Problèmes d'ordonnancement intégré entre la production et le transport avec stocks intermédiaires et prise en compte de dates dues

Deyun Wang

N° d'ordre : 178 Année 2012

Université de Technologie de Belfort-Montbéliard

Ecole Doctorale Sciences Physiques pour l'Ingénieur et Microtechniques

THÈSE

Présentée en vue de l'obtention du titre de

DOCTEUR

de

l'Université de Technologie de Belfort-Montbéliard

Spécialité : Automatique

Par

DEYUN WANG

Integrated Scheduling of Production and Transportation Operations with Stage-Dependent Inventory Costs and Due Dates Considerations

---

Soutenue le 26 avril 2012 devant le jury composé de:

Rapporteurs :	Imed KACEM	-	Professeur, Université Paul Verlaine - Metz
	Laurent GENESTE	-	Professeur, Ecole Nationale d'Ingnieurs de Tarbes
Examinateurs :	Farouk YALAOUI	-	Professeur, Université de Technologie Troyes
	Christian TAHON	-	Professeur, Université de Valenciennes et du Hainaut-Cambrésis
Directeur :	Abdellah EL MOUDNI	-	Professeur, Université de Technologie de Belfort-Montbéliard
Co-directeur :	Olivier GRUNDER	-	Maître de Conferences, Université de Technologie de Belfort-Montbéliard

Thèse préparée au sein du laboratoire Systèmes et Transport, UTBM, Belfort, France

ABSTRACT

Increasing global competition in the business world and heightened expectations of customers have forced companies to consider not only the pricing or product quality, but reliability and timeliness of the deliveries as well. In manufacturing-centric industries such as automotive and electronics, distribution and inventory costs constitute the second and third largest cost components following the production costs. Therefore, industrial and logistics companies need to continuously search for ways to lower the inventory level and distribution cost. This trend has created a closer interaction between the different stages of a supply chain, and increased the practical usefulness of the integrated models.

This thesis considers two categories of integrated scheduling problems. One is Integrated Scheduling of Production-Distribution-Inventory problems (ISPDI problems) and the other is Integrated Scheduling of Production-Inventory-Distribution-Inventory problems (ISPIDI problems). Jobs are first processed on a single machine in the production stage, and then delivered to a pre-specified customer by a capacitated transporter. Each job has a distinct due date, and must be delivered to customer before this due date. Each production batch requires a setup cost and a setup time before the first job of this batch is processed. Each round trip between the factory and customer requires a delivery cost as well as a delivery time. Moreover, it is assumed that a job which is completed before its departure date or delivered to the customer before its due date will incur a corresponding inventory cost. Our objective is to minimize the total cost involving setup, inventory and delivery costs while guaranteeing a certain customer service level.

For ISPDI problems, we firstly provide a mixed integer programming model for the case of multi-product, single-stage situation, and develop an improved Genetic algorithm (GA) for solving it. Then, we extend this model to a single-product, multi-stage model, and provide two

methods, dominance-related greedy algorithm and GA, for solving it. For ISPIDI problems, we establish a general non-linear model for the case of single-product situation and devise a special case from the general model. Then we provide an optimality property between the production and delivery schedules for the special case. Finally, a heuristic approach is developed for solving it. For each problem under study, in order to evaluate the performance of the proposed algorithms, some interesting lower bounds on the corresponding objective functions are established according to different methods such as lagrangian relaxation method, classical bin-packing based method. Computational results show the efficiency of the proposed models and algorithms in terms of solution quality and running time.

Keyword: Coordinated scheduling; Production and distribution; Integration; Supply chain management; Heuristic; Batching; Inventory; Complexity; Mixed integer programming

RÉSUMÉ

L'augmentation de la concurrence économique internationale et les attentes accrues des clients ont imposé aux entreprises de prendre en compte non seulement le prix ou la qualité du produit, mais également la fiabilité et la rapidité des livraisons. Dans les industries ayant une composante manufacturière dominante telles que l'automobile et l'électronique, la distribution et les coûts de stockage constituent les deuxième et troisième catégories de coûts les plus importantes après les coûts de production. Par conséquent, les entreprises industrielles et de logistique recherchent continuellement des méthodes pour réduire le niveau des stocks et les coûts de distribution. Cette tendance a créé une interaction plus forte entre les différentes étapes de la chane logistique, et augmente de ce fait l'utilité pratique des modèles intégrés.

Cette thèse considère deux catégories de problèmes d'ordonnancement intégré. La première catégorie est l'ordonnancement intégré de la production, distribution et stockage (Integrated Scheduling of Production-Distribution-Inventory, ISPDI) et la deuxième est l'ordonnancement intégré de la production, stockage, distribution et stockage (Integrated Scheduling of Production-Inventory-Distribution-Inventory, ISPIDI). Au niveau de la production, les tâches à réaliser sont traitées sur une seule machine et regroupées par lot de production, ce qui nécessite un coût et un temps de réglage. Elles doivent ensuite être livrées à un client prédéfini par un transporteur à capacité limitée, avant des dates dues données. Chaque aller-retour du transporteur entre l'usine et le client implique un coût de livraison et des délais de livraison. De plus, on suppose que les tâches qui sont terminées avant leur date de départ ou qui sont livrées au client avant leur date due entraînent un coût de stockage supplémentaire. Notre objectif est de minimiser le coût total comprenant les coûts de reglage, de stockage et de transport, tout en garantissant un niveau de service donné pour le client.

Pour les problèmes ISPDI, nous avons d'abord fourni un modèle de programmation mixte entière pour le problème multi-produits, à un seul niveau, et avons développé un algorithme génétique amélioré pour le résoudre. Puis, nous avons modifié ce modèle pour prendre en compte le cas mono-produit, multi-niveau, et avons proposé deux méthodes, un algorithme hybride et un algorithme génétique, pour le résoudre. Pour les problèmes ISPIDI, nous avons établi un modèle général non-linéaire dans le cas mono-produit, et avons traité un cas spécifique du cas général. Puis nous avons démontré une propriété d'optimalité qui lie les ordonnancements de production et de livraison dans le cas particulier, pour finalement proposer une approche heuristique pour le résoudre. Pour chaque problème étudié et afin d'évaluer les performances des algorithmes proposés, des limites inférieures intéressantes sur les fonctions objectifs correspondantes ont été établies selon des méthodes différentes telles que la méthode de relaxation lagrangienne ou des méthodes basées sur les bornes inférieures du problème de bin packing. Les résultats des expérimentations montrent l'efficacité des modèles et algorithmes proposés en termes de qualité de la solution et de temps d'exécution.

Mots clefs: Ordonnancement intégré; Production et distribution; Planification de la chane logistique; Meta-Heuristique; Lots; Complexité; Programmation en nombres entiers mixte

PREFACE

The work presented in the thesis entitled “Integrated Scheduling of Production and Transportation Operations with Stage-Dependent Inventory costs and Due Dates Considerations” has been prepared by Deyun Wang during the period September 2008 to April 2012, in the Laboratoire Systèmes et Transports (SET), at Université de Technologie de Belfort-Montbéliard (UTBM).

The Ph.D. project has been supervised by my advisor professors Abdellah EL Moundni and Olivier Grunder. This thesis addresses two main integrated scheduling problems, namely ISPDI and ISPIDI problems, and provides several heuristic approaches for solving them. The results of the thesis show the efficiency of the proposed formulations and the solving methods. The thesis is submitted as a partial fulfillment of the requirement for obtaining the Ph.D. degree at the Université de Technologie de Belfort-Montbéliard (UTBM).

ACKNOWLEDGEMENT

Though the following dissertation is an individual work, I could never have reached the heights or explored the depths without the help, support, guidance and efforts of a lot of people. First and foremost, I would like to express my sincere gratitude to my thesis director Prof. Dr. Abdellah EL Moudni for giving me the opportunity to conduct my PhD research project and untiring help during my difficult moments. I would like to thank to my supervisor Assoc. Prof. Dr. Olivier Grunder for his technical advice, encouragement and insightful comments throughout my dissertation work. Without him this thesis would definitely not have been possible. I am also thankful for the excellent example he has provided as a successful research worker. I will miss all my meeting time with Dr. Olivier Grunder that served as my source of inspiration and replenishment in trying times and all this time will be cherished by me in the future.

I also want to thank my dissertation committee members, Dr. Christian Tahon, Dr. Laurent Geneste, Dr. Imed Kacem and Dr. Farouk Yalaoui for agreeing to serve on my committee and for also making valuable and timely suggestions that helped improve the outcome of my research.

My warm thanks are due to Prof. Dr. Kejun Zhu, Head of the School of Economics and Management, China University of Geosciences, Wuhan, China, who introduced me to the filed of Operational Research and provided me many chances to participate in projects. Without his inspiration and help, my career path would not be clear as it is now.

I also would like to express my appreciation to all my friends and colleagues at the laboratory SET who made my stay here for the last four years very memorable. Special thanks go to Kahina Ait Ali, Fayez Shakil Ahmed, Imad Matraji, Adnen Elamraoui, and Adeel Mehmood. A very special thank you to philippe Descamps for offering me the convenience for my each

meeting time with Dr. Olivier Grunder. Thanks also go out to the secretary of laboratory SET Ms. Ariane Glatigny for her help with administrative matters. Furthermore, I have to thank the library for providing me lots of helpful references related to my work. I would like to acknowledge and thank the China Scholarship Council and the Université de Technologie de Belfort-Montbéliard for supporting me financially through my doctoral studies.

Last but not least, I would like to express my very special thanks to my parents Hongxue Wang, Xueyun Li and my sisters Jinhua Wang and Jinguo Wang. Their understanding and faith in me and my capabilities, their love, encouragement, and eternal support have motivated me all the time. Last but not least, I would like to thank my girl friend Ting Yang for her love, patience and continuous support throughout all my years here at the Université de Technologie de Belfort-Montbéliard.

TABLE OF CONTENTS

LIST OF PUBLICATIONS

This thesis contains some unpublished material, but is mainly based on the following 8 publications. In the text, these publications are referred to as [PX].

International journals [3]

[PJ1] O. Grunder, D. Y. Wang and A. E. Moudni, "Production Scheduling Problem with Delivery Considerations in a Mono-product Supply Chain Environment to Minimize the Total Joint Cost," in *European Journal of Industrial Engineering*. ***(Accepted, in press)***

[PJ2] D. Y. Wang, O. Grunder and A. E. Moudni, "Single-item Production-delivery Scheduling Problem with Stage-dependent Inventory Costs and Due Dates Considerations," in *International Journal of Production Research*. ***(Accepted, in press)***

[PJ3] D. Y. Wang, O. Grunder and A. E. Moudni, "Using Genetic Algorithm for Lot sizing and Scheduling Problem with Arbitrary Job Volumes and Distinct Job Due Dates Considerations," in *International Journal of Systems Science*. ***(Under Review)***

International conferences [5]

[PC1] O. Grunder, D. Y. Wang and A. E. Moudni, "Lot Sizing with Delivery and Earliness Penalties Problem: The Multiple Stage Case," in 13^{th} *IFAC Symposium on Information Control Problem in Manufacturing*, Moscow, Russia, June 3–5, 2009.

[PC2] D. Y. Wang, O. Grunder and A. E. Moudni, "A Nested Algorithm for Lot Sizing and Delivery Problem," in *8 ème Congrès International de Génie Industriel*, Tarbes, France, Juin 10–12, 2009.

[PC3] O. Grunder, D. Y. Wang and A. E. Moudni, "Lot Sizing Problem with Consideration of Delivery and Earliness Penalties in Mono-product Supply Chain Environment," in 12^{th} *Large Scale Systems: Theory and Applications*, Lille, France, July 12–14, 2010.

[PC4] D. Y. Wang, O. Grunder and A. E. Moudni, "Production Scheduling Problems with Intermediate Inventory and Delivery Considerations," in *International Conference on Industrial Engineering and Systems Management*, Metz, France, May 25–27, 2011.

[PC5] D. Y. Wang, O. Grunder and A. E. Moudni, "A Tabu Search Approach for Integrated Scheduling Problem of Production and Distribution with Arbitrary Job Sizes and Due Dates Considerations," in 14^{th} *IFAC Symposium on Information Control Problems in Manufacturing*, Bucharest, Romania, May 23–25, 2012.

LIST OF FIGURES

LIST OF TABLES

INTRODUCTION

Motivation

Increasing global competition in the business world and heightened expectations of customers have forced companies to consider not only the pricing or product quality, but reliability and timeliness of the deliveries as well. In manufacturing-centric industries such as automotive and electronics, transportation and inventory costs constitute the second and third largest cost components following the production costs. Therefore, industrial and logistics companies need to continuously search for ways to lower the inventory level and distribution cost. This trend has created a closer interaction between the stages of a supply chain and has increased the practical usefulness of integrated models.

This integrating production and delivery schedules without intermediate inventory considerations is very common in the supply chain of time-sensitive products. For example, see the newspapers printing and distribution example provided by Buer et al. (1999), mail processing and distribution example provided by Wang et al. (2005), and industrial adhesive materials production and delivery example provided by Devapriya et al. (2006). In all these examples existed in practical life, because of the time-sensitive characteristics of these products, orders should be delivered to the customers directly without intermediate inventory, hence, integrated scheduling of production and product distribution is imperative. However, for many years companies and researchers consider the production and transportation subproblems in a separate and sequential manner with little or no integration where the production subproblem was firstly optimized and then, the transportation schedule for finished products was arranged according to the optimal strategy of production schedule. Obviously, such a separate and sequential approach will not necessarily yield a global optimal solution.

In some existing supply chains, production and distribution are indirectly connected through an intermediate stage of finished product inventory which works as a buffer to balance the abilities of the production and delivery stages, and hence the intermediate inventory is a non-negligible element when the companies tend to integrate production and transportation activities. Because in a supply chain, the rate of production and the speed of transportation are commonly not matched, thus from the whole system point of view, the consideration of the existence of intermediate inventory may efficiently balance their abilities and consequently improve the performance of the entire supply chain. As a practical example of the proposed problem, we can consider a scheduling issue existed commonly in the iron and steel industry. There is an oven that must heat different pieces of work at a given high temperature, then the finished pieces of work should be transported to next plant for painting by a capacitated transporter. In this case keeping the required temperature of the oven while it is empty may clearly be too costly (can be treated as setup cost), therefore a large production batch will result in lower setup cost, however, a large production batch may exceed the capacity of the transporter. Consequently, these pieces of work beyond the transporter capacity will stay at the factory (or intermediate inventory) waiting for the next delivery, and thus generate an intermediate inventory cost. However, many companies and researchers have studied this problem without taking into account the intermediate inventory. They implicitly assume that the production batch size is limited by the capacity of the transporter; i.e., a finished production batch can be delivered in one delivery batch. This assumption will result in worse performance for the production stage when the setup is relatively large and the manufacturing rate is far larger than that of transportation.

Moreover, in today's competitive environment, the most important objective for supply chains is to meet the customers' demand in a timely fashion, i.e., to deliver the right product to the right place at the right time for the right price. The delay in the delivery of the product may not only incur a tardiness penalty due to customer dissatisfaction, a possible contractual cost for late delivery and potential loss of reputation, but also lead to failure of the supply chains which are aimed for the proper and timely flow of the inventory. On the other hand, the finished products which are delivered to customer before its deadline could result in additional storage or insurance costs, or even product deterioration.

Therefore, simultaneously considering different and sometimes conflicting objectives from different participants, or different departments within the same participant in a supply chain becomes very crucial for most of the businesses that exist today. In this thesis we mainly study the following two problems: (1) integrated scheduling problems for a make-to-order production-distribution-inventory system; (2) integrated scheduling problems for a make-to-order production-inventory-distribution-inventory system.

ISPDI and ISPIDI Problems

Potts and Kovalyov (2000) classify the relevant scheduling models into two variants depending on two different assumptions. The first is *batch availability*, under which a job only becomes available when the complete batch to which it belongs has been processed. For example, this situation occurs if the jobs in a batch are placed on a pallet, and the pallet is only moved from the machine when all of these jobs are processed. An alternative assumption is *job availability* (or *item availability*), in which a job becomes available immediately after its processing is completed. Unless stated otherwise, we adopt the assumption of batch availability.

Our work mainly considered two categories of integrated scheduling problems in which the first one is **I**ntegrated **S**cheduling of **P**roduction-**D**elivery-(Customer) **I**nventory problem (*ISPDI* problem) and the second one is **I**ntegrated **S**cheduling of **P**roduction-(Intermediate) **I**nventory-**D**elivery-(Customer) **I**nventory problem (*ISPIDI* problem). The two categories of problems share the same machine environment which is described as follows. At the beginning of a planning horizon, the manufacturer has received an order of processing a set of independent and non-preemptive jobs associated with distinct due dates specified by the customer. It is assumed that the jobs that need to be processed are available at time 0. Jobs are first processed on a single machine in the production stage, and then delivered to the customers by a capacitated vehicle. It is assumed that each job has a constant processing time, and each production batch requires a setup cost as well as a setup time before the first job of this batch is processed. A job becomes available for delivery only when the production batch to which it belongs is completely finished. All the jobs processed consecutively without setup in between constitute a production batch.

The different aspects on the two categories of problems can be stated in more detailed fashion as follows. For the first category of problems, the completed jobs are delivered to the customer directly without intermediate inventory by a capacitated vehicle. Concerning the second category of problems, because of the existence of the intermediate inventory, when the production rate is larger than that of distribution, the completed jobs will first be stored in this intermediate inventory waiting for delivery, hence they will incur a finished product inventory cost which is a non-negligible part of the total cost.

The two categories of problems share the same distribution environment which is described as follows. Each job should be delivered to the customer before its due date, and each round trip between the factory and customer requires a delivery cost as well as a delivery time. Moreover, it is assumed that a job which is completed before its departure date or delivered to the customer before its due date will incur a corresponding inventory cost (WIP inventory, finished product inventory or customer inventory cost). We estimate both the total logistics cost and the customer service level. The logistics cost is measured by actual expenses of operations. The customer service performance is expressed in terms of the deadline of each job, i.e., each job must be delivered to the customer before its deadline.

Thesis Outline

The contents of this thesis are organized into 4 chapters.

In Chapter 1, we provide a comprehensive literature review of the production-distribution scheduling problems and production-inventory-distribution scheduling problems. The multi-product ISPDI problem with arbitrary job volumes and distinct due dates considerations is discussed in Chapter 2. We show that this problem is NP-hard, and formulate it as a mixed integer programming model. We then propose an improved genetic algorithm for solving the model. In order to evaluate the performance of the proposed genetic algorithm, we provide a lower bound based on the classical bin-packing problem. Based on the consideration that the inventory cost depends much on the product itself, the proposed model has been extended to the model where each job is associated with a distinct unit inventory cost. We formulate this extended problem

as a non-linear model, and then propose a Tabu-based method for solving it. Finally, based on the lower bound proposed above, we analyze the average-case and worst-case performances of the proposed Tabu-based method. In the Chapter 3, we address the single-product, multi-stage ISPDI problem. We select a supply chain environment which is composed of multiple supply links as the studied object. Particularly, we assume that the production start dates of jobs in one supply link equal to the due dates of the jobs in its previous supply link. In each link of the supply chain, we study an integrated scheduling problem of production and distribution. We provide the NP-hardness proof for the problem through a reduction from the knapsack problem. Then a genetic algorithm and a dominance related greedy approach are developed for solving this model. Finally, by comparing with a lower bound, we do the analysis of the performances for the two proposed algorithms. In the Chapter 4, we investigate the single-product ISPIDI problem where the production and distribution stages are indirectly linked through an intermediate stage of finished product inventory. In specific, we assume that the intermediate stage works as a buffer to balance the production rate and the speed of distribution. Moreover, this intermediate inventory allows the jobs to be rescheduled for transportation process after completion on the machine. The proposed problem is proved to be NP-hard by a reduction from the knapsack problem. We formulate the problem as a non-linear model in a general way and provide some properties. Based on the general model, we derive a special instance and provide an efficient property between the production and distribution schedules. Then, we develop an efficient heuristic algorithm for solving the special instance. In order to evaluate the performance of proposed heuristic algorithm, we establish a basic branch and bound approach and a lower bound based on the lagrangian decomposition method. Finally, we analyze the average-case performance of the proposed heuristic algorithm in terms of both solution quality and computational time. At last, in the Chapter of conclusion, we make some concluding remarks based on the computational results and analysis, and suggest directions for future research.

1. LITERATURE REVIEW

1.1 Introduction

This section presents a literature review on the coordinated scheduling of production-distribution, and production-inventory-distribution problems. There is a vast literature on the machine scheduling problems. Extensive reviews of classical machine scheduling models, as well as contributions reported in this field, can be found in Cheng and Sin (1990), Drexl and Kimms (1997), Allahverdi et al. (1999), Gordon et al. (2002), Mndez et al. (2006), Tang et al. (2001), Potts and Kovalyov (2000), Levner et al. (2010), Allahverdi et al. (2008), and Koulamas (2010), among others. However, comparing to the classical machine scheduling problems, the integrated scheduling of production and distribution has not received enough attention.

In a recent review paper, Bhatnagar and Chandra (1993) study the integrated optimization of organizations. They distinguish two levels on it, integration between functions, which they call the General Coordination problem, and integration within the same function at different echelons in the organization. They classify the research on General Coordination problem into three categories: (1) supply and production planning, (2) production and distribution planning, and (3) inventory and distribution planning. Because our work falls into the last two categories where the transportation process is explicitly considered, thus, we will here only cover the literature that explicitly involve both production and transportation activities at the operational level. Based on the problems studied in this thesis, we will mainly review two categories of problems in which the first one is production-distribution scheduling problems and the second is production-inventory-distribution scheduling problems.

1.2 Production-Distribution Problems

Integrating production and outbound delivery schedules is very critical and common in the supply chain of time-sensitive products, see Chen (2010). For example, see the newspapers printing and distribution example provided by Buer et al. (1999), mail processing and distribution example provided by Wang et al. (2005), and industrial adhesive materials production and delivery example provided by Devapriya et al. (2006). Therefore, how to effectively integrate the production and delivery stages at the operational level so as to lower the operational costs and improve customer service becomes very important to the success of a company. However, most of the existing models on the production-distribution scheduling problems only study strategic or tactical levels of decisions, and very few have addressed integrated decisions at the operational level, see Chen (2004). Chandra and Fisher (1994) emphasize the need for studying this integrated scheduling issues at the operational level. They consider an integrated scheduling problem where a plant produces and stores the products until they are delivered to the customers by a fleet of trucks. They provide two solutions. The first solution solves the production scheduling and vehicle routing problems separately, but the second one solves the problem in a coordinated manner. Their computational results show that the reduction in total operating cost from coordination could reach to 20%. Chen and Vairaktarakis (2005) and Pundoor and Chen (2005) also show that there is significant benefit by using the optimal integrated production-distribution schedule compared to the schedule generated by a separate and sequential scheduling approach in the context of the models they consider.

Chen (2004) provides a review on the models that involve explicitly both production and distribution operations. They refer to this kind of models as *explicit production-distribution* (EPD) models. They then classify various existed EPD problems according to three dimensions: (A) decision level, (B) integration structure, and (C) problem parameters, which are described in detail as follows.

