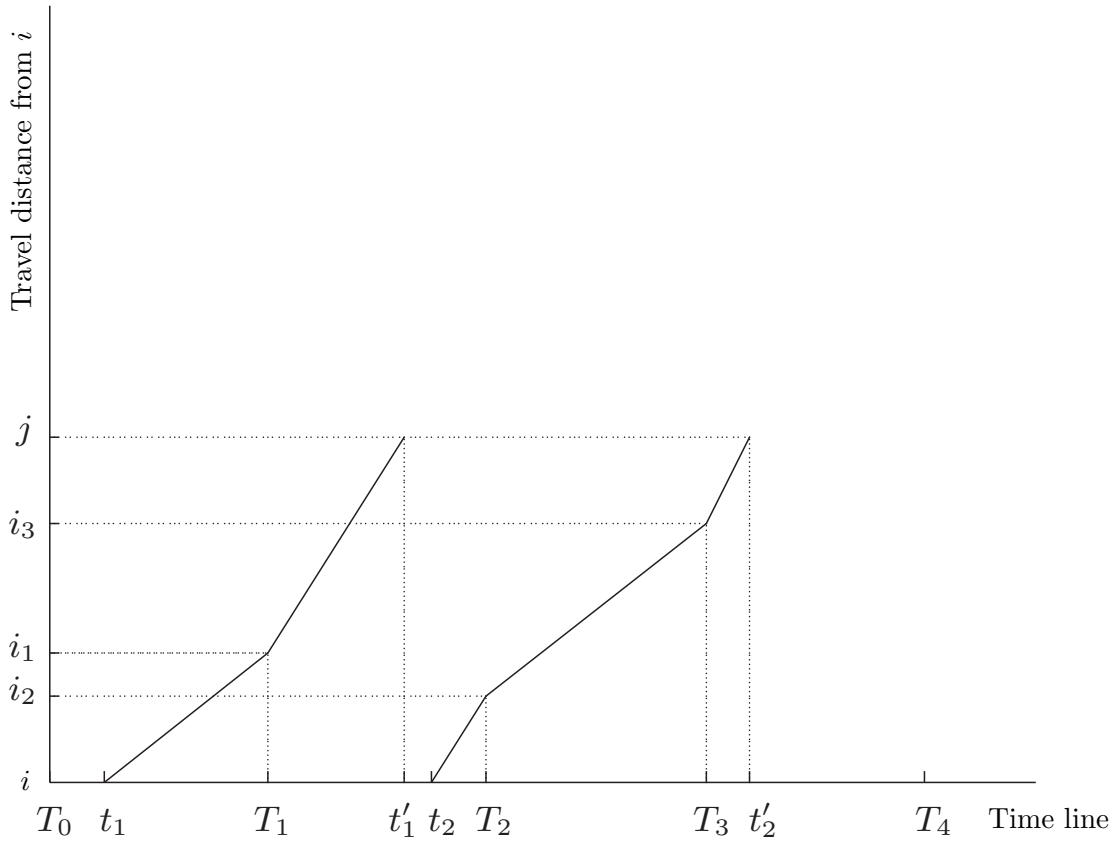


Fig. 5.8: Illustration of changes of travel speed on a non-reserved lane (i, j). Time horizon is divided as $[T_0, T_1, T_2, T_3, T_4) = [0, 1, 2, 3, 4)$. If one vehicle departs i at time $t_1 = 0.25$, it arrives j at time $t_1' = 1.625$, and the travel time on (i, j) is 1.375; if another vehicle departs i at time $t_2 = 1.75$, it arrives j at time $t_2' = 3.2$, and the travel time on (i, j) is 1.45.

and $v_{ij3} = 2.5$ for each time interval, respectively. If a vehicle departs node i at time $t_1 = 0.25 \in [T_0, T_1)$, it travels on (i, j) with a speed of v_{ij0} until it arrives point i_1 at time T_1, which is the boundary of time interval $[T_0, T_1)$. The traveled distance between i and i_1 is 0.75 $(= v_{ij0} \times (T_1 - t_1))$ and the distance between i_1 and j is 1.25 $(= D_{ij} - v_{ij0} \times (T_1 - t_1))$. Then, the vehicle continues traveling on (i, j) with a speed of v_{ij1} until it arrives point j at time t_1' $(= 1.625 = T_1 + (D_{ij} - v_{ij0} \times (T_1 - t_1))/v_{ij1})$. The travel time on (i, j) is 1.375 $(= t_1' - t_1)$. For the same example, if another vehicle departs node i at time $t_2 = 1.75 \in [T_1, T_2)$, it travels on (i, j) with a speed of v_{ij1} until it arrives point i_2 at time T_2. The traveled distance between i and i_2 is 0.5 $(= v_{ij1} \times (T_2 - t_2))$ and the distance between i_2 and j is 1.75. Then, it continues traveling on (i, j) with a speed of v_{ij2} until it arrives point i_3 at time T_3. The traveled distance between i and i_3 is 1.5 $(= 0.5 + v_{ij2} \times (T_3 - T_2))$ and the distance between

1: Input: departure time t_0 at node i and $t_0 \in [T_q, T_{q+1})$, length D_{ij} of lane (i,j). Denote v_{ijq} as the travel speed during time interval $[T_q, T_{q+1}, q \in Q)$.
2: Set $t := t_0$, $D := D_{ij}$, and $t' := t + D/v_{ijq}$.
3: **while** $(t' > T_{q+1})$ **do**
4: Set $D := D - v_{ijq}(T_{q+1} - t)$,
5: Set $t := T_{q+1}$,
6: Set $t' := t + D/v_{ij(q+1)}$,
7: Set $q := q + 1$.
8: **end while**
9: Return $t' - t_0$.

Fig. 5.9: Procedure of calculating travel time on a non-reserved lane (i,j) for a given departure time at node i.

i_3 and j is 0.5. It continues traveling on (i,j) with a speed of v_{ij3} until it arrives j at time t_2' $(= 3.2 = T_3 + (D_{ij} - v_{ij1} \times (T_2 - t_2) - v_{ij2} \times (T_3 - T_2))/v_{ij3})$. The travel time on (i,j) is 1.45 $(= t_2' - t_2)$. The above two examples shown that different departure time at node i will result different link travel time on the non-reserved lane (i,j). For any non-reserved lane (i,j) in the network and a given departure time t_0 at node i, the travel time on (i,j) can be calculated by the procedure given by Ichoua et al. [40]. It is presented in Fig. 5.9. For details of this procedure, readers could refer to [40]. Moreover, it is pointed out in [40] that the travel time is a piecewise linear continuous function of the departure time at node i. Nevertheless, the authors did not explain how to obtain this piecewise linear function. In the following, we will explain how to obtain the piecewise linear function.

For each non-reserved lane (i,j) in the network for the LRP-TS, the corresponding travel time $\tau_{ij}^*(t)$ is a piecewise linear function of the departure time t at node i. The *breakpoints* $(t_m, \tau_{ij}^*(t_m)),\ m = 1, \ldots, np$, are some special points of the travel time function $\tau_{ij}^*(t)$, where t_m is the departure time at node i, $\tau_{ij}^*(t_m)$ is the corresponding travel time on (i,j), and np is the number of the breakpoints. They are defined as the points where the slope of the function changes. If the breakpoints of the piecewise linear function are known, then the function can be obtained by connecting these breakpoints. To find the breakpoints, a theorem will be presented as follows.

Recall the time horizon is divided into nq time intervals as $[T_0,\ T_1,\ \ldots,\ T_{nq})$ for the LRP-TS, where $T_q, q \in Q = \{0,\ 1, \ldots,\ nq\}$, is the boundary of the time interval. The travel time function $\tau_{ij}^*(t)$ corresponding to the non-reserved lane (i,j) in the

network is a piecewise linear function of the departure time t at node i. All the breakpoints $(t_m, \tau_{ij}^*(t_m))$, $m = 1, \ldots, np \leq 2nq$ have the following characteristics.

Theorem 7 *For each breakpoint* $(t_m, \tau_{ij}^*(t_m))$, $m = 1, \ldots, np \leq 2nq$, *of the travel time function on the non-reserved lane* (i, j), *one of the following two cases is satisfied: (1), the departure time* t_m *at node* i *is the boundary of certain time interval; or (2), the arrived time* $t_m + \tau_{ij}^*(t_m)$ *at node* j *is the boundary of certain time interval.*

The proof of this theorem is given in Appendix I. It tells us that the breakpoints of the travel time function have some special characteristics. Then we can find these breakpoints according to these characteristics. For the breakpoints corresponding to first case, the departure times t_m at i are the boundaries of the time intervals, and the travel time $\tau_{ij}^*(t_m)$ can be calculated by the procedure in Fig. 5.9. For the breakpoints corresponding to second case, the arrived times $t_m + \tau_{ij}^*(t_m)$ at node j are the boundaries of the time intervals, similar to the procedure in Fig. 5.9, the departure times t_m can be obtained, then the travel time $\tau_{ij}^*(t_m)$ can also be obtained.

To illustrate how to find the breakpoints, an example is given as follows. The considered time horizon is supposed as $[0, 4)$ and all the tasks must be started and completed within the time horizon. It is divided as $[T_0, T_1, T_2, T_3, T_4) = [0, 1, 2, 3, 4)$. The travel speed on a non-reserved lane (i, j) of length 2 for each time interval are $[v_{ij0}, v_{ij1}, v_{ij2}, v_{ij3}] = [1, 2, 1, 2.5]$. For the breakpoints corresponding to first case, the departure times $t_m, m = 1, 2, 3, 4$ are $T_0 = 0$, $T_1 = 1$, $T_2 = 2$, and $T_3 = 3$, the travel time on (i, j) can be calculated by the procedure in Fig. 5.9 and $\tau_{ij}^*(t_1) = 1.5$, $\tau_{ij}^*(t_2) = 1$, $\tau_{ij}^*(t_3) = 1.4$, and $\tau_{ij}^*(t_4) = 0.8$. Then the breakpoints $(t_m, \tau_{ij}^*(t_m)), m = 1, 2, 3, 4$ corresponding to first case are obtained. Then, we consider the breakpoints corresponding to second case, whose arrived time at j are respective $T_2 = 2$, $T_3 = 3$, and $T_4 = 4$ ($T_0 = 0$ and $T_1 = 1$ are not considered because the earliest departure time at i is $T_0 = 0$ and the travel time is 1.5, then the earliest arrived time at j is 1.5). Similar to the procedure in Fig. 5.9, we can know that for the departure times at node i are $t_5 = 1$, $t_6 = 1.5$, $t_7 = 3.2$, then the travel times on (i, j) are $\tau_{ij}^*(t_5) = 1$, $\tau_{ij}^*(t_6) = 1.5$, $\tau_{ij}^*(t_7) = 0.8$, and the arrived time at j are exactly T_2, T_3, and T_4. Then the breakpoints corresponding to second case are obtained. Now all the breakpoints $(t_m, \tau_{ij}^*(t_m)), m = 1, \ldots, 7$ are obtained.

To obtain the travel time function, the breakpoints are depicted in the Fig. 5.10 and are connected from left to right, then the travel time function is obtained. It can

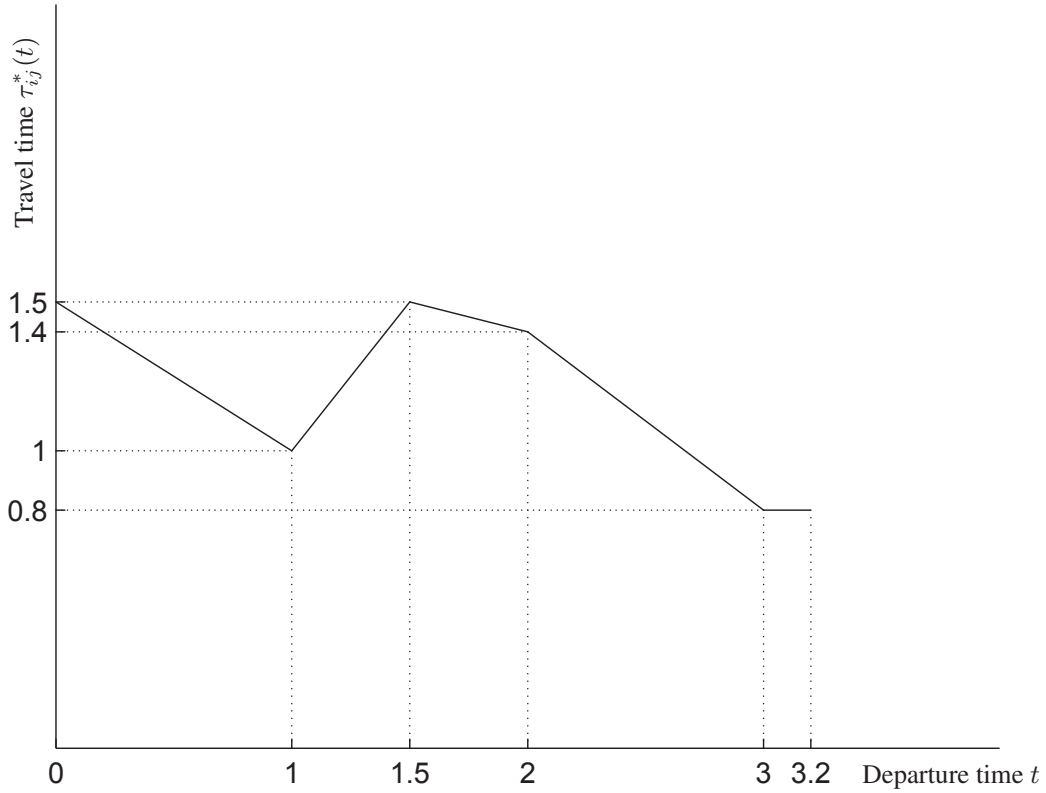


Fig. 5.10: An example of travel time function of a non-reserved lane (i, j) with length of 2. Time horizon is divided as $[T_0, T_1, T_2, T_3, T_4) = [0, 1, 2, 3, 4)$ and the travel speeds for each time interval are $[v_{ij0}, v_{ij1}, v_{ij2}, v_{ij3}] = [1,\ 2,\ 1,\ 2.5]$.

be represented as:

$$\tau^*_{ij}(t) = \begin{cases} -0.5t + 1.5, & 0 \le t < 1 \\ t, & 1 \le t < 1.5 \\ -0.2t + 1.8, & 1.5 \le t < 2 \\ -0.6t + 2.6, & 2 \le t < 3 \\ 0.8, & 3 \le t \le 3.2 \end{cases} \tag{5.27}$$

For different non-reserved lane, the corresponding travel time function is different. Thus, the travel time function on non-reserved lane (i, j) is denoted as $\tau^*_{ij}(t)$ in the problem formulation in the next section.

5.3.2 Problem description and formulation

The LRP-TS is described as follows. Given a network $G(N, A)$, a set of tasks and corresponding source-destination pairs, The LRP-TS is to set reserved lanes from the network and to design paths for the time-guaranteed tasks, with the objective of minimizing the total impact of reserved lanes on the network. The travel time on a reserved lane is assumed a constant because it can be only used by the tasks' vehicles

and the travel speed on it is smooth. But the travel time on a non-reserved lane (i, j) is a piecewise linear function of the departure time at node i as the example in Fig. 5.10. Because it is assumed that the travel speed on a non-reserved lane (i, j) can be changed for different time intervals. Therefore, different departure time at i will result in different travel time on (i, j). For each non-reserved lane, if its length and the travel speeds for each time interval are known, the travel time function can be calculated based on the procedure in Fig. 5.9. To formulate the LRP-TS, the travel time function for each $(i, j) \in A$ are pre-calculated and are known parameter in the LRP-TS.

The notations of the LRP-TS are given as follows.

Sets and input parameters

A: set of directed arcs $(i, j), i \neq j, i, j \in N$
K: set of transportation tasks, $k \in K$
N: set of nodes
a_{ij}: traffic impact if a lane in $(i, j) \in A$ is reserved
d_k: destination node of task $k \in K$
p_k: prescribed travel duration to complete task $k \in K$
s_k: source node of task $k \in K$
st_k: starting time for task k at node s_k, $k \in K$
τ_{ij}: travel time on a reserved lane in $(i, j) \in A$
$\tau^*_{ij}(t)$: link travel time function on a non-reserved lane in $(i, j) \in A$ when the task departs at node i at time t

Decision variables

t_{ki}: departure time of task k at node j; $t_{kj} = 0$ if task k does not visit node j, $\forall k \in K, \forall i \in N$
x_{kij}: $x_{kij} = 1$, if a lane in link (i, j) is in the path of task k and this lane is reserved; and otherwise $x_{kij} = 0$, $\forall k \in K, \forall (i, j) \in A$.
y_{kij}: $y_{kij} = 1$, if a lane in link (i, j) is in the path of task k and this lane is not reserved; and otherwise $y_{kij} = 0$, $\forall k \in K, \forall (i, j) \in A$.
z_{ij}: $z_{ij} = 1$, if there is a reserved lane in link (i, j); and otherwise $z_{ij} = 0$, $\forall (i, j) \in A$

Then the LRP-TS is formulated as the following programming model.

$$P_s: \min \sum_{(i,j)\in A} a_{ij} z_{ij} \tag{5.28}$$

$$\text{s.t.} \sum_{i:(s_k,i)\in A} (x_{ks_ki} + y_{ks_ki}) = 1, \qquad \forall k \in K, \tag{5.29}$$

$$\sum_{i:(i,d_k)\in A} (x_{kid_k} + y_{kid_k}) = 1, \qquad \forall k \in K, \tag{5.30}$$

$$\sum_{i:(i,j)\in A}(x_{kij}+y_{kij})=\sum_{i:(j,i)\in A}(x_{kji}+y_{kji}), \qquad \forall k\in K,\forall j\in N\backslash\{s_k,d_k\}, \quad (5.31)$$

$$x_{kij}\le z_{ij}, \qquad \forall k\in K,\forall (i,j)\in A, \quad (5.32)$$

$$y_{kij}+z_{ij}\le 1, \qquad \forall k\in K,\forall (i,j)\in A, \quad (5.33)$$

$$t_{ks_k}=st_k, \qquad \forall k\in K, \quad (5.34)$$

$$t_{kj}=\sum_{i:(i,j)\in A}\left((t_{ki}+\tau_{ij})x_{kij}+(t_{ki}+\tau^*_{ij}(t_{ki}))y_{kij}\right), \qquad \forall k\in K,\forall j\in N\backslash\{s_k\}, \quad (5.35)$$

$$t_{kd_k}-t_{ks_k}\le p_k, \qquad \forall k\in K, \quad (5.36)$$

$$t_{ki}\ge 0, \qquad \forall k\in K,\forall i\in N, \quad (5.37)$$

$$x_{kij},y_{kij}\in\{0,\ 1\}, \qquad \forall k\in K,\forall (i,\ j)\in A, \quad (5.38)$$

$$z_{ij}\in\{0,\ 1\}, \qquad \forall (i,\ j)\in A. \quad (5.39)$$

The model P_s for LRP-TS is similar to model P_t for LRP-TT in section 5.2.1. To simplify the description, we only explain the differences between them. First, there are four types of decision variables in P_s, whereas there are five types of decision variables in P_t. Because in the LRP-TT, the travel time on a non-reserved lane is dependent on the time interval in which the departure time at node i is. Hence, we need one type of decision decision variables (b_{kiq}) and constraints (5.7)–(5.9) to indicate this information when modeling LRP-TT. Second, the constraints (5.35) in P_s and (5.11) in P_t are different. The item $\tau^*_{ij}(t_{ki})$ in (5.35) represents the travel time on a non-reserved lane (i,j) when the departure time at node i is t_{ki}. The item τ''_{ijq} in (5.11) is a constant and it represents the travel time on a non-reserved lane (i,j) when the departure time at node i is in time interval $[T_q,T_{q+1})$. It can be predictable that model P_s is more difficult to solve than model P_t because the non-constant item $\tau^*_{ij}(t_{ki})$ is involved in model P_s. For details of model P_t, readers are referred to section 5.2.1.

The complexity of the LRP-TS is given by the following theorem.

Theorem 8 *The LRP-TS is NP-hard.*

Proof: If the travel speed on a non-reserved lane $(i,j)\in A$ is unchanged over the whole time horizon, the dynamic link travel time will be reduced to the static link travel time. If the travel speed on each non-reserved lane is very small, then the travel time on it will be greater than the prescribed travel time to complete any task $k\in K$, which implies that each lane in the task paths must be reserved, otherwise the concerned tasks cannot be completed within the prescribed travel time. In this case, the LRP-TS is reduced to the LRP in chapter 3, which has been proved NP-hard. Therefore, the LRP-TS is NP-hard. □

5.3.3 Model reformulation

Note that constraint (5.35) is not linear. It can be reformulated in the same way described in section 5.2.2. To simplify the description, the reformulations of (5.35) are presented as follows. For details of the transformation, readers are referred to section 5.2.2.

1) Task k does not visit node j, i.e., $\sum_{i:(i,j)\in A} x_{kij} = 0$, and $\sum_{i:(i,j)\in A} y_{kij} = 0$. Then (5.35) can be reformulated as follows:

$$t_{kj} \leq M \sum_{i:(i,j)\in A} (x_{kij} + y_{kij}), \qquad \forall k \in K, \forall j \in N\backslash\{s_k\}, \tag{5.40}$$

where M is a large constant.

2) Task k visits node j via a reserved lane in (i,j), i.e., $\sum_{i:(i,j)\in A} x_{kij} = 1$, and $\sum_{i:(i,j)\in A} y_{kij} = 0$. Then (5.35) can be reformulated as follows:

$$\begin{array}{l} t_{kj} - t_{ki} - \tau_{ij} \leq M(1 - x_{kij}), \\ t_{kj} - t_{ki} - \tau_{ij} \geq M(x_{kij} - 1), \end{array} \qquad \forall k \in K, \forall j \in N\backslash\{s_k\}, (i,j) \in A. \tag{5.41}$$

3) Task k visits node j via a non-reserved lane in (i,j), i.e., $\sum_{i:(i,j)\in A} x_{kij} = 0$, and $\sum_{i:(i,j)\in A} y_{kij} = 1$. Then (5.35) can be reformulated as follows:

$$\begin{array}{l} t_{kj} - t_{ki} - \tau^*_{ij}(t_{ki}) \leq M(1 - y_{kij}), \\ t_{kj} - t_{ki} - \tau^*_{ij}(t_{ki}) \geq M(y_{kij} - 1), \end{array} \qquad \forall k \in K, \forall j \in N\backslash\{s_k\}, (i,j) \in A. \tag{5.42}$$

Then P_s can be reformulated as an equivalent model as follows:

$$\begin{array}{ll} P'_s: & \min \sum_{(i,j)\in A} a_{ij} z_{ij} \\ \text{s.t.} & \text{constraints (5.29)} - \text{(5.34) and (5.36)} - \text{(5.42)}. \end{array}$$

Note that P'_s is not a linear model since item $\tau^*_t(t_{ki})$ in (5.42) is not linear.

5.3.4 Tabu search algorithm

Tabu search (TS) proposed by Golver, is a local serach-based metaheuristic. It has been successfully implemented for some transportation problems with dynamic factors [12] [32] [40]. Generally speaking, tabu search is an iterative search strategy that starts from an *initial solution*. At each iteration, a *neighborhood* is generated around the current solution. The neighborhood is a subset of feasible solutions which can

1: Input network (N, A), set of reserved lanes RA, source node s, and destination node d. Denote LT_{ij} as the travel time on lane $(i, j) \in A$. Let $l(s, j), j \in N$ denote the shortest travel time from s to j, and $pred(j)$ denote the precedent node of j in the shortest path from s to j.
2: Set $l(s, s) := 0$ and $l(s, j) := +\infty$ for each $j \in N \setminus \{s\}$. Set $pred(s) := \emptyset$.
3: **while** (some arc $(i, j) \in A$ satisfies $l(s, j) > l(s, i) + LT_{ij}$) **do**
4: Set $l(s, j) := l(s, i) + LT_{ij}$,
5: Set $pred(j) := i$.
6: **end while**
7: Return $l(s, d)$ as the shortest travel time from s to d.

Note: If (i, j) is reserved, LT_{ij} is of value τ_{ij}; otherwise it is of value $\tau^*_{ij}(l(s, i))$, which $l(s, i)$ is the departure time at node i.

Fig. 5.11: Modified label-correcting algorithm for shortest path in a network with dynamic link travel time.

be reached by an operation called *move* from the current solution. Then the best admissible solution in the neighborhood becomes current solution. To avoid the possibility of cycling in the search, a *tabu list* is created to record recently visited solutions and forbid moving to them for a number of iterations (called *tabu tenure*). However, forbidden moves can be overridden when *aspiration criteria* are satisfied. Finally, tabu search is stopped when some termination rules are met.

As mentioned in section 5.3.1, one characteristic of the LRP-TS is that the FIFO property is satisfied. Moreover, it was shown that the label-correcting algorithm for shortest path can be applied to networks with FIFO property [45]. Thus, a modified label-correcting algorithm is proposed to find shortest path in a network with dynamic link travel time. It will be extensively used to generate initial solution and evaluate the feasibility of moves in the TS algorithm. In the remainder of this section, the modified label-correcting algorithm is firstly described. Then the initial solution, neighborhood move, tabu list, aspiration criteria, and termination rules for the TS algorithm are addressed, respectively.

Modified label-correcting algorithm

The modified label-correcting algorithm is presented in Fig. 5.11. It is based on the algorithm for networks with static link travel time [4]. The only difference between two algorithm is the travel time, denoted as LT_{ij}, on lane (i, j). In [4], the algorithm deals with static networks and LT_{ij} is a constant for each lane $(i, j) \in A$. Whereas

in the modified algorithm presented in Fig. 5.11, the value of LT_{ij} is dependent on the reservation status of the lane (i, j). If it is reserved, then LT_{ij} is a constant of value τ_{ij}; otherwise, LT_{ij} is dependent on the departure time $l(s, i)$ at node i and it is of value $\tau^*_{ij}(l(s, i))$. Then, given a network (N, A) with FIFO property and a set of reserved lanes (denoted as RA), the modified label-correcting algorithm in Fig. 5.11 can find a shortest path for a given source-destination pair. This algorithm will be used to generate initial solution and check the feasible of moves in the TS algorithm.

Initial solution

The initial feasible solution for TS is constructed by a greed heuristic. The idea of the greed heuristic is to set reserved lanes one by one until a time-guaranteed path is found for each task. To reduce the computational effort, we firstly compute a set of candidate reserved lanes for each task as follows.

For $\forall(i, j) \in A$, let $l(s_k, i)$ denote the shortest travel duration from s_k to i in an exclusively reserved path and $l(j, d_k)$ denote the shortest travel duration from j to d_k in an exclusively reserved path. For $\forall k \in K$, define set A_k as follows:

$$A_k = \{\ (i, j) \in A \mid l(s_k, i) + \tau_{ij} + l(j, d_k) > p_k\ \}. \tag{5.43}$$

The lane (i, j) in A_k implies that if it is in the path of task k, the travel time constraint will be violated. Thus, to ensure the time-guaranteed tasks, it will not be selected for the path of task k. Then the set of candidate reserved lanes for task $k \in K$ is defined as $\overline{A}_k = A \setminus A_k$ and we only consider the lanes from it to be reserved for taks k. In the implementation, the lanes in $\overline{A}_k$ are ordered in decreasing impact a_{ij} and are selected to be reserved one by one until a time-guaranteed path is found for each $k \in K$. The overall procedure for constructing initial solution presented in Fig. 5.12 is described as follows.

For each task $k \in K$, its set of reserved lanes RA_k is initialized as $\emptyset$ and the set of candidate reserved lanes $\overline{A}_k$ is defined. Denote the current shortest travel duration of task k from s_k to d_k as L_k, which is initialized as $+\infty$. At each iteration, we select the lane $(\bar{i}, \bar{j})$ with minimal impact from $\overline{A}_k$ and delete it from $\overline{A}_k$. Then we compute the shortest travel duration L'_k from s_k to d_k by the modified label-correcting algorithm in Fig. 5.11. If $L'_k < L_k$, we updated $L_k = L'_k$ and add lane $(\bar{i}, \bar{j})$ to set RA_k; otherwise it means that the reservation of lane $(\bar{i}, \bar{j})$ does not help reduce the travel duration, and we do not add it to RA_k. The iteration for task k will termination if $L_k \leq p_k$, which means that a time-guaranteed path is found for task k. If a time-guaranteed path

1: **for** ($k = 1$ to $|K|$) **do**
2: Initialize $RA_k \coloneqq \emptyset$ and $L_k = +\infty$. Define A_k by (5.43) and $\overline{A}_k = A \setminus A_k$.
3: **while** ($L_k > p_k$) **do**
4: Select the lane $(\bar{i}, \bar{j})$ with minimal impact of reserved lane from $\overline{A}_k$. Then set $\overline{A}_k \coloneqq \overline{A}_k \setminus \{(\bar{i}, \bar{j})$.
5: Reserve the lanes in RA_k, as well lane $(\bar{i}, \bar{j})$.
6: Compute the shortest travel duration L'_k from s_k to d_k by the modified label-correcting algorithm in Fig. 5.11.
7: If $L'_k < L_k$, update $L_k \coloneqq L'_k$ and $RA_k \coloneqq RA_k \bigcup \{(\bar{i}, \bar{j})\}$.
8: **end while**
9: **end for**
10: Return the initial solution.

Fig. 5.12: Procedure for constructing initial solution.

is found for each task $k \in K$, the procedure for constructing initial solution is stopped.

Neighborhood and move

A *neighborhood* is defined as a subset of feasible solutions which can be reached from the current solution by an operation called *move*. In the implement, moves are defined as changing the reserving status of lanes and three types of moves are considered. For each move, the set of reserved lanes RA is updated and the move is evaluated by $\sum_{(\bar{i},\bar{j})\in RA} a_{ij}$. To reduce the computational effort, instead of checking the feasibility of each move in the neighborhood, only some candidate moves are checked. Details of the three types of moves and strategy for checking the feasibility of moves are described as follows.

The first type of move (add move) is to add a new reserved lane (i', j') to the network, i.e., $RA = RA \cup \{(i', j')\},\ (i', j') \notin RA$. Each move of this type is feasible because adding a reserved lane in the network will not increase the shortest travel duration for any task $k \in K$.

The second type of move (delete move) is to reset a previously reserved lane $(\bar{i}, \bar{j})$ as non-reserved, i.e., $RA = RA \setminus \{(\bar{i}, \bar{j})\},\ (\bar{i}, \bar{j}) \in RA$. To reduce the computational effort for checking the feasibility of the move, a subset K' of tasks is defined as follows:

$$K' = \{\ k \mid (\bar{i}, \bar{j}) \in RA_k,\ k \in K\ \}, \tag{5.44}$$

where RA_k is the set of reserved lanes in the path of task $k \in K$. Resetting a previously reserved lane $(\bar{i}, \bar{j})$ as non-reserved may cause the path of task k totally

1: Input the move and its type. Let (i', j') be the lane involves in the first type move, $(\bar{i}, \bar{j})$ be the lane involves in the second type move, and $(\bar{i}, \bar{j})$ and (i', j') be the lanes involves in the third type move. Let RA denote the set of reserved lanes and RA_k denote the set of reserved lane in the path of task $k \in K$.
2: If the move is of the first type, it is feasible and apply the modified label-correcting algorithm in Fig. 5.11 to find the shortest path for each task $k \in K$. Goto step 6.
3: If the move is of second type, update $RA = RA \setminus \{(\bar{i}, \bar{j})\}$; otherwise update $RA = RA \cup \{(i', j')\} \setminus \{(\bar{i}, \bar{j})\}$.
4: Define K' by (5.44): $K' = \{\ k \mid (\bar{i}, \bar{j}) \in RA_k,\ k \in K\ \}$.
5: For each $k \in K'$, find the shortest travel duration L_k for task k by the modified label-correcting algorithm. For each $k \in K'$, if $L_k \leq p_k$, the move is feasible; otherwise, the move is not feasible.
6: If the move is feasible, return the corresponding task paths.

Fig. 5.13: Procedure for checking the feasibility of move.

changed and its shortest travel duration L_k should be recomputed by the modified label-correcting algorithm. For each $k \in K'$, if $L_k \leq p_k$, the move is feasible; otherwise it is not feasible.

The third type of move (swap move) is to add a new reserved lane (i', j') and reset a previously reserved lane $(\bar{i}, \bar{j})$ as non-reserved, i.e., $RA = RA \cup \{(i', j')\} \setminus \{(\bar{i}, \bar{j})\}$, $(i', j') \notin RA$, $(\bar{i}, \bar{j}) \in RA$. The check for the feasibility of the move is the same as that for the second type of move.

It is easy to see that the number of the third moves is enormous and it is very time consuming to check the feasibility of all the moves in the neighborhood. To reduce the computational effort, the search strategy is implemented as follows. The values of all moves of three types are firstly calculated by $\sum_{(i,j)\in RA} a_{ij}$ and the moves are ordered in decreasing value. Then we selected the move as candidate move with minimal value orderly until it is feasible and non-tabued, or results a better feasible solution than the current best solution.

Tabu list and tabu tenure

To avoid trapping in local minimum, a *tabu list* is constructed to record the information which is used to forbid the search from returning back to previously visited solutions. As a move is performed in the TS, the lane(s) associated with the move become(s) tabu in the following nTS iterations by being added to the tabu list. For

the move of first type, the lane (i', j') introduced to RA is tabu. For the second type, the lane $(\overline{i}, \overline{j})$ removed from RA is tabu. For the third type, both (i', j') and $(\overline{i}, \overline{j})$ are tabu. The value of tabu tenure $nbTS$ is randomly generated in the range nbL and $2nbL$, where nbL is the average number of lanes in the task paths in the initial solution.

Aspiration Criteria

Aspiration criterion is a strategy that can override tabu moves which may lead to an unexplore search space. In our implementation, we employ a simple aspiration criterion which removes a move from the tabu list when it results in a new solution better than the current best solution. Then the current best solution is updated as the new solution.

Stopping criteria

The preliminary experiments shown that the current best solution usually can be found within 50 iteration. The total number of iterations of the proposed TS algorithm is set as 100. The algorithm will also be terminated if the current best solution is not improved for consecutive 50 iterations.