(A) Decision Level: the EPD models can be classified into two following types according to their decision level: (A1) tactical EPD models which mainly involve decisions such as: how much to produce and how much to ship in a time period, how much inventory to keep, etc.,

(A2) Operational EPD models which mainly involve detailed scheduling level decisions such as: when and on which machine to process a job, when and by which vehicle to deliver a job, which route to take for a vehicle, etc.

(B) Integration Structure: the integration between production and distribution operations can be divided into the following three types of structures: (B1) integration of production and outbound transportation, (B2) integration of inbound transportation and production, and (B3) integration of inbound transportation, production and outbound transportation.

(C) Problem Parameters: there are three variations on these parameters considered in the EPD literature: (C1) one time period, (C2) infinite horizon with constant demand rate, and (C3) finite horizon but with multiple time periods and dynamic demand.

According to the three dimensions A, B, C of model characteristics described above, they classify the EPD problems into five problem classes as follows.

Class 1. Production-Transportation Problems – A1, B1, C1

Class 2. Joint Lot Sizing and Finished Product Delivery Problems – A1, B1, C2

Class 3. Joint Raw Material Delivery and Lot Sizing Problems – A1, B2, C2

Class 4. General Tactical Production-Distribution Problems – A1, B1 or B3, C1 or C3

Class 5. Joint Job Processing and Finished Job Delivery Problems – A2, B1, C3

Then, he reviews recent work in the area of integrated scheduling problems for each class of problem mentioned above. Chen (2010) also provides a survey of models and results in the area of integrated scheduling of production and distribution. He presents a unified model representation scheme, classifies existing models into several different classes, and for each class of the models gives an overview of the optimality properties, computational tractability, and solution algorithms for the various problems studied in the literature. Extensive reviews of integrated scheduling of production-distribution or production-inventory-distribution models, as well as contributions reported in this field, can be found in Potts and Wassenhove (1992), Potts and

Kovalyov (2000), Thomas and Griffin (1996), Webster and Baker (1995) and Sarmiento and Nagi (1999), among others.

The integrated scheduling of production-distribution problems can be divided into two categories according to the different types of objective functions. In the first category, only the time-related objectives such as makespan are considered; while in the second one, both the time-related and cost-related objectives are considered.

1.2.1 Integrated models with time-related objectives

There are many such integrated scheduling models (e.g., Li et al. (2005), Sung and Kim (2002), Potts (1980)) without involving delivery cost-related performances. Lee and Chen (2001) study the machine scheduling problems with explicit transportation considerations. They identify two types of transportation situations in their models. The first type, type-1, involves transporting a semi-finished job from one machine to another for further processing. The second type, type-2, involves transporting a finished job to the customer or warehouse. Both transportation capacity and transportation times are taken into account explicitly in their model. They classify the computational complexity of various scheduling problems with type-1 or type-2 transportation by either proving their NP-hardness or providing polynomial algorithms. Chang and Lee (2004) consider an extension of Lee and Chen's work where each job is assumed to occupy a different amount of storage space in the vehicle during delivery. They show that the problems that jointly consider production and delivery with the consideration that each job may require a different amount of space during transport are intractable, and provide heuristics for some cases of the problem. Zhong et al. (2007) study the similar problem to the one studied by Chang and Lee (2004) with the objective of minimizing the makespan. For the first problem, in which jobs are processed on a single machine and delivered by one vehicle to a customer, they propose a best possible approximation algorithm with a worst-case ratio arbitrarily close to $3/2$. For the second problem, which differs from the first problem in that jobs are processed by two parallel machines, they devise an improved algorithm with a worst-case ratio $5/3$. However, they just consider the time-related objectives such as makespan, the maximum lateness and the sum of

completion times, etc., without considering any cost-related objectives such as delivery cost, and setup cost, etc.

Soukhal et al. (2005) investigate flow shop scheduling models that explicitly consider constraints on both transportation and buffer capacities. They assume that the finished jobs need to be transferred from the processing facility and delivered to one and only one customer or warehouse by a capacitated vehicle. They establish some new complexity results for some special cases of the problem. For the makespan objective function, they prove that this problem is strongly NP-hard when the capacity of a truck is limited to two or three parts with an unlimited buffer at the output of the each machine, and that the problem with additional constraints, such as blocking, is also strongly NP-hard. Lu et al. (2008) consider an integrated scheduling problem involving release dates and job delivery, where only one vehicle of capacity c is employed to deliver these jobs to a single customer. They define that the delivery completion time of a job as the time at which the delivery batch containing the job is delivered to the customer and the vehicle returns to the machine. Their objective is to minimize the makespan, i.e., the maximum delivery completion time of the jobs. When preemption is allowed to all jobs, they give a polynomial-time algorithm for this problem. When preemption is not allowed, they show that this problem is strongly NP-hard for each fixed $c \geq 1$. They also provide a $5/3$ approximation algorithm for this problem, and the bound is tight. Liu and Lu (2011) study the same problem by introducing an improved approximation solving method which is better than that given in the literature reviewed by them.

Qi (2009) study a problem similar to the one studied by Qi (2005) with the objective of minimizing the arrival time of the last delivered job to the customer. They show that the problem is NP-hard in the strong sense, and propose an $O(n)$ time heuristic with a tight performance bound of 2. They identify some polynomially solvable cases of the problem, and develop heuristics with better performance bounds for some special cases of the problem. Li and Ou (2005) develop and analyze a three-stage integrated scheduling problem involving pickup, production, and delivery functions. In specific, they assume that there is a capacitated pickup and delivery vehicle that travels between the machine and the storage area. Their objective is to minimize the makespan of the schedule. They show that the problem is strongly NP-hard in general but is

solvable in polynomial time when the job processing sequence is predetermined, and propose a heuristic algorithm for the general problem.

Tang and Liu (2009b) address two scheduling problems for a two-machine flowshop where a single machine is followed by a batching machine. They assume that there is a transporter in the first problem between machines for delivering the jobs, and there are deteriorating jobs in the second problem to be processed on the single machine. For the first problem with minimizing the makespan, they propose a mixed integer programming model, and show that the problem is strongly NP-hard.Then they devise a heuristic algorithm for solving this problem. For the second problem, they develop the optimal algorithms with polynomial time for minimizing the makespan, the total completion time and the maximum lateness, respectively. Tang and Gong (2008) study a coordinated scheduling problem of hybrid batch production on a single batching machine and two-stage transportation connecting the production. They assume that there is a crane available in the first-stage transportation that transports jobs from the warehouse to the machine, and there is a vehicle available in the second-stage transportation to deliver jobs from the machine to the customer. Their objective is to minimize the sum of the makespan and the total setup cost. They show that the problem is NP-hard. Then they propose a heuristic algorithm for the general problem and analyze its tight worst-case bound. They also devise a polynomial time algorithm for a case where the job transportation times are identical on the crane or the vehicle. Liu (2011) considers the similar problem to the one studied by Tang and Gong (2008). He proposes two genetic algorithms for this scheduling problem, with different result representations. Tang and Gong (2009) consider a coordinated scheduling problem of production and transportation in which each job is delivered to a single batching machine for further processing. They assume that there are a number of vehicles that transport jobs from the holding area to the batching machine, and each vehicle can transport only one job at a time. The batching machine can process a batch of jobs simultaneously provided that the batch size is less than the machine capacity. Their objective is to find a joint schedule of production and transportation such that the sum of the total completion time and the total processing cost is optimized. Gong and Tang (2011) study a coordinated scheduling problem in which a single transporter can deliver several jobs as a batch between machines. It is assumed that each job

is associated with a distinct physical volume. Their objective is to minimize the makespan. For the jobs with the same size of physical volume, they present a heuristic approach with an absolute worst-case ratio of 2 and a polynomial-time optimal algorithm for a special case with given job sequence. For the jobs having different size of physical storage space, they devise a heuristic algorithm with an absolute worst-case ratio of $7/3$ and asymptotic worst-case ratio of $20/9$.

Li et al. (2011) study an integrated scheduling problem where a set of jobs are processed in batches on an unbounded parallel-batch machine, and then the completed jobs are delivered by a capacitated vehicle from the machine to their specified customers. In specific, the model assumes that jobs of the same family have identical size in a transportation vehicle and belong to a specified customer, and that jobs from different families cannot be transported together by the vehicle in a delivery batch. Their objective is to find a joint schedule to minimize the time when the vehicle finishes delivering the last delivery batch to its customer and returns to the machine. They first show that the problem is NP-hard, and then develop for the problem a heuristic algorithm involving a worst-case performance ratio of $3/2$.

Most of the papers reviewed above do not consider the cost-related objectives in their models. For more details on the scheduling problem without cost-related objective considerations, the reader is referred to Li and Yuan (2009), Tang and Liu (2009a), Zdrzalka (1995), and Behnamian et al. (2012), among others.

1.2.2 Integrated models with both time and cost-related objectives

The cost objective function is a very important factor of system performance measurement, since it reflects the amount of energy that the system consumes. Here the energy may be related to anything valuable, such as oil, gas and time, etc. Specially, with the popularization of "energy saving and carbon reduction" concept, reduction of energy consumption becomes a crucial factor to the success of a company in modern society. Cheng et al. (1996) consider a single machine batch delivery problem with objective of minimizing the sum of total weighted earliness and total delivery cost, where the earliness of a job is defined as the difference be-

tween the delivery time of this job and the completion time of this job on the machine. They show that this problem and the parallel machine scheduling problem are closely related and develop polynomial time algorithms for a special case. However, they do not consider any due date constraints on jobs. Cheng et al. (1997) study the similar problem except that they assume that a constant setup time is required before the process of each job on the batch. They show that some cases of the problem are NP-hard and provide dynamic programming approaches and heuristics for solving them. Min et al. (2007) consider a machine scheduling problem where the jobs need to be delivered to customers in batches after processing on machine. They define that the delivery date of a batch equals the completion time of the last job in the batch, and the delivery cost depends on the number of deliveries. Their objective is to minimize the sum of the total weighted flow time and delivery cost. They provide the NP-hardness proofs for the problem, and show that, if the number of batches is B, the problem remains strongly NP-hard when $B \leq U$ for a variable $U \geq 2$ or $B \geq U$ for any constant $U \geq 2$. For the case of $B \leq U$, they develop a dynamic programming algorithm that runs in pseudo-polynomial time for any constant $U \geq 2$. Moreover, they also provide optimal algorithms for some special cases.

Hall et al. (2000) and Yang (2000) analyze various integrated scheduling problems of production and distribution with the assumptions of infinite vehicle capacities, a sufficient number of vehicles and no delivery costs. Wang and Cheng (2000) study a parallel machine scheduling problem in which a set of n independent and simultaneously available jobs are first to be processed on a number of m parallel machines, and then the completed jobs need to be delivered in batches to customers. They adopt the same assumptions on the delivery date and delivery cost as that of Min et al. (2007). Their objective is to minimize the sum of the total flow time and the delivery costs. They first show that the problem is NP-complete in the ordinary sense even when $m = 2$, and NP-complete in the strong sense when m is arbitrary. Then they develop a dynamic programming algorithm for solving the problem. They also provide two polynomial time algorithms for the special cases where the job assignment is given or the job processing times are equal. Qi (2005) considers a problem where the raw material used for manufacturing jobs is delivered in batches to a single machine and the raw material delivery and job sequencing decisions are considered simultaneously. His objective is to minimize the sum of delivery

and flow-time costs.

Hall and Potts (2003) consider a variety of scheduling, batching and transportation problems within a supply chain environment with the objective of minimizing the total scheduling and transportation cost. It is assumed that each batch will be shipped to only one downstream destination. For each problem, they either provide a dynamic programming algorithm or demonstrate that the problem is intractable. One of the problems identified by Hall and Potts (2003) is that of batching and sequencing on a single machine under the batch availability assumption, in order to minimize the sum of flow times and delivery costs. Mazdeh et al. (2007) consider this problem with the same objective and devise a branch-and-bound solution scheme based on some structural properties of the problem. However, the transportation capacity is not considered in their models.

Chen and Lee (2008) investigate a general two-stage scheduling problem, in which jobs of different importance are processed by one first-stage processor and then, in the second stage, the completed jobs need to be batch delivered to various pre-specified destinations in one of a number of available transportation modes. Their objective is to minimize the sum of weighted job delivery time and total transportation cost. They draw an overall picture of the problem complexity for various cases of problem parameters accompanied by polynomial algorithms for solvable cases. On the other hand, they propose for an approximation approach of performance guarantee for the most general case. Cheng and Wang (2010) study the machine scheduling problems with job class setup and delivery considerations. They assume that a setup time is required for a job if it is the first job to be processed on a machine or its processing on a machine follows a job that belongs to another class. The processed jobs need to be delivered in batches to their respective customers. Their objective is to minimize the weighted sum of the last arrival time of jobs to customers and the delivery cost. For the problem of processing jobs on a single machine and delivering them to multiple customers, they develop a dynamic programming approach to solve the problem optimally. For the problem of processing jobs on parallel machines and delivering them to a single customer, they propose a heuristic and analyze its performance bound. Mazdeh et al. (2011) address scheduling problem in which a set of jobs are processed on a single machine, and then delivered in batches to one customer or to another

machine for further processing. The objective adopted is to minimize the sum of weighted flow times and delivery costs. They establish some structure properties for this problem, and then they devise a branch-and-bound solution method based on these properties.

Wang and Cheng (2009a) investigate the identical parallel-machine scheduling problem in which both job class setups for job processing and product delivery are required. They define that a setup time is incurred for a job if it is the first job to be processed on a machine or its processing on a machine follows a job that belongs to another class. Finished jobs need to be delivered in batches by a capacitated vehicle to their respective customers. Their objective is to minimize the weighted sum of the last arrival time of the jobs to the customers and the delivery cost. They develop heuristics for the problem and analyze their performance bounds.

1.2.3 Integrated models with due-date constraints

Another line of research related to our work focus on problems in which jobs are associated with due date (or deadline) constraints. The delay in the delivery of the product may not only incur a tardiness penalty due to customer dissatisfaction, a possible contractual cost for late delivery and potential loss of reputation, but also lead to failure of the supply chains which are aimed for the proper and timely flow of the inventory. On the other hand, the finished products which are delivered to customer before its deadline could result in additional storage or insurance costs, or even product deterioration. Therefore, the integrated scheduling problem involving due date considerations becomes very crucial for most of the businesses that exist today.

Panwalkar et al. (1982) study the machine scheduling problem in which all jobs have a common due date. The objective is to determine the optimal value of this due date and an optimal sequence to minimize a total penalty function which is based on the due date value and the earliness or lateness of each job in the selected sequence. Seidmann et al. (1982) consider the optimal assignment of due dates for a single processor scheduling problem in which each job can have a distinctive due date. Their objective is to select optimal due dates and optimal sequence. These two papers started extensive research in the area of due date assignment. Due date decisions have a direct impact on customer service level. Implementing due date decisions

involves both production and delivery stages. However, as Chen (2010) points out, almost all the existing due date setting models such as Cheng and Gupta (1989) ignore scheduling issues associated with delivery of finished orders.

Most researchers study the production scheduling problems involving due dates without considering delivery process, e.g., De et al. (1990), Hall et al. (1991), Pan et al. (2001), Rabadi et al. (2004), Hassin and Shani (2005), Supithak et al. (2010), Chen (1997), Chen and Powell (1999). See recent related comprehensive reviews of Baker and Scudder (1990), Lauff and Werner (2004) and Gordon et al. (2002). To our best knowledge, there are only very few researchers consider the production-distribution problems with due dates considerations. Yuan (1996) considers the single machine scheduling with a common due date, earliness-tardiness, and batch delivery costs. He provides the NP-hardness proofs for the problem under study. Herrmann and Lee (1993), Chen (1996), and Cheng et al. (1996) study machine scheduling problems with jobs delivered in batches after being processed in the manufacturing unit. It is assumed that each delivery batch requires a certain transportation cost. However, they do not consider the transportation times and due date constraints on jobs. Yang (2000) addresses a model similar to the one studied by Cheng et al. (1996), but with given batch delivery dates. Eksioglu (2002) and Liu (2003) study various integrated scheduling problems without taking WIP inventory and customer inventory costs into consideration.

Pundoor and Chen (2005) consider the production-distribution system with one supplier and one or more customers. It is assumed that each order requested by the customer is associated with a different due date. Their objective is to optimize a combined objective function that considered both the maximum tardiness and total distribution cost. They show that for an arbitrary number of customers, the problem under study is NP-hard even in the special case where the processing times and the due dates are agreeable, and propose a fast heuristic for solving the problem. Similar to the papers Chandra and Fisher (1994), and Fumero and Vercellis (1999), they also demonstrate that there is distinct advantage of using the integrated production-distribution approach as compared to the two sequential approaches that try to optimize production and delivery sequentially with no or only partial integration.

Lee (2001) considers a multi-machine two-stage manufacturing system with respecting the fol-

lowing three objectives concurrently: (1) meeting customers' due dates, (2) minimizing inventory cost, and (3) minimizing machining cost. It is assumed that each order is an indivisible scheduling element that needs to be delivered to customers on the due date. Wang and Wang (2010) study a make-to-order production-distribution problem with single supplier and multiple customers. They assume that each order is associated with a different deadlines and needs to be delivered to the corresponding customer before its deadline. Their objective is to find a joint schedule of order processing at the supplier and order delivery from the supplier to the customers such that the total distribution cost is minimized. They consider the solvability of three cases of the problem and provide efficient algorithms for solving them. However, they do not consider any time-related objective functions.

Zhong et al. (2010) study an integrated production and delivery scheduling problem faced by a make-to-order company with a commit-to-delivery business model. At the beginning of a planning horizon, the company has accepted a set of orders and committed a delivery date for each order. The company needs to process these orders on a dedicated production line and deliver the finished orders to the respective customers by a third-party logistics provider. They assume that the shipping cost of an order charged by the third-party logistics provider increases linearly with the order size and decreases linearly with the shipping time requested. Their objective is to determine a production schedule for the accepted orders and a shipping mode for delivering each completed order so that the total shipping cost is minimum subject to the constraint that all the orders are completed and delivered to their customers on or before the respective committed delivery dates. They show that the problem under study is strongly NP-hard. Then they develop a polynomial-time heuristic approach and show that its worst-case performance ratio is bounded by 2 and that this bound is tight.

There are many supply chain environments which involves more than one supply links. As a practical example of the proposed problem, we can consider a scheduling issue existed in the paper industry. At the beginning of a planning horizon, a customer requires a certain amount of colorful paper bags and sends his requirement to a paper bag manufacturer. Each order has a due date constraint specified by the customer. After the manufacturer receives the order sent by the customer, he will need to buy roll papers from a roll paper manufacturer to finish

the order sent by the paper bag customer. This problem becomes more and more important with the development of economic globalization. The work PJ1 chooses a supply chain which involves multiple supply links as the studied object. Each supply link is composed of one supplier, one capacitated transporter and one customer. In each supply link of the supply chain, they consider an integrated scheduling problem in which a given set of identical jobs are first processed on a single machine, and then batch delivered to a pre-specified customer directly without intermediate inventory by a capacitated transporter. Each job has a due date specified by the customer in the current supply link. It is supposed that a job which is finished before its departure date or delivered to the customer before its due date will incur an earliness penalty which is equivalent to a corresponding inventory cost. The objective is to find a coordinated production and delivery schedule for each supply link such that the total joint cost of the supply chain is minimized. They show that this problem is NP-hard in the maximum capacity of the transporters. Then a dominance related greedy algorithm and a genetic algorithm are proposed. In order to evaluate the efficiency of the proposed heuristics, they propose a simple branch and bound approach for the small size problems and a lower bound of the objective value for large size problems.

1.3 Production-Inventory-Distribution Problems

In some existing supply chains, production and distribution are often indirectly linked by an intermediate stage of finished product inventory, and hence the intermediate inventory is a non-negligible element when the companies tend to integrate production and transportation activities. Since in a supply chain, the abilities of the two main logistical stages, i.e. the rate of production and the speed of delivery, are commonly not matched. In this case, from the whole system point of view, the consideration of the existence of intermediate inventory may efficiently balance their abilities and consequently improve the performance of the entire supply chain. However, many companies and researchers have studied this problem without taking into account the intermediate inventory. They implicitly assume that the production batch size is limited by the capacity of the vehicle, i.e., a finished production batch can be delivered in one

delivery batch. This assumption will result in worse performance for the production stage when the setup is relatively large and the manufacturing rate is far larger than that of transportation. This problem is significant as it addresses the issue of striking a proper balance between the rate of production, the level of inventory and the speed of delivery. In this section, we review the papers on integrated optimization of production-inventory-distribution problems. Since the manufacturer, inventory and customer are three key components of a supply chain. Thus, this problem can also be called supply chain scheduling which has been one of the most important and widely discussed topics in manufacturing research area over the last ten years. However, most of the papers consider the production-inventory scheduling problems existed in the production stage without delivery consideration or the inventory-distribution scheduling problems existed in the distribution stage without production scheduling. For the production-inventory scheduling problems, see, for example, Fleischmann (1990), Drexl and Kimms (1997), Ouenniche and Boctor (2001), Dobson and Yano (1994), Fleischmann (1994), Ferretti et al. (2006) and Fandel and Hegene (2006), among others. For the inventory-distribution scheduling problems, see, for example, Hanczar (2010), Rodriguez and Vecchietti (2010), and Zhao et al. (2010), among others.

However, there are few papers which consider the three key stages of a supply chain in an integrated way. Only recently, models that integrate production scheduling, inventory control, and distribution arrangement appear in the literature. Glover et al. (1979) study an integrated system of production, distribution, and inventory planning. They show that this integrated system has saved approximately 18 million dollars during its first three years of implementation for a major national company. This result shows the potential cost benefits of integrating decisions of production scheduling, inventory control, and distribution arrangement. Fumero and Vercellis (1999) propose an integrated optimization model for the planning problem of production and distribution, in which the products need to be first processed and then delivered with limited available resources, for both production system and a homogeneous distribution fleet. The tradeoff is among production setup cost, inventory cost and transportation cost. Similar to the method used in Chandra and Fisher (1994), they also develop two methods: (1) synchronized approach, and (2) decoupled approach. Their computational results indicate the efficiency of the

proposed synchronized approach and the substantial advantage of the synchronized approach over the decoupled approach.

Hahm and Yano (1992) consider the problem of determining the frequency of production of a single component and the frequency of delivery of that component to a customer which uses this component at a constant rate. Their objective is to minimize the average cost per unit time of production setup costs, inventory holding costs at both supplier and customer, and transportation costs. They prove that the ratio between the production interval and delivery interval must be an integer in an optimal solution. Then, based on this optimality property, they devise an optimal solution procedure for solving the problem. Torabi et al. (2006) study the lot and delivery scheduling problem in a simple supply chain where a single supplier manufactures multiple components on a flexible flow line and delivers them directly to an assembly facility. They assume that all of parameters such as demand rates for the components are deterministic and constant over a finite planning horizon. Their objective is to find a lot and delivery schedule that would minimize the average of inventory holding, setup, and transportation costs per unit time for the supply chain. Then they formulate the problem as a mixed integer nonlinear programming model and propose an optimal enumeration method to solve this model. Due to the difficulty of obtaining the optimal solution in reasonable computing time for medium and large-scaled problems, they also develop a hybrid genetic algorithm. However, both of them do not consider delivery time in their models.