Overall algorithm

The proposed TS algorithm is presented in Fig. 5.14. Steps 1 and 2 are to obtain an initial solution and initialize parameters. n_1 and n_2 represent the total number of iterations in the TS and number of consecutive iterations in which the best solution is not improved. Steps 5 and 6 are to select the candidate move. Steps 7–11 means that if the candidate move results in a better solution than the current best solution and it is feasible, then the current best solution is updated. Steps 12–21 means that the candidate move does not result in a better solution than the current best solution. If it is not in the tabu list and feasible, the current solution is updated and a new iteration repeats; otherwise we goto back to step 6 and select a new candidate move. The TS algorithm is terminated until the maximal number of iteration is $iterMax = 100$ or the current best solution is not improved for consecutive $iterNoImprove = 50$.

5.3.5 Computational results

In this section, the performance of the proposed TS algorithm was evaluated on randomly generated problem instances. The algorithm was coded in Visual C++ and all the experiments were carried out on a PC with 3.0 GHz CPU and 4.0 GB RAM. Each problem set includes five instances.

1: Apply the procedure presented in Fig. 5.12 to find an initial solution.
2: Set current solution sol_c and current best solution sol_{best} as initial solution. Set current best objective value $val_{best} \coloneqq +\infty$. Set $n_1 \coloneqq 0$ and $n_2 \coloneqq 0$.
3: **while** ($n_1 < iterMax$ and $n_2 < iterNoImprove$) **do**
4: Set $n_1 \coloneqq n_1 + 1$.
5: Generate neighborhood moves. Calculate the values of moves and order the moves in decreasing values. Set $nb \coloneqq 0$.
6: Set $nb \coloneqq nb + 1$ and set candidate move as the move with nb-th minimal value.
7: **if** (value of candidate move $<$ val_{best}) **then**
8: Apply the procedure in Fig. 5.13 to check the feasibility of the move.
9: **if** (candidate move is feasible) **then**
10: Update sol_c and sol_{best} as the solution corresponding to the candidate move. Update val_{best}. Set $n_2 = 0$.
11: **end if**
12: **else**
13: **if** (candidate move is not on the tabu list) **then**
14: Apply the procedure in Fig. 5.13 to check the feasibility of the move.
15: **if** (candidate move is feasible) **then**
16: Update sol_c as the corresponding solution of candidate move. Set $n_2 = n_2 + 1$.
17: **else**
18: Goto step 6.
19: **end if**
20: **else**
21: Goto step 6.
22: **end if**
23: **end if**
24: **end while**
25: Return sol_{best} and val_{bset} as the current best solution and objective value.

Fig. 5.14: Tabu search algorithm for the LRP-TS.

Table 5.5: notations used in the numerical results.

$\|N\|$:	number of nodes in the network
$\|K\|$:	number of tasks
CT_s:	CPU time (second) by TS algorithm
CT_0:	CPU time (second) by CPLEX
$G_U(\%)$:	$G_U(\%) = 100 \times (Z(TS) - Z(CPLEX))/Z(CPLEX)$
$G_L(\%)$:	$G_U(\%) = 100 \times (Z(TS) - Z_{LB}(CPLEX))/Z_{LB}(CPLEX)$

The problem instances are randomly generated in a similar way as described in section 5.2.4. The graph $G(N, A)$ is generated based on the network model proposed by Waxman [81]. The time horizon is set as $[0, 12]$ and is divided into $|Q|$ time intervals. Denote v_r and $v_{ijq}, \forall (i, j) \in A, \forall q \in Q$ as the speed on a reserved lane and on a non-reserved lane, respectively. v_r is set as 60 and v_{ijq} is uniformly generated from $[30, 50]$. Then the link travel time τ_{ij} on a reserved lane is defined as $Length_{ij}/v_r$. The link travel time $\tau^*_{ij}(t)$ on a non-reserved lane is calculated by the procedure presented in Fig. 5.9. The average link travel time on a non-reserved lane through the whole horizon is defined as $\overline{\tau}^*_{ij} = Length_{ij}/(\sum_{q \in Q} v_{ijq}/|Q|)$. Then the impact of reserved lane is set as $a_{ij} = r_a \overline{\tau}^*_{ij}$, where r_a is a given parameter. The prescribed travel duration is set as $p_k = dis(s_k, d_k) + r_p(dis'(s_k, d_k) - dis(s_k, d_k))$, where $dis(s_k, d_k)$ and $dis'(s_k, d_k)$ are the shortest travel duration from s_k to d_k in an exclusively reserved path and in an exclusively non-reserved path. In the default case, $r_a \in [0.2, 0.3]$, $r_p = 0.7$.

The performance of the proposed algorithm is compared with the direct use of CPLEX to the resolution of model P'_t. For small sized problems, we compare the computational time of finding an optimal solution by the proposed TS algorithm and CPLEX. The CPU time for largest size instance by TS is nearly 1800 seconds, then the maximal running time for CPLEX is set as 3600 seconds. For large sized instances, CPLEX cannot find an optimal solution (even a feasible solution) within 3600 seconds. Then we compare the upper bound (denoted as $Z(TS)$) obtained by the TS algorithm and the upper bound (denoted as $Z(CPLEX)$). we also compare the $Z(TS)$ and lower bound (denoted as $Z_{LB}(CPLEX)$) by CPLEX. With the notations given in Table 5.5, computational results are reported in Table 5.6.

In Table 5.6, the proposed TS algorithm can find a feasible solutio for all the problem sets 1–17, whereas CPLEX can find an optimal solution for small sized problem sets 1–7. As the size of the problem increases, the computational time by

Table 5.6: Computational results of problems with different sizes.

Set	$\|N\|$	$\|K\|$	CT_s	CT_0	$G_U(\%)$	$G_L(\%)$
1	25	7	1.62	27.48	0.04	0.04
2	30	7	3.00	55.84	0.52	0.52
3	35	7	3.69	82.77	2.36	2.36
4	40	7	5.84	167.52	0.93	0.93
5	45	7	11.19	189.93	1.84	1.84
6	50	7	14.88	314.48	1.72	1.72
7	55	10	27.78	1058.15	5.42	5.42
8	60	10	62.43	-	2.41	21.44
9	60	15	100.15	-	1.23	30.51
10	70	15	158.76	-	-7.88	49.24
11	70	20	212.27	-	-10.10	38.71
12	80	20	405.16	-	-5.93	33.72
13	80	25	559.03	-	-	52.91
14	90	25	807.38	-	-	56.17
15	90	30	1133.66	-	-	56.42
16	100	30	1435.07	-	-	50.07
17	100	35	1765.37	-	-	54.72

CPLEX increase quickly and it cannot solve the problem sets optimally within 3600 seconds for sets 8–17. CPLEX can only find a feasible solution for sets 8–12 and it cannot find a feasible solution for sets 13–17.

It can be seen that the computational time CT_0 by CPLEX increases much quickly from 27.48 seconds for set 1 to 1058.15 seconds for set 7, whereas the time CT_s by the proposed TS algorithm increases slowly from 1.62 seconds for set 1 to 27.78 seconds for set 7. Note that problem sets 1–7 are solved by CPLEX optimally, the upper bound and lower bound by CPLEX is also the optimal value of the problem. Thus, G_U is equal to G_L for sets 1–7. The gaps $G_U(\%)$ between the TS upper bound and optimal value range from 0.04% to 5.42% for sets 1–7, which shown that the feasible solutions obtained by the TS algorithm are close to the optimal solutions for small sized problems.

For sets 8–12, CPLEX can only find an feasible solution within 3600 seconds. For sets 8 and 9, the feasible solutions found by CPLEX are better than those found by the TS algorithm since the gaps $G_U(\%)$ are positive. However, for sets 10–12, $G_U(\%)$ are negative, which implies that the feasible solutions found by the TS algorithm are better than those found by CPLEX. The gaps G_L between the upper bound by TS algorithm and the lower bound by CPLEX range from 21.44% to 49.24%. It increases quickly, as compared with the results for sets 1–7.

For even larger sized problem sets 13–17, CPLEX cannot find a feasible solution whereas the TS algorithm can. The gaps G_L range from 50.07% to 56.17%, which shown that the differences between upper bound found by the TS algorithm and the lower bound found by CPLEX are large.

It is true that this part of work is not sufficient, due to a restriction of limited research time. The preliminary computational result shown that the proposed algorithm can found near optimal solution for small sized problems. However, as the size of problem increases, the gap between upper bound by the proposed algorithm and the lower bound by CPLEX increases sharply. Therefore, it is necessary to conduct more investigate for this problem. Methods for obtaining better lower bound should be explored in future work. The upper bound can also be improved by some hybrid methods.

5.4 Conclusions

In this chapter, we have studied the lane reservation problem with time-dependent travel time (LTP-TT) and lane reservation problem with time-dependent travel speed

(LTP-TS). In both of the problems, the link travel time on reserved lanes is a constant, but the link travel time on non-reserved lanes is not a constant.

In the LTP-TT, the link travel time on non-reserved lanes is a step function of the departure time at the beginning node of the link. An mixed integer nonlinear programming model was firstly formulated, and it was then transformed into an equivalent linear model. Then a cut-and-solve based algorithm, in which partial integral relaxation strategy and new piercing cut generation strategy were proposed, was developed for the LRP-TT. Computational results shown that the proposed algorithm can solve nearly all the problem sets more quickly than CLPEX. Sensitive experiment on different setting of parameters (impact of reserved lanes and prescribed travel duration of the tasks) shown that the overall performance of the proposed algorithm outperformed CPLEX fro tested problem instances.

In the LTP-TS, the link travel speed on a non-reserved lane is assumed a step function of the time, which means that the vehicles may travel at different speed on it. Then the link travel time can be computed and it is a piecewise linear continuous function based on the assumption of the step function of link travel speed. Moreover, the FIFO property is satisfied in the LRP-TS. The problem was demonstrated NP-hard. A tabu search algorithm was developed to find near-optimal solutions. A modified label-correcting algorithm is extensively used to generate the initial solution and check the feasibility of moves in the proposed TS algorithm. Preliminary computational result shown that the TS algorithm can efficiently find near-optimal solutions with "good" quality for small sized problems. However, for larger sized problems, the gap between upper bound and lower bound is very large. Methods for improving both of them should be explored in the future.

Chapter 6

Conclusions and perspectives

This thesis studied several lane reservation problems which aim to design time-guaranteed task paths via optimally selecting and reserving lanes from a transportation network, with the objective of minimizing the total traffic impact of reserved lanes. For each of the studied problems, mathematical models are formulated, their complexities are demonstrated, and appropriate resolution methods, including exact cut-and-solve method, cut-and-solve and cutting plane combined method, and tabu search method, are proposed. Some problems' properties are also explored to help solve the problems. The performance of the proposed algorithms is compared with a direct use of state-of-the-art solver CPLEX on randomly generated problem instances.

We firstly studied a lane reservation problem (LRP) for future automated truck freight transportation with static link travel time. An integer linear programming model was formulated and the complexity of the problem was proved NP-hard. A cut-and-solve based algorithm was developed to find an optimal solution of the problem. New strategies of generating piercing cuts were proposed for the cut-and-solve method according to the characteristics of the problem. Numerical computational results on randomly generated problem instances shown that the computational time by the proposed algorithm can efficiently solve the tested problems than CPLEX since the computational time by the proposed algorithm is only 42% of that by CPLEX on average for different sized problems. Sensitive experiments of problems with different types of impact, and different average node degree are also shown the effectiveness of the proposed algorithm.

The LRP was extended to the capacitated lane reservation problem (CLRP) with considering residual capacity constraint on non-reserved lanes. For the CLRP, an integer linear programming model was formulated and its complexity was proved NP-hard. Then a cut-and-solve and cutting plane combined approach was proposed for the problem. The embedded cutting plane method in the proposed algorithm permits

to accelerate the convergence of the algorithm. Computational results demonstrated that the proposed algorithm is more efficient to solve the tested problems since it takes about an average of 23% computational time of CPLEX for different sized problems. Sensitive experiments for parameters of residual capacity, impact of reserved lanes, and prescribed travel duration of the tasks shown that the performance is stable with different setting of the these parameters.

Finally, lane reservation problem with time-dependent travel time (LRP-TT), and lane reservation problem with time-dependent travel speed (LRP-TS) were studied. The common characteristic of these two problems is that: the link travel time on reserved lanes is a constant, but the link travel time on non-reserved lanes are not constants any more. In the LRP-TT, the link travel time on a non-reserved lane (i, j) is assumed as a step function of the departure time at node i. A mixed integer nonlinear programming model was firstly formulated and then it was transformed into a linear one. The problem was proved NP-hard. A cut-and-solve based approach with partial integral relaxation strategy and new piercing cut generation strategy is proposed for the LRP-TT. Computational results shown that the proposed algorithm can find an optimal solution of tested problems more quickly than CLPEX for different tested sized problems. It was can be seen that the proposed algorithm takes about an average of 68%, 74%, and 71% computational time of CPLEX for problems with different number of time intervals, different impact of reserved lanes, and prescribed travel duration of the tasks, respectively.

In the LRP-TS, the link travel speed on a non-reserved lane is assumed as a step function of the time. Therefore, the link travel time on a non-reserved lane (i, j) is calculated and it is a piecewise linear continuous function of the departure time at node i. The "first-in-first-out" (FIFO) property is also satisfied. It enables us to develop a tabu search algorithm in which the shortest paths of the tasks are computed by the modified label-correcting shortest path algorithm. It becomes much difficult to solve the problem due to the continuous function of link travel time on non-reserved lanes. Preliminary computational results shown that the proposed tabu search algorithm can find near-optimal solutions with "good" quality for tested small sized problems. However, the gaps between the upper bound and lower bound for large sized problems are large.

To summary this thesis, there is still much work to do in the future. First of all, the impact of single reserved lane is assumed as already know parameter for the studied problems. Actually, this parameter is very complex as many factors, such as type of the reserved lanes, location of the reserved lanes in the network, reservation

time periods, are related with it. It is necessary to systemically study this issue. The results of this related work may provide useful information for our future work.

It becomes more and more difficult to solve the considered problems, especially the last two problems. Because of the introduction of the time-dependent link travel time and time-dependent link travel speed, the link travel time on a non-reserved lane is not a constant in the LRP-TT and LRP-TS. These problems become more complex to be solved as compared with the previous problems. Not only spatial decision (decision of task paths and reserved lanes), but also temporal decision (arrived time of tasks at the nodes) are considered in both problems. To solve large sized problems, properties related to these two levels of decisions should more investigated. For example, certain lanes, which are very far from the source or destination of the task, are unlikely visited by the task. The theoretical study may help solve the problems.

It can be seen that the proposed algorithm becomes less effective to solve the LRP, CLRP and LTP-TT, as the ratios of the CPU time by the proposed algorithm and those by CPLEX increase. Then, more appropriate decomposing strategy for the current problem and generation strategy for the piercing cut in the cut-and-solve method should be developed. Some hybrid methods can be developed according to the analysis of properties of the problem.

It seems that it is much more difficult to solve the LRP-TS than the previous three problems. The possible reason is that the travel time on a non-reserved lane is a continuous function. Due to the complexity of the problem, the size of problems that can be solved is relatively small. Though the computational results shown that the proposed algorithm can find "good" upper bound for small sized problems, the gaps between the upper bound and the lower bound are not satisfied. More efficient methods should be explored to obtain better values of them. Since the travel time is a continuous function, some analytical method may be developed.

In the LRP-TT and LRP-TS, the number of time intervals is not large because of the complexity. In future work, it should be considered more number of time intervals to make the problems closer to realistic situations, obviously the developed resolution method should be sufficiently robust. In addition, the problems in this thesis only consider which lanes should be reserved (spatial level). Future work may additionally consider which time periods the lane reservation strategy should be performed (temporal level).

Appendix I

In this appendix, we present the proof of the theorem in section 5.3.1 as follows. Suppose the time horizon is divided into nq time intervals as $[T_0,\ T_1,\ \ldots,\ T_{nq})$ for the LRP-TS, where $T_q, q \in Q = \{0,\ 1, \ldots,\ nq\}$, is the boundary of the time interval. The travel time function $\tau_{ij}^*(t)$ corresponding to the non-reserved lane (i, j) in the network is a piecewise linear function of the departure time t at node i. Then its breakpoints have the following characteristics.

Theorem 7 *For each breakpoint $\overline{P} = (\bar{t}, \tau_{ij}^*(\bar{t}))$ of the travel time function on the non-reserved lane (i, j), one of the following two cases is satisfied: (1), the departure time $\bar{t}$ at node i is the boundary of certain time interval $T_q, q \in Q$; or (2), the arrived time $\bar{t} + \tau_{ij}^*(\bar{t})$ at node j is the boundary of certain time interval $T_q, q \in Q$.*

Proof: Suppose a vehicle A departs at node i at time $\bar{t}$, the corresponding travel time on (i, j) is $\tau_{ij}^*(\bar{t})$, and the arrived time at node j is $\bar{t} + \tau_{ij}^*(\bar{t})$. Suppose the departure time $\bar{t}$ at node i is in time interval $[T_{q_1}, T_{q_1+1})$, the arrived time $\bar{t} + \tau_{ij}^*(\bar{t})$ at node j is in $[T_{q_2}, T_{q_2+1})$, and $q_2 \geq q_1$.

Let $\Delta t > 0$ be a very small positive number satisfying $\bar{t} + \Delta t \in [T_{q_1}, T_{q_1+1})$. Then vehicle A travels a distance of $v_{ijq_1}\Delta t$ from $\bar{t}$ to $\bar{t} + \Delta t$, where v_{ijq_1} is the travel speed during time interval $[T_{q_1}, T_{q_1+1})$. Since the arrived time at j is $\bar{t} + \tau_{ij}^*(\bar{t})$, then the vehicle travels a distance of $D_{ij} - v_{ijq_1}\Delta t$ from $\bar{t} + \Delta t$ to $\bar{t} + \tau_{ij}^*(\bar{t})$, where D_{ij} is the length of lane (i, j). The travel information of vehicle A is presented in Fig. A.1.

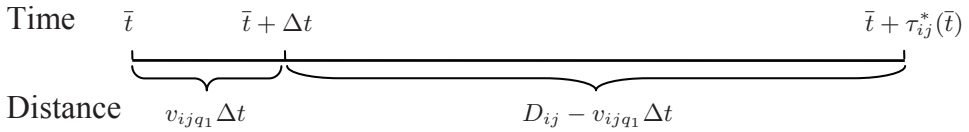


Fig. A.1: Travel information of vehicle A.

Now suppose vehicle B departs at node i at time $\bar{t} + \Delta t$, the corresponding travel time on (i, j) is $\tau_{ij}^*(\bar{t} + \Delta t)$, and the arrived time at node j is $\bar{t} + \Delta t + \tau_{ij}^*(\bar{t} + \Delta t)$. Then vehicle B travels a distance of $D_{ij} - v_{ijq_1}\Delta t$ from $\bar{t} + \Delta t$ to $\bar{t} + \tau_{ij}^*(\bar{t})$, and it travels a distance $v_{ijq_1}\Delta t$ form $\bar{t} + \tau_{ij}^*(\bar{t})$ to $\bar{t} + \Delta t + \tau_{ij}^*(\bar{t} + \Delta t)$. The travel information of vehicle B is presented in Fig. A.2.

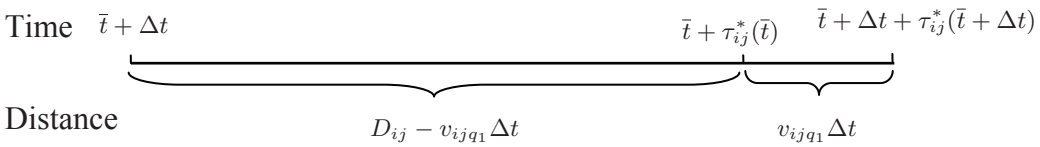


Fig. A.2: Travel information of vehicle B.

Note that the distance traveled by vehicle A form $\bar{t}$ to $\bar{t}+\Delta t$ is equal to the distance traveled by vehicle B from $\bar{t}+\tau_{ij}^*(\bar{t})$ to $\bar{t}+\Delta t+\tau_{ij}^*(\bar{t}+\Delta t)$.

Denote points $\overline{P}=(\bar{t},\tau_{ij}^*(\bar{t}))$ and $P_r=(\bar{t}+\Delta t,\tau_{ij}^*(\bar{t}+\Delta t))$. The slope S_r between $\overline{P}$ and P_r is:

$$\begin{aligned} S_r &= \frac{\tau_{ij}^*(\bar{t}+\Delta t)-\tau_{ij}^*(\bar{t})}{\bar{t}+\Delta t-\bar{t}} = \frac{\bar{t}+\Delta t+\tau_{ij}^*(\bar{t}+\Delta t)-(\bar{t}+\tau_{ij}^*(\bar{t}))-\Delta t}{\Delta t} \\ &= \frac{\bar{t}+\Delta t+\tau_{ij}^*(\bar{t}+\Delta t)-(\bar{t}+\tau_{ij}^*(\bar{t}))}{\Delta t}-1 \end{aligned} \tag{A1}$$

Similarly, suppose vehicle C departs at node i at time $\bar{t}-\Delta t$, the corresponding travel time on (i,j) is $\tau_{ij}^*(\bar{t}-\Delta t)$, and the arrived time at node j is $\bar{t}-\Delta t+\tau_{ij}^*(\bar{t}-\Delta t)$. Denote point $P_l=(\bar{t}-\Delta t,\tau_{ij}^*(\bar{t}-\Delta t))$. The slope S_l between P_l and $\overline{P}$ is:

$$\begin{aligned} S_l &= \frac{\tau_{ij}^*(\bar{t})-\tau_{ij}^*(\bar{t}-\Delta t)}{\bar{t}-(\bar{t}-\Delta t)} = \frac{\bar{t}+\tau_{ij}^*(\bar{t})-(\bar{t}-\Delta t+\tau_{ij}^*(\bar{t}-\Delta t))-\Delta t}{\Delta t} \\ &= \frac{\bar{t}+\tau_{ij}^*(\bar{t})-(\bar{t}-\Delta t+\tau_{ij}^*(\bar{t}-\Delta t))}{\Delta t}-1 \end{aligned} \tag{A2}$$

Because $\overline{P}$ is a breakpoint of the travel time function, then according to the definition of breakpoint, the right slope S_r at $\overline{P}$ is not equal to the left slope S_l at $\overline{P}$. In the following we will prove that if $\bar{t}+\tau_{ij}^*(\bar{t})$ is not the boundary of certain time interval, then $\bar{t}$ must be the boundary of certain time interval.

Note that $\bar{t}\in[T_{q_1},T_{q_1+1})$ and $\bar{t}+\tau_{ij}^*(\bar{t})\in[T_{q_2},T_{q_2+1})$, $q_2\geq q_1$. If $\tau_{ij}^*(\bar{t})$ is not the boundary of certain time interval, then $\bar{t}+\tau_{ij}^*(\bar{t})\in(T_{q_2},T_{q_2+1})$. Because Δt is very small, the distance $v_{ijq_1}\Delta t$ is also small. That means vehicle B takes very short time to travel this distance and the arrived time $\bar{t}+\Delta t+\tau_{ij}^*(\bar{t}+\Delta t)$ at node j of vehicle B is in time interval $[T_{q_2},T_{q_2+1})$. Therefore, vehicle B travels a distance of $v_{ijq_1}\Delta t$ at an unchangeable speed of v_{ijq_2} from $\bar{t}+\tau_{ij}^*(\bar{t})$ to $\bar{t}+\Delta t+\tau_{ij}^*(\bar{t}+\Delta t)$. Then,

$$\begin{aligned} S_r &= \frac{\bar{t}+\Delta t+\tau_{ij}^*(\bar{t}+\Delta t)-(\bar{t}+\tau_{ij}^*(\bar{t}))}{\Delta t}-1 = \frac{v_{ijq_1}\Delta t/v_{ijq_2}}{\Delta t}-1 \\ &= \frac{v_{ijq_1}}{v_{ijq_2}}-1 \end{aligned} \tag{A3}$$

Similar, the arrived time $\bar{t}-\Delta t+\tau_{ij}^*(\bar{t}-\Delta t)$ at node j of vehicle C is also in time interval $[T_{q_2},T_{q_2+1})$. Then $\bar{t}$ must be the boundary of certain time interval. If it is not, we have $\bar{t}\in(T_{q_1},T_{q_1+1})$ and $\bar{t}-\Delta t\in(T_{q_1},T_{q_1+1})$. Note that the distance

traveled by vehicle C form $\bar{t}-\Delta t$ to $\bar{t}$ is equal to the distance traveled by vehicle A from $\bar{t}-\Delta t+\tau_{ij}^*(\bar{t}-\Delta t)$ to $\bar{t}+\tau_{ij}^*(\bar{t})$. Therefore, vehicle A travels a distance of $v_{ijq_1}\Delta t$ at an unchangeable speed of v_{ijq_2} from $\bar{t}-\Delta t+\tau_{ij}^*(\bar{t}+\Delta t)$ to $\bar{t}+\tau_{ij}^*(\bar{t})$. Then,

$$\begin{aligned} S_l &= \frac{\bar{t}+\tau_{ij}^*(\bar{t})-(\bar{t}-\Delta t+\tau_{ij}^*(\bar{t}-\Delta t))}{\Delta t}-1=\frac{v_{ijq_1}\Delta t/v_{ijq_2}}{\Delta t}-1 \\ &= \frac{v_{ijq_1}}{v_{ijq_2}}-1 \end{aligned} \tag{A4}$$

Eq. (A3) and (A4) imply that $S_r = S_l$, which means that the slope does not change at point $\overline{P}$. This is contradictory to the definition of breakpoint. Therefore, $\bar{t}$ must be the boundary of certain time interval. The theorem is proved. □

Résumé en français

Chapitre 1 Introduction

Le transport joue un rôle de plus en plus prépondérant dans l'organisation de nos sociétés et le processus d'intégration de l'économie mondiale. Il est donc crucial de viser sa bonne gestion. Mais, l'urbanisation rapide et le nombre sans cesse croissant de véhicules causent de nombreux problèmes, tels que l'inefficacité ou l'inadéquation de certain modes de transport, l'augmentation du temps de voyage, le gaspillage de l'énergie, et la dégradation de l'environnement. La solution conventionnelle, qui consiste à augmenter la capacité du réseau de transport en créant de nouvelles routes ou de nouvelles voies sur les routes existantes, n'est généralement pas viable pour des raisons économiques et de contraintes spatiales. Il est donc nécessaire d'explorer de nouvelles solutions faisant appel à une meilleure gestion et organisation du trafic.

Parmi les solutions alternatives, la stratégie de réservation des voies semble être prometteuse. Ainsi, des voies peuvent être réservées pour les usagers prioritaires. Seuls certains types de véhicules sont autorisés à utiliser les voies réservées. Ces voies peuvent alors offrir un environnement de voyage sans congestion. Le concept de réservation des voies a été présenté comme une stratégie de gestion du trafic et a été mis en place dans plusieurs pays. Il y a relativement peu de travaux sur l'optimisation du choix de ces voies sur un réseau de transport. Cette thèse s'inscrit dans cette perspective. Elle a pour objectif de fournir une aide à la décision pour la mise en place de stratégies de réservation des voies.

Dans cette thèse, quatre nouveaux problèmes de réservation des voies sont étudiés. Le but est de minimiser l'impact global sur le trafic de l'opération de réservation des voies. Les problèmes considérés acquièrent un caractère plus général et plus réaliste en y intégrant différents facteurs. Nous étudions dans un premier temps, le problème "lane reservation problem" (LRP) avec un temps de parcours statique sur la voie réservée. Par la suite, le problème "capacitated lane reservation problem" (CLRP) prend en compte la capacité des voies. Enfin, les problèmes "lane reservation problems

with time-depdedent travel time" (LRP-TT) et "lane reservation problems with time-depdedent travel speed" (LRP-TS) avec un temps de parcours dynamique sur les voies non réservées sont étudiés. Pour chacun des problèmes, de nouveaux modèles mathématiques (programmation linéaire en nombres entiers ou programmation non linéaire en nombres mixtes, du plus basique au plus élaboré) et méthodes de résolution ("cut-and-solve", la méthode de coupe, et la recherche tabou) sont développés. Les performances des algorithmes proposés sont évaluées par des expériences numériques.

Cette thèse est organisée comme suit. Un état de l'art sur la stratégie de réservation des voies est présenté dans le chapitre 2. Certains problèmes de transport sont introduits et certaines méthodes de optimisation, y compris "cut-and-solve", méthode de coupe, recherche tabou, sont décrites. Dans le chapitre 3, une approche basée sur la méthode "cut-and-solve" est développée pour le LRP. Une approche combinée des méthodes de "cut-and-solve" et de coupe est proposée pour le CLRP dans le chapitre 4. Le chapitre 5 considère les problèmes LRP-TT et LRP-TS. La méthode de "cut-and-solve" est développée pour le LRP-TT et la méthode de recherche tabou est développée pour le LRP-TS.

Chapitre 2 Etat de l'art

Dans ce chapitre, nous présentons d'abord le principe d'utilisation de la réservation des voies dans des réseaux de transport afin de réduire les congestions. La réservation des voies aux usagers prioritaires a été mise en place dans certains pays depuis bientôt quarante ans. Les voies sont souvent réservées pour les bus et les taxis [22] [49] [50]. Dans certains pays d'Amérique du Nord, une voie est réservée aux véhicules avec au moins deux passagers [51] [80]. Cette voie est destinée à encourager le covoiturage afin de réduire le nombre de véhicules sur la route. Une alternative est d'offrir une réservation premium à des usagers non prioritaires contre un droit de péage élevé pour utiliser efficacement la voie réservée [59].

Le concept de réservation de la voie a été étudié dans la littérature. Les études ont les points communs suivants : 1), Elles sont axées sur la performance d'une voie réservée dans la région locale des réseaux de transport; 2), les méthodes d'étude sont soit basées sur l'analyse de données empiriques, soit utilisent des expériences de simulation via des simulateurs de trafic. Il n'y avait quasiment pas de travaux sur l'optimisation du choix de ces voies au niveau d'un réseau de transport, ce qui conforte la nécessité du travail réalisé dans le cadre de cette thèse.

Ensuite, quelques problèmes classiques de transport sont présentés, comme le problème de multiflot de coût minimum, le problème de localisation des installations, le problème de tournées avec fenêtres de temps, et leurs modèles mathématiques détaillés. Il est indiqué que les problèmes de réservation des voies ne peuvent pas être transformés en aucun d'entre eux. D'où l'intérêt scientifique d'étudier ces problèmes.

Enfin, les méthodes de résolution sont présentées. D'abord, la méthode "cut-and-solve" est introduite. Cette méthode a été proposée par Climer et Zhang [16]. C'est une méthode exacte itérative qui peut trouver une solution optimale pour une problème de programmation en nombres entiers ou mixtes. Pour un problème, à la n-ème itération de la méthode, le "current problem" (CP_n) est décomposé en deux sous-problèmes : le "sparse problem" (SP_n) et le "remaining problem" (RP_n) par une "piercing cut" (PC_n). Le SP_n peut être résolu exactement et fournit une borne supérieure du problème original. La meilleure borne supérieure du problème original est ensuite mise à jour. Le SP_n n'est pas pris en compte dans les itérations ultérieures. La résolution du modèle relaxé du RP_n permet de fournir une borne inférieure du RP_n. Si la borne inférieure du RP_n est supérieure ou égale à la meilleure borne supérieure du problème original, elle est alors la valeur optimale et le processus d'itération s'arrête. Sinon, le RP_n devient le CP_{n+1} pour la prochaine itération. La définition de PC_n est importante pour la méthode "cut-and-solve" car la résolution du SP_n dépend de la PC_n. En plus, une borne inférieure serrée du RP_n peut accélérer la convergence de la "cut-and-solve".

La méthode de coupe intégrée est une méthode itérative qui peut améliorer la borne inférieure de la programmation linéaire en nombres entiers. A la première itération, un modèle relaxé du problème est résolu et une solution fractionnaire est obtenue. D'abord on recherche les "cover inequalities" (CIs) valides pour toutes les solutions réalisables du problème original mais violées par la solution fractionnaire. Ces CIs permettent de séparer toutes les solutions réalisables de la solution fractionnaire. Ensuite, ces CIs sont ajoutées au modèle relaxé précédent. Un nouveau modèle relaxé est obtenu et résolu. Ce processus continue jusqu'à ce qu'aucune CIs ne soit trouvée ou que la solution fractionnaire soit une solution réalisable du problème original (aussi une solution optimale réalisable du problème original). L'algorithme de la recherche de CIs est un algorithme de séparation. Des détails de la méthode de coupe pour CIs peuvent être trouvés dans la littérature [42].

La méthode de "tabu search" est une stratégie de recherche itérative. A chaque itération de la méthode, un voisin "neighborhood" est généré autour de la solution courante "current solution" (pour la première itération, on prend la solution initiale

"initial solution"). Le "neighborhood" est un ensemble de solutions réalisables, qui peut être atteint par une opération appelée "move" à partir de "current solution". La meilleure solution admissible dans ce "neighborhood" est choisie et devient la "current solution" pour la prochaine itération. La meilleure solution est mise à jour si la meilleure solution admissible est meilleure qu'elle. Pour interdire de retomber dans le minimum local auquel on vient d'échapper, le mécanisme ("tabu list") est créé pour interdire de revenir aux dernières "moves" explorées. L'itération de la "tabu search" est répétée jusqu'à ce qu'un nombre maximal d'itérations soit atteint.

Chapitre 3 Problème de réservation des voies

Dans ce chapitre, nous étudions un problème de réservation des voies "lane reservation problem" (LRP), motivé par le transport de marchandises par l'utilisation de camions automatisés. Ces camions automatisés du futur peuvent former un peloton avec le premier conduit par l'homme et les autres suivent en mode automatique [61] [77]. Pour assurer un environnement approprié et sûr pour ces camions automatisés, il est préférable de leur réserver des voies. Cela consiste à sélectionner certaines voies ordinaires existantes dans un réseau de transport et de les convertir en des voies réservées à l'usage spécifique de camions automatisés. Les voies réservées peuvent offrir un itinéraire sans congestion et la durée de transit est alors maîtrisée. Cette stratégie apparaît comme plus économique et plus flexible que celle qui consisterait à construire de nouvelles infrastructures. C'est une stratégie intelligente de gestion du trafic.

Cependant, les voies réservées ont un impact négatif sur les autres usagers sur les voies adjacentes. Ces voies utilisées par les usagers généraux peuvent être encombrées et la durée de leur trajet se trouvera plus longue. Il faut donc bien déterminer quelles voies du réseau devraient être réservées afin de minimiser l'impact de ces réservations de voies. Le "lane reservation problem" (LRP) dans ce chapitre se formalise comme suit. On a un réseau $G(N, A)$, où N est un ensemble de nœuds et A est un ensemble d'arcs orientés. Étant donné un ensemble de tâches de transport sachant qu'une paire source-destination correspond à chaque tâche, le LRP consiste à réserver des voies dans le réseau et à concevoir un chemin pour chaque tâche afin qu'elle puisse être achevée dans un délai prescrit. Le critère est de minimiser l'impact total de toutes les voies réservées sur le trafic.