Lejeune (2006) addresses the coordinated planning and scheduling of the inventory, production and distribution operations in a three-stage supply chain. The first stage which he calls supplier is in charge of the procurement of raw materials and/or components. The second stage which he calls production represents the manufacturing and/or assembly of the finished goods. The third stage which he calls distribution represents the transportation of the completed goods to a customer or to a distribution center. The objective adopted by him is to construct a sustainable inventory-production-distribution plan enabling it to minimize its costs while satisfying the customer's demand. After modeling the problem as a mixed integer programming model, he develops an algorithm based on variable neighborhood decomposition search. Selvarajah and Steiner (2009) study the upstream supplier's batch scheduling problem in a supply chain, which

is defined by Hall and Potts (2003). In this problem, the supplier has to manufacture multiple products and deliver them to customers in batches. There is an associated delivery cost with each batch. Their objective of the supplier is to minimize the total inventory holding and delivery costs. They propose simple approximation approaches for this strongly NP-hard problem, which find a solution that is guaranteed to have a cost at most $3/2$ times the minimum. They also prove that the approximation algorithms have worst-case bounds that vary parametrically with the data and that for realistic parameter values are much better than $3/2$.

Sawik (2009) considers a long-term, integrated scheduling of material manufacturing, material supply and product assembly in a customer driven supply chain. He provides a mixed integer programming model for this problem and presents two approaches for solving this model. The computational results show the efficiency of their proposed approaches. Wang and Cheng (2009b) study a logistics scheduling problem with raw material supply and product delivery considerations. Their objective is to find a joint schedule such that the sum of WIP inventory cost and transport cost is minimized. Here the transport cost includes both supply and delivery costs. For the special case of the problem where all the jobs have identical processing times, they show that the inventory cost function can be unified into a common expression for various batching schemes. Based on this characteristic and other optimal properties, they propose an algorithm with the time complexity of $O(n)$ to solve this special case. For the general problem, they consider several special cases, develop their optimal properties, and propose polynomial-time approaches to solve them optimally. However, they do not consider intermediate inventory cost in their models. Wang and Cheng (2009c) study the problem similar to the one studied by Wang and Cheng (2009b) with the objective of minimizing the makespan. In their model, it is assumed that the warehouse, the factory and the customer are located at three different sites. They show that the problem under study is NP-hard in the strong sense, and develop several heuristics for the general problem and for some special cases. However, they do not take account of any inventory cost such as WIP inventory cost.

Pundoor and Chen (2009) consider an integrated production and transportation scheduling problem existed in a two-stage supply chain consisting of one or more suppliers, a warehouse, and a customer. Each supplier produces a different product at a constant rate. There is a setup time

and a setup cost per production run for each supplier. They assume that the finished products are first transported from the suppliers to the warehouse and are then sent from the warehouse to the customer. The customer's demand for each product is constant over time. Their objective is to find jointly a cyclic production schedule for each supplier, a cyclic delivery schedule from each supplier to the warehouse, and a cyclic delivery schedule from the warehouse to the customer so that the customer demand for each product is satisfied without backlog at the least total production, inventory and distribution cost. They take two production and delivery scheduling policies into account. They derive either an exact or a heuristic solution approach for the problem under each policy. They also evaluate the value of the warehouse by comparing their model with a model that does not have the warehouse in the supply chain (i.e., the products are delivered directly from the suppliers to the customer).

Kang and Kim (2010) investigate a two-level supply chain in which a supplier serves a number of retailers in a given geographic region and determines a replenishment plan for each retailer by using the information on demands of final customers and inventory levels of the retailers. In their problem, they assume that the deliveries are carried out by homogeneous capacitated vehicles, and each vehicle can visit multiple retailers in a single delivery trip. Their objective is to determine the replenishment quantities and timing for the retailers as well as the amount of products delivered to the retailers by each vehicle for minimizing the sum of the fixed vehicle cost, retailer-dependent material handling cost, and inventory holding cost of the whole supply chain. They provide some heuristic algorithms by simultaneously considering inventory and transportation decisions.

The models reviewed above only consider one kind of inventory, i.e., either intermediate inventory which connects production and delivery or customer inventory. However, in a supply chain, according to different phases of product lifecycle, the inventories existed in different stages of the supply chain can be classified into different categories. Moreover, the inventory holding cost represents a combination of the cost of capital, the cost of physical storage and the cost of losses due to spoilage; hence, it highly depends on the inventory type (or value). Therefore, it is much more reasonable to calculate the inventory costs according to different types of inventories. Lee and Yoon (2010) consider an integrated production-and-delivery scheduling problem that in-

corporates stage-dependent (WIP and finished-goods) inventory holding costs. Their objective is to find the coordinated schedule of production and delivery such that the total cost of the associated WIP inventory, finished product inventory and delivery is minimized. Particularly, in their model, it is assumed that both the WIP inventory cost and finished product inventory cost are characterized in terms of the weighted flow time, and the delivery cost is proportional to the required number of delivery batches. They show that the problem under study is NP-hard in the strong sense and propose three heuristic algorithms for solving this problem. In order to evaluate the performance of the proposed algorithms, they develop a lower bound based on the lagrangian decomposition method.

Grunder (2010) considers a single-product batch scheduling problem with the objective of minimizing the sum of production, transportation and holding cost. Particularly, he assumes that the setup times depend on the batch sizes. He firstly shows that the problem is NP-hard in a general case, and then proposes a dynamic programming approach based on a dominance relation property. Yeung et al. (2011) study a two-echelon supply chain scheduling problem in which there is a supplier, and a manufacturer who receives orders from the customers and then orders supplies from the supplier to produce the products. The manufacturer can accept only some of the orders because of the production and delivery time constraints in the supply chain. Their objective is to maximize the profit, subject to sizes of the orders, time-dependent storage costs, and transportation costs of dual delivery models in the supply chain. They formulate the problem as a two-machine multiple common time windows flow shop scheduling problem, and propose fast pseudo-polynomial dynamic programming algorithms for the problems.

Even though the models reviewed above consider the production, inventory, and delivery functions simultaneously at an operational level, few of them allowed the intermediate inventory which connects the production and distribution stage to work as a buffer for resequencing and rebatching the jobs. That is to say, most of the integrated models of production scheduling, inventory control, and product distribution implicitly assume that the batch size is limited by the capacity of the vehicle; i.e., after one batch is processed by a machine, it can be entirely delivered by the vehicle to the customer. Agnetis et al. (2006) investigate an integrated scheduling problem in a two-stage supply chain which is consisted of one supplier and several manufactur-

ers. They assume that both the supplier and each manufacturer have an ideal schedule, determined by their own costs and constraints, and that an interchange cost is incurred by the supplier or a manufacturer whenever the relative order of two jobs in its actual schedule is different from that in its ideal schedule. They also assume the existence of an intermediate storage buffer for resequencing the jobs between the two stages. Their objective is to find an optimal schedule for supplier, an optimal schedule for manufacturer, and optimal schedules for both such that the total interchange cost, or the sum of total interchange cost and buffer storage cost is minimized. They provide polynomial time algorithms for all the supplier's and manufacturer's problems, as well as for a special case of the joint scheduling problem.

The work PJ2 studies an integrated scheduling problem for a single-item, make-to-order supply chain system consisting of one supplier, one capacitated transporter and one customer. In specific, they assume the existence in the production stage of an intermediate inventory which works as a buffer to balance the production rate and the transportation speed. Each job has a due date specified by the customer, and must be delivered to customer before its due date. Moreover, it is assumed that a job which is finished before its departure date or arrives at the customer before its due date will incur a stage-dependent corresponding inventory cost (WIP inventory, finished-good inventory or customer inventory cost). Their objective is to find a coordinated production and delivery schedule such that the sum of setup, delivery and inventory costs is minimized. They formulate the problem as a non-linear model in a general way and provide some properties. Then we derive a precise instance from the general model, and develop a heuristic algorithm for solving this precise instance. In order to evaluate the performance of the heuristic algorithm, they propose a simple branch and bound approach (B&B) for the small size problems, and a lower bound based on the Lagrangian relaxation method for the large size problems. To the best of our knowledge, this paper is the only one who allows the existence of an intermediate inventory between the two stages working for resequencing the jobs.

1.4 Summary

Our work differs from the models mentioned above mainly in the following three aspects.

In our work, since it is assumed that each production batch requires a setup cost before the first job of this batch is processed, thus the manufacturer wishes to group as many jobs as possible (as one batch) to minimize the total setup cost. Grouping the jobs together implies that some completed jobs may have to wait for other jobs to be completed so they can be finished in the same production batch. Hence, some previously completed jobs will incur the WIP inventory holding cost. Thus, Minimizing the WIP inventory holding cost becomes a very important element when integrating the two logistics stages, i.e. Production and delivery stages. However, most of the papers on integrating of production-distribution problems reviewed above do not consider the WIP inventory holding cost in their models.

In the model of production-inventory-distribution problem studied by this thesis, since there is no limitation on the production rate but a capacity limitation on the transporter, thus, the rate of production and the speed of transportation may not be matched. Consequently, the intermediate inventory will work as a buffer to balance the two logistics stages. The existence of the intermediate inventory allows the jobs to be resequencd after processing in the production stage. Except the work studied by Agnetis et al. (2006), all of the papers reviewed above do not consider this specific function of the intermediate inventory.

In our work, the objective adopted involves both logistics cost and the customer service level. The total operational cost is minimized while guaranteeing a certain customer service level, i.e., each job must be delivered to the pre-specified customer before its due date. Most of the papers involving due-date constraints reviewed above either only consider the time related objective functions or do not take the delivery process into account.

Some others that do study the integrated scheduling problems differ from ours in the model structure and assumptions, and very few of them addresses the problem from a distribution cost and batching point of view. Even among such models, either the due dates are identical or it is assumed that job delivery can be carried out instantaneously without any limit on the batch size. None of the models address the problem with distinct due dates, transportation times and costs, delivery batch size limit, and stage-dependent inventories simultaneously.

2. MULTI-PRODUCT ISPDI PROBLEM

2.1 Multi-Product ISPDI Problem with Unit Job Holding Cost

2.1.1 Introduction

Integrating production and outbound delivery schedules is very critical and common in the supply chain of time-sensitive products, see Chen (2010). This integrating issues are very common in many time-sensitive product supply chain systems. For example, see the newspapers printing and distribution example provided by Buer et al. (1999), mail processing and distribution example provided by Wang et al. (2005), and industrial adhesive materials production and delivery example provided by Devapriya et al. (2006). Therefore, how to effectively integrate these two logistics stages at the operational level so as to lower the operational costs and improve customer service becomes very important to the success of a company. However, most of the existing models on the production-distribution problems only study strategic or tactical levels of decisions, and very few have addressed integrated decisions at the operational level, see Chen (2004).

In this section, we consider the first category of problem (ISPDI problem) where the production and distribution are very closely connected and no finished product inventory is held between them. A schematic diagram of this system is given in Fig. 1. At the beginning of the horizon, the customer requires a set of jobs, and sends the requirement to the supplier. The supplier need to firstly batch process these jobs (materials) on a batching machine at production stage, and then delivered the finished jobs directly to the pre-specified customer at the subsequent delivery stage by a capacitated vehicle without intermediate inventory stage. The capacity of the transporter is measured by a certain amount of volume. Each job is associated with a distinct

due date specified by the customer and a distinct volume. Delay is not allowed, i.e., each job has to be delivered to the customer before its deadline. The processing time of a batch is a constant independent of the jobs it contains. In production, a constant setup time as well as a constant setup cost is required before the first job of this batch is processed. In delivery, a constant delivery time as well as a constant delivery cost is needed for each round-trip between the factory and customer. Moreover, it is supposed that a job which arrives at customer before its due date will incur a customer inventory cost.

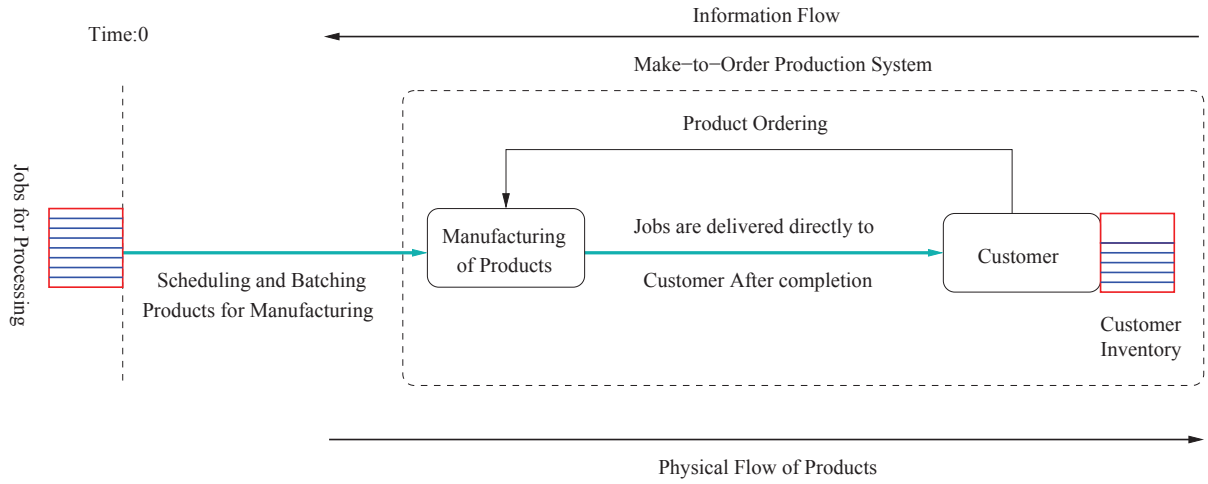

Fig. 1. *Production-Distribution-Inventory Problem*

Our work differs from others mainly in the following two aspects. The first one is that we considered setup, WIP inventory, distribution, customer inventory and due date constraints simultaneously in our model. The second one is that we estimate both the total logistics cost and the customer service level. The logistics cost is measured by actual expenses of operations. The customer service performance is expressed in terms of the deadline of each job, i.e., each job must be delivered to the customer before its due date.

This section is organized as follows. In Subsection 2.1.2, we formally describe the problem and formulate it as a mixed integer programming model (MIP). In Subsection 2.1.3, we proposed a genetic algorithm (GA), for solving the MIP model. In Subsection 2.1.4, we derived a lower bound for the evaluation of the proposed algorithm. At last, in Subsection 2.1.5 and Subsection 2.1.6, we summarized this study.

2.1.2 Problem Description and Formulation

The problem is described as follows. There is a set of jobs ($N = \{1, 2, \ldots, n\}$) to be processed by a batching machine. Each job has a distinct due date d_i and a distinct volume v_i. Specially, the batching machine can process several jobs simultaneously, as a batch, provided that the total volume of these jobs is less than the machine capacity c (i.e. compatible jobs). The processing time of a batch on the batching machine is a constant p_t independent of the jobs it contains. Moreover, in production, if a job is the first job to be processed on the machine or its processing on the machine follows a job from another batch, then a constant setup time s_t as well as a constant setup cost s_c is required before this job can be processed.

After processing, the finished jobs need to be delivered by a vehicle which has the same capacity c as the machine to a pre-specified customer and each job has to be delivered to customer before its due date, i.e., delay is not allowed. Each round-trip between the factory and customer requires a constant delivery time η_t as well as a constant delivery cost η_c. Further, we suppose that a job which arrives at the customer before its due date will incur a customer inventory cost. The objective is to find a coordinated production and delivery scheme such that the sum of setup, delivery and customer inventory cost is minimized.

Since the machine and the vehicle share the same capacity, so each processed batch at the factory can be delivered totally to the customer, i.e., the optimal schedules do not require sequence changes between the machine and vehicle. Thus, in this study, all the batches are processed on the machine and vehicle in the same order. The problem under study requires two distinct, but dependent, decisions to be made: (1) scheduling decision: sequence in which the jobs are to be processed, and (2) batching decision: which job in which batch. An example is depicted in Fig. 2. The problem is complicated by the fact that these two decisions are dependent on each other. The two decisions are dependent because the batch size depends on the jobs in the batch. In a problem with same job size, the jobs can be rearranged according to rules of *SPT, EDD*, etc., which will be the optimal processing sequence in the optimal schedule. However, in our research, because of the limitation of vehicle capacity, it is impossible to initially determine the optimal processing sequence of jobs, thus, we should first determine the optimal processing

sequence of the jobs and then divide these jobs into batches. The problem under study is NP-hard. Because when the unit holding cost in the customer area is 0, i.e. $\beta = 0$, our problem will be equivalent to the batch formation problem of minimizing the number of batches. The batch formation problem is actually a bin-packing problem which is NP-hard. Since our problem can be reduced to a bin-packing problem, thus it is also NP-hard.

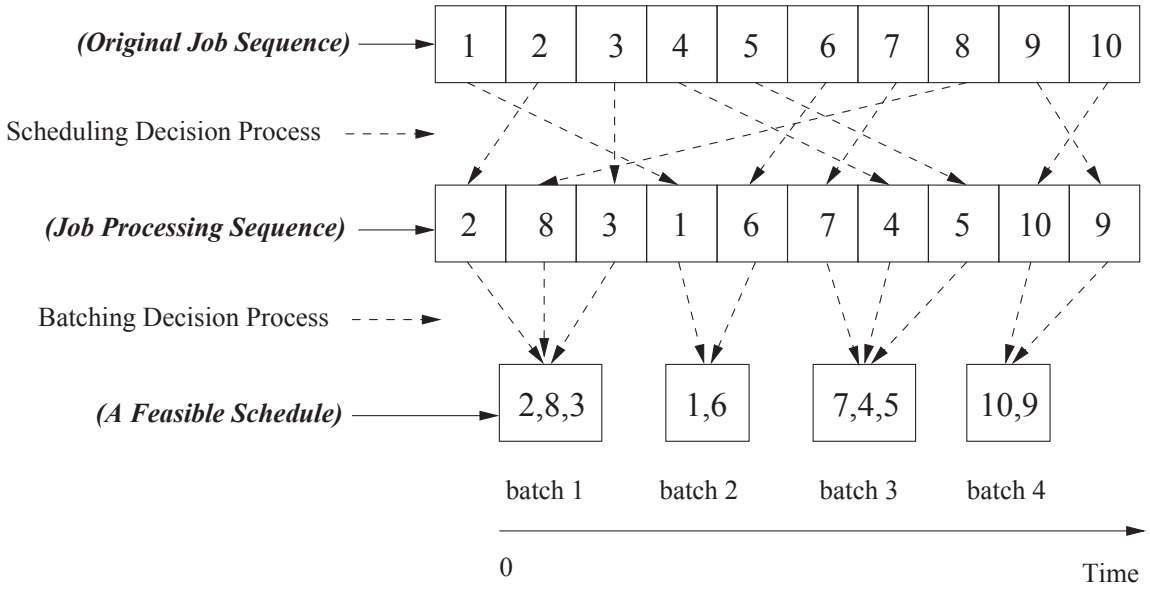

Fig. 2. *Illustration of an example of the problem.*

Although the mathematical programming formulation is not an efficient solution method, it is a natural way to attack scheduling problems. Thus, we formulate our problem to be a mixed integer programming model.

The following notations will be used throughout the section:

Parameters:

1. N: set of all jobs, $N = \{1, 2, \ldots, n\}$, where n is the total number of jobs;
2. K: set of all batches, $K = \{1, 2, \ldots, u\}$, where u is the total number of batches;
3. b_k: index for kth batch, $k = 1, 2, \ldots, u$;
4. i: index for jobs, $i = 1, 2, \ldots, n$;

5. j: index for positions in the processing sequence, $j = 1, 2, \ldots, n$;
6. d_i: due date of the ith job;
7. v_i: volume of the ith job;
8. δ_i: batching decision variable: 1, job i and the job which is assigned to the next position of job i should be batched together as one batch; 0, otherwise;
9. J_{ij}: job i which is assigned to position j;
10. c: common capacity of the machine and vehicle;
11. p_t: batch processing time on machine;
12. β: unit holding cost in customer area;
13. η_t, η_c: round-trip delivery time and cost of the vehicle, respectively;
14. s_t, s_c: setup time and cost, respectively;
15. M: a sufficiently large positive constant;

Decision variables:

1. x_{ijk}: 1, if job i is assigned to the jth position in the processing sequence and belongs to the kth batch; 0, otherwise;
2. C_j: completion (arrival) time (at the customer) of the job which is assigned to jth position in the sequence;

With the notations mentioned above, we build the mixed integer programming model as follows:

$$Min\ Z = (s_c + \eta_c).u + \beta.\left\{\sum_{j=1}^{n}\left(\sum_{i=1}^{n}\sum_{k=1}^{n} d_i.x_{ijk} - C_j\right)\right\} \tag{1}$$

Subject to:

$$\sum_{i=1}^{n}\sum_{k=1}^{n} x_{ijk} = 1, \qquad \forall j, \tag{2a}$$

$$\sum_{j=1}^{n}\sum_{k=1}^{n} x_{ijk} = 1, \qquad \forall i, \qquad (2b)$$

$$\sum_{i=1}^{n}\sum_{j=1}^{n} x_{ijk}.v_i \leqslant c, \qquad \forall k, \qquad (2c)$$

$$\sum_{i=1}^{n} x_{i11} = 1, \qquad (2d)$$

$$u = \sum_{i=1}^{n}\sum_{k=1}^{n} k.x_{ink}, \qquad (2e)$$

$$C_j \leqslant \sum_{i=1}^{n}\sum_{k=1}^{n} d_i.x_{ijk}, \qquad \forall j, \qquad (2f)$$

$$C_{j+1} - C_j \geqslant 0, \qquad \forall j, \qquad (2g)$$

$$C_{j+1} - C_j \leqslant \left(2 - \sum_{i=1}^{n}\left(x_{ijk} + x_{i,j+1,k}\right)\right).M, \qquad \forall j,k, \qquad (2h)$$

$$C_{j+1} - C_j \geqslant \max\left\{(s_t + p_t), \eta_t\right\}.\left\{\sum_{i=1}^{n}(x_{ijk} + x_{i,j+1,k+1}) - 1\right\}, \qquad \forall j,k, \qquad (2i)$$

$$\sum_{i=1}^{n} x_{ijk} - \sum_{i'=1}^{n}(x_{i',j+1,k} + x_{i',j+1,k+1}) \leqslant 0, \qquad \forall j,k, \qquad (2j)$$

$$C_j \geqslant 0, \qquad \forall j, \qquad (2k)$$

$$x_{ijk} \in \{0,1\}, \qquad \forall i,j,k, \qquad (2l)$$

The objective function Eq.(1) minimizes the total joint cost, i.e. the sum of setup, delivery and customer inventory cost. Constraint (2a) specifies that each job i can be assigned to exactly one position j in the processing sequence. Constraint (2b) ensures that each job must be scheduled exactly once. Constraint (2c) guarantees that the number of jobs scheduled in one batch cannot exceed the capacity of the vehicle. Constraint (2d) indicates that the job which is assigned to the first position in the sequence should be in the first batch. Constraint (2e) defines the total number of batches in a feasible schedule. Constraint (2f) guarantees that each job has to be delivered to customer before its due date. Constraint (2g) and (2h) define that two jobs which are assigned to two consecutive positions of one batch in the processing sequence will have the same completion time. Constraint (2i) defines the property of the completion time of two consecutive batches. They indicate that one batch can be processed by the batching machine

only after its previous batch has been completely finished, and one batch can be transported by the vehicle only after its previous batch has been completely delivered. Constraint (2j) indicates that the two jobs which are assigned to two consecutive positions j and $j+1$ will either be in the same batch or in two consecutive batches. Constraint (2k) and (2l) define the range of the variables.

Although the mixed integer programming model provides the optimal solution, variables and constraints increase drastically when the number of jobs increases. Moreover, NP-hard characteristic indicates that the existence of a polynomial time algorithm to solve our scheduling problem is impossible. Hence, developing fast heuristic algorithm for yielding near-optimal solutions is justifiable. In the next section, a genetic algorithm is presented for solving the problem. Before proposing the genetic algorithm, we firstly show some straightforward properties to our model.

(1) There exists an optimal schedule for the problem such that there is no idle time between the first and the last processed jobs in each batch.

(2) There exists an optimal schedule for the problem in which the departure time of each batch on the vehicle is made immediately at the completion time of this batch on the batching machine.

2.1.3 An Improved Genetic Algorithm

Genetic algorithm (GA) is an intelligent stochastic optimization technique based on the evolutionary ideas of natural selection and genetic. GA has been widely studied, experimented and applied in many fields in engineering worlds since its introduction by Holland (1975). GA starts with an initial population of solutions. Each solution in the population is called a chromosome (or individual) which represents a solution in the search space. The chromosomes are evolved through successive iterations, called generations, by genetic operators (selection, crossover and mutation) that mimic the evolution principles assigned to each chromosome according to a problem specific objective function. Generation by generation, the new chromosome, called offspring, are created and survive with chromosomes in the current population, called parents,

to form a new population. We formally describe the GA proposed in this section as follows:

Step 1. Initialize the probability of crossover operator (p_c) and mutation operator (p_m). Set the generation counter $g = 0$. Generate a number of $popsize$ chromosomes as an initial population $pop(0)$ by both random and artificial way. Then, carry out the repair operator (described in Subsection 2.1.3.5) on each chromosome of the newly generated population $pop(0)$.

Step 2. If the termination condition is met, then stop the algorithm and choose the best solution in the population as the final solution of the problem.

Step 3. Evaluate the fitness values of the chromosomes in the population $pop(g)$

Step 4. Selection: Select chromosomes in current population $pop(g)$ for creation of the next generation by a way of roulette wheel.

Step 5. Crossover: Generate a random number α in the uniform distribution on the interval [0, 1]. If $\alpha \geqslant p_c$, apply a suitable crossover operator to the two chosen chromosomes and generate an offspring. Carry out the repair operator on the newly generated offspring. Update the population by replacing the worst chromosome in the population by this offspring. Go to Step 2.

Step 6. Mutation: If $\alpha \leqslant p_m$, generate an offspring from the first of the two chosen chromosomes by a mutation process, carry out the repair operator on the newly generated offspring.

Step 7. Local improvement: Carry out the local improvement procedure, and update the population by replacing the worst chromosome in the population by this offspring. Let $g = g + 1$ and go to Step 2.

The following parts are the descriptions and specifics of the main elements of GA.

2.1.3.1 Encoding scheme and decoding scheme

For chromosome representation, there are a variety of encoding methods, and the most commonly used ones are binary coding and real number coding methods. GA coding has an important impact on algorithm performance such as the searching capability and the diversity of the population. Combining with the characteristics of the scheduling problems in this section, we choose both the real number coding method and binary coding methods. For our problem, a member of the population is a string (or permutation) of genes in which each gene is composed of two parts in which the upper half with an integer number indicates the initial job index (job position in the natural sequence) and the lower half with a binary number indicates the batching (or merging) decisions. We define that the binary number 1 represents that the current job should be batched (or merged) with its next consecutive job so as to be assigned to one batch; 0, otherwise. With this type of chromosome encoding method, each chromosome is composed of two parts in which the upper half permutation of integers represents the job processing sequence, the lower half permutation of binary numbers represents the batching decision. Here, the processing sequence of jobs is represented by the order number of genes from the left side to the right side.

For example, for a problem with a given set of 7 jobs, $N = \{1,2,\ldots,7\}$, a feasible chromosome structure is presented as shown in Fig. 3. In Fig. 3, the upper half permutation of integers indicates that the processing sequence of jobs is (3,1,5,2,7,4,6), the lower half permutation of binary numbers indicates that job 3, 1 and 5 are assigned to the first batch, job 2 is assigned to the second batch, and job 7, 4 and 6 are assigned to the third batch. Thus, the solution represented by the chromosome shown in Fig. 3 is $(3,1,5),(2),(7,4,6)$.

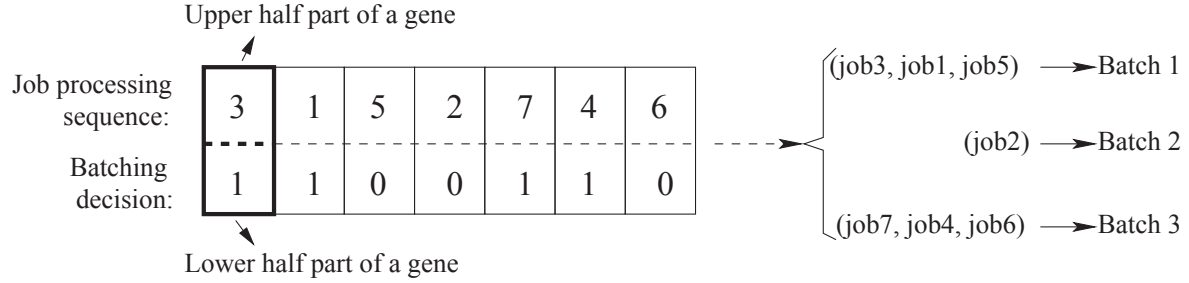

Fig. 3. *Illustration of a feasible chromosome.*

Decoding scheme

The purpose of decoding process is to translate chromosomes (permutations of genes) into a full schedule which can be evaluated by the fitness function which is presented in the Subsection 2.1.3.4. The full schedule of our problem should include the following information: (1) the processing sequence of jobs on machine; (2) which job is assigned to which batch (batching information); (3) the completion time (arrival time) of each job at the customer. Based on this consideration, we propose a two-stage algorithm to deal with the translation process. The first stage of the algorithm will obtain the job processing sequence and batching information, and the second stage will obtain the arrival time at customer of each job. For ease of description, we define that job $[j]$ as the job which is assigned to the jth position of the processing sequence. For example, in the chromosome presented in Fig. 3, the job which is assigned to the first position of the processing sequence is denoted by job $[1]$, i.e. job 3. Moreover, since the jobs which are assigned to one batch will have the same completion time, therefore, we here only need to calculate the completion time of each batch which is denoted by $C_k^{'}$, $k = 1,2,\ldots,u$. So now this decoding scheme can be described as follows:

Stage 1: Obtain job processing sequence and batching information.

Step 1. Initialize the index of job position j and index of batch k to the value 0 and 1, respectively.

Step 2. Assign job $[j]$ to the batch k.

Step 3. For each j in turn, $j < n-1$.

Step 4. If δ_j=1, assign job $[j+1]$ to batch k. $j = j+1$. Go to Step 3.

Step 5. Else, $k = k+1$, assign job $[j+1]$ to batch k. $j = j+1$. Go to Step 3.

Stage 2: Obtain the completion time of each job.

Step 1. For each batch k, $1 \leqslant k \leqslant u$.

Step 2. Initialize the completion time of batch k, $C_k^{'}$, to be $+\infty$,

Step 2.1 For each job i in batch k.

Step 2.2 If $C_k^{'} > d_i$, then let $C_k^{'} = d_i$. Go to Step 1.

Step 3. For each batch k in turn, $k = u, u-1, \ldots, 2$.

Step 4. If $C_{k-1}^{'} > C_k^{'} - \eta_t$, then let $C_{k-1}^{'} = C_k^{'} - \eta_t$.

For example, assume that the permutation with a processing sequence (3,5,1,2,6,4) and a batching decision sequence (1,0,0,1,1,0) is a feasible chromosome produced by GA operator. The due date associated with each job i, $i = 1, 2 \ldots, n$, is assumed to be as follows: $d_1 = 1150, d_2 = 1210, d_3 = 1100, d_4 = 1250, d_5 = 1500, d_6 = 1200$. The round-trip delivery time η_t is 100. We now take the chromosome as an example to illustrate how the decoding scheme works.

Stage 1: Obtain job processing sequence and batching information.

Step 1. Initialize the index of job position j to be 1, and the index of batch k to be 1.

Step 2. Assign job 3 to batch 1.

Loop: *Step* 3 - *Step* 5, we obtain:

δ_1=1, assign job 5 to the third batch. Let $j = 2$,

δ_2=0, let $k = 2$ and assign job 1 to the second batch. Let $j = 3$.

δ_3=0, let $k = 3$ and assign job 2 to the third batch. Let $j = 4$.

δ_4=1, assign job 6 to the third batch. Let $j = 5$.

δ_5=1, assign job 4 to the third batch. Stop.

Hence, we obtain the job processing sequence and batching information is (3,5), (1), (2,6,4).

Stage 2: Obtain the completion time of each job.

Loop 1: *Step* 1 - *Step* 2, we obtain:

$C_{b_3}^{'} = \min\{d_2, d_6, d_4\} = 1200$.

$C_{b_2}^{'} = \min\{d_1\} = 1150$.

$C_{b_1}^{'} = \min\{d_3, d_5\} = 1100$.

Loop 2: *Step* 3 - *Step* 4, we obtain:
$C_3' = 1200$.
$C_2' > C_3' - \eta_t$, let $C_2' = C_3' - \eta_t = 1100$.
$C_1' > C_2' - \eta_t$, let $C_1' = C_2' - \eta_t = 1000$.

Hence, the completion time of each job i is as follows (according to the processing sequence):
$C_3 = C_5 = 1000, C_1 = 1100, C_2 = C_6 = C_4 = 1200$.

2.1.3.2 Initialize population

For standard GA, in theory, the global convergence nature of GA can guarantee robustness of initial population of GA, but in practice, because the convergence constraint can not be satisfied outright, consequently, this result in that the effectiveness and efficiency of the algorithm depend much on the quality of initial population. Therefore, we use both artificial and random way to generate initial population. Artificial method can guaranty the quality of the initial population to a certain extent. Random way can guaranty the diversity of the initial population. In this study, we proposed two artificial chromosomes which were introduced into the initial population as two possible good original searching points.

The first artificial chromosome is generated by the following method:

Step 1. Rearrange the jobs according to the increasing order of the due dates;

Step 2. For $1 \leqslant i \leqslant n$

Step 3. Set δ_i=0; Go to Step 2.

The chromosome generated by this method represents a solution where the jobs are processed according to the increasing order of due dates, and every job forms a separate batch. This chromosome can obtain a good objective value when the interval between two consecutive due dates is not smaller than the round-trip delivery time.

The second artificial chromosome is generated by the following method:

Step 1. Rearrange the jobs according to the increasing order of due dates;

Step 2. For $1 \leqslant i < n$;

Step 3. If $v_i + v_{i+1} \leqslant c$, then set δ_i=1;

Step 4. Else, set δ_i=0, $i = i + 1$; Go to Step 2.

The chromosome generated by this method represents a solution with a small number of batches which can obtain a good objective value when the transportation cost is relatively important.

Then, the other chromosomes in the initial population are generated by a random way. However, since the chromosomes are randomly generated, it does not always produce a feasible solution. Hence, a repair operator described in Subsection 2.1.3.5 has to be carried out after a chromosome is generated in order to keep its legitimacy.

Further, for the value of the population size *popsize*, there is no clear indication about how many chromosomes we should generate using the two methods mentioned above for a population. After some experiments, 300 is chosen as the size of the population for GA in the simulation process of this study.

2.1.3.3 Genetic operators

(1) Selection operation

The core idea of selection is to choose the good-quality chromosomes for copying them to the next generation and choose the poor-quality chromosomes for eliminating by maximum probability, so that the average population fitness is improved. Thus the biological "survival of the fittest" is reflected. In this study, we adopt "roulette wheel of reserving elites" method in which chromosomes are given a probability of being selected that is directly proportionate to their fitness, two chromosomes are then chosen randomly based on these probabilities, see Goldberg (1989). "Reserving elites" means that before each new generation is built, copy the best chromosome in its previous generation to the new generation. In this way, the best chromosomes generated in each generation can survive to the end of the algorithm.

(2) Crossover operation

The main purpose of crossover operation is to recombine the features of two randomly selected parents from the population with the aim of producing better offsprings. Regarding to the permutation-based representation, several crossover operators have been proposed by Iyer and Saxena (2004), Poon and Carter (1995), Wang and Wu (2004). Among them, the following crossover operators have been widely used: partially matched crossover intending to keep the absolute positions of genes and linear order crossover intending to preserve relative positions. Here, we designed three crossover operators based on these two kind of crossover methods, and they are selected with equal probabilities.

Crossover A as shown in Fig. 4 is a linear order crossover which intends to preserve relative positions of the genes, and works on the upper half part (job processing sequence) of each chromosome. The function of crossover A aims to change the job processing sequence. The details of crossover operator A is shown as follows:

Step 1. Select a subsequence of job positions in the job processing sequence of a parent with a random size within 1 to $n-1$.

Step 2. Select the job positions which includes the same jobs as the job positions found in Step 1 include.

Step 3. Produce the offsprings by placing the jobs in the selected job positions of one parent to the selected positions in the other parent through making a left-to-right scan.

Crossover B as shown in Fig. 5 is a partially matched crossover which intends to keep the absolute positions of genes, and works on the lower half part (batching decision sequence) of each chromosome. The function of crossover B aims to change the division decisions (number of batches). The details of crossover operator B is shown as follows:

Step 1. Randomly choose two crossover points in the batching decision sequence with a size within 1 to $n-1$.

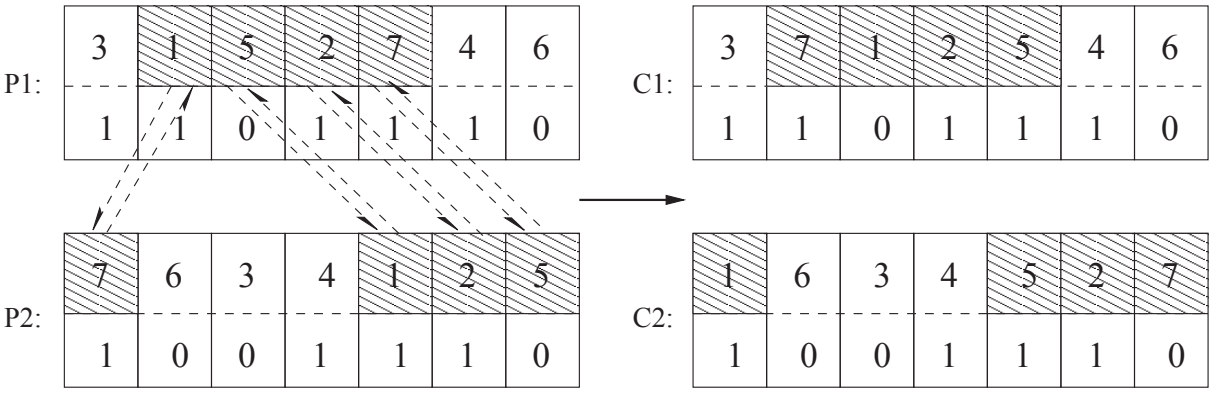

Fig. 4. *Illustration of the crossover operator A.*

Step 2. Take out the subsequence of batching decision between the two crossover points in each parent. Produce the offsprings by swapping the two subsequences of batching decision.

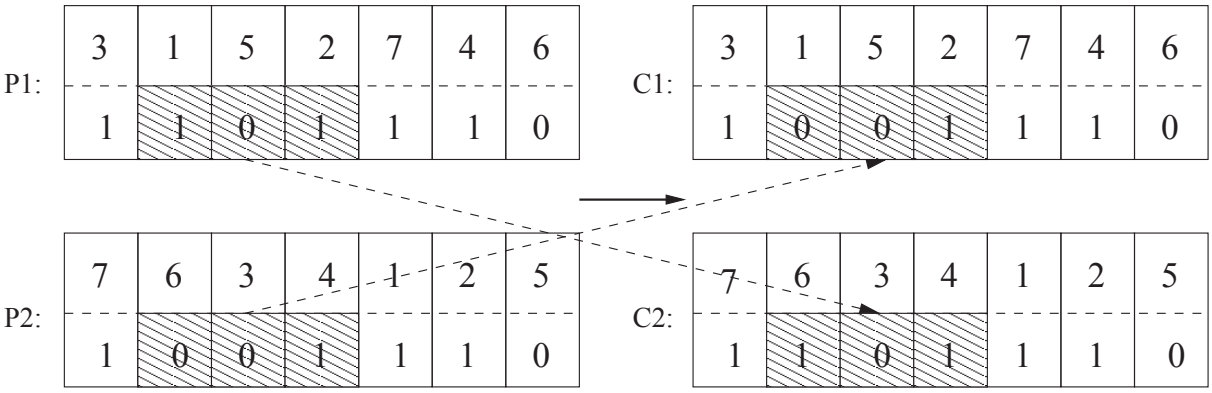

Fig. 5. *Illustration of the crossover operator B.*

Crossover C is an integrative operation of Crossover A and Crossover B, it intends to change both the job processing sequence and batch decision sequence.

(3) Mutation operation

Mutation is used to produce small perturbations on chromosomes to promote diversity of the population. There are several mutation operators such as swapping, inversion, insertion and shift mutation (see Gen and Cheng (1997)). In this study, we use two mutation operators which are explained as follows: (1) The first mutation operator, as shown in Fig. 6, randomly selects

two positions in the job processing sequence of a chromosome, and then invert the subsequence between these two positions; (2) The second mutation operator, as shown in Fig. 7, randomly

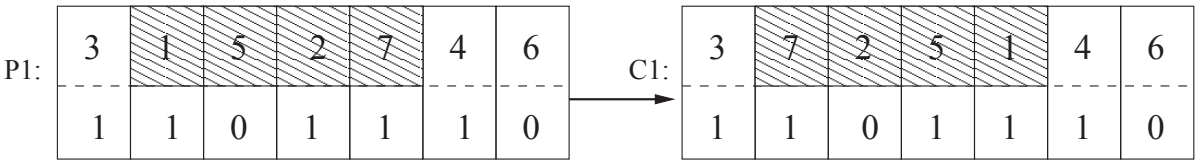

Fig. 6. *Illustration of the mutation operators A and B.*

selects a mutation point in the batching decision sequence of a chromosome, if the value the batching decision located in this position is 1, then change it to be 0; otherwise, change it to be 1.

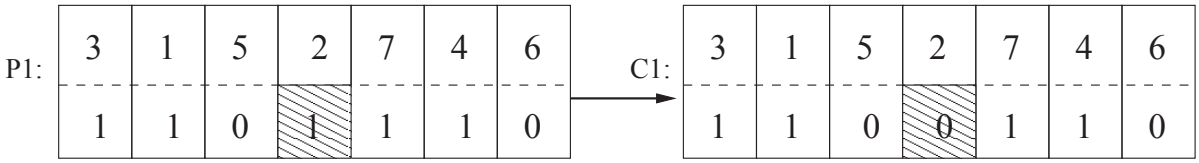

Fig. 7. *Illustration of the crossover operator C.*

2.1.3.4 Fitness function and stopping criterion

The fitness value is the measure of goodness of a solution with respect to the original objective function and the degree of infeasibility. For the cost minimization problem we have considered, candidate solutions with lower costs imply better solutions and vice versa. Therefore, for each chromosome h, its fitness value f_h can be evaluated by the reciprocal of the objective function Eq. 1, i.e.

$$f_h = \frac{1}{\left(s_c + \eta_c\right).u + \beta.\left\{\sum_{j=1}^{n}\left(\sum_{i=1}^{n}\sum_{k=1}^{n} d_i.x_{ijk} - C_j\right)\right\}} \tag{3}$$

Genetic operation is a repeated iterative search method. It progressively approaches but never arrives at the best solution. Thus conditions of termination are needed to be setup. The most common way of termination is to setup maximal generations. Once the objective function reaches the optimal value, termination can be done by control of deviation. The second method

of termination is to monitor the variation of fitness of the best individual. Once it shows negligible variation in a certain number of generations, termination can be done. Here, we use two stop criteria mentioned above as the termination condition of the algorithm. The algorithm terminates when at least one of these two conditions mentioned above is met.

2.1.3.5 Repair operator

The function of repair operator is to check the legitimacy (feasibility) of the chromosomes in the population. In a feasible solution, the total volume of the jobs in each batch should be less than the capacity of the vehicle, see Eq.(2c). So in the initialization of the population or a newly generated population, every chromosome should meet the capacity constraint. If a chromosome does not satisfy the capacity constraint, it would be repaired to be feasible. This correction mechanism of the repair operator is designed to move jobs from the over-capacity batches to other batches with surplus capacity. Because the jobs in the same batch have the same processing times, so the sequence of jobs in one batch has no effect on the objective value. Therefore, for ease of presentation, we here suppose that all the jobs in one batch are arranged according to the increasing order of their due dates. The repair operator can be described as follows:

Step 1. Decode a chromosome to a solution (feasible or infeasible) by the decoding method described in Subsection 2.1.3.1.

Step 2. For $1 \leqslant k \leqslant u$

Step 3. If $Vol(b_k) > c$

Step 3.1 If $Vol(b_k) - c + Vol(b_{k+1}) < c$, assign the last job in the batch b_k, denoted by j_{b_k}, to the batch b_{k+1} by setting $\delta_{j_{b_k-1}} = 0$ and $\delta_{j_{b_k}} = 1$, respectively.

Step 3.2 Else, insert a new batch in position $(k+1)$, denoted by b_{k+1}, and assign the last job of batch b_k to the new batch b_{k+1} by setting $\delta_{j_{b_{k-1}}} = 0$. Let $u = u + 1$. Go to Step 2.

Here, $Vol(b_k)$ represents the total volume of jobs which are assigned to the batch b_k.

2.1.3.6 Local improvement

In order to improve the performance of the GA, we introduce a local improvement procedure based on the steepest descent method into the proposed GA. Before proposing the local improvement procedure, we firstly propose a neighborhood generation method as follows:

Step 1. Randomly select two genes from a chromosome.

Step 2. Produce a new chromosome by swapping the two selected genes.

Step 3. Check the legitimacy of the new chromosome by the repair operator described in the subsection 2.1.3.5.

With the neighborhood generation method mentioned above, we design the following local improvement procedure to improve the performance of the GA.

Step 1. For $1 \leqslant i \leqslant \delta$.

Step 2. Generate a neighborhood of the chromosome by the neighborhood generation method mentioned above.

Step 3. If the fitness values of the two chromosomes satisfy: $f_{neighborhood} > f_{chromosome}$, then replace the chromosome by the neighborhood.

Step 4. Else, $i = i + 1$. Go to Step 1.