Pour bien étudier le problème, certaines hypothèses sont formulées comme suit : 1), il y a au moins deux voies sur chaque arcs permettant ainsi la réservation d'une

voie si nécessaire; 2), il y a au plus une voie réservée sur chaque arc parce que les voies réservées peuvent être partagées entre tous les usagers des tâches; 3), les camions de chaque tâche se déplacent sur un chemin exclusivement réservé (chaque voie dans le chemin est réservée) à partir de sa source vers sa destination pour une raison de sécurité.

Modèle Mathématique

Pour formuler le problème, les notations sont présentées comme suit.

Paramètres

A : l'ensemble des arcs orientés $(i,j), i \neq j, i,j \in N$
K : l'ensemble des tâches de transport $k \in K$
N : l'ensemble des nœuds
a_{ij} : l'impact de la réservation d'une voie $(i,j) \in A$
d_k : le nœud destination de la tâche $k \in K$
p_k : la durée de trajet maximale souhaitée de la tâche $k \in K$
s_k : le nœud source de la tâche $k \in K$
τ_{ij} : la durée de traversée de la voie réservée $(i,j) \in A$

Variables de décision

x_{kij} $x_{kij} = 1$, si la tâche $k \in K$ traverse la voie réservée $(i,j) \in A$; $x_{kij} = 0$, sinon
z_{ij} $z_{ij} = 1$, si la voie $(i,j) \in A$ est réservée; $z_{ij} = 0$, sinon

Le LRP est formulé comme le problème de programmation linéaire suivant :

$$P_l: \quad \min \sum_{(i,j)\in A} a_{ij} z_{ij} \tag{3.1}$$

$$\text{s.t.} \sum_{i:(s_k,i)\in A} x_{ks_ki} = 1, \qquad \forall k \in K, \tag{3.2}$$

$$\sum_{i:(i,d_k)\in A} x_{kid_k} = 1, \qquad \forall k \in K, \tag{3.3}$$

$$\sum_{i:(j,i)\in A} x_{kji} = \sum_{i:(i,j)\in A} x_{kij}, \qquad \forall k \in K, \forall j \in N \setminus \{s_k, d_k\}, \tag{3.4}$$

$$\sum_{(i,j)\in A} \tau_{ij} x_{kij} \leq p_k, \qquad \forall k \in K, \tag{3.5}$$

$$x_{kij} \leq z_{ij}, \qquad \forall k \in K, \forall (i,j) \in A, \tag{3.6}$$

$$x_{kij} \in \{0,1\}, \qquad \forall k \in K, \forall (i,j) \in A, \tag{3.7}$$

$$z_{ij} \in \{0,1\}, \qquad \forall (i,j) \in A. \tag{3.8}$$

La fonction objectif (3.1) consiste à minimiser l'impact global sur le trafic de toutes

les voies réservées. (3.2)–(3.4) indiquent qu'il y a exactement un chemin pour chaque tâche $k \in K$ à partir de sa source s_k vers sa destination d_k. (3.5) est la contrainte de délai. Elle indique que la tâche doit être achevée dans le délai p_k. (3.6) indique que la tâche k ne peut pas passer la voie réservée (i, j) si cette dernière n'est pas réservée, c'est-à-dire, si $z_{ij} = 0$, alors $x_{kij} = 0$. (3.7) et (3.8) sont les contraintes sur la nature des variables de décision.

Le LRP est NP-difficile, car un cas particulier de celui-ci (si les sources de toutes les tâches sont du même nœud et le délai p_k est très grand) peut être réduit au problème de l'arbre de Steiner orienté, qui est également connu pour être NP-difficile [44].

Approche de résolution

Pour résoudre le problème, un pré-traitement est d'abord effectué pour réduire l'espace des solutions du modèle P_l et un nouveau modèle P'_l serré est obtenu. Notons $l(j, d_k)$ (resp. $l(s_k, j)$) comme étant la plus courte durée de trajet à partir de nœud j vers le nœud d_k (resp. à partir de nœud s_k vers le nœud j) dans un chemin exclusivement réservé (il peut être calculé par les algorithmes de plus court chemin). Pour chaque tâche $k \in K$, les ensembles A_k et A'_k sont définis comme suit :

$$A_k = \{\ (s_k, j) \mid \tau_{s_k j} + l(j, d_k) > p_k,\ \forall (s_k, j) \in A\ \},$$
$$A'_k = \{\ (j, d_k) \mid l(s_k, j) + \tau_{j d_k} > p_k,\ \forall (j, d_k) \in A\ \}.$$

Les voies dans les ensembles A_k ou A'_k ne seront pas dans le chemin de la tâche k, sinon la tâche k ne pourra pas être achevée dans le délai p_k. Les variables correspondantes peuvent donc être fixées à zéro. Le nouveau modèle P'_l est défini comme suit :

$$P'_l : \quad \min \sum_{(i,j)\in A} a_{ij} Z_{ij}$$
$$\text{s.t.} \quad \text{Constraintes } (3.2) - (3.8)$$
$$x_{k s_k j} = 0, \qquad \forall k \in K, (s_k, j) \in A_k, \tag{3.9}$$
$$x_{k j d_k} = 0, \qquad \forall k \in K, (j, d_k) \in A'_k. \tag{3.10}$$

Le P'_l est égal au P_l, mais l'espace des solutions de P'_l est réduit car certaines variables de décision sont fixées à zéro. La méthode de "cut-and-solve" va donc résoudre le modèle P'_l au lieu du modèle P_l.

L'approche de résolution pour le LRP est basée sur la méthode de "cut-and-solve". C'est une méthode exacte itérativede programmation en nombres entiers qui

peut trouver une solution optimale . A la n-ème itération de la méthode, le "current problem" (CP_n) est décomposé en deux sous-problèmes: le "sparse problem" (SP_n) et le "remaining problem" (RP_n) par une "piercing cut" (PC_n) comme suit :

$$PC_n : \quad \sum_{z_{ij} \in V_n} z_{ij} \geq 1,$$

où $V_n = \{\ z_{ij} \mid \psi(z_{ij}) > h_n,\ \forall (i,j) \in A\ \}$, $\psi(z_{ij})$ est la coût réduit de la variable de décision z_{ij}, et h_n est une constante connue. Par cette PC_n, le SP_n peut être obtenu en ajoutant la contrainte $\sum_{z_{ij} \in V_n} z_{ij} = 0$ au problème CP_n et le RP_n peut être obtenu en ajoutant la contrainte $\sum_{z_{ij} \in V_n} z_{ij} \geq 1$ au problème CP_n. Le CP_n est défini comme le P'_l pour $n = 1$ et RP_{n-1} pour $n \geq 2$. Les SP_n et RP_n sont définis comme suit :

$$\begin{aligned}
SP_n : \quad & \min \sum_{(i,j) \in A} a_{ij} Z_{ij} \\
\text{s.t.} \quad & \text{Constraintes } (3.2) - (3.10) \\
& \sum_{z_{ij} \in V_m} z_{ij} \geq 1, \qquad m = 1, 2, \ldots, n-1. \qquad (3.11) \\
& \sum_{z_{ij} \in V_n} z_{ij} = 0. \qquad (3.12) \\
RP_n : \quad & \min \sum_{(i,j) \in A} a_{ij} Z_{ij} \\
\text{s.t.} \quad & \text{Constraintes } (3.2) - (3.10), \text{ et } (3.11) \\
& \sum_{z_{ij} \in V_n} z_{ij} \geq 1. \qquad (3.13)
\end{aligned}$$

Pour améliorer la performance de la méthode "cut-and-solve", deux ensembles sont définis :

$$\begin{aligned}
U_n &= \{\ z_{s_k j} \mid z^*_{s_k j} < \max_{j:(s_k,j) \in A} z^*_{s_k j},\ \forall k \in K, (s_k, j) \in A\ \}, \\
U'_n &= \{\ z_{i d_k} \mid z^*_{i d_k} < \max_{i:(i,d_k) \in A} z^*_{i d_k},\ \forall k \in K, (i, d_k) \in A\ \},
\end{aligned}$$

où les $z^*_{s_k j}$ et $z^*_{i d_k}$ sont les valeurs des variables de décision dans la solution optimale du problème de la relaxation linéaire de la CP_n. Une nouvelle définition de V_n est présentée comme suit :

$$V_n = (\{\ z_{ij} \mid \psi(z_{ij}) > h_n,\ \forall (i,j) \in A\ \} \cup U_n \cup U'_n) \cap V_{n-1}, \qquad (3.14)$$

où $V_0 = \{z_{ij} \mid \forall (i,j) \in A\}$. Selon la définition de (3.20), $V_1 \supseteq \cdots \supseteq V_{n-1} \supseteq V_n,\ n \geq 2$, est satisfaite. Les SP_n et RP_n sont donc réduits à des nouveaux "sparse problem"

(SP'_n) et "remaining problem" (RP'_n), qui sont définis comme suit :

$$
\begin{aligned}
SP'_n: \quad & \min \sum_{(i,j)\in A} a_{ij} Z_{ij} \\
\text{s.t.} \quad & \text{Constraintes } (3.2) - (3.10) \text{ et } (3.12) \\
& \sum_{z_{ij}\in(V_{n-1}\setminus V_n)} z_{ij} \geq 1. \qquad (3.15) \\
RP'_n: \quad & \min \sum_{(i,j)\in A} a_{ij} Z_{ij} \\
\text{s.t.} \quad & \text{Constraintes } (3.2) - (3.10) \text{ et } (3.13).
\end{aligned}
$$

Il est prouvé que le SP'_n (resp. RP'_n) est égal au SP_n (RP_n), mais le SP'_n (resp. RP'_n) a moins de contraintes que SP_n (resp. RP_n).

Le SP'_n peut être résolu exactement par le logiciel commercial CPLEX et fournit une borne supérieure du problème original. La meilleure borne supérieure est ensuite mise à jour. Le SP'_n n'est pas pris en compte dans les prochaines itérations. Il est difficile de résoudre le RP'_n exactement. La relaxation linéaire du RP'_n est donc résolu et une borne inférieure du RP'_n est obtenue par CPLEX. Le RP'_n est ensuite devenue le CP_{n+1} dans la prochaine itération. Après chaque itération, la taille du "current problem" devient plus petite. Cette phase est répétée jusqu'à ce que le critère d'arrêt soit atteint : la borne inférieure du RP'_n est supérieure ou égale à la meilleure borne supérieure. La valeur optimale du problème original est alors obtenue.

Expériences numériques

L'algorithme proposé a été évalué par 62×5 instances générées au aléatoirement. Le temps CPU nécessité par l'algorithme proposé est comparé à celui de la résolution directe du problème par CPLEX. Les expériences numériques montrent que : 1), toutes les instances sont résolues exactement par l'algorithme proposé. Il peut trouver une solution optimale pour la plus grande instance constituée 150 nœuds et 30 tâches en 1543.43 secondes, comparativement, CPLEX a nécessité prend 4058.54 secondes pour la même instance. 2), les pré-traitement et les nouvelles définitions des "sparse problem" et "remaining problem" sont efficaces pour accélérer la convergence de l'algorithme proposé. L'algorithme proposé est 2.84 fois plus rapide que le même algorithme sans pré-traitement et 1.52 fois plus rapide que le même algorithme sans utiliser les nouvelles définitions des "sparse problem" et "remaining problem". 3), les temps CPU moyenss de l'algorithme proposé sont réduisent respectivement de 42%, 31%, et 36% de ceux de CPLEX pour les instances avec différentes tailles, différents

impacts des voies réservées et différents degrés moyens des noeuds du réseau (défini par 2 × nombre d'arcs/nombre de nœuds).

Chapitre 4 Problème de réservation des voies avec contrainte de capacité

Dans ce chapitre, nous étudions le "capacitated lane reservation problem" (CLRP) pour les événements exceptionnels, tels que les grandes manifestations sportives. En raison du trafic dense lors de ces occasions, il est difficile d'accomplir les tâches de transport dans les délais. La stratégie de réservation des voies est donc nécessaire.

Le CLRP est une extension du LRP étudié dans le chapitre 3, car il considère la capacité résiduelle de chaque voie non réservée dans le réseau, qui est ignorée dans le LRP. La capacité résiduelle d'une voie non réservée est le flux résiduel de la voie qui peut être utilisé par les tâches spéciales sans causer plus de temps de parcours ou de congestion sur cette voie. C'est-à-dire, le flux total des tâches se déplaçant sur la voie non réservée ne peut pas être supérieur à sa capacité résiduelle. Pour une voie réservée, elle peut être partagée par toutes les tâches, parce que les véhicules généraux ne peuvent pas l'utiliser, et la capacité de celle-ci est suffisante pour que chaque tâche puisse l'utiliser.

Ici, on a un réseau $G(N, A)$, où N est un ensemble de nœuds, et A est un ensemble d'arcs orienté, et étant donné un ensemble de tâches de transport associant une paire source-destination à chaque tâche. Le CLRP dans ce chapitre consiste à réserver des voies dans le réseau et à concevoir un chemin pour chaque tâche afin qu'elle puisse être achevée dans un délai prescrit. Le flux total des tâches se déplaçant sur chaque voie non réservée ne peut pas être supérieur à sa capacité résiduelle. Le but est de minimiser l'impact total de toutes les voies réservées sur le trafic.

Les hypothèses pour le CLRP sont les mêmes que celles pour le LRP, mais ces deux problèmes se différencient sur un point : le chemin pour chaque tâche n'est pas nécessairement exclusivement réservé, c'est-à-dire, il peut exister des tronçons sans voie réservée dans le chemin.

Modèle Mathématique

Les notations pour le CLRP sont présentées comme suit.

Paramètres

A : l'ensemble des arcs orientés $(i, j), i \neq j, i, j \in N$

K : l'ensemble des tâches de transport $k \in K$

N : l'ensemble des nœuds
a_{ij} : l'impact de la réservation d'une voie $(i,j) \in A$
c_{ij} : la capacité résiduelle d'une voie non réservée $(i,j) \in A$
d_k : le nœud destination de la tâche $k \in K$
fl_k : flux de trafic de la tâche $k \in K$
p_k : la durée de trajet maximale souhaitée pour la tâche $k \in K$
s_k : le nœud source de la tâche $k \in K$
τ_{ij} : la durée de traversée de la voie réservée $(i,j) \in A$
τ'_{ij} : la durée de traversée de la voie non réservée $(i,j) \in A$

Variables de décision
x_{kij} $x_{kij} = 1$, si la tâche $k \in K$ traverse la voie réservée $(i,j) \in A$; $x_{kij} = 0$, sinon
y_{kij} $y_{kij} = 1$, si la tâche $k \in K$ traverse la voie non réservée $(i,j) \in A$; $y_{kij} = 0$, sinon
z_{ij} $z_{ij} = 1$, si la voie $(i,j) \in A$ est réservée; $z_{ij} = 0$, sinon

Le CLRP est formulé comme un problème de programmation linéaire :

$$P_c : \quad \min \sum_{(i,j)\in A} a_{ij} z_{ij} \tag{4.1}$$

$$\text{s.t.} \quad \sum_{i:(s_k,i)\in A} (x_{ks_ki} + y_{ks_ki}) = 1, \qquad \forall k \in K, \tag{4.2}$$

$$\sum_{i:(i,d_k)\in A} (x_{kid_k} + y_{kid_k}) = 1, \qquad \forall k \in K, \tag{4.3}$$

$$\sum_{i:(j,i)\in A} (x_{kji} + y_{kji}) = \sum_{i:(i,j)\in A} (x_{kij} + y_{kij}), \quad \forall k \in K, \forall j \in N \setminus \{s_k, d_k\}, \tag{4.4}$$

$$\sum_{(i,j)\in A} \left(\tau_{ij} x_{kij} + \tau'_{ij} y_{kij}\right) \le p_k, \qquad \forall k \in K, \tag{4.5}$$

$$x_{kij} \le z_{ij}, \qquad \forall k \in K, \forall (i,j) \in A, \tag{4.6}$$

$$y_{kij} + z_{ij} \le 1, \qquad \forall k \in K, \forall (i,j) \in A, \tag{4.7}$$

$$\sum_{k\in K} fl_k y_{kij} \le c_{ij}(1 - z_{ij}), \qquad \forall (i,j) \in A, \tag{4.8}$$

$$x_{kij}, y_{kij} \in \{0,\ 1\}, \qquad \forall k \in K, \forall (i,j) \in A, \tag{4.9}$$

$$z_{ij} \in \{0,\ 1\}, \qquad \forall (i,j) \in A. \tag{4.10}$$

La fonction coût (4.1) minimise l'impact global sur le trafic de toutes les voies réservées. (4.2)–(4.4) indiquent qu'il y a exactement un chemin pour chaque tâche $k \in K$ à partir de sa source s_k vers sa destination d_k. (4.5) est la contrainte de délai. Elle indique que la tâche doit être achevée dans le délai p_k. (4.6) indique que la tâche k ne peut pas passer la voie réservée (i,j) si cette dernière n'est pas réservée,

c'est-à-dire, si $z_{ij} = 0$, alors $x_{kij} = 0$. (4.7) indique que la tâche k peut passer la voie non réservée (i, j) si cette dernière n'est pas réservée, c'est-à-dire, si $y_{kij} = 1$, alors $z_{ij} = 0$. (4.8) indique que le flux total des tâches se déplaçant sur chaque voie non réservée (i, j) ne peut pas être supérieur à la capacité résiduelle c_{ij}. (4.9) et (4.10) sont les contraintes sur les variables de décision.