This local improvement procedure is conducted after the mutation operator, if a better neighborhood has been found within δ iterations, then replace the newly generated chromosome by the neighborhood, else, stop the local improvement procedure and keep the original chromosome.

2.1.4 Bin-Packing Problem Based Lower Bound Derivation

Since it is difficult to obtain an optimal solution in reasonable computing time even for the situation with 20 jobs, so we try to establish an efficient lower bound (LB) for evaluating the

proposed algorithm. Based on this consideration, we propose the following procedure to generate a lower bound on the objective value as a comparison.

We divide the proposed problem into two subproblems in which the first one (P_1) is associated with the objective function of

$$Z_1 = (s_c + \eta_c).u$$

, and the second one (P_2) is associated with the objective function of

$$Z_2 = \beta.\left\{\sum_{j=1}^{n}\left(\sum_{i=1}^{n}\sum_{k=1}^{n} d_i.x_{ijk} - C_j\right)\right\}$$

. Then we establish the lower bounds for the problem P_1 and P_2, denoted by l_1 and l_2, respectively. Consequently, the lower bound of the problem proposed in this section *LB* will be obtained by $l_1 + l_2$.

Lower bound of P_1

We assume that there is no customer holding cost for jobs, i.e. $\beta = 0$, consequently, the problem becomes:

$$\min Z_1 = (s_c + \eta_c).u \tag{4}$$

Subject to:

$$\sum_{i=1}^{n} x_{ik}.v_i \leqslant c, \ \ \forall k, \tag{5}$$

This is a classic Bin-packing problem. For this problem, Martello and Toth (1990) have proposed an efficient lower bound which is described as follows:

Theorem 1. *Given any instance I of Bin-packing problem, and any integer* α*,* $0 \leq \alpha \leq c/2$*, let*

$$J_1 = \{j \in N : v_j > c - \alpha\}$$
$$J_2 = \{j \in N : c - \alpha \geq v_j > c/2\}$$
$$J_3 = \{j \in N : c/2 \leq v_j \leq \alpha\}$$

then,

$$L(\alpha) = |J_1| + |J_2| + \max\left\{0, \left\lceil \frac{\sum_{j\in J_3} v_j - (|J_1|.c - \sum_{j\in J_2} v_j)}{c} \right\rceil \right\} \tag{6}$$

is a lower bound of $Z(I)$.

So obviously the best lower bound can be found by the following equation:

$$L_2 = \max\{L(\alpha) : 0 \leq \alpha \leq c/2, \alpha \; integer\} \tag{7}$$

They pointed out that L_2 can be determined efficiently by the following way:

Theorem 2. *Let* V *be the set of all the distinct values* $v_i \geq c/2$. *Then*

$$L_2 = \begin{cases} n & if\ V = \emptyset \\ \max\{L(\alpha) : \alpha \in V\} & otherwise \end{cases} \tag{8}$$

They further proved that L_2 can be computed in $O(n)$ time if the jobs are sorted according to the decreasing order of their volumes and developed a pseudo-code based on which our lower bound is calculated.

Lower bound of P_2

According to the lower bound generation method mentioned above, we here build a new problem ($P_2^{'}$) with the following assumptions:

1. The round-trip delivery cost is 0, e.g. $\eta_c^{'} = 0$;
2. The job volumes are 1, e.g. $v_i^{'} = 1$ with $i = 1, 2, \ldots, n$;
3. The capacity of the transporter $c^{'} = \left\lfloor \frac{c}{\min\{v_i\}} \right\rfloor$;

Then, we obtain a new problem ($P_2^{'}$) which is described as follows:

$$\min Z_2^{'} = \beta.\left\{ \sum_{i=1}^{n} \sum_{k=1}^{n} d_i.x_{ik} - C_i \right\} \tag{9}$$

Subject to:

$$\sum_{k=1}^{n} x_{ik} = 1, \qquad \forall i, \qquad (10a)$$

$$\sum_{i=1}^{n} x_{ik}.v_i \leqslant c^{'}, \qquad \forall k, \qquad (10b)$$

$$\sum_{i=1}^{n} x_{i1} = 1, \qquad (10c)$$

$$u = \sum_{k=1}^{n} k.x_{nk}, \qquad (10d)$$

$$C_i \leqslant d_i, \qquad \forall i, \qquad (10e)$$

$$C_{i+1} - C_i \geqslant 0, \qquad \forall i, \qquad (10f)$$

$$C_{i+1} - C_i \leqslant (2 - (x_{ik} + x_{ik})).M, \qquad \forall i,k, \qquad (10g)$$

$$C_{i+1} - C_i \geqslant \max\left\{(s_t + p_t), \eta_t\right\}.\left\{(x_{ik} + x_{i,k+1}) - 1\right\}, \qquad \forall i,k, \qquad (10h)$$

$$x_{ik} - (x_{i+1,k} + x_{i+1,k+1}) \leqslant 0, \qquad \forall i,k, \qquad (10i)$$

$$C_i \geqslant 0, \qquad \forall i, \qquad (10j)$$

$$x_{ik} \in \{0,1\}, \qquad \forall i,k, \qquad (10k)$$

The new problem $P_2^{'}$ is equivalent to the problem studied by Baptiste (2000). Thus after rearrange the jobs according to the increasing order of the jobs' due dates, the problem $P_2^{'}$ could be solved by the dominance related dynamic programming scheme provided by Baptiste (2000).

With the above assumption, we obtain the following proposition:

Theorem 3. *Given any instance I, assume* π *and* $\pi^{'}$ *are two optimal solutions of the problem* $P_2(I)$ *and* $P_2^{'}(I)$*, respectively, and* $f(\pi)$ *and* $f(\pi^{'})$ *are the evaluations of* π *and* $\pi^{'}$*, respectively. Then,* $f(\pi^{'})$ *is lower bound of* $f(\pi)$*.*

Proof. *(By contradiction)* Suppose $f(\pi) \leqslant f(\pi^{'})$. Since in problem P_2, the transporter can held a maximum number of $\left\lfloor \frac{c}{\min\{v_i\}} \right\rfloor$ with $i = 1,2,\ldots,n$ jobs because of the limitations of job volumes. However, in the new problem $P_2^{'}$, with the assumptions (1) and (2), the transporter

can always held a number of $\left\lfloor \frac{c}{\min\{v_i\}} \right\rfloor$ with $i = 1,2,\ldots,n$ jobs. Therefore, a solution feasible for the problem P_2 will be also feasible for the problem $P_2^{'}$. Thus we can always replace the solution $\pi^{'}$ by π to generate a better solution. This contradicts the assumption that the solution $\pi^{'}$ is the optimal solution of the problem $P_2^{'}$ which completes the proof. ■

We obtain this lower bound by two steps of approximation in which the first one is to assume that the customer holding cost is 0, and the second one is to find a lower bound for the relaxed problem. Therefore, the deviation will be accumulated after the two approximation steps. Therefore, even though this lower bound is efficient for the classic Bin-packing problem, however, this lower bound is not very efficient for our problem when the transporter capacity is relatively larger than the job volumes.

2.1.5 Experiment and Computational Results

In this section, the computational experiments are carried out to test the performance of the model presented in Section 2.1.2 and the proposed GA. The proposed GA is coded in JAVA language and implemented on the computer with 4Gb RAM and 512KB L2 cache. As a comparison, CPLEX solver is used to exactly solve the model with small-scale random instances, and a lower bound is proposed to evaluate the efficiency of the GA for large scale problem instances.

The parameters of the GA are summarized as follows:

- Population size $popsize$: 300
- Termination condition: 300 iterations or fitness of the best individual did not change for a certain number of generations
- Crossover probability p_c: 0.7
- Mutation probability p_m: 0.1
- Variable δ in local improvement procedure: 30

2.1.5.1 Random instances with small sizes

We create 5 random instances with small sizes based on the following parameter settings. The batch processing time on the batching machine is randomly generated from the uniform distribution with range [1, 5]. The volumes of the different jobs are randomly generated from the uniform distribution with range [1, 30]. The setup time and setup cost are 50 and 100, respectively. The vehicle's capacity is 50, further, its round-trip delivery time and cost are 100 and 200, respectively. The unit holding cost in the customer area is 1.0. For each combination, we randomly generated 50 problem instances and take the average value (Avg.Value) and average cpu time (Avg.CpuT) which are defined in the Subsection 2.1.5.2 for the performance test of the proposed GA. We run the CPLEX solver and the GA using the 5 instances and the results

Table 1. *Results of random instances with small sizes.*

Instance	Size (n)	CPLEX			GA	
No.		Value	CpuT (s)		Value	CpuT (s)
1	5	792.74	0.30	Avg.	792.74	2.13
				Max.	792.74	2.41
2	7	936.74	4.10	Avg.	936.74	2.55
				Max.	936.74	2.83
3	9	1139.62	513.30	Avg.	1139.62	3.18
				Max.	1139.62	4.07
4	11	1369.69	7989.40	Avg.	1369.69	3.69
				Max.	1369.69	4.58
5	15	1613.76	55305.50	Avg.	1618.91	4.41
				Max.	1637.55	4.80

are shown in Table 1. We observe that our GA runs much faster than the CPLEX solver. Although the CPLEX solver finds the optimal solution, the computational time of CPLEX grows exponentially as the instance size increases. The computational time of the proposed GA is very short. Moreover, the GA can obtain optimal or near optimal solutions for all of the situations.

2.1.5.2 Random instances with large sizes

To test the performance of the GA thoroughly, we conduct experiments using random instances with large problem sizes. We consider three scenarios where the capacity of the transporter was generated from a discrete uniform distribution in the interval [50,100], [100, 150] and [150, 200], respectively. For each scenario, we considered three cases with small, middle and large job volumes, which were randomly generated from the uniform distributions in the intervals [1, 10], [1, 20] and [1, 30], respectively. In each case, we set the number of jobs as 30, 50, 70, 100 and 150. The job processing time, setup time and setup cost were randomly generated from a discrete uniform distribution in the interval [1,5], [10, 40] and [200, 400], respectively. The round-trip delivery time and cost of the transporter were generated from the uniform distribution with range [50, 150] and [300, 600], respectively. Unit customer holding cost were randomly generated from the uniform distribution with range [0.001, 0.005]. The due date associated with each job j was generated from the uniform distribution with range [10000, 12000].

Considering the different transporter capacity values, number of jobs, and delivery costs, we tested 45 situations of the problem. For each situation, we randomly generated 50 problem instances for the performance test of the GA. Based on the derived lower bound, the error ratio is defined as *ER=(GAS-LB)/LB* $ER = (GAS - LB)/LB$, where GAS denotes the evaluation of the solution generated by the proposed GA, and average error ratio is defined as *Avg.ER=(*$\sum ER$*)/number of instances tested for a parameter combination*. The average running time ($Avg.CpuT$) is calculated by *Avg.CpuT=*$\sum$*(CpuT of each instance)/number of instances tested for a parameter combination*. The computational results are displayed in Table 2-4.

Tables 2, 3 and 4 show clearly that the average error ratios of GA of all the situations were no more than 14%, which demonstrate that the proposed GA is capable of generating near-optimal solutions within a reasonable amount of CPU time. As seen in each table, the average error ratios appear in an increasing trend as the value of n increases. One of its reasons may be that the lower bound increases as n increases, but the growth rate is a little smaller than that of the objective value of GA, hence the difference between the objective value generated by GA and the lower bound may grow with the increase of n. They also indicate that the average ratios

Table 2. *Results of random instances with large sizes for* $c \in [50, 100]$.

Size		$v_i \in [1, 10]$		$v_i \in [1, 20]$		$v_i \in [1, 30]$	
n		ER (%)	CpuT (s)	ER (%)	CpuT (s)	ER (%)	CpuT (s)
30	Avg.	0.65	10.34	2.04	14.48	4.02	16.81
	Max.	1.97	22.84	20.11	29.59	18.52	24.88
50	Avg.	0.60	25.22	4.35	30.26	7.90	32.45
	Max.	1.45	36.71	16.86	36.39	16.77	38.68
70	Avg.	1.43	43.29	5.24	42.29	9.14	44.00
	Max.	14.49	48.68	14.86	47.07	15.25	50.62
100	Avg.	2.64	60.57	5.46	56.18	10.73	60.59
	Max.	12.36	65.85	13.70	62.23	11.48	68.93
150	Avg.	3.26	86.82	7.55	81.31	13.23	89.77
	Max.	11.00	93.63	14.46	89.14	15.22	100.07

Table 3. *Results of random instances with large sizes for* $c \in [100, 150]$.

Size		$v_i \in [1, 10]$		$v_i \in [1, 20]$		$v_i \in [1, 30]$	
n		ER (%)	CpuT (s)	ER (%)	CpuT (s)	ER (%)	CpuT (s)
30	Avg.	0.92	5.36	0.47	6.92	1.82	7.61
	Max.	1.39	10.10	1.21	15.65	9.41	14.94
50	Avg.	1.17	11.65	0.44	17.42	4.03	20.52
	Max.	3.77	19.17	0.92	24.15	12.69	25.68
70	Avg.	1.10	23.41	1.39	29.20	4.50	29.60
	Max.	2.95	33.47	14.69	32.43	11.71	33.11
100	Avg.	1.18	37.50	2.47	39.90	4.69	40.20
	Max.	2.83	42.18	13.15	43.12	9.95	44.10
150	Avg.	1.94	54.65	4.36	57.68	6.34	59.14
	Max.	13.85	59.06	10.05	63.61	11.00	64.48

appear in an increasing trend as the variation range of the jobs' volumes increases. It may be interpreted that there is large differences between job volumes when the job volume was generated in a large variation range, hence, according to the lower bound calculation method mentioned above, the transporter can be made full use of to hold jobs, i.e., the remaining space of the transporter will be stuffed by the small jobs. Consequently, the number of batches decrease, which leads to a worse lower bound. Moreover, with comparison of the three tables, we

Table 4. *Results of random instances with large sizes for* $c \in [150, 200]$.

Size		$v_i \in [1, 10]$		$v_i \in [1, 20]$		$v_i \in [1, 30]$	
n		ER (%)	CpuT (s)	ER (%)	CpuT (s)	ER (%)	CpuT (s)
30	Avg.	3.02	3.46	0.87	5.48	0.45	7.95
	Max.	5.87	7.00	1.67	9.54	0.63	14.13
50	Avg.	1.74	55.89	1.77	14.80	0.46	20.62
	Max.	3.49	67.13	10.14	21.94	1.01	29.88
70	Avg.	1.88	19.20	1.61	24.55	2.04	29.12
	Max.	4.21	29.75	11.38	30.94	14.19	32.05
100	Avg.	1.58	35.11	2.35	38.23	2.76	40.52
	Max.	4.90	39.93	13.86	41.17	11.76	44.32
150	Avg.	2.41	54.37	3.06	54.95	4.34	58.67
	Max.	6.77	58.37	13.55	57.87	10.54	66.25

also can observe that the maximum error ratios did not appear in certain trend as the variation range of the transporter capacity increases, this indicate that the changes of the capacity has no influences on the performance of GA. However, the maximum error ratios of GA of all the situations were no more than 26%, which indicate that the reliability and stability of the GA.

2.1.6 Summary

The coordination of production and distribution is an important issue in manufacturing and logistics management. In this section, we have tackled a coordinated scheduling problem of

production and distribution with customer inventory cost considerations. Particularly, it is assumed that each job is associated with an arbitrary volume and a distinct due date. Our objective is to find a coordinated scheme such that the sum of the setup, delivery and customer holding cost is minimized.

In order to clearly describe the problem, a mixed integer programming model is presented. We then showed that the problem is NP-hard and proposed GA approach to solve it. Finally, we evaluated the performance of the GA for both small size and large size problems. For small size problems, we compared our proposed GA with CPLEX solver, and the computational results show that the proposed GA can find the optimal or near optimal solution in a very short running time. For large size problems, we derived a lower bound as a comparison, and the results indicate the efficiency of the proposed GA in practice.

2.2 Multi-Product ISPDI Problem with Arbitrary Job Holding Cost

2.2.1 Introduction

The model studied in this Section is an extension of the model formulated in the Section 2.1. Since the inventory holding cost represents a combination of the cost of capital, the cost of physical storage and the cost of losses due to spoilage, etc; hence, it highly depends on the inventory type as well as the product itself. Therefore, it is much more reasonable to calculate the inventory costs according to different types of jobs. In the Section 2.1, we studied the integrating scheduling model in which each job has an identical unit inventory cost. In this part, based on the consideration mentioned above, we extended the model presented in Section 2.1 to the one in which each job is associated with a distinct unit inventory cost.

This Section is organized as follows: We described and formulated the problem in Subsection 2.2.2. Then, we proposed a Tabu search approach (TS) in Subsection 2.2.3. Finally, in Subsection 2.2.4 and Subsection 2.2.5, we make some concluding remarks based on the computational results and suggest directions for future research.

2.2.2 Non-linear Problem Formulation

The problem description is same as that presented in the Section 2.1 except that the job unit holding cost in this problem is arbitrary, i.e., each job is associated with a distinct unit customer holding cost. Based on the characteristics of the problem, we formulate the problem as a non-linear model. Before the model is presented, the parameters and variables used in the model are firstly described below.

- J: the set of all jobs, $J = \{j_1, j_2, \ldots, j_n\}$, where n is the total number of jobs;
- i: index for jobs, $i = 1, 2, \ldots, n$;
- d_i: due date of the job i;
- v_i: volume of the job i;
- β_i: unit holding cost in customer area for job i;
- c: capacity of the transporter;
- λ_c, λ_t: setup cost and setup time, respectively;
- η_c, η_t: round-trip delivery cost and time, respectively.
- p_t: processing time of the batching machine;
- u: the total number of batches;
- b: index for batches, $b = 1, 2, \ldots, n$;
- C_i: completion (arrival) time of the job i;
- C_b^B: completion (arrival) time of batch b;
- b_i: the number of batch to which the job i is assigned;

We now formulate the problem as follows:

$$\min Z = (\lambda_c + \eta_c).u + \left\{ \sum_{i=1}^{n} \beta_i.(d_i - C_i) \right\} \tag{11}$$

Subject to:

$$C_i \leq d_i, \quad \forall i \in [1,n]; \tag{12a}$$

$$\sum_{i=1}^{n} v_i \leq c, \quad b_i = b, b \in [1,u]; \tag{12b}$$

$$C_b^B = C_i, \quad \forall i \in [1,n]/b_i = b, b \in [1,u]; \tag{12c}$$

$$C_b^B \leq C_{b+1}^B - \max\{\lambda_t + p_t, \eta_t\}, \; b \in [1,u]; \tag{12d}$$

The objective function Eq. (11) minimizes the sum of setup, delivery and customer inventory cost. Constraint (12a) guarantees that each job has to be delivered to customer before or on its due date. Constraint (12b) ensures that the number of jobs scheduled in one batch cannot exceed the capacity of the transporter. Constraint (12c) indicates that two jobs which are assigned to the same batch will have the same completion time. Constraint (12d) defines the property of the completion time of two consecutive batches. They indicate that one batch can be processed by the batching machine only after its previous batch has been completely finished, and one batch can be delivered by the transporter only after its previous batch has been completely transported.

Since the problem addressed in the Section 2.1 is NP-hard, therefore, this problem can also be proved to be NP-hard by the same method described in Section 2.1.2. Based on this problem characteristic, we, in the next section, establish a tabu search approach (TS) for solving this problem.

2.2.3 Tabu Search Algorithm

The Tabu search approach (TS), proposed by Glover (1989) and Glover (1990), is a meta-heuristic global optimization method for large combinatorial optimization problems. It is different from the well-known hill-climbing local search methods in the sense that it does not become trapped in local optima. TS has been utilized for various scheduling problems (for example, see Chen et al. (2007), Hertz and Widmer (1996) and Xu et al. (2010), among others).

The success of TS in above studies motivated us to develop a TS approach for the proposed problem in this study. The proposed TS is described below with the following notations: s_0, s_c and s^* are the initial, the current and the best solutions, respectively; $f(s)$ indicates the evaluation of the solution s, $N(s)$ is the neighborhood of s, $\overline{N}(s)$ denotes the admissible subset of $N(s)$; s_{iter} denotes the best solution in $\overline{N}(s)$. The details of the algorithm are presented as follows:

Algorithm 1 Steps of the proposed TS approach.

1: Initialization parameters: $v_j, d_j, \lambda_t, \lambda_c, \eta_t, \eta_c, c, \beta_j, n$;
2: Set $f(s^*)$=0, *TabuTenure*=k ;
3: Generate the initial solution s_0 ;
4: Conduct the correction operator on s_0 ;
5: $s \leftarrow s_0$;
6: **while** termination criterion is not met **do**
7: Find the set of candidate neighbors $N(s)$;
8: Find $s_{iter} \in \overline{N}(s)$;
9: Update the *TabuList* ;
10: Update s^* and $f(s^*)$;
11: **end while**

2.2.3.1 Solution representation and initial solution

We represent a solution by a vector of batch numbers in which the ith entry indicates the number of the batch to which the job i is assigned to. As an example, assume that the representation of a feasible solution to a problem with 8 jobs, J=$\{j_1, j_2, \ldots, j_8\}$, is [1,1,1,2,5,1,3,4]. This representation indicates that jobs j_1, j_2, j_3 and j_6 are assigned to the first batch, the jobs j_4, j_5, j_7 and j_8 are assigned to the second, fifth, third and fourth batch, respectively. Hence, the processing sequence and batch information of the jobs can be expressed as (j_1-j_2-j_3-j_6)-(j_4)-(j_7)-(j_8)-(j_5).

TS starts with an initial feasible solution and tries to improve it iteratively. For our problem, we construct an initial solution s_0 where each job is treated as a separate batch. For example, for a

problem with 8 jobs, J=$\{j_1, j_2, \ldots, j_8\}$, the initial solution will be [1,2,3,4,5,6,7,8].

2.2.3.2 Move operator and neighborhood definition

The problem under study requires two distinct, but dependent, decisions to be made: (1) number of batches, and (2) which job in which batch. Therefore, an efficient neighborhood generation method should consider both of the two decisions mentioned above. With this consideration, we design the following neighborhood generation method (See Algorithm 2). The main idea of the neighborhood generation method is that we firstly select a random job (denoted by j_i) and then modify the number of batch to which the job is assigned as a randomly selected batch number (denoted by $b_i^{'}$ with $1 \leq b_i^{'} \leq n$) given that the capacity of transporter is not exceeded, finally we conduct a correction operator which is described in the Section 2.2.3.5 on this modified solution in order to keep its legitimacy. Assume s_c is the current solution, then this neighborhood generation method can be described as follows (See Algorithm 2): This neighborhood generation

Algorithm 2 Neighborhood Generation Method.

1: For a current solution s_c, randomly generate a pair of numbers $(i, b_i^{'})$ in the range of [1,n];
2: **if** batchSize[$b_i^{'}$]+ $v_i \leq c$ **then**
3: Obtain new neighbor $s_c^{'}$ by setting $b_i = b_i^{'}$;
4: **else**
5: Go to Step 1 ;
6: **end if**
7: Conduct the correction operator on the newly generated neighbor $s_c^{'}$;

method can find a neighbor by changing the number of batches, the number of batch to which one job is belongs or the two previous decisions simultaneously. For example, assume that [1,1,3,2,2,3,4,1] is a feasible current solution. Depends on the randomly generated job index and the batch number, i.e. $(i, b_i^{'})$, the neighbor of the current solution could be:

(1) For $(i, b_i^{'})$=(3,6): after carrying out the correction operator on this newly generated neighbor, we obtain [1,1,5,2,2,3,4,1]. (Creation a new batch (batch 5)).

(2) For (i, b_i')=(7,3): the neighbor will be [1,1,3,2,2,3,3,1]. (Delete one batch).

(3) For (i, b_i')=(4,1): the neighbor will be [1,1,3,1,2,3,4,1]. (Keep the current number of batches).

From this example mentioned above, we can observe that this neighborhood generation method can efficiently search the good neighbors for a given solution. Moreover, the experiment conducted in Subsection 2.2.4 also proved that the efficiency of the neighborhood generation method.

2.2.3.3 Tabu list and tabu tenure

In the TS approach, the tabu list (denoted by *TabuList*) keeps the most recent moves in order to avoid local optima. In our implementation, the tabu list is formed with the k most recently moves which are dependent upon the k pairs of randomly generated numbers (i, b_i'). In the tabu list, the most recent pairs of randomly generated numbers are kept so that the moves determined by these pairs of numbers are not conducted again. For example, when the current solution is [1,1,3,2,2,3,4,1] and the best solution found in the neighborhood is [1,1,5,2,2,3,4,1], then the pair of entries corresponding to order pairs (3,5) will be tabu during the tabu tenure.

2.2.3.4 Aspiration and termination criteria

The aspiration criteria are given as an opportunity to override the tabu status. Because the tabus are sometimes too powerful, they may prohibit good moves which will lead to a better solution. Therefore, it is necessary to override the tabu status when some move in the tabu list may lead to an overall stagnation of the search process. In our implementation, a tabu move is accepted when it results in a solution with an objective value better than that of the current best-known solution.

The TS terminates when at lease one of the following two conditions is met:

(1) When the number of iterations reaches to maximum iterations (denoted by *maxIteration*).

(2) The evaluation has not been changed for a given number of iterations.

2.2.3.5 Correction operator

The function of correction operator is to check the legitimacy (feasibility) of the solutions. In the feasible solution, each batch should include at least one job, i.e., empty batch should not existed. However, according to neighborhood generation method mentioned above, a neighborhood solution with empty batches may be generated. For example, the current solution is [1,1,3,2,2,3,4,1] and a pair of randomly generated numbers is $(3,8)$, thus according to the neighborhood generation method, the newly generated neighbor will be [1,1,8,2,2,3,4,1]. In this newly generated neighbor, batches b_5, b_6 and b_7 are empty, so it is necessary to delete these empty batches in order to avoid the idle time. Based on this consideration, the correction mechanism of the correction operator is designed to ensure the continuity of the batch numbers by deleting the empty batches. Take the newly generated neighbor mentioned above as an example, the correction operator will delete batch b_5, b_6 and b_7 and consequently a legal neighbor [1,1,5,2,2,3,4,1] is obtained. The correction operator can be described as follows (See Algorithm 6):

2.2.4 Experiment and Computational Results

In this section, the computational experiments are carried out to test the performance of the proposed TS approach. The proposed TS is coded in JAVA language and implemented on the computer with 4Gb RAM and 512KB L2 cache. As a comparison, a basic branch and bound (B&B) algorithm is used to exactly solve the model with small-scale random instances, and then a lower bound is proposed to evaluate the efficiency of the TS for large scale problem instances.

The parameters of the TS are summarized as follows:

- Termination condition: reach to a maximum number of 500 iterations (i.e. *maxIteration*=500) or evaluation of the best individual did not change for 50 generations
- Size of the neighborhood $N(s)$: 500
- Tabu tenure k : k=5, if $n \leq 50$; k=10, if $n > 50$.

Algorithm 3 Steps of the proposed correction operator.

```
1: b ← 1, maxBatchIndex ← n ;
2: while b ≤ maxBatchIndex do
3:    if batchSize[b] == 0 then
4:       for j ← 1 to n do
5:          if b_j > b then
6:             b_j=b_j-1 ;
7:          end if
8:       end for
9:       maxBatchIndex=maxBatchIndex-1 ;
10:      for b' ← 1 to maxBatchIndex do
11:         update batchSize[b'] ;
12:      end for
13:   else
14:      b++ ;
15:   end if
16: end while
```

2.2.4.1 Random instances with small sizes

We create 5 random instances with small sizes based on the following parameter settings. The batch processing time on the batch machine is randomly generated from the uniform distribution with range [1, 5]. The volumes of the different jobs are randomly generated from the uniform distribution with range [10, 50]. The setup time and setup cost are [50,100] and [200,500], respectively. The vehicle's capacity is generated in the range [50,100], further, its round trip delivery time and cost are generated from the range [50,100] and [300,500], respectively. The unit holding costs of different jobs in the customer area are generated from the uniform distribution with range [1, 5]. The due date associated with each job is generated by the same way described in the section 2.2.4.2. For each combination, we randomly generated 50 problem instances and take the average value (Avg.Value) and average cpu time (Avg.CpuT) which are defined in the

section 2.2.4.2 for the performance test of the proposed TS.

In the proposed B&B procedure, each job i with $1 \leq i \leq n$, is attempted to be assigned to each possible position in a given partial solution, (There are two kinds of possibilities in which the first one is that job i is attempted to be assigned to each possible batch of a given partial solution; The second one is that job i, as a newly generated batch, is attempted to be added in each possible position of the given partial solution.) Moreover, in order to increase the efficiency of the B&B procedure, we define that the exploration of the current solution is stopped if its partial evaluation is larger than the evaluation of the best solution found so far. We run the

Table 5. *Results of random instances with small sizes.*

Size	B&B			TS	
(n)	Value	CpuT (s)		Value	CpuT (s)
5	2112.78	0.47	Avg.	2112.78	0.41
			Max.	2112.78	0.59
7	2501.63	0.51	Avg.	2501.63	0.44
			Max.	2501.63	0.60
9	3245.91	0.68	Avg.	3246.27	0.56
			Max.	3246.85	0.76
11	5617.15	25.77	Avg.	5618.62	0.81
			Max.	5619.11	1.45
15	6322.17	10541.87	Avg.	6323.64	4.41
			Max.	6325.90	4.80

B&B solver and the TS using the 5 instances and the results are shown in Table 5. We observe that our TS runs much faster than the B&B solver. Although the B&B solver finds the optimal solution, the computational time of B&B grows exponentially as the instance size increases. The computational time of the proposed TS is very short. Moreover, the TS can obtain optimal or near optimal solutions for all of the situations.

2.2.4.2 Random instances with large sizes

To test the performance of the TS thoroughly, we conduct experiments using random instances with large problem sizes. We consider three scenarios where the capacity of the transporter were generated from a discrete uniform distribution in the interval [100,120], [120, 150] and [150, 200], respectively. For each scenario, we considered three cases with small, middle and large job volumes, which were randomly generated from the uniform distributions in the intervals [30, 50], [30, 80] and [30, 100], respectively. In each case, we set the number of jobs as 30, 50, 70, 100 and 150. The job processing time, setup time and setup cost were randomly generated from a discrete uniform distribution in the interval [1, 5], [10, 40] and [200, 400], respectively. The round-trip delivery time and cost of the transporter were generated from the uniform distribution with range [50, 150] and [300, 600], respectively; Unit customer holding cost were randomly generated from the uniform distribution with range [0.001, 0.005]. The due date associated with each job j was generated from the uniform distribution with range [10000, 12000].

Considering the different transporter capacity values, number of jobs, and delivery costs, we tested 45 situations of the problem. For each situation, we randomly generated 50 problem instances for the performance test of the heuristic algorithm. The error ratio (ER), average error ratio ($Avg.ER$) and average running time ($Avg.CpuT$) have the same definition as described in 2.1.5.2. The computational results are displayed in Tables 6-8. As seen in each table from Tables 6-8, the average error ratios appear in an increasing trend as the value of n increases. One of its reasons may be that the lower bound increases as n increases, but the growth rate is a little smaller than that of the objective value of the TS approach, hence the difference between the objective value generated by the heuristic and the lower bound may grow with the increase of n. They also indicate that the average ratios appear in a decreasing trend in a general point of view as the variation range of the jobs' volumes increases. It may be interpreted that when the job volumes are generated from the large variation range, the number of jobs that the transporter can held decreases which leads a better lower bound, consequently, the deviation decreases. Tables 6, 7 and 8 show clearly that the average error ratios of TS of all the situations were no more

Table 6. *Results of random instances with large sizes for* $c \in [100, 120]$.

Size		$v_i \in [30, 50]$		$v_i \in [30, 80]$		$v_i \in [30, 100]$	
n		ER (%)	CpuT (s)	ER (%)	CpuT (s)	ER (%)	CpuT (s)
30	Avg.	9.87	1.54	2.87	2.04	2.24	1.98
	Max.	18.43	2.89	7.32	3.34	6.03	3.24
50	Avg.	10.10	2.53	3.76	2.20	1.56	3.63
	Max.	14.17	3.08	8.95	3.92	3.69	9.40
70	Avg.	10.53	4.51	5.08	4.88	3.24	4.82
	Max.	15.99	5.07	9.84	6.36	7.77	6.28
100	Avg.	13.21	7.51	7.09	8.07	3.85	7.98
	Max.	18.03	8.29	13.32	9.82	7.01	9.43
150	Avg.	14.81	13.01	9.94	15.17	5.88	13.51
	Max.	18.12	14.33	14.90	17.27	9.11	15.86

Table 7. *Results of random instances with large sizes for* $c \in [120, 150]$.

Size		$v_i \in [30, 50]$		$v_i \in [30, 80]$		$v_i \in [30, 100]$	
n		ER (%)	CpuT (s)	ER (%)	CpuT (s)	ER (%)	CpuT (s)
30	Avg.	6.41	1.22	6.12	6.92	3.39	1.55
	Max.	12.62	1.75	9.22	15.65	7.81	2.23
50	Avg.	7.67	2.13	7.38	2.34	6.67	2.57
	Max.	14.51	2.40	10.77	2.85	12.21	3.35
70	Avg.	7.32	3.32	7.35	3.63	6.47	4.12
	Max.	10.90	3.83	11.28	4.27	10.67	4.95
100	Avg.	8.70	5.74	8.62	6.24	8.11	7.02
	Max.	11.39	6.28	11.26	7.13	12.08	7.89
150	Avg.	8.75	10.57	10.43	11.33	9.92	12.39
	Max.	11.42	11.55	13.18	12.51	15.17	15.58

than 11% with rare exceptions in the situation where job volumes are generated from the range [30,50] and the capacity are generated from [100,120], which demonstrates that the proposed TS is capable of generating near-optimal solutions within a reasonable amount of CPU time.

Table 8. *Results of random instances with large sizes for* $c \in [150, 200]$.

Size		$v_i \in [30, 50]$		$v_i \in [30, 80]$		$v_i \in [30, 100]$	
n		ER (%)	CpuT (s)	ER (%)	CpuT (s)	ER (%)	CpuT (s)
30	Avg.	6.43	1.19	2.91	1.24	6.39	1.25
	Max.	16.75	2.07	11.18	2.12	10.26	2.05
50	Avg.	5.90	1.95	6.58	1.95	6.11	2.22
	Max.	10.23	2.32	12.00	2.48	11.37	2.62
70	Avg.	6.48	3.28	6.34	3.29	7.37	3.75
	Max.	10.87	3.74	10.05	3.77	13.69	4.18
100	Avg.	6.50	5.98	6.64	5.58	7.38	5.99
	Max.	12.44	6.39	9.77	6.30	12.48	7.03
150	Avg.	7.32	10.61	8.11	10.60	8.90	11.17
	Max.	11.27	11.32	10.25	11.05	10.73	11.93

It deserves to note that this lower bound is obtained by two steps of approximation, so the deviation ratio will be accumulated after the two approximation steps as mentioned in subsection 2.4 of section 2. Therefore, it is worth considering this point when analyzing the performance of TS.

2.2.5 Summary

In this section, we have extended the integrated model studied in section 2.1 to a model where each job is associated with a distinct unit customer holding cost. We firstly formulated the problem as a non-linear model and proposed a tabu search approach to solve it. Then, we evaluated the performance of the tabu search approach for both small size and large size problems. For small size problems, we compared our proposed tabu search approach with a basic branch and

bound solver, and the computational results show that the proposed tabu search algorithm can find the optimal or near optimal solution in a very short running time. For large size problems, we used the a lower bound presented in section 2.1 as a comparison, and the results indicate the efficiency of the proposed tabu search algorithm in practice.

3. SINGLE-PRODUCT, MULTI-STAGE ISPDI PROBLEM

3.1 Introduction

There are many supply chain environments which involves more than one supply links. As a practical example of the proposed problem, we can consider a scheduling issue existed in the paper industry. At the beginning of a planning horizon, a customer requires a certain amount of colorful paper bags and sends his requirement to a paper bag manufacturer. Each order has a due date constraint specified by the customer. After the manufacturer receives the order sent by the customer, he will need to buy roll papers from a roll paper manufacturer to finish the order sent by the paper bag customer. This example mentioned above involves two supply links in which the first supply link is composed of the paper bag manufacturer which plays a role of supplier and the final customer, the second is composed of the roll paper manufacturer which plays a role of supplier and the paper bag manufacturer which plays now a role of customer. The objective of the practical example may be minimization of the total logistics costs involving setup, inventory and distribution costs. This integrated scheduling issues becomes more and more important with the popularity of globalization and the concept of "Division of Labor". However, few of researchers have addressed this problems. Even though some models such as Pundoor and Chen (2005), Lee (2001), Zhong et al. (2010) and Wang and Wang (2010) consider the integrated scheduling problem for a so called supply chain environment, most of them only study single supply link situation.

The problem addressed in this chapter is an extension of the problem studied by Grunder (2010). We extended the single-supply-link model studied by Grunder (2010) into a multi-stage model. We take a supply chain environment which is composed of multiple supply links as the studied object. In each link of the supply chain, we study a general two-stage scheduling problem, in

which a given set of simultaneous identical jobs are batch processed by one first-stage processor and then, in the second stage, the completed jobs need to be batch delivered directly to a pre-specified customer by a capacitated transporter without intermediate inventory. In specific, each job is associated with a distinct due date specified by the customer. Moreover, it is supposed that a job which is finished before its delivery date or delivered to the customer before its due date will incur an earliness penalty which is equivalent to the corresponding inventory cost.

This chapter is organized as follows: In Section 3.2, we formulate the problem in a general way and show that it is NP-hard in the maximum capacity of the transporter. In Section 3.3, we establish a dominance relationship for the general model and propose a heuristic procedure. Then we introduce a genetic algorithm in Section 3.4 for solving the problem. In Section 3.5, we show some computational results and make some discussions. Finally, in Section 3.6 we conclude the chapter.

3.2 Problem Formulation

The problem is described as follows. The studied supply chain is a modular system composed of $m+1$ facilities organized in a number of m connected supply links (see Fig. 8). The supply

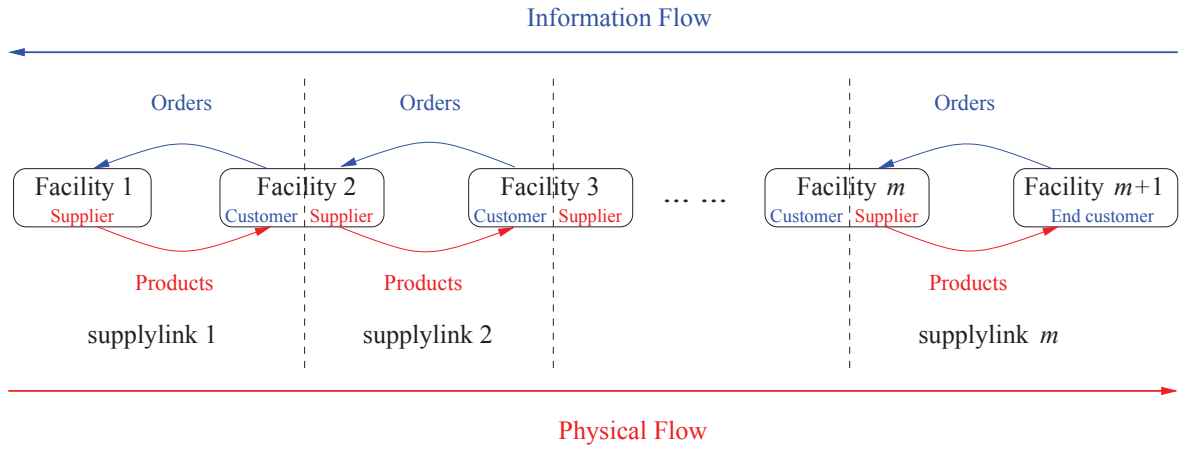

Fig. 8. *Studied system: linear supply chain composed of $m+1$ facilities*

link s is an elementary supply chain composed of a supplier (facility h), a transporter and a customer (facility $h+1$). Each facility h ($1 < h \leq m$) receives parts from the previous facility

$h-1$, processes them and delivers them to the next facility $h+1$. In the system, the first facility is a supplier of material for the whole supply chain, and the last facility is the end customer. Take into consideration again the practical example mentioned above, in the supply chain with 3 facilities (customer, paper bag manufacturer and roll paper manufacturer), the last facility is a customer who requires a certain amount of colorful paper bags before a certain date and sends his requirement to the second facility who is a paper bag manufacturer. After the manufacturer receives the order sent by the customer, he will need to buy roll papers from a roll paper manufacturer (first facility) to finish the order sent by the paper bag customer.

We assume that n identical jobs are requested by the end customer (facility $m+1$) and each job is associated with a due date. All jobs processed in one supply link share the same processing time. Initially, jobs need to be batch processed by facility h, and then delivered to facility $h+1$ by a capacitated transporter. Here, we define a set of jobs which are produced and delivered in one trip as one batch. Moreover, we assume that a job becomes available for production or delivery only when the batch to which it belongs has been completely finished (*Batch availability*). Without loss of generality, we assume that the jobs in a given supply link are numbered according to earliest due date first rule, i.e., jobs are arranged according to the increasing order of their due date values. Each batch has to be processed by the facility h before being delivered to the facility $h+1$ by the transporter. Each trip from the facility h (manufacturer) to $h+1$ (customer) requires a delivery time which depends on the number of jobs in this delivery batch. Here, we define the delivery time from the manufacturer to customer includes the loading and unloading time of the jobs. Each trip from the facility $h+1$ to h requires a delivery time which is a constant for one supply link but differs for different supply links. We assume that each job in the supply link s has to be delivered to customer before its due date which is taken as the production starting date of this job in the next supply link $s+1$. Furthermore, the due dates of the jobs in the final supply link m correspond to the due dates of the order sent by the end customer, and are given parameters of the problem.

Some symbols used in this chapter are listed as follows:

- n: number of jobs

- m: number of supply links
- j: index of job, $1 \leq j \leq n$
- h: index of facility, $1 \leq h \leq m+1$
- s: index of supply link, $1 \leq s \leq m$
- $d_{s,j}$: due date of job j in supply link s
- $C_{s,j}$: completion time of job j in supply link s
- σ_s: a feasible schedule of n jobs for supply link s
- $Q_{\sigma_s}(=|\sigma_s|)$: number of batches in schedule σ_s
- p_s: unit processing time of job in supply link s
- q: index of batch, $q = 1,2,\ldots,Q_{\sigma_s}$
- $\sigma_{s,q}$: batch size of the batch q in supply link s
- c_s: capacity of the transporter in supply link s
- $\tau_{s,b}$: delivery time of a batch (including loading and unloading time) with batch size b from the manufacturer to customer in supply link s
- $\tau_{s,0}$: delivery time from the customer to manufacturer in supply link s
- $\lfloor q \rfloor_s$ $(=\sum_{k=1}^{q-1}\sigma_{s,k}+1)$, $\lceil q \rceil_s$ $(=\sum_{k=1}^{q}\sigma_{s,k})$: the first and last job of the q^{th} batch, respectively

Moreover, we assume that the transporter can deliver only specific quantities of products, which is the case when the products are packed in groups or handled by pallets. We note $\mathscr{C}_s = \{c_{s,1}, c_{s,2}, ..., c_{s,\mu_s}\}$ as the set of the μ_s possible quantities of parts that can be delivered by the transporter for the supply link s, with $1 \leq c_{s,1} < c_{s,2} < ... < c_{s,\mu_s}$ and $\mu_s = |\mathscr{C}_s|$. Consequently, the total number of delivered parts could exceed the number of needed parts. If this is the case, we consider that the surplus is lost.

We define a *partial solution* as a sequence of batches in which the number of delivered jobs is less than n. For a given supply link s, a solution $(\sigma_{s,q})_{q=1..Q_{\sigma_s}}$ is extended from a partial solution $(\sigma'_{s,q})_{q=1..Q_{\sigma'_s}}$, if the number of delivered jobs for σ_s is n, and the last batches of σ_s are identical to those of σ'_s. That is to say, for each $q \in [1..Q_{\sigma'_s}]$, $\sigma_{s,q+Q_{\sigma_s}-Q_{\sigma'_s}} = \sigma'_{s,q}$. For example, consider a problem of 18 jobs with 4 supply links (see Fig. 10 presented in Subsection 3.4.1). Here, any feasible sequence of batches that includes a number less than 18 jobs can be treated as a partial solution, e.g., $\sigma'_s = ((8,5),(3,6,4),(11,2),(6,1,3,3))$, and any feasible sequence of batches that starts from partial solution σ'_s and includes σ'_s will be an extended solution from σ'_s, e.g., $\sigma'_s = ((8,5,5),(3,6,4,5),(11,2,5),(6,1,3,3,5))$. Moreover, the best solution extended from σ'_s is denoted by $\hat{\sigma}'_s$ and verifies the following expression: $f(\hat{\sigma}'_s) = min\{f(\sigma_s)/\sigma_s \; extends \; \sigma'_s\}$

The costs factors considered in this study includes the supplier holding costs, transportation costs and customer holding costs. Since the production costs and setup costs are constant for the n identical jobs for each supply link, so they will not be considered here.

We assume that the sequence of transport lots is same as that of production batches which can be justified in a just-in-time context. Moreover, the holding costs are generally non decreasing along the supply chain from the original supplier to the final customer. Therefore, the production of a batch has to be finished in order to be entirely loaded and delivered to the customer, mainly because the jobs are identical and no setup cost is required for the production batches. Consequently, the different dates of production and transportation are computed backwards from the last supply link to the first one. For a given supply link, the dates of batches are determined starting from the last delivered batch by applying equations in the reverse order to obtain the dates of the previous batches. This strategy enables to obtain the latest dates of the batches for the whole system, which induces the minimal cost for a given sequence of batches in the case of a single supply link as proved in Elmahi et al. (2006).