Le CLRP est NP-difficile, car le LRP est un cas particulier de celui-ci (dans le cas ou la capacité résiduelle de chaque voie non réservée est très petite, alors chaque voie dans le chemin de la tâche doit être réservée), qui a été également démontré NP-difficile dans chapitre 3.

Approche de résolution

Pour résoudre le problème, un pré-traitement est d'abord effectué pour réduire l'espace des solutions du modèle P_c et un nouveau modèle P'_c serré est obtenu. Notons $l(j, d_k)$ (resp. $l(s_k, j)$) comme la plus courte durée de transit à partir de j vers le nœud d_k (resp. à partir de nœud s_k vers le nœud j) dans un chemin exclusivement réservé. Pour chaque tâche $k \in K$, les ensembles A_k et A'_k sont définis comme suit :

$$A_k = \{\ (s_k, j) \mid \tau_{s_k j} + l(j, d_k) > p_k,\ \forall (s_k, j) \in A\ \},$$
$$A'_k = \{\ (j, d_k) \mid l(s_k, j) + \tau_{j d_k} > p_k,\ \forall (j, d_k) \in A\ \}.$$

Les voies dans les ensembles A_k ou A'_k ne sont pas dans le chemin de la tâche k, sinon la tâche k ne peut pas être achevée dans le délai p_k. Les variables correspondantes peuvent donc être fixées à zéro. Le modèle P'_l est défini comme suit :

$$\begin{aligned}
P'_c: \quad & \min \sum_{(i,j)\in A} a_{ij} Z_{ij} \\
\text{s.t.} \quad & \text{Constraintes } (4.2) - (4.10) \\
& x_{k s_k j} + y_{k s_k j} = 0, && \forall k \in K, (s_k, j) \in A_k, && (4.11) \\
& x_{k j d_k} + y_{k j d_k} = 0, && \forall k \in K, (j, d_k) \in A'_k. && (4.12)
\end{aligned}$$

Pour résoudre le CLRP, une approche combinée des méthodes de "cut-and-solve" et de coupe est proposée. La méthode de coupe est intégrée dans l'algorithme proposé pour obtenir une borne inférieure serrée pour le "remaining problem" (RP_n). Le "piercing cut" (PC_n) de la "cut-and-solve" est défini comme suit :

$$PC_n: \quad \sum_{z_{ij} \in V_n} z_{ij} \geq 1,$$

où $V_n = \{\ z_{ij} \mid \psi(z_{ij}) > h_n,\ \forall (i,j) \in A\ \}$, $\psi(z_{ij})$ est le coût réduit de la variable de décision z_{ij}, et h_n est une constante connue. Le "sparse problem" (SP_n) et RP_n sont définis comme suit :

$$
\begin{aligned}
SP_n: \quad & \min \sum_{(i,j)\in A} a_{ij} Z_{ij} \\
\text{s.t.} \quad & \text{Constraintes } (4.2) - (4.12) \\
& \sum_{z_{ij}\in V_m} z_{ij} \geq 1, \qquad m = 1, 2, \ldots, n-1. \qquad (4.13) \\
& \sum_{z_{ij}\in V_n} z_{ij} = 0. \qquad (4.14)
\end{aligned}
$$

$$
\begin{aligned}
RP_n: \quad & \min \sum_{(i,j)\in A} a_{ij} Z_{ij} \\
\text{s.t.} \quad & \text{Constraintes } (4.2) - (4.12),\ \text{et } (4.13) \\
& \sum_{z_{ij}\in V_n} z_{ij} \geq 1. \qquad (4.15)
\end{aligned}
$$

Le SP_n est résolu exactement par le logiciel commercial CPLEX et fournit une borne supérieure du problème original. La meilleure borne supérieure est ensuite mise à jour.

La méthode de coupe intégrée est une méthode itérative qui peut améliorer la borne inférieure du problème de programmation linéaire en nombres entiers RP_n. A la première itération de la méthode de coupe, un modèle de la relaxation linéaire du RP_n est résolu et une solution fractionnaire est obtenue. Les "cover inequalities" (CIs) correspondant aux contraintes (4.5) et (4.8) sont obtenues par les algorithmes de séparation et sont définies respectivement comme suit :

$$\sum_{(i,j)\in A_x} x_{kij} + \sum_{(i,j)\in A_y} y_{kij} \leq |A_x| + |A_y| - 1, \qquad \forall k \in K, \qquad (4.16)$$

$$\sum_{k\in C} y_{kij} \leq (|C| - 1)(1 - z_{ij}), \qquad (i,j) \in A, \qquad (4.17)$$

où A_x et A_y sont des sous-ensembles de A, et C est le sous-ensemble de K. Le algorithme de séparation pour (4.5) et (4.8) sont basés sur l'algorithme dans la littérature [42]. Les lecteurs intéressés peuvent se référer à ce travail. Les CIs trouvées par le algorithme de séparation sont satisfaites par toutes les solutions réalisables du problème original, mais elles sont violées par la solution fractionnaire. Ensuite ces CIs sont ajoutées au modèle relaxé précédent. La borne supérieure du RP_n peut donc être améliorée. Ce processus se poursuit jusqu'à ce qu'aucune CIs ne soit trouvée par l'algorithme de séparation.

Comme pour la méthode dans chapitre 3, l'itération de la "cut-and-solve" est répétée jusqu'à ce que le critère d'arrêt soit atteint : la borne inférieure du RP_n est supérieure ou égale à la meilleure borne supérieure. La valeur optimale du problème original est alors obtenue.

Expériences numériques

La performance de l'algorithme proposé a été évaluée sur des instances générées au aléatoirement. Les expériences numériques montrent que : 1), toutes les instances sont résolues exactement par l'algorithme proposé. Celui-ci peut trouver une solution optimale pour la plus grande instance de 120 nœuds du réseau. 2), la méthode de coupe améliore la première borne inférieure de 6.92% en moyenne pour les instances à 100 nœuds et 5–40 tâches. Les temps CPU moyens sont de 289.01 secondes pour la méthode combinée "cut-and-solve" et méthode de coupe, et 598.83 secondes pour la méthode "cut-and-solve". 3), les temps CPU moyens pour l'algorithme proposé réduisent respectivement de 21.55%, 42.76%, 65.66%, et 57.48% de ceux obténus par CPLEX pour des instances avec différents nombres de nœuds, différents capacités résiduelles des voies non réservées, différents impacts des voies réservées et différents délais prescrits pour réaliser la tâche.

Chapitre 5 Problème de reservation des voies avec temps de parcours variable

Le LRP du chapitre 3 et le CLRP du chapitre 4 concernent la réservation des voies dans un réseau avec un temps de parcours sur la voie supposé constant. En réalité, le temps de parcours sur une voie varie au cours du temps en raison de plusieurs facteurs tels que les heures de pointe, le flux de trafic et les conditions météorologiques. La prise en compte d'un temps de parcours sur une voie variable a été considérée dans certains problèmes de transport [13], [37]. Pour rendre le problème plus proche des situations réelles, il est nécessaire de considérer le temps de parcours dynamique dans les problèmes de réservation des voies. Dans ce chapitre, nous étudions deux problèmes de réservation des voies avec le temps de parcours variable : "lane reservation problem with time-dependent travel time" (LTP-TT) et "lane reservation problem with time-dependent travel speed" (LTP-TS). Dans ces deux problèmes, le temps de parcours sur les voies non réservées n'est plus constant, il peut virer au fonction du temps.

5.1 Problème de reservation des voies avec temps de parcours variable

Dans le LRP-TT, l'horizon de temporel est subdivisé en plusieurs petits intervalles, le temps de parcours sur une voie non réservée (i, j) est une fonction de l'heure de départ au nœud i. Le temps de parcours est une constante si l'heure de départ au nœud i est dans le même intervalle de temps, mais il change pour différents intervalles de temps. Autrement dit, le temps de parcours sur une voie non réservée est une fonction de l'étape, chaque étape correspond à un intervalle de temps. Le temps de parcours sur une voie réservée est une constante.

Le problème est décrit comme suit. On a un réseau $G(N, A)$, et un ensemble de tâches de transport associé à des paires source-destination. Le LRP-TT a pour objectif de réserver des voies dans le réseau afin de respecter les temps de transport pour chaque tâche en minimisant l'impact de toutes les voies réservées.

Modèle Mathématique

Les notations pour le LRP-TT sont présentées comme suit.

Paramètres

A :	l'ensemble des arcs orientés $(i, j), i \neq j, i, j \in N$
K :	l'ensemble des tâches de transport $k \in K$
N :	l'ensemble des nœuds
Q :	l'ensemble des indices des intervalles de temps
$[T_q, T_{q+1})$:	intervalle de temps, $q \in Q$
a_{ij} :	l'impact de la réservation d'une voie $(i, j) \in A$
d_k :	le nœud destination de la tâche $k \in K$
p_k :	la durée de trajet maximale souhaitée pour la tâche $k \in K$
s_k :	le nœud source de la tâche $k \in K$
st_k :	l'heure de départ de la tâche $k \in K$ au nœud s_k
τ_{ij} :	la durée de traversée de la voie réservée $(i, j) \in A$
τ''_{ijq} :	la durée de traversée de la voie non réservée $(i, j) \in A$ si l'heure de départ au nœud i est dans l'intervalle de temps $[T_q, T_{q+1}), q \in Q$

Variables de décision

b_{kiq}	$b_{kiq} = 1$, si la tâche k part du nœud i dans l'intervalle $[T_q, T_{q+1})$; $b_{kiq} = 0$, sinon
t_{ki}	l'heure de départ de la tâche $k \in K$ au nœud $i \in N$; si la tâche k ne visite pas le nœud i
x_{kij}	$x_{kij} = 1$, si la tâche $k \in K$ traverse la voie réservée $(i, j) \in A$; $x_{kij} = 0$, sinon

y_{kij}	$y_{kij} = 1$, si la tâche $k \in K$ traverse la voie non réservée $(i,j) \in A$; $y_{kij} = 0$, sinon
z_{ij}	$z_{ij} = 1$, si la voie $(i,j) \in A$ est réservée; $z_{ij} = 0$, sinon

Le LRP-TT est formulé comme le problème d'optimisation suivant :

$$P_t: \min \sum_{(i,j)\in A} a_{ij} z_{ij} \tag{5.1}$$

$$\text{s.t.} \sum_{i:(s_k,i)\in A} (x_{ks_ki} + y_{ks_ki}) = 1, \qquad \forall k \in K, \tag{5.2}$$

$$\sum_{i:(i,d_k)\in A} (x_{kid_k} + y_{kid_k}) = 1, \qquad \forall k \in K, \tag{5.3}$$

$$\sum_{i:(i,j)\in A} (x_{kij} + y_{kij}) = \sum_{i:(j,i)\in A} (x_{kji} + y_{kji}), \qquad \forall k \in K, \forall j \in N\backslash\{s_k, d_k\}, \tag{5.4}$$

$$x_{kij} \le z_{ij}, \qquad \forall k \in K, \forall (i,\ j) \in A, \tag{5.5}$$

$$y_{kij} + z_{ij} \le 1, \qquad \forall k \in K, \forall (i,\ j) \in A, \tag{5.6}$$

$$\sum_{q\in Q} b_{kiq} = 1, \qquad \forall k \in K, \forall i \in N\backslash\{d_k\}, \tag{5.7}$$

$$t_{ki} \ge \sum_{q\in Q} b_{kiq} T_q, \qquad \forall k \in K, \forall i \in N\backslash\{d_k\}, \tag{5.8}$$

$$t_{ki} < \sum_{q\in Q} b_{kiq} T_{q+1}, \qquad \forall k \in K, \forall i \in N\backslash\{d_k\}, \tag{5.9}$$

$$t_{ks_k} = st_k, \qquad \forall k \in K, \tag{5.10}$$

$$t_{kj} = \sum_{i:(i,j)\in A} ((t_{ki} + \tau_{ij}) x_{kij} + (t_{ki} + \sum_{q\in Q} b_{kiq} \tau''_{ijq}) y_{kij}), \qquad \forall k \in K, \forall j \in N\backslash\{s_k\}, \tag{5.11}$$

$$t_{kd_k} - t_{ks_k} \le p_k, \qquad \forall k \in K, \tag{5.12}$$

$$t_{ki} \ge 0, \qquad \forall k \in K, \forall i \in N, \tag{5.13}$$

$$b_{kiq} \in \{0,\ 1\}, \qquad \forall k \in K, \forall i \in N, \forall q \in Q, \tag{5.14}$$

$$x_{kij},\ y_{kij} \in \{0,\ 1\}, \qquad \forall k \in K, \forall (i,\ j) \in A, \tag{5.15}$$

$$z_{ij} \in \{0,\ 1\}, \qquad \forall (i,\ j) \in A. \tag{5.16}$$

La fonction objectif (5.1) consiste à l'impact global sur le trafic de toutes les voies réservées. Les conditions (5.2)–(5.4), (5.5) et (5.6) sont respectivement similaires à (4.2)–(4.4), (4.6) et (4.7) du modèle du CLRP étudié dans chapitre 4. (5.7)–(5.9) indiquent que si $b_{kiq} = 1$, alors l'heure de départ de la tâche k au nœud i est dans l'intervalle de temps $[T_q, T_{q+1})$. (5.10) indique que la tâche k part du nœud s_k à l'heure st_k. (5.11) indique que l'heure de départ de la tâche k au nœud j. (5.12)

indique que la durée de trajet à partir du nœud s_k vers le nœud d_k ne doit pas être supérieure au délai p_k. (5.13)–(5.16) sont les contraintes sur les variables de décision.

Le LRP-TT est NP-difficile, car un cas particulier de celui-ci (si le temps de parcours τ''_{ijq} est très grand pour les couples $(i,j) \in A$ et $q \in Q$, alors chaque voie dans le chemin de la tâche doit être réservée) peut être réduit au LRP, qui a été aussi démontré NP-difficile dans chapitre 3.

Linéarisation du modèle

Le modèle P_t n'est pas linéaire. Il est transformé en un modèle linéaire équivalent par linéarisation de la contrainte (5.11). Pour chaque tâche $k \in K$ et chaque nœud $j \in N \setminus \{s_k\}$, il y a trois cas : 1), la tâche k ne visite pas j; 2), la tâche k visite j via une voie réservée (i,j); 3), la tâche k visite j via une voie non réservée (i,j). Pour chaque cas, (5.11) peut être transformée en les contraintes suivantes (5.17), (5.18), et (5.19) :

$$t_{kj} \leq M \sum_{i:(i,j)\in A} (x_{kij} + y_{kij}), \qquad \forall k \in K, \forall j \in N\backslash\{s_k\}, \tag{5.17}$$

$$\left.\begin{array}{l} t_{kj} - t_{ki} - \tau_{ij} \leq M(1 - x_{kij}), \\ t_{kj} - t_{ki} - \tau_{ij} \geq M(x_{kij} - 1), \end{array}\right\} \qquad \forall k \in K, \forall j \in N\backslash\{s_k\}, (i,j) \in A, \tag{5.18}$$

$$\left.\begin{array}{l} t_{kj} - t_{ki} - \sum\limits_{q\in Q} b_{kiq}\tau''_{ijq} \leq M(1 - y_{kij}), \\ t_{kj} - t_{ki} - \sum\limits_{q\in Q} b_{kiq}\tau''_{ijq} \geq M(y_{kij} - 1), \end{array}\right\} \qquad \forall k \in K, \forall j \in N\backslash\{s_k\}, (i,j) \in A, \tag{5.19}$$

où M est une très grande constante. Le modèle linéaire P'_t peut donc être représenté comme suit : (5.1)–(5.10) et (5.12)–(5.19).

Approche de résolution

D'abord, un pré-traitement est effectué. Notons $l(j,d_k)$ (resp. $l(s_k,j)$) comme la plus courte durée de transit à partir de j vers destination du nœud d_k (resp. à partir du nœud s_k vers le nœud j) dans un chemin exclusivement réservé. L'ensemble $N_k, k \in K$, est défini comme suit :

$$N_k = \{j \mid l(s_k,j) + l(j,d_k) > p_k, j \in N\}.$$

Les variables correspondantes peuvent être fixées à zéro :

$$\left.\begin{array}{c} \sum\limits_{i:(i,j)\in A} (x_{kij} + y_{kij}) = 0, \\ \sum\limits_{i:(j,i)\in A} (x_{kji} + y_{kji}) = 0, \\ t_{kj} = 0, \end{array}\right\} \qquad \forall k \in K, \forall j \in N_k. \tag{5.20}$$

Pour $\forall k \in K, \forall j \in N \backslash \{s_k\}$, l'heure plus tôt au nœud j est soit zéro, soit $st_k + l(s_k, j)$. Ces deux cas peuvent être représentés comme suit :

$$t_{kj} \geq (st_k + l(s_k, j)) \sum_{i:(i,j)\in A} (x_{kij} + y_{kij}), \qquad \forall k \in K, \forall j \in N \backslash \{s_k\}. \tag{5.21}$$

(5.20) et (5.21) peuvent alors être ajoutées comme de nouvelles contraintes au modèle P'_t et un nouveau modèle P''_t est obtenu. Il est défini comme suit :

$$\begin{aligned} P''_t : \quad & \min \sum_{(i,j)\in A} a_{ij} z_{ij} \\ \text{s.t.} \quad & \text{constraintes } (5.2) - (5.10) \text{ et } (5.12) - (5.21). \end{aligned}$$

L'approche de résolution pour le LRP-TT est basée sur la méthode de "cut-and-solve", mais la génération de "piercing cut" (PC_n) est différente de celle dans les chapitres précédents. Au lieu de la relaxation linéaire (toutes les variables entières sont relaxées en des variables continues), une relaxation partielle (i.e., variables entières x_{kij} et z_{ij} sont relaxées en des variables continues, et b_{kiq} et y_{kij} ne sont pas relaxées) est appliquée au "current problem" (CP_n) pour générer la PC_n. Car si la relaxation linéaire est appliquée, la borne inférieure du "remaining problem" (RP_n) est très mauvaise, ce qui ne contribue pas à la convergence de la méthode "cut-and-solve". Par la suite, le modèle de la relaxation partielle du CP_n est résolu, une solution fractionnaire sol^*_n est obtenue. K_n est défini comme l'ensemble des tâches qui ont un multi-chemin à partir de leurs sources vers les destinations. Pour chaque $k \in K_n$, une voie critique (i_k, j_k) avec la plus grande valeur dans le multi-chemin de la tâche k dans sol^*_n est trouvée. Ensuite, V_n est défini comme l'ensemble des voies critiques (i_k, j_k) des tâches $k \in K_n$. La PC_n est defini comme suit :

$$PC_n : \quad \sum_{(i_k j_k)\in V_n} x_{ki_kj_k} + y_{ki_kj_k} \leq h_n - 1,$$

où h_n est un nombre entier donné. Le "sparse problem" (SP_n) et RP_n sont définis comme suit :

$$\begin{aligned} SP_n : \quad & \min \sum_{(i,j)\in A} a_{ij} Z_{ij} \\ \text{s.t.} \quad & \text{constraintes } (5.2) - (5.10) \text{ et } (5.12) - (5.21) \\ & \sum_{(i_k j_k)\in V_m} x_{ki_kj_k} + y_{ki_kj_k} \leq h_n - 1, \qquad m = 1, 2, \ldots, n-1. \qquad (5.22) \\ & \sum_{(i_k j_k)\in V_n} x_{ki_kj_k} + y_{ki_kj_k} \geq h_n. \qquad (5.23) \end{aligned}$$

$$RP_n: \quad \min \sum_{(i,j)\in A} a_{ij} Z_{ij}$$

$$\text{s.t.} \quad \text{constraintes } (5.2)-(5.10), (5.12)-(5.21), \text{ et } (5.22)$$

$$\sum_{(i_k j_k)\in V_n} x_{k i_k j_k} + y_{k i_k j_k} \leq h_n - 1 \tag{5.24}$$

Comme dans les chapitres précédents, l'itération de la "cut-and-solve" est répétée jusqu'à ce que la borne inférieure du RP_n devienne supérieure ou égale à la meilleure borne supérieure.

Expériences numériques

La performance de l'algorithme proposé a été évaluée par des instances générées au aléatoirement. Les expériences numériques montrent que : 1), l'algorithme proposé peut résoudre exactement la plus grande instance constituée de 95 nœuds du réseau. 2), les temps CPU moyens de l'algorithme proposé sont réduisent de 65.50%, 74.44%, et 67.19% de ceux de CPLEX pour les instances avec différents nombres d'intervalles de temps, différents impacts des voies réservées et différents délais prescrits pour compléter la tâche.

5.2 Problème de reservation des voies avec vitesse variable

Le problèm "lane reservation problem with time-dependent travel speed" (LRP-TS) est similaire au LRP-TT sauf sur un point. Dans le LRP-TT, le temps de parcours sur une voie non réservée (i, j) est une constante si l'heure de départ au nœud i est dans le même intervalle de temps. Dans le LRP-TS, la vitesse des véhicules est une constante si l'heure est dans le même intervalle de temps. Le temps de parcours sur une voie non réservée (i, j) dans le LRP-TS est donc relatif à la vitesse sur la voie, pas uniquement l'heure de départ au nœud i. D'abord, le temps de parcours sur une voie non réservée est calculée selon la procédure dans [40]. C'est une fonction continue de l'heure de départ au nœud i linéaire par morceaux. De plus, on suppose que le principe "first-in-first-out" est satisfait.

Le LRP-TS vise à de réserver des voies dans le réseau $G(N, A)$ afin de respecter les temps de transport pour chaque tâche $k \in K$ en minimisant l'impact de toutes les voies réservées. Le temps de parcours sur une voie non réservée est une fonction continue linéaire par morceaux et le temps de parcours sur une voie réservée est une constante.

Les notations pour le LRP-TS sont présentées comme suit.

Paramètres

A :	l'ensemble des arcs orientés $(i,j), i \neq j, i,j \in N$
K :	l'ensemble des tâches de transport $k \in K$
N :	l'ensemble des nœuds
a_{ij} :	l'impact de la réservation d'une voie $(i,j) \in A$
d_k :	le nœud destination de la tâche $k \in K$
p_k :	la durée de trajet maximale souhaitée pour la tâche $k \in K$
s_k :	le nœud source de la tâche $k \in K$
st_k :	l'heure de départ de la tâche $k \in K$ au nœud s_k
τ_{ij} :	la durée de traversée de la voie réservée $(i,j) \in A$
$\tau^*_{ij}(t)$:	la durée de traversée de la voie non réservée $(i,j) \in A$ si l'heure de départ au nœud i est t

Variables de décision

t_{ki}	l'heure de départ de la tâche $k \in K$ au nœud $i \in N$; $t_{ki} = 0$, si la tâche k ne visite pas le nœud i
x_{kij}	$x_{kij} = 1$, si la tâche $k \in K$ traverse la voie réservée $(i,j) \in A$; $x_{kij} = 0$, sinon
y_{kij}	$y_{kij} = 1$, si la tâche $k \in K$ traverse la voie non réservée $(i,j) \in A$; $y_{kij} = 0$, sinon
z_{ij}	$z_{ij} = 1$, si la voie $(i,j) \in A$ est réservée; $z_{ij} = 0$, sinon

Le LRP-TT est formulé comme le problème d'optimisation suivant :

$$P_s : \min \sum_{(i,j)\in A} a_{ij} z_{ij} \tag{5.25}$$

$$\text{s.t.} \sum_{i:(s_k,i)\in A} (x_{ks_ki} + y_{ks_ki}) = 1, \qquad \forall k \in K, \tag{5.26}$$

$$\sum_{i:(i,d_k)\in A} (x_{kid_k} + y_{kid_k}) = 1, \qquad \forall k \in K, \tag{5.27}$$

$$\sum_{i:(i,j)\in A} (x_{kij} + y_{kij}) = \sum_{i:(j,i)\in A} (x_{kji} + y_{kji}), \qquad \forall k \in K, \forall j \in N \backslash \{s_k, d_k\}, \tag{5.28}$$

$$x_{kij} \leq z_{ij}, \qquad \forall k \in K, \forall (i,j) \in A, \tag{5.29}$$

$$y_{kij} + z_{ij} \leq 1, \qquad \forall k \in K, \forall (i,j) \in A, \tag{5.30}$$

$$t_{ks_k} = st_k, \qquad \forall k \in K, \tag{5.31}$$

$$t_{kj} = \sum_{i:(i,j)\in A} \left((t_{ki} + \tau_{ij}) x_{kij} + (t_{ki} + \tau^*_{ij}(t_{ki})) y_{kij} \right), \qquad \forall k \in K, \forall j \in N \backslash \{s_k\}, \tag{5.32}$$

$$t_{kd_k} - t_{ks_k} \leq p_k, \qquad \forall k \in K, \tag{5.33}$$

$$t_{ki} \geq 0, \qquad \forall k \in K, \forall i \in N, \tag{5.34}$$

$$x_{kij}, y_{kij} \in \{0,\ 1\}, \qquad \forall k \in K, \forall (i,\ j) \in A, \tag{5.35}$$

$$z_{ij} \in \{0,\ 1\}, \qquad \forall (i,\ j) \in A. \tag{5.36}$$

Le modèle P_s du LRP-TS est similaire au modèle P_t du LRP-TT mais diffère par les deux points suivants : 1), il y a quatre types de variables de décision dans P_s et cinq types de variables de décision dans P_t. Parce que le temps de parcours sur une voie non réservée dans LRP-TT dépend de l'intervalle de temps, nous avons donc besoin de variables de décision b_{kiq} et des contraintes (5.7)–(5.9) pour indiquer cette information dans le modèle P_t; 2), les contraintes (5.11) dans P_t et (5.32) dans P_s sont différentes. τ''_{ijq} dans (5.11) est une constante et $\tau^*_{ij}(t_{ki})$ dans (5.32) est une fonction de t_{ki}. Il est plus difficile de résoudre le modèle P_s que le modèle P_t en raison de la fonction $\tau^*_{ij}(t_{ki})$.

Le LRP-TS est NP-difficile, car un cas particulier de celui-ci (si le temps de parcours $\tau^*_{ij}(t)$ est très grand pour tout $(i,j) \in A$ et $t \in [T_q, T_{q+1}), q \in Q$, alors chaque voie dans le chemin de la tâche doit être réservée) peut être réduit au LRP, qui a été aussi démontré NP-difficile dans chapitre 3.

Reformulation du modèle

Le modèle P_s ne peut pas être résolu par CPLEX. Il peut être transformé en un modèle équivalent par reformulation de la contrainte (5.32) selon les trois cas suivants : 1), la tâche k ne visite pas j; 2), la tâche k visite j via une voie réservée (i,j); 3), la tâche k visite j via une voie non réservée (i,j). Pour chaque cas, (5.32) peut être transformée en les contraintes suivantes (5.37), (5.38), et (5.39) :

$$t_{kj} \le M \sum_{i:(i,j)\in A} (x_{kij} + y_{kij}), \qquad \forall k \in K, \forall j \in N\backslash\{s_k\}, \tag{5.37}$$

$$\left.\begin{array}{l} t_{kj} - t_{ki} - \tau_{ij} \le M(1 - x_{kij}), \\ t_{kj} - t_{ki} - \tau_{ij} \ge M(x_{kij} - 1), \end{array}\right\} \qquad \forall k \in K, \forall j \in N\backslash\{s_k\}, (i,j) \in A. \tag{5.38}$$

$$\left.\begin{array}{l} t_{kj} - t_{ki} - \tau^*_{ij}(t_{ki}) \le M(1 - y_{kij}), \\ t_{kj} - t_{ki} - \tau^*_{ij}(t_{ki}) \ge M(y_{kij} - 1), \end{array}\right\} \qquad \forall k \in K, \forall j \in N\backslash\{s_k\}, (i,j) \in A. \tag{5.39}$$

où M est une très grande constante. Le modèle P'_s est donc représenté comme suit :

$$\begin{array}{ll} P'_s: & \min \displaystyle\sum_{(i,j)\in A} a_{ij} z_{ij} \\ \text{s.t.} & \text{constraintes } (5.26) - (5.31) \text{ et } (5.33) - (5.39). \end{array}$$

Bien que le modèle P'_s ne soit pas strictement linéaire en raison de la fonction τ^*_{ij}, il est linéaire par morceaux et peut être résolu par CPLEX.

Approche de résolution

L'approche de résolution pour le LRP-TS est fondée sur la méthode de "tabu search", qui est une stratégie de recherche itérative. A chaque itération de la méthode, un "neighborhood" est généré autour de la "current solution" (pour la première itération, il est pris comme "initial solution"). Le "neighborhood" est un ensemble de solutions réalisables, qui peut être créé par une opération appelée "move" à partir de "current solution". La meilleure solution admissible dans ce "neighborhood" est choisi et devient la "current solution" pour la prochaine itération. La meilleure solution est mise à jour si la meilleure solution admissible est meilleure qu'elle. Pour éviter de retomber dans le minimum local auquel on vient d'échapper, le mécanisme ("tabu list") est créé pour interdire de revenir aux dernières "moves" explorées.

La solution initiale est construite sur la base d'un algorithme modifié pour la recherche du plus court chemin dans un réseau avec des temps de parcours dynamiques sur les voies non réservées. L'idée est de les transformer en voies réservées une à une jusqu'à ce que chaque tâche puisse être réalisée dans le délai prescrit.

Les solutions dans le "neighborhood" sont relatives à trois types de "move" : 1), ajouter une nouvelle voie réservée (i', j') au réseau; 2), supprimer une voie réservée $(\overline{i}, \overline{j})$ du réseau; 3), ajouter une nouvelle voie réservée (i', j') au réseau et supprimer une voie réservée $(\overline{i}, \overline{j})$ du réseau. Le chemin le plus court pour chaque tâche est recalculé par l'algorithme modifié pour la recherche du plus court chemin. Si la plus courte durée de parcours de chaque tâche n'est pas supérieure au délai prescrit, le "move" est réalisable. Ensuite, la meilleure solution admissible est choisie et les voies correspondant (i', j') et/ou $(\overline{i}, \overline{j})$ sont interdites pour quelques itérations prochaines. L'itération de la "tabu search" est répétée jusqu'à ce qu'un nombre maximal d'itération soit atteint.

Expériences numériques

La performance de l'algorithme proposé a été évaluée sur des instances générées au aléatoirement. Les résultats numériques montrent que l'algorithme proposé peut résoudre des problèmes jusqu'à 100 nœuds et 35 tâches avec un temps de calcul CPU raisonnable (moins de 30 minutes). Pour les instances de petite taille (25–55 nœuds et 7–10 tâches), l'écart maximal entre la borne supérieure et la solution optimale est 5.42%. Pour les instances de grande taille (80–100 nœuds et 25–35 tâches), CPLEX ne peut pas trouver une solution réalisable. L'écart maximal entre la borne supérieure donnée par l'algorithme proposé et la borne inférieure obtenue par CPLEX est de

56.42%.

Chapitre 6 Conclusions et perspectives

Dans cette thèse, nous avons étudié quatre problèmes de réservation des voies dans les réseaux de transport. L'objectif de la thèse est de fournir une aide à la décision pour mettre en place la stratégie de réservation des voies. D'abord, nous avons considéré le LRP avec le temps de parcours statique sur une voie dans chapitre 3. Ensuite, le CLRP dans chapitre 4 prend en compte la capacité des voies. Enfin, nous avons considéré dans le chapitre 5 les LRP-TT et LRP-TS avec le temps de parcours dynamique sur les voies non réservées. Pour les quatre problèmes, nous avons développé de nouveaux modèles mathématiques du plus basique au plus élaboré (programmation linéaire en nombres entiers ou programmation non linéaire en nombres mixtes). Après l'étude des propriétés de chaque problème, nous avons proposé des méthodes de résolution : la méthode fondée sur "cut-and-solve" pour le LRP et LRP-TT; la méthode combinée "cut-and-solve" et méthode de coupe pour le CLRP; et la méthode utilisant la recherche tabou pour le LRP-TS. Les algorithmes proposés permettent de trouver des solutions optimales ou proches de l'optimum. Les performances des algorithmes proposés sont évaluées par des expériences numériques.

Plusieurs études restent à faire pour la recherche future. D'abord, l'impact de la seule voie réservée est considéré comme paramètre d'entrée des problèmes. En fait, ce paramètre est très complexe car de nombreux facteurs, comme le type des voies et l'emplacement des voies réservées dans le réseau, y contribuent. Il est nécessaire d'étudier systématiquement ce paramètre. Il devient difficile de résoudre le LTP-TT et LRP-TS pour des instances de grande taille, en particulier le LTP-TS. Un algorithme efficace doit être développé pour trouver une borne inférieure de meilleure que celle obtenue par CPLEX enfin d'évaluer la recherche tabou proposée dans chapitre 5.

Enfin, avec le volume de transport croissant au quotidien et le processus d'intégration de l'économie mondiale, bien gérer le trafic devient de plus en plue important. La stratégie de réservation des voies fournit une alternative pour les gestionnaires. Comme pour toute stratégie de gestion de la circulation, il est nécessaire de bien examiner tous les effets positifs et négatifs avant de l'appliquer les décissions dans la vie réelle. Enfin, cette stratégie doit aussi s'intégrer dans la conception de systèmes de transport intelligents.

Bibliography

[1] Approximation algorithm. `http://en.wikipedia.org/wiki/Approximation_algorithm`.

[2] HOV lane/current evaluation results. Oregon Department of Transportation, 2001.

[3] A. F. Abdelghany, K. F. Abdelghany, H. S. Mahmassani, and P. M. Murray. Dynamic traffic assignment in design and evaluation of high-occupancy toll lanes. *Transportation Research Record: Journal of the Transportation Research Board*, 1733:39–48, 2000.

[4] R. Ahuja, T. Magnanti, and J. Orlin. Network flows: theory, algorithms, and applications. 1993.

[5] K. S. Al-Sultan and M. A. Al-Fawzan. A tabu search approach to the uncapacitated facility location problem. *Annals of Operations Research*, 86:91–103, 1999.

[6] V. T. Arasan and P. Vedagiri. Bus priority on roads carrying heterogeneous traffic: a study using computer simulation. *European Journal of Transport and Infrastructure Research*, 8(1):45–64, 2008.

[7] V. T. Arasan and P. Vedagiri. Study of the impact of exclusive bus lane under highly heterogeneous traffic condition. *Public Transport*, 2(1–2):135–155, 2010.