The batch availability assumption implies that a job has to wait until the remaining jobs in its production batch are completely finished. This will, subsequently, incur an inventory cost named WIP holding cost in this time interval. As the transportation process of a batch starts at the end of the production process of this batch, we consider that the corresponding WIP holding time as well as WIP holding cost depends only on the size of this batch. We assume,

consequently, that the supplier's cost for the q^{th} batch of supply link s is given by the following expression : $\gamma_s(\sigma_{s,q})$. Similarly, the cost expression for the delivery of the q^{th} batch of supply link s is given by: $\eta_s(\sigma_{s,q})$. For ease of explanation, we define $\alpha_s(b)$ $(=\gamma_s(b)+\eta_s(b))$ as the total supplier and transporter cost for a batch size b. We suppose that α_s is a non-decreasing function of the batch size b. Consequently, the total supplier's holding cost and transporter's cost of the s^{th} supply link can be expressed as: $\sum_{q=1}^{Q_{\sigma_s}} \alpha_s(\sigma_{s,q})$.

Moreover, for a given supply link s, it is supposed that a job j which arrived to the customer before its due date will incur an earliness penalty (corresponding to an equivalent inventory cost) given by: $\beta_{sj}(C_{s,j})$. It is assumed that $\forall i \leq j, (\beta_{si} - \beta_{sj})$ is a non-decreasing function, see Baptiste (2000). Consequently, that two consecutive jobs j and $j+1$ are either in the same batch or in two consecutive batches q and $q+1$.

So now, our problem is to define the joint production and delivery schedule for each supply link in order to minimize the sum of inventory and delivery cost of the whole supply chain.

With the assumption mentioned above, in order to obtain an optimal solution, the main tasks are obviously to determine the number of batches and batch size in each supply link s. Therefore, we define the decision variables in supply link s as the number of batches Q_{σ_s} and its size $\sigma_{s,q}$. Since the order is sent by the customer in the last supply link, therefore, the due dates of jobs in the last supply link, $d_{m,j}$, are treated as parameters of the problem.

Consequently, the formulation of the problem is to find a solution $\sigma = (Q_{\sigma_s}, \sigma_{s,q})_{s=1..m}$ that minimizes the function:

$$min\ f(\sigma) = \sum_{s=1}^{m} \left\{ \sum_{q=1}^{Q_{\sigma_s}} \alpha_s(\sigma_{s,q}) + \sum_{j=1}^{n} \beta_{s,j}(C_{s,j}) \right\} \tag{13}$$

Subject to:

$$\sum_{q=1}^{Q_{\sigma_s}} \sigma_{s,q} \geq n \qquad s = 1...m \tag{14a}$$

$$C_{s,j} \leq d_{s,j} \qquad s = 1...m, j = 1...n \tag{14b}$$

$$C_{s,\lceil q+1 \rceil_s} \geq C_{s,\lceil q \rceil_s} + \tau_{s,0} + \tau_{s,\sigma_{s,q+1}} \qquad s = 1...m, q = 1...Q_{\sigma_s} - 1 \tag{14c}$$

$$d_{s-1,j} = C_{s,j} - (\tau_{s,\sigma_{s,q}} + \sigma_{s,q}.p_s) \qquad s = 2...m, j = 1...n, \lfloor q \rfloor_s \leq j \leq \lceil q \rceil_s \tag{14d}$$

$$\sigma_{s,q} \in \mathscr{C}_s \qquad s = 1...m, q = 1...Q_{\sigma_s} \qquad (14e)$$

For ease of reference, we note this problem as the Single-Product, Multi-Stage ISPDI problem (SM-ISPDI). The constraints (14a) state that the number of delivered jobs should be at least equal to n. The constraints (14b) state that the job j should be completed before its due date. The constraints (14c) state that the interval between two consecutive delivery batches should at least equal to one round trip delivery time of the transporter. The constraints (14d) state that the due dates of a supply link are the production starting dates of its next supply link. Finally, the constraints (14e) are the variables constraints.

The considered problem is complex as it involves scheduling considerations from the customer's point of view and batching decisions from the aspects of the supplier and the transporter. Moreover the different supply links are connected to each other, which increases the overall complexity of the problem.

When the customer's holding cost is zero, the dimension of the problem is reduced to the number of different capacities μ_s that a transporter can deliver. In this case, the SM-ISPDI problem is equivalent to a knapsack problem and is NP-hard. This is the main idea for proving the following proposition.

Theorem 4. *The SM-ISPDI problem is NP-hard in* $\max_{s=1}^{m} |\mathscr{C}_s|$.

Proof. We prove SM-ISPDI problem is NP-hard by showing that it is a reduction of m KNAPSACK problems which are NP-hard, see Karp (1972).

KNAPSACK: Given an instance of the knapsack problem of size κ: $(a_i)_{i=1..\kappa}$, $(b_i)_{i=1..\kappa}$, and 2 numbers A and B, does there exist $(y_i)_{i=1..\kappa}$ such that $\sum_{i=1}^{\kappa} a_i.y_i \geq A$ and $\sum_{i=1}^{\kappa} b_i.y_i \leq B$?

We assume that the customer cost function $\beta_{s,j}$ is zero, Then rewrite the initial formulation of the problem by introducing the new variables $x_{s,k}$, for $s = 1..m$ and $k = 1..\mu_s$, which represents the number of batches that hold exactly $c_{s,k}$ parts, i.e., $x_{s,k} = |\{q \in \{1..Q_{\sigma_s}\}/\sigma_{s,q} = c_{s,k}\}|$. Then, the formulation of the SM-ISPDI problem is reduced to the following problem (P'):

$$\min \sum_{s=1}^{m} \left\{ \sum_{k=1}^{\mu_s} \alpha_s(c_{s,q}).x_{s,k} \right\} \qquad (15)$$

subject to:

$$\forall s = 1...m, \sum_{k=1}^{\mu_s} c_{s,k}.x_{s,k} \geq n \tag{16}$$

As the variables $x_{s,k}$ are independent for different values of s (constraints 16), so the problem (P') is equivalent to the sum of the m following problems $(P''_s)_{s=1..m}$: *min* $\sum_{k=1}^{\mu_s} \alpha_s(c_{s,q}).x_{s,k}$, under the constraint: $\sum_{k=1}^{\mu_s} c_{s,k}.x_{s,k} \geq n$. The recognition version of (P''_s) can be formulated as follows: "Does there exist a solution $(x_{s,k})_{k=1..\mu_s}$, under the constraint: $\sum_{k=1}^{\mu_s} \alpha_s(c_{s,k})x_{s,k} \leq B$".

This is the formulation of a knapsack problem by applying the following changes: $\kappa = \mu_s$, $A = n$, $a_k = c_{s,k}$ and $b_k = \alpha_s(c_{s,k})$. The worst case corresponds to: $\kappa = \max_{s=1}^{m} |\mathscr{C}_s|$. This completes the proof. ■

Proposition 4 shows that the capacity of the transporter has an important impact on the complexity of the problem. Moreover, this proposition is proved without taking into account the customers' holding costs. However these parameters increase the complexity of the problem as they make the connection between the different supply links. We conjecture consequently that the SM-ISPDI problem should also be NP-Hard in the number of jobs n, but this problem is still open.

3.3 Dominance Relation and Related Algorithm

Note that multiple supply links arise in the proposed problem and each supply link includes a sub scheduling problem. As only the due dates in the final supply link are given parameters of the SM-ISPDI problem, we point out that the due dates of the previous supply links are variables and dependent on the schedule of the following supply links. It is not easy to simultaneously handle all the supply links. Therefore, we will consider the modularity property of the supply chain to propose efficient algorithms starting from the final supply link, backwards to the first one.

We will first prove that there exists a dominance relation for the case $m = 1$. This property will then be used to propose two solution methods. The first one is an exact algorithm based on a generalized dynamic programming scheme, which can solve small to large instances of the

problem for a single supply link. The second one is a heuristic approach that will be able to produce good solutions for the general case ($m > 1$).

3.3.1 Dominance relation for $m = 1$

In this subsection we propose a dominance relation to prune the search space. In order to illustrate the dominance precisely, we consider a SM-ISPDI problem instance (P_1) with $m = 1$. Two partial solutions σ and σ' are given, in which the number of batches are Q_σ and $Q_{\sigma'}$ respectively for supply link 1. The partial solution σ dominates σ' if the best solution extended from σ is better than the best one extended from σ' i.e., $f(\hat{\sigma}) \leqslant f(\hat{\sigma'})$. This relation will be noted: $\sigma \preccurlyeq \sigma'$.

Theorem 5. *Considering a SM-ISPDI problem instance with a single supply link (P_1), two partial solutions σ and σ' of P_1, σ dominates σ' if the following conditions are satisfied:*

$$\sum_{q=1}^{Q_\sigma} \sigma_{1,q} = \sum_{q=1}^{Q_{\sigma'}} \sigma'_{1,q} \tag{17a}$$

$$\sum_{q=1}^{Q_\sigma} \alpha_1(\sigma_{1,q}) \leqslant \sum_{q=1}^{Q_{\sigma'}} \alpha_1(\sigma'_{1,q}) \tag{17b}$$

$$\sum_{j=1}^{n} \beta_{1,j}(C_{1,j}) \leqslant \sum_{j=1}^{n} \beta_{1,j}(C'_{1,j}) \tag{17c}$$

$$C_{1,1} - \tau_{1,\sigma_{1,1}} \geqslant C'_{1,1} - \tau_{1,\sigma'_{1,1}} \tag{17d}$$

Proof. Proof Suppose to the contrary that $f(\hat{\sigma}) > f(\hat{\sigma'})$. Now consider the solution σ^* extended from σ and beginning with the first $q_0 = Q_{\hat{\sigma'}} - Q_{\sigma'}$ batches of $\hat{\sigma'}$:

$$\begin{cases} \sigma^*_{1,q} & = \hat{\sigma'}_{1,q} & 1 \leqslant q \leqslant q_0 \\ \sigma^*_{1,q} & = \sigma_{1,q-q_0} & q_0 < q \leqslant q_0 + Q_\sigma \\ q_0 & = Q_{\hat{\sigma'}} - Q_{\sigma'} \end{cases} \tag{18}$$

From equations (17b) and (18), we may write that:

$$\sum_{q=1}^{Q_{\sigma^*}} \alpha(\sigma^*_q) \leq \sum_{q=1}^{Q_{\hat{\sigma'}}} \alpha(\hat{\sigma'}_q) \tag{19}$$

From the customer point of view, we will focus on the new jobs j ($j = 1, ..., j_0$) that have been added to the partial solution, with: $j_0 = \lceil q_0 \rceil_s = n - \sum_{q=1}^{Q_\sigma} \sigma_{1,q}$. In a just in time context, the constraints (14b) and (14c) give the following expression for the completion time of the last job added j_0 : $C^{\sigma^*}_{1,j_0} = min(C^{\sigma^*}_{1,j_0+1} - \tau_{1,\sigma^*_{1,q_0+1}} - \tau_{1,0}, d_{1,j_0})$,

As $C^{\sigma^*}_{1,j_0+1} = C^{\sigma}_{1,1}$ and $\sigma^*_{1,q_0+1} = \sigma_{1,1}$ we obtain: $C^{\sigma^*}_{1,j_0} = min(C^{\sigma}_{1,1} - \tau_{1,\sigma_{1,1}} - \tau_{1,0}, d_{1,j_0})$.

In the same way, $C^{\hat{\sigma}'}_{1,j_0} = min(C^{\sigma'}_{1,1} - \tau_{1,\sigma'_{1,1}} - \tau_{1,0}, d_{1,j_0})$.

From these two expressions, we conclude that $C^{\sigma^*}_{1,j_0} \geq C^{\hat{\sigma}'}_{1,j_0}$. As the jobs before j_0 share the same batches for both sequences σ^* and $\hat{\sigma}'$, the completion time of the corresponding jobs of solution σ^* are greater than those of solution $\hat{\sigma}'$, thus: $\forall j \in [1, .., j_0], C^{\sigma^*}_{1,j} \geq C^{\hat{\sigma}'}_{1,j}$. The storage cost of the customer is a non-increasing function of the completion time, thus we obtain:

$$\sum_{j=1}^{j_0} \beta_{s,j}(C^{\sigma^*}_{s,j}) \leqslant \sum_{j=1}^{j_0} \beta_{s,j}(C^{\hat{\sigma}'}_{s,j}) \tag{20}$$

The two expressions (19) and (20) lead to: $f(\sigma^*) \leq f(\hat{\sigma}')$. Consequently, there exists a complete solution σ^* extended from the partial solution σ which is better than $\hat{\sigma}'$, which contradicts our assumption. ■

In this theorem, we establish a dominance relationship for single supply link situation between two partial solutions with the same number of delivered jobs. Based on this property, we first propose a dynamic programming based solution to optimally solve the SM-ISPDI problem with a single supply link. We then propose a general solution method for the SM-ISPDI problem based on this property.

3.3.2 Generalized dynamic programming approach for $m = 1$

As mentioned above, in order to build the optimal solution for a supply link subproblem of SM-ISPDI problem, we propose a generalized dynamic programming scheme, denoted by "Latest Loading Dates method" (LLD). The main idea of the LLD approach (see Algorithm 4) is described as follows. Start from a single delivered part and end with all the delivered parts. Each level k (including a number of k jobs) will be composed of the dominant partial solutions which

deliver a number of k parts. The dominated solutions will not be retained as they will lead to worse evaluations than the dominant ones. The process to build all the dominant solutions for level k consists of considering the retained solutions of all previous levels from level 1 to level $k-1$. For each retained solution of level k' less than k, a new solution of level k is built by simply adding a batch of $k-k'$ parts, if this is possible. With this procedure, the number of possible solutions for level k will grow very quickly, as this is the sum of all the dominant solutions of the previous levels. To reduce the size of this set of solutions, the dominance criteria is applied among the solutions of the given level. After reaching level n, the solution with the minimal evaluation is the optimal solution of the SM-ISPDI problem with a single supply link.

3.3.3 Dominance-related greedy (DRG) in the general case

For the general case ($m > 1$), the idea is to use the modular structure of the original system in order to apply successively the LLD algorithm (Algorithm 4) from the last supply link (m) backward to the first one. The algorithm (Algorithm 5) starts from the optimal solution of the last supply link associated with the final due dates specified by the final customer. With this solution, the starting production dates of the jobs will be chosen as the due dates of jobs in the previous supply link. Then apply the LLD algorithm to solve the optimization problem of the supply link $m-1$. Repeat the procedure mentioned above until the first supply link is solved. The final solution for the whole supply chain is thus the sum of the m solutions obtained from the supply link problems. This algorithm will be denoted by "dominance related greedy" algorithm (DRG).

DRG is straightforward and shortsighted in its approach in the sense that it always find the optimal solution for the current supply link on the basis of information at hand without worrying about the effect that these decisions may have in the future. Therefore, this mechanism can not guarantee an optimal solution. Genetic algorithms are generally the best and most robust kind of evolutionary algorithms. Therefore, in the next section, we propose a genetic algorithm approach for solving the problem. The solution generated by the genetic algorithm will be an interesting comparison reference for evaluation of the solution qualities provided by DRG.

Algorithm 4 Latest Loading Date Algorithm for $m = 1$.

Initialization: $SetOfDominanceSolutions[0] \Leftarrow \{$solution with 0 delivered jobs$\}$;
for $k = 1$ to n **do**
  $SetOfDominanceSolutions[k] \Leftarrow 0$;
  for $b \in \mathscr{C}$ and $b \leq k$ **do**
    for each $sol \in SetOfDominanceSolutions[k]$ **do**
      insert batch b at beginning of sol ;
      $dominant \Leftarrow true$;
      for each $soldom$ in $SetOfDominanceSolutions[0]$ **do**
        if $soldom \preceq sol$ **then**
          $dominant \Leftarrow false$;
        else
          if $sol \preceq soldom$ **then**
            remove $soldom$ from $SetOfDominanceSolutions[k]$;
          end if
        end if
      end for
      if $dominant$ **then**
        add sol to $SetOfDominanceSolutions[k]$;
      end if
    end for
  end for
end for
$bestSolution \Leftarrow$ getBestSolution ($SetOfDominanceSolutions[n]$) ;

3.4 Genetic Algorithm

As a global optimization method, genetic algorithm (GA) has been used successfully to find optimal or near-optimal solutions for a wide variety of optimization problems since its introduction by Holland (1975). GA starts with an initial set of solutions, called population. Each

Algorithm 5 Dominance Related Algorithm in General Case.

1: Initialization: due dates of SL $m \Leftarrow$ final due dates;
2: **for** $s = m$ to 1 **do**
3: solve the supply link problem s with LLD;
4: **if** $s > 1$ **then**
5: due dates of SL $s-1 \Leftarrow$ production starting dates of SL s ;
6: **end if**
7: **end for**

solution in the population is called a chromosome (or individual), which represents a point in the search space. The chromosomes are evolved through successive iterations, called generations, by genetic operators (selection, crossover and mutation) that mimic the principles assigned to each individual according to a problem-specific objective function. Generation by generation, the new individuals, called offspring, are created and survive with chromosomes in the current population, called parents, to form a new population. Since in the genetic operation process, the offsprings are randomly generated, it does not always produce the feasible solutions. Hence, a repair operator has to be carried out after a chromosome is generated in order to keep its legitimacy. The repair operator will check each gene of the chromosome, if the gene's value of some chromosome is larger than the transporter capacity, this gene will be divided into two new genes. Then take the new generated gene as the new starting point, repeat the check process, until each gene's value of the chromosome is smaller or equal than the transporter capacity. We take the reciprocal of the total cost, i.e. $1/f(\sigma)$, as the fitness function to evaluate each individual. Moreover, the algorithm is stopped when the number of iterations meets the maximum generations or the fitness value of the best individual in a population is unchangeable for a certain number of iterations.

The flow diagram of GA steps is shown in Fig. 9. We denote *pop_size* as the population size, and *cross_size* and *mut_size* as the number of chromosomes selected to undergo the crossover and mutation operation, respectively. The algorithm selects $pop\_size/2$ couples of individuals by "roulette wheel of reserving elites method", see Gen and Cheng (1997). For each couple of individuals, a crossover operation is applied with a probability p_c to produce the

offsprings, otherwise both parents are duplicated to produce the offsprings. Then we perform a mutation operation on the offsprings with a probability of p_m. The offsprings are then directly added to the new generation. The number of individuals that undergo a crossover or mutation operation is given by the following expressions: $cross\_size = round(pop\_size \times p_c)$, $mut\_size = round(pop\_size \times p_m)$.

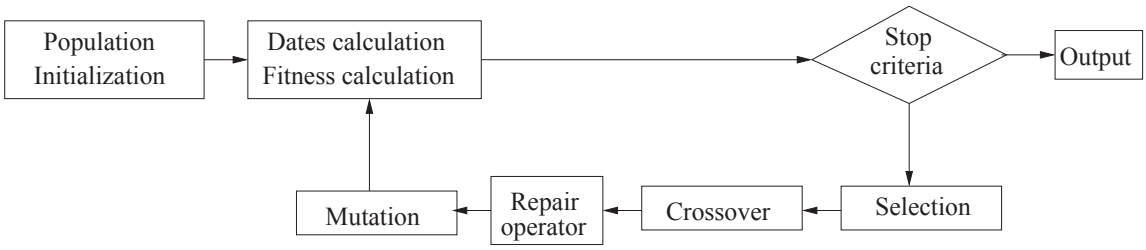

Fig. 9. *The flow diagram of GA*

The following parts are the descriptions and specifics of the main elements of GA:

3.4.1 Chromosome representation and initial population

For chromosome representation, there are a variety of encoding methods, and the most commonly used ones are binary coding and real number coding methods. Based on the characteristics of the scheduling problems under study, this work adopted the real number coding method. In one chromosome, the value of each gene represents a batch size, i.e., number of jobs. A solution of the whole problem is composed of the solutions of the m supply link problems. The solution of the supply link s is composed of a sequence of Q_s genes. A feasible chromosome of a problem with four supply links and 18 jobs is shown in Fig. 10. In this chromosome, the

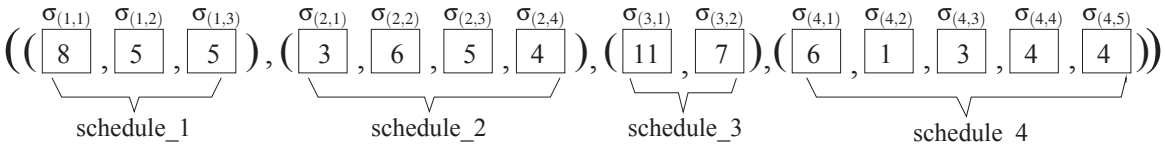

Fig. 10. *An example of the chromosome structure: 4 supply links with 18 jobs.*

solution of the whole problem is ((8,5,5),(3,6,5,4),(11,7),(6,1,3,4,4)) which is composed of the following four subsolutions of the supply link problems: (8,5,5),(3,6,5,4),(11,7),(6,1,3,4,4).

In order to improving the quality of the initial population, the following four specific design chromosomes are introduced into the initial population:

(1) The first one is a "full delivery capacitated" solution in which each batch is a fully loaded, except for the first one which is assigned with the remaining jobs ($n - c_s.E(\frac{n}{c_s})$). This solution corresponds to the case where the transporter delivery costs are dominant compared to the holding costs of the supplier and the customer. The number of deliveries should be consequently reduced as much as possible in this case.

(2) In the second one, each gene is assigned with only one job. This solution is interesting in the case where the transporter's deliveries are fast compared to the due dates interval and have low cost compared to the holding costs of the parts.

(3) The third one is obtained by doing a dichotomy process on the "full delivery capacitated" solution. The so called "dichotomy process" starts to divide, if it is possible, the last batch in this chromosome into two parts. If the new chromosome is better than the original one, then keep this division and stay on the current gene, if not, cancel this division and go to the previous gene. By this way, gene by gene from the last one to the first one, a whole chromosome is generated which will be at least as good as the "full delivery capacitated" solution.

(4) The fourth one uses a different building scheme which is related to a greedy approach. The procedure starts with a batch which is assigned with only one job, then compare the two partial solutions built by the following two construction measures in which the first one is that another job is added into this batch and the second one is that a new batch with only one job is added before the first batch, and take the better partial solution as the new starting partial solution. Then start with the newly constructed batch in the new partial solution and repeat the comparison between the two construction measures mentioned above. By this means, job by job, the whole solution will be constructed after the nth job has been considered.

The other chromosomes are generated by random manner with respecting the transporter capacity constraint.

3.4.2 Genetic operators

(1) Crossover operators

In GA, the main purpose of crossover operation is to recombine the features of two randomly selected parents from the population with the aim of producing better offsprings. Regarding to the permutation-based representation, several crossover operators have been proposed in Iyer and Saxena (2004) and Wang and Wu (2004). Among them, the following crossover operators have been widely used: partially matched crossover intending to keep the absolute positions of genes and linear order crossover intending to preserve relative positions. Here, we used these two operators in this study, and they will be chosen randomly.

The first crossover operator aims to swap two sequences of genes in the same supply link of two parents. It works as follows:

Step 1. Select a random integer s in the interval $[1, m]$, then randomly select two sequences of genes, π'_s and π''_s respectively, in the supply link s of the two parents.

Step 2. Produce the first offspring by replacing π'_s of the first parent with π''_s. The second offspring is obtained from the second parent with the reverse operation.(i.e., π''_s is replaced by π'_s).

In this first operator, the number of exchanged positions corresponds to the number of batches for the supply link s.