[8] N. Azi, M. Gendreau, and J.-Y. Potvin. An exact algorithm for a single-vehicle routing problem with time windows and multiple routes. *European journal of operational research*, 178(3):755–766, 2007.

[9] N. Bansal, A. Blum, S. Chawla, and A. Meyerson. Approximation algorithms for deadline-tsp and vehicle routing with time-windows. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, Chicago, USA, 2004. ACM.

[10] M. Burris and E. Sullivan. Benefit-cost analysis of variable pricing projects: QuickRide HOT lanes. *Journal of Transportation Engineering*, 132(3):183–190, 2006.

[11] I. Chabini. A new algorithm for shortest paths in discrete dynamic networks. In *Proceedings of the 8th IFAC Symposium on Transportation Systems*, Chania, Greece, 1997.

[12] M.-S. Chang, S.R. Chen, and C.-F. Hsueh. Real-time vehicle routing problem with time windows and simultaneous delivery/pickup demands. *Journal of the Eastern Asia Society for Transportation Studies*, 5:2273–2286, 2003.

[13] Z. Chen and H. Xu. Dynamic column generation for dynamic vehicle routing with time windows. *Transportation Science*, 40(1):74–88, 2006.

[14] D. Choi and W. Choi. Effects of an exclusive bus lane for the oversaturated freeway in Korea. In *Institute of Transportation Engineers 65th Annual Meeting*, 1995.

[15] F. A. Chudak and D. B. Shmoys. Improved approximation algorithms for the uncapacitated facility location problem. *SIAM Journal on Computing*, 33(1):1–25, 2003.

[16] S. Climer and W. Zhang. Cut-and-solve: An iterative search strategy for combinatorial optimization problems. *Artificial Intelligence*, 170(8-9):714–738, 2006.

[17] J.-F. Cordeau, G. Laporte, M. W. P. Savelsbergh, and D. Vigo. *Vehicle routing*. Handbooks in Operations Research and Management Science. North-Holland, Amsterdam, 2007.

[18] H. Crowder, E.L. Johnson, and M. Padberg. Solving large-scale zero-one linear programming problems. *Operations Research*, 31(5):803–834, 1983.

[19] E.W. Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959.

[20] C. Dobre, V. Cristea, and L. Iftode. ILRSH: Intelligent lane reservation system for highway(s). In *2012 Sixth International Conference on Complex, Intelligent, and Software Intensive Systems*, pages 747–754, Palermo, Italy, Jul. 2012.

[21] Marco Dorigo. *Optimization, learning and natural algorithms*. PhD thesis, Politecnico di Milano, Italy, 1992.

[22] S. Falbel, P. Rodriguez, H. Levinson, K. Younger, and S. Misiewicz. Bus rapid transit plans in New York's capital district. *Journal of Public Transportation*, 9:23–49, 2006.

[23] Y. Fang, F. Chu, S. Mammar, and A. Che. A cut-and-solve based algorithm for optimal lane reservation with dynamic link travel times. *International Journal of Production Research* (accepted).

[24] Y. Fang, F. Chu, S. Mammar, and A. Che. An optimal algorithm for automated truck freight transportation via lane reservation strategy. *Transportation Research Part C: Emerging Technologies*, 26:170–183, 2013.

[25] Y. Fang, F. Chu, S. Mammar, and M. Zhou. Optimal lane reservation in transportation network. *IEEE Transactions on Intelligent Transportation Systems*, 13(2):482–491, 2012.

[26] Federal Highway Administration. `http://ops.fhwa.dot.gov/publications/fhwahop09029/sec2_operational.htm`.

[27] R.W. Floyd. Algorithm 97: shortest path. *Communications of the ACM*, 5(6):345, 1962.

[28] C. Fuhs and J. Obenberger. Development of high-occupancy vehicle facilities: Review of national trends. *Transportation Research Record*, 1781:1–9, 2002.

[29] V. Gabrel, A. Knippel, and M. Minoux. Exact solution of multicommodity network optimization problems with general step cost functions. *Operations Research Letters*, 25(1):15–23, 1999.

[30] G. Gallo and S. Pallottino. Shortest path algorithms. *Annals of Operations Research*, 13(1):1–79, 1988.

[31] N. Garg and J. Köenemann. Faster and simpler algorithms for multicommodity flow and other fractional packing problems. *SIAM Journal on Computing*, 37(2):630–652, 2007.

[32] M. Gendreau, F. Guertin, J.-Y. Potvin, and E. Taillard. Parallel tabu search for real-time vehicle routing and dispatching. *Transportation science*, 33(4):381–390, 1999.

[33] F. Glover. Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research*, 13(5):533–549, 1986.

[34] F. Glover. Tabu search: A tutorial. *Interfaces*, 20(4):74–94, 1990.

[35] F. Glover, R. Glover, and D. Klingman. Computational study of an improved shortest path algorithm. *Networks*, 14(1):25–36, 1984.

[36] Fred Glover. Tabu search–part I. *ORSA Journal on computing*, 1(3):190–206, 1989.

[37] A. Haghani and S. Jung. A dynamic vehicle routing problem with time-dependent travel times. *Computers & operations research*, 32(11):2959–2986, 2005.

[38] J.H. Holland. *Adaptation in natural and artificial systems*. Ann Arbor, MI: University of Michigan Press, 1975.

[39] F. K. Hwang, D. S. Richards, and P. Winter. *The Steiner tree problem*. North Holland, Amsterdam, The Netherlands, 1992.

[40] S. Ichoua, M. Gendreau, and J.-Y. Potvin. Vehicle dispatching with time-dependent travel times. *European Journal of Operational Research*, 144(2):379–396, 2003.

[41] Liviu Iftode, Stephen Smaldone, Mario Gerla, and James Misener. Active highways (position paper). In *IEEE 19th International Symposium on Personal, Indoor and Mobile Radio Communications*, pages 1–5, Cannes, France, Sept. 2008.

[42] K. Kaparis and A.N. Letchford. Separation algorithms for 0-1 knapsack polytopes. *Mathematical programming*, 124(1):69–91, 2010.

[43] S. Karim. The effect of bus lane on the travel time of other modes using floating car method. In *Proceedings of the Eastern Asia Society for Transportation Studies*, pages 135–149, Fukuoka, Japan, Oct. 2003.

[44] R.M. Karp. *Reducibility among combinatorial problems*. Complexity of Computer Computations. Plenum Press, New York, 1972.

[45] D. E. Kaufman and R. L. Smith. Fastest paths in time-dependent networks for intelligent vehicle-highway systems application. *Journal of Intelligent Transportation Systems*, 1(1):1–11, 1993.

[46] S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi. Optimization by simmulated annealing. *Science*, 220(4598):671–680, 1983.

[47] J. Kwon and P. Varaiya. Effectiveness of California's high occupancy vehicle (HOV) system. *Transportation Research Part C: Emerging Technologies*, 16(1):98–115, 2008.

[48] T. Leighton, F. Makedon, S. Plotkin, C. Stein, E. Tardos, and S. Tragoudas. Fast approximation algorithms for multicommodity flow problems. *Journal of Computer and System Sciences*, 50(2):228–243, 1995.

[49] H. S. Levinson, S. Zimmerman, J. Clinger, and J. Gast. Bus rapid transit: synthesis of case studies. *Transportation Research Record*, 1841:1–11, 2003.

[50] H. S. Levinson, S. Zimmerman, J. Clinger, and S. C. Rutherford. Bus rapid transit: An overview. *Journal of Public Transportation*, 5(2):1–30, 2002.

[51] R. B. Machemehl, T. W. Rioux, A. Tsyganov, and P. Poolman. Freeway operational flexibility concepts. Technical report, NO. 1844-1, Center for Transportation Research, University of Texas at Austin, Austin, Texas, 2001.

[52] C Maclennan. Priority for public transport and other high occupancy vehicles on urban roads. *Routes/Roads*, Special II(10.07 A):5–38, 1995.

[53] C. Malandraki and M. S. Daskin. Time dependent vehicle routing problems: Formulations, properties and heuristic algorithms. *Transportation Science*, 26(3):185–200, 1992.

[54] C. Malandraki and R. B. Dial. A restricted dynamic programming heuristic algorithm for the time dependent traveling salesman problem. *European Journal of Operational Research*, 90(1):45–55, 1996.

[55] P. Martin, J. Perrin, R. Lambert, and P. Wu. Evaluate effectiveness of high occupancy vehicle (HOV) lanes. Technical report, NO. UTL-1001-48, Civil & Environmental Engineering Department, University of Utah, Salt Lake City, Utah, 2002.

[56] P. M. Murray, H. S. Mahmassani, and K. F. Abdelghany. Methodology for assessing high-occupancy toll-lane usage and network performance. *Transportation Research Record: Journal of the Transportation Research Board*, 1765:8–15, 2001.

[57] B. G. Perez, G.-C. Sciara, and P. Brinckerhoff. *A guide for HOT lane development.* Federal Highway Administration, US Department of Transportation, `http://ntl.bts.gov/lib/jpodocs/repts_te/13668_files/chapter_1.htm`, 2003.

[58] D. T. Pham and D. Karaboga. *Intelligent Optimisation Techniques: Genetic Algorithms, Tabu Search, Simulated Annealing and Neural Networks.* Springer, London, 2000.

[59] R. W. Poole Jr and C. K. Orski. HOT lanes: a better way to attack urban highway congestion. *Regulation*, 23(1):15–20, 2000.

[60] J. Princeton and S. Cohens. Impact of a dedicated lane on the capacity and the level of service of an urban motorway. *Procedia Social and Behavioral Sciences*, 16:196–206, 2011.

[61] R. Ramakers, K. Henning, S. Gies, D. Abel, and H. Max. Electronically coupled truck platoons on german highways. In *2009 IEEE International Conference on Systems, Man and Cybernetics*, pages 2409–2414, Oct. 2009.

[62] N. Ravi, S. Smaldone, L. Iftode, and M. Gerla. Lane reservation for highways (position paper). In *Proceeding of IEEE International Conference on Intelligent Transportation Systems*, pages 795–800, Seattle, WA, USA, Aug. 2007.

[63] M. G. C. Resende and R. F. Werneck. A hybrid multistart heuristic for the uncapacitated facility location problem. *European Journal of Operational Research*, 174(1):54–68, 2006.

[64] S. Sarin, A. S. Sarna, and B. Sharme. Experience with bus lanes under mixed traffic conditions. In *Institute of Transportation Engineers 53rd Annual Meeting*, 1983.

[65] S. Schijns and P. Eng. Brisbane, Australia–HOV metropolis? In *10th International Conference on High–Occupancy Vehicle Systems*, Dallas, Texas, Aug. 2000.

[66] A. S. Shalaby. Simulating performance impacts of bus lanes and supporting measures. *Journal of transportation engineering*, 125(5):390–397, 1999.

[67] S. E. Shladover. Truck automation operational concept alternatives. In *2010 IEEE Intelligent Vehicles Symposium*, pages 1072–1077, San Diego, USA, Jun. 2010.

[68] D. B. Shmoys, É. Tardos, and K. Aardal. Approximation algorithms for facility location problems. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, El Paso, Texas, USA, 1997.

[69] N. Smith and D. Hensher. The future of exclusive busways: the Brazilian experience. *Transport Reviews*, 18(2):131–152, 1998.

[70] M. M. Solomon. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, 35(2):254–265, 1987.

[71] E. Sullivan and M. Burris. Benefit-cost analysis of variable pricing projects: SR-91 express lanes. *Journal of transportation engineering*, 132(3):191–198, 2006.

[72] D. Tcha and B. Lee. A branch-and-bound algorithm for the multi-level uncapacitated facility location problem. *European Journal of Operational Research*, 18(1):35–43, 1984.

[73] J. A. Tomlin. Minimum-cost multicommodity network flows. *Operations Research*, 14(1):45–51, 1966.

[74] S. Tragantalerngsak, J. Holt, and M. Rönnqvist. Lagrangian heuristics for the two-echelon, single-source, capacitated facility location problem. *European Journal of Operational Research*, 102(3):611–625, 1997.

[75] H.-S. J. Tsao and J. L. Botha. An automated highway system dedicated to intercity trucking: Design options, operating concepts, and deployment. *Journal of Intelligent Transportation Systems: Technology, Planning, and Operations*, 7(2):169–196, 2002.

[76] H.S.J. Tsao and J.L. Botha. Definition and evaluation of bus and truck automation operations concepts. Technical report, No. UCB-ITS-PRR-2003-19, California Partners for Advanced Transit and Highways (PATH), Institute of Transportation Studies (UCB), UC Berkeley, 2003.

[77] S. Tsugawa, S. Kato, and K. Aoki. An automated truck platoon for energy saving. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4109–4114, San Francisco, CA, USA, Aug. 2011.

[78] K. F. Turnbull. HOV project case studies: history and institutional arrangements. Technical report, NO. DOT-T-92-13, Texas Transportation Institute, Texas A & M University System, College Station, Texas, 1990.

[79] K. F. Turnbull. HOV and HOT lanes in the United States. In *Proceedings of European Transport Conference*, Strasbourg, France, Sept. 2005.

[80] K. F. Turnbull and T. DeJohn. New Jersey I-80 and I-287 HOV lane case study. Technical report, NO. FHWAOP-00-018, Texas Transportation Institute, Texas A & M University System, College Station, Texas, 2000.

[81] B. M. Waxman. Routing of multipoint connections. *IEEE Journal on Selected Areas in Communications*, 6(9):1617–1622, 1988.

[82] L. Wei and T. Chong. Theory and practice of bus lane operation in Kunming. *DISP*, 151:68–72, 2002.

[83] Y. Wu, C. Chu, F. Chu, and N. Wu. Heuristic for lane reservation problem in time constrained transportation. In *Proceedings of Automation Science and Engineering*, pages 543–548, Bangalore, India, Aug. 2009.

[84] Y. Wu and N. Wu. An approximate algorithm for the lane reservation problem in time constrained transportation. In *2010 2nd International Conference on Advanced Computer Control (ICACC)*, pages 192–196, Shenyang, China, Mar. 2010.

[85] H. Yang and W. Wang. An innovative dynamic bus lane system and its simulation-based performance investigation. In *Proceedings of Intelligent Vehicles Symposium*, Xi'an, China, Jun. 2009.

[86] Zhen Yang, Feng Chu, and Haoxun Chen. A cut-and-solve based algorithm for the single-source capacitated facility location problem. *European Journal of Operational Research*, 221(3):521–532, 2012.

[87] A. K. Ziliaskopoulos and H. S. Mahmassani. Time-dependent,shortest-path algorithm for real-time intelligent vehicle highway system applications. *Transportation Research Record*, (1408):94–100, 1993.

Yunfei FANG

Doctorat : Optimisation et Sûreté des Systèmes

Année 2013

Etude des problèmes de réservation de voies dans un réseau de transport

Le concept de réservation de voies a été présenté comme une stratégie de gestion du trafic et a de nombreuses applications dans la vie réelle. Des études antérieures dans la littérature se concentrent principalement sur l'impact de la réservation de voies dans une région locale d'un réseau de transport. Dans cette thèse, les problèmes de réservation de voies sont étudiés dans le but de minimiser l'impact total sur le trafic par la réservation de voies dans un réseau de transport. Nous avons d'abord étudié le problème de réservation de voies (LRP) pour les poids lourds automatisé avec temps de déplacement statique. Ce travail est généralisé au problème de réservation de voies avec une capacité limitée de la voie (CLRP) pour les grands événements spéciaux. Enfin, le problème de réservation de voies avec le temps de déplacement dynamique (LRP-TT) et le problème de réservation de voies avec la vitesse de déplacement dynamique (LRP-TS) sont étudiés. Pour chacun des problèmes étudiés, les modèles mathématiques appropriés sont formulés, leurs complexités sont démontrées. Différentes méthodes de résolution sont explorées, y compris une méthode exacte cut-and-solve, une méthode de cut-and-solve et plan de coupe combinée et une méthode de recherche tabou. Les performances des algorithmes proposés sont évaluées par des instances générées aléatoirement. Les résultats numériques ont montré que les algorithmes proposés sont plus efficaces que le logiciel commercial CPLEX.

Mots clés : réservations de voie - problèmes de transport (programmation) - optimisation combinatoire - transport de marchandises.

Study of Lane Reservation Problems in a Transportation Network

The concept of lane reservation has been introduced as a traffic management strategy and has many applications in real life. Previous studies in the literature mainly focus on the impact of lane reservation in a local region of transportation network. In this thesis optimal lane reservation problems are studied with the objective to minimize impact on total traffic by optimally setting reserved lanes in a transportation network. We firstly focus on the lane reservation problem (LRP) for automated truck freight transportation with static link travel time. This primary work has been extended to the capacitated lane reservation problem (CLRP) for large-scale special events. Finally, lane reservation problem with time-dependent travel time (TTLRP), and lane reservation problem with time-dependent travel speed (TSLRP) are studied. For each of the considered problems, appropriate mathematical models are formulated, their complexities are demonstrated. Different resolution methods are explored including exact cut-and-solve method, cut-and-solve and cutting plane combined method, and Tabu-search method. The performance of the proposed algorithms is evaluated by randomly generated instances. Numerical results have shown that the proposed algorithms are more effective to solve the considered problems than the reference commercial package CPLEX.

Keywords: reservation systems - transportation problems (programming) - combinatorial optimization - freight and freightage.

Thèse réalisée en partenariat entre :

utt
université de technologie
Troyes

Ecole Doctorale "Sciences et Technologies"