The second crossover operator aims to swap two sets of consecutive genes for several supply links of two randomly selected parents. It works as follows:

Step 1. Generate a random integer s in the interval $[1, m]$, and select two sequences of genes from supply link 1 to s, π'_s and π''_s respectively, in the two parents.

Step 2. As in the first crossover operator, produce the offsprings by exchanging the sequences π'_s and π''_s in the two parents.

In this crossover operator, the number of exchanged positions corresponds to the sum of the number of batches for the supply links from 1 to s.

(2) Mutation operators

Mutation is used to produce small perturbations on chromosomes to promote diversity of the population. There are several mutation operators such as swapping, inversion, insertion and shift mutation (see Gen and Cheng (1997)).

In this study, we use three mutation operators in which the first mutation operator as shown in Fig. 11 is a swap operator which works to swap two randomly selected batches of one solution of a supply link. The steps of the mutation operator are shown as follows:

Step 1. Select randomly a chromosome and an integer s in the interval $[1, m]$.

Step 2. Choose randomly two batches q and q' in the supply link s, with $q \neq q'$.

Step 3. Produce offspring by exchanging the positions of the two batches, i.e., exchange the value of $\sigma_{s,q}$ and $\sigma_{s,q'}$.

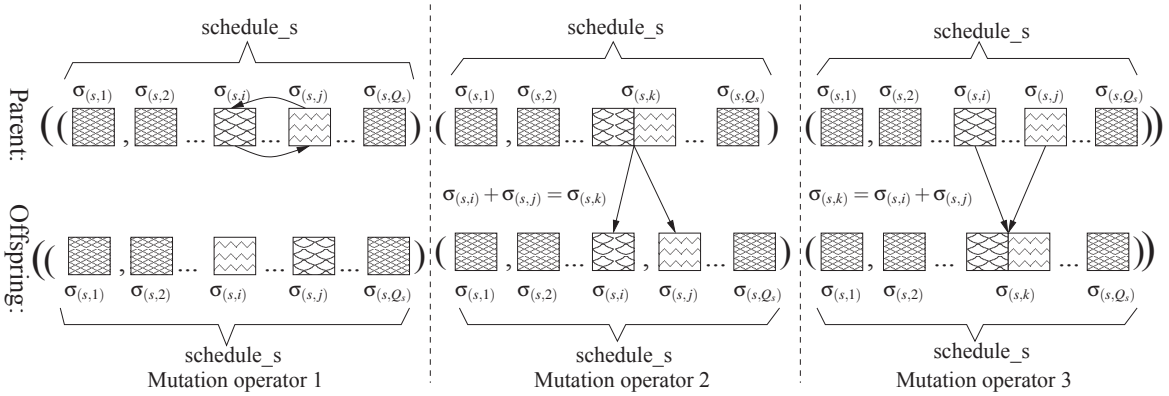

Fig. 11. *Illustration of the mutation operators.*

The second mutation operator as shown in Fig. 11 is a split mutation operator which works to divide a randomly selected batch of one solution of a supply link into two separate batches. The steps of this mutation operator are shown as follows:

Step 1. Select randomly a chromosome and an integer s in the interval $[1,m]$, then choose a batch q for which $\sigma_{s,q} \geqslant 2$,

Step 2. Then create two batches q' and q'' to replace the previous batch q in the following way: $\sigma_{s,q'} + \sigma_{s,q''} = \sigma_{s,q}$. The batch size of batch q' is randomly selected in the interval $[1, \sigma_{s,q}]$, and the batch size of batch q'' is $(\sigma_{s,q} - \sigma_{s,q'})$.

The third mutation operator as shown in Fig. 11 is a fusion operator which works to merge two successive batches of one solution of a supply link into one new batch. The steps of this mutation operator are shown as follows:

Step 1. Select randomly a chromosome and an integer s in the interval $[1,m]$, then choose two consecutive batches q and $q' = q+1$ from supply link s.

Step 2. If $\sigma_{s,q} + \sigma_{s,q+1} \leqslant c$, then create a new batch which includes the batch q and $q+1$ to replace the two previous batches, i.e. $\sigma'_{s,q} = \sigma_{s,q} + \sigma_{s,q+1}$.

3.5 Experiment and Computational Results

In this section, the computational experiments are carried out to test the performance of the two proposed heuristics which are coded in JAVA language and implemented on an Intel(R) Core(TM)2 Quad CPU Q9550 @ 2.83GHz with 3,2G RAM and 12M L2 cache. As a comparison, a simple branch and bound (BBP) approach is employed to exactly solve small scale problem instances, and a lower bound is developed to evaluate the efficiency of the proposed heuristics for large scale problem instances. To increase the efficiency of the BBP procedure, we explore the solution space corresponding to the $m-1$ last supply link problems, and then apply the optimal algorithm LLD (Algorithm 4) for the first supply link problem. The exploration of the current solution is stopped if its partial evaluation is larger than the evaluation of the best solution found so far. In this case, another branch of the solution space is explored.

The two processes for generating both the small size and large size problem instances share some common parameters which are defined as follows: The transporter capacity c_s is chosen

randomly between one third and two thirds of the number of jobs. Furthermore, the round-trip delivery time $\tau_{s,b}$ and delivery cost $\eta_s(b)$ are constants for any batch size b, and randomly generated in the range [200,400] unit of times (*ut*) and [500,1000] monetary unit (*mu*), respectively.

The different parameters used for generating the small size and large size problem instances are stated in more detailed fashion in the following Subsection 3.5.1 and 3.5.2, respectively.

3.5.1 Problem instances with small sizes

We construct 5 classes of experiments with small sizes in order to measure the quality of the solutions found by DRG and GA. As a comparison, a basic Branch and Bound algorithm presented above is used to optimally solve the small problem instances.

compared to the optimal solution which will be built by a branch and bound approach. For these experiments, the number of supply links m is 2 and 3 and the number of jobs n is 5, 15 and 25, respectively. The due dates $(d_{m,j})_{j=1..n}$ are uniformly separated with values generated in the interval [0,50].

For each scenario, the supplier processing time p_s for any job is unitary. We assume that the holding cost $\gamma_s(b)$ of a given batch of size b for the supplier is linear in the corresponding holding times of the parts in this batch. The unit holding cost in the supplier area has been selected in the range [0.01,0.05] *mu*. In the same way, the holding cost $\beta_{sj}(C_{s,j})$ of a job j for the customer is linear in the holding time of job j. We assume that the unit holding cost of the customer is more expensive than that of the supplier with a maximum deviation of 10% in order to truly reflect the real case. Since the holding cost represents a combination of the cost of capital, the cost of physical storage and the cost of losses due to spoilage; hence, it highly depends on the inventory type. Moreover, the value of the production is added according to the supply chain from the original material supplier to the last customer in the supply chain. Therefore, it is logical to set the unit customer holding cost to be a little higher than unit supplier holding cost.

Moreover, we calibrate the population size and the number of iterations for GA as 500. The crossover and mutation probabilities are set to be 0.6 and 0.1, respectively. The values are

appropriate for such a problem to find high quality solutions in a reasonable time. Finally, when the iterations meets the maximum number of generations or the fitness value of the best chromosome is unchangeable for 50 generations, GA is stopped.

The ratio that the proposed heuristic algorithms obtains optimal solution is defined as follows:

$$OptimR = \frac{times\ that\ Heu\ obtains\ optimal\ solution}{times\ of\ experiments\ tested\ for\ a\ situation} \times 100\%$$

where the Heu indicates the proposed heuristic algorithm (DRG or GA). The worst error ratio is defined as $MaxER = (WorstHeu - BBP)/BBP$, where $WorstHeu$ denotes the worst evaluation of the solution generated by DRG or GA.

For each class of experiment, 50 instances were generated and solved using the two heuristics presented above. The computational results are shown in Table 9.

It is observed that GA is optimal for 2-supply-link instances, and for 3-supply-link, 5-part instances. For the DRG procedure, the optimality ratio decreases slowly as the number of parts increases, however the maximum error ratio is always lower than 4%. This results indicate that despite the optimal solution is not always reached, the DRG algorithm generally provides near-optimal solutions. Another important point is that the running time of DRG is very short and less than 100 ms, GA needs less than 1 minute while BBP requires several hours to find the optimal solution even for small size instances.

Table 9. *Results of random instances with small sizes.*

Instance	Size	*DRG*		*GA*	
No.	$(m \times n)$	$OptimR(\%)$	$MaxER(\%)$	$OptimR(\%)$	$MaxER(\%)$
1	2×5	88,00	0,13	100,00	0,00
2	2×15	68,00	0,54	100,00	0,00
3	2×25	61,90	3,96	100,00	0,00
4	3×5	64,00	0,23	100,00	0,00
5	3×15	45,45	1,10	90,91	0,77

3.5.2 Random instances with large sizes

To test the performances of the two proposed heuristic algorithms thoroughly, we conduct experiments using random instances with large sizes. We consider nine scenarios where the number of supply links m is 2, 3 and 4, and the number of jobs n is 100, 200 and 300. For each scenario, the processing time associated with a job is assumed to be unitary, and 100 problem instances were generated for test. For the second part of the experimentation process, we have increased significantly the population size and the number of iterations of GA up to 1000 as the size of the instances are larger. The crossover and mutation probabilities are set again to be 0.6 and 0.1, respectively.

The error ratio is defined as $ER = (Heu - LB)/LB$, where Heu denotes the evaluation of the solution generated by DRG or GA, and $LBdenotes$ represents the lower bound. The average error was defined as $Avg.ER = (\sum ER)/times\ of\ experiments\ tested\ for\ a\ situation.$ The running time of a single experiment is denoted by $cputime$ and the average running time is calculated by: $Avg.CpuT = \sum(cputime)/times\ of\ experiments\ tested\ for\ a\ situation.$

Since it is difficult to obtain an optimal solution in reasonable computing time even for the situation with 3 supply links and 20 jobs, so we evaluated the two proposed heuristics by the following straightforward lower bound. Let the unit customer holding cost in the supply links from 1 to $m-1$ be 0. So the lower bound can be derived as the evaluation of the optimal solution of the mth supply link plus the evaluation of the optimal solutions of all the remaining supply links from 1 to $m-1$. If the customer holding cost is set to be 0 for all supply links except the last one, then all supply links are disconnected and the optimal solution of the whole system is the sum of the local optimum of each supply link. The advantage of this lower bound is that it can be obtained very quickly, however its main drawback is that its quality decreases as the customer holding costs increase. Nevertheless, this lower bound is an interesting reference to measure the relative performances of both solvers DRG and GA.

According to the importance of delivery cost and customer holding cost, We divided our experiments into two series. The first one with low customer holding cost (range [0.001,0.005] *mu*), in this case, the delivery cost is much more important than the customer holding cost, therefore,

the solution found by the heuristic algorithms will have a trend of generating a solution with small number of batches in order to reduce the delivery cost. The second one with higher customer holding cost (range [0.001,0.05] *mu*), in this case, the heuristic algorithms should balance the cost for delivery and inventory in the search process to get a good enough solution. By this two experiments designed above, we can do the test for the performance of the proposed heuristic algorithms. The experimental results for the first configuration with lower holding costs are displayed in Table 10, and for the second configuration with higher holding costs in Table 11.

Table 10. *Large-size problem instances with lower holding costs.*

Instance	Size	*DRG*		*GA*	
No.	$(m \times n)$	$Avg.ER(\%)$	$Avg.CpuT(ms)$	$Avg.ER(\%)$	$Avg.CpuT(ms)$
1	2×100	2.31	389.32	1.62	31'397.14
2	2×200	6.69	1594.04	4.86	36'834.58
3	2×300	9.83	3409.46	6.91	43'072.56
4	3×100	2.73	550.66	2.01	46'013.8
5	3×200	10.67	2239.1	7.23	54'923.92
6	3×300	12.63	5532.98	9.51	62'340.12
7	4×100	3.08	702.26	2.22	63'117.5
8	4×200	10.73	2983.88	6.81	73'807.96
9	4×300	14.40	7514.78	14.03	84'303.56

From the results presented in Tables 10 and 11, we can observe that the average error ratios of DRG and GA are roughly in the same ranges of value for the different experiments. The GA seems however to be more efficient than DRG when the size of the problem instances range from small to medium. From the aspect of CPU processing time, although DRG is much faster than GA, the processing time of DRG seems to have an exponential trend when the number of parts increase, while the processing time of GA is linear as the number of parts increase.

When the holding cost along the supply chain is low, it is observed from the Table 10 that the average deviation ratios of both heuristic algorithms for all the situations were no more than 15%, which indicates that the performances of GA and DRG are good for the randomly

generated problem instances. When the holding cost along the supply chain is higher, it is

Table 11. *Large-size problem with higher holding costs.*

Instance	Size	*DRG*		*GA*	
No.	$(m \times n)$	$Avg.ER(\%)$	$Avg.CpuT(ms)$	$Avg.ER(\%)$	$Avg.CpuT(ms)$
1	2×100	9.19	367.34	6.58	31'261.6
2	2×200	16.24	1166.82	13.76	37'101.04
3	2×300	20.18	3532.52	18.07	43'592.4
4	3×100	14.21	536.7	12.43	45'967.98
5	3×200	23.53	1825.6	22.92	55'200.7
6	3×300	26.50	3924.38	24.81	63'037.96
7	4×100	13.92	689.3	13.03	63'455.4
8	4×200	29.30	2376.06	31.44	74'349.86
9	4×300	34.49	4914.41	39.11	84'857.10

observed from the Table 11 that the average deviation ratios of both heuristic algorithms for all the situations were no more than 40%. This overall increase in the average error ratios can be explained by the worse quality of the proposed lower bound. When the number of parts or the number of supply links increases, the holding costs are becoming higher, and consequently the quality of the proposed lower bound decreases.

On further investigation of Table 11, we notice that when the dimension of the problems increases (in terms of the number of supply links and number of parts), the quality of the solutions provided by DRG becomes better than those found by GA. This observation highlights that problems with different sizes require different solving methods to deal with.

3.6 Summary

In this study we have investigated a multi-stage, lot-sizing and delivery scheduling problem in a supply chain environment with batch-size dependent delivery times and costs. The objective

is to determine the batch sizes and batch sequence to minimize the total joint cost involving the delivery and holding costs. We showed that the problem is NP-hard in the maximum capacity of the transporters in a general case, and presented a dominant relationship between partial solutions with the same number of delivered parts.

Then, we proposed two heuristic algorithms to solve this problem in which the first one is a heuristic procedure DRG based on a dynamic programming algorithm for each supply link, the second one is GA. Experiments have been conducted to compare the performance of the two proposed heuristic algorithms. Results show that they are efficient to solve the considered problem. DRG provides solutions of high quality in a very short time. This may be explained by the characteristics of DRG. DRG is dynamic programming based algorithm, for each given supply link problem, it is a real dynamic programming approach. It starts from solving the last supply link, one by one, optimally solved each supply link up to the first one. Moreover, it is based on an exact algorithm, so it performs efficiently in term of solution quality. However, its performance on solution quality depends much on the nature of the problem. GA solves the problem from an overall point of view and can find the optimal solution for majority of the small size instances, however DRG never found optimal solution for small size problem instances. For large size instances, GA performs better than DRG for most of the tested problem instances.

4. SINGLE-PRODUCT ISPIDI PROBLEM

4.1 Introduction

In some existing supply chains, production and distribution are often indirectly linked by an intermediate stage of finished product inventory, and hence the intermediate inventory is a non-negligible element when the companies tend to integrate production and transportation activities. Since in a supply chain, the abilities of the two main logistical stages, i.e. the rate of production and the speed of delivery, are commonly not matched. In this case, from the whole system point of view, the consideration of the existence of intermediate inventory may efficiently balance their abilities and consequently improve the performance of the entire supply chain. This problem is significant as it addresses the issue of striking a proper balance between the rate of production, the level of inventory and the speed of delivery.

As a practical example of the proposed problem, we can consider a scheduling issue existed commonly in the iron and steel industry. There is an oven that must heat different pieces of work at a given high temperature, then the finished pieces of work should be transported to next plant for painting by a capacitated transporter. In this case keeping the required temperature of the oven while it is empty may clearly be too costly (can be treated as setup cost), therefore a large production batch will result in lower setup cost, however, a large production batch may exceed the capacity of the transporter. Consequently, these pieces of work beyond the transporter capacity will stay at the factory (or intermediate inventory) and generate an intermediate inventory cost.

Many researchers such as Pundoor and Chen (2005), Chen and Lee (2008) and Hall and Potts (2003) have studied the integrated scheduling problems without taking into account the inter-

mediate inventory which works as a buffer to balance the production rate and the speed of transportation. That is to say, most of the integrated models of production scheduling and product distribution implicitly assume that the batch size is limited by the capacity of the vehicle; i.e., after one batch is processed by a machine, it can be entirely delivered by the vehicle to the customer. This assumption will result in worse performance for the production stage when the setup is relatively large and the manufacturing rate is far larger than that of transportation.

In this chapter, we study the second category of problems (ISPIDI problem) where the production and distribution are indirectly linked through an intermediate stage of finished product inventory. In specific, we assumed that the intermediate stage worked as a buffer to balance the production rate and the speed of distribution. Moreover, this intermediate inventory allowed the jobs to be rescheduled for transportation process after completion on the machine. This specific assumption makes our work differ from the others such as Chang and Lee (2004), Li and Ou (2005), Tang and Liu (2009b), Gong and Tang (2011) and Tang and Gong (2008). A schematic diagram of the supply chain is given in Fig. 12.

This chapter is organized as follows. In Section 4.2, we formally describe the problem and introduce some notations, then we show some straightforward optimality properties. In Section 4.4 and 4.5, we study a precise instance of the proposed general model and propose a heuristic based on some optimality properties. In Section 4.6, we develop a lower bound on the optimal solution of the precise model based on the lagrangian relaxation method. At last, in Section 4.7 and Section 4.8, we conclude this chapter.

4.2 Problem Description and Formulation

The problem is described as follows. This paper studies an integrated scheduling problem for a make-to-order supply chain environment where the production and distribution are indirectly linked through an intermediate stage of finished product inventory, see Fig. 12. At the beginning of a planning horizon, the supplier has received an order for processing a set of n identical jobs ($J = \{1,2,\ldots,n\}$) by a single machine which has no capacity limitation. Each job j has a due date (or deadline) d_j specified by the customer and a constant processing time p_t. Here,

"identical jobs" means that all of the jobs share the same attributes. "Jobs have different due dates" means that each job has to be delivered to the customer before its due date. Jobs are first

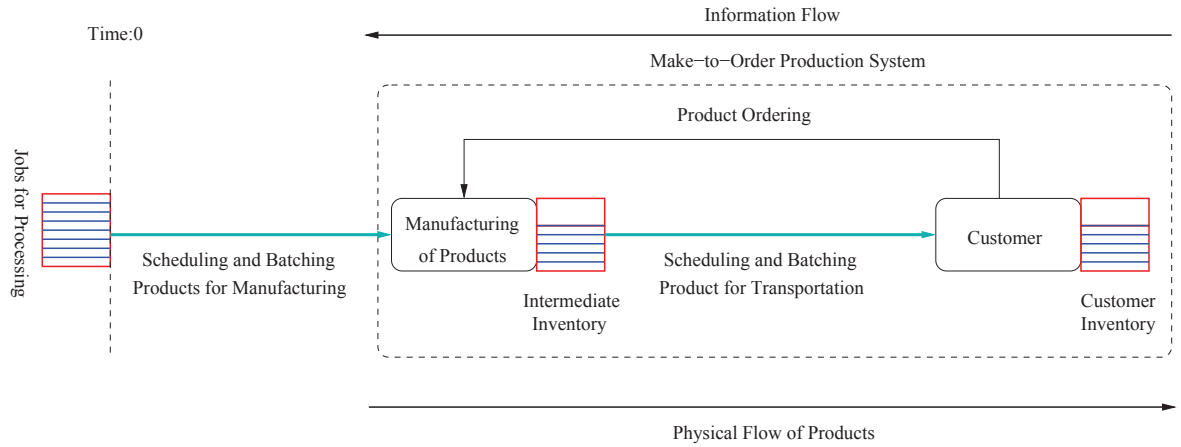

Fig. 12. *Production-Inventory-Distribution-Inventory Problem*

processed on the machine in production stage, and then delivered to a pre-specified customer in delivery stage. It is assumed that each production batch requires a setup cost $\gamma_c(\omega)$ as well as a setup time $\gamma_t(\omega)$ before the first job of this batch is processed, where ω is the number of the loaded jobs in the corresponding production batch, both γ_c and γ_t are non-decreasing functions of ω. The setup cost here as well as the setup time is related to activities, such as heating, cooling and replacing, which are associated with the restarting of the machine after idleness and the opportunity cost represents the loss of utilization of the machine and labor. We define that all the jobs processed consecutively without setup in between constitute a production batch. Further, it is assumed that a job becomes available for delivery only when the production batch to which it belongs is completely finished.

In delivery stage, because of the existence of an intermediate inventory in the factory, all the jobs in a completed production batch are firstly stored in this intermediate inventory waiting for delivery by a vehicle of capacity c to a pre-specified customer. Each job must be delivered to the customer before its deadline. The cost for delivering jobs from the factory to the intermediate inventory is assumed to be 0. Each round trip between the factory and customer requires a delivery cost $\eta_c(\sigma)$ as well as a delivery time $\eta_t(\sigma)$, where σ is the number of the loaded jobs in the delivery trip, both η_t and η_c are non-decreasing functions of σ. Moreover, we suppose that a

job which is finished before its departure date or arrives at the customer before its due date will incur a stage-dependent corresponding inventory cost (WIP inventory cost, finished-inventory cost or customer inventory cost). So now, our problem is to find a coordinated production and delivery solution such that the sum of setup, inventory and delivery cost is minimized.

The jobs are identical, therefore the job sequence has no effect on the objective function. Consequently, in order to obtain an optimal solution, the main tasks are obviously to determine the number of production and delivery batches, respectively, and the number of jobs in each production and delivery batch.

The following notations will be used throughout the paper:

- B_k^p, B_h^d: the kth production batch and hth delivery batch, respectively;
- $k(j)$: the job j which is assigned to the production batch k;
- $C_{k(j)}^p$: the completion time at the factory of job j which is assigned to the production batch k;
- C_k^p: the completion time at the factory of production batch B_k^p;
- $h(j)$: the job j which is assigned to the delivery batch h;
- $C_{h(j)}^d$: the arrival time at the customer of job j which is assigned to the delivery batch h;
- $t_{h(j)}^d$: the departure time from the factory of job j which is assigned to the delivery batch h;
- C_h^d: the arrival time at the customer of delivery batch B_h^d;
- t_h^d: the departure time from the factory of delivery batch B_h^d;
- $\psi = [B_1^p, B_2^p, \ldots, B_u^p]$: a production solution that processes all the jobs on a single machine, where u is the number of production batches in a production solution;
- $\omega_k = |B_k^p|$: the number of jobs in B_k^p for $k = 1, 2, \ldots, u$;

- $\varphi = [B_1^d, B_2^d, \ldots, B_v^d]$: a delivery solution that transports all the jobs from the factory to the customer, where v is the number of delivery batches in a delivery solution;
- $\sigma_h = |B_h^d|$: the number of jobs in B_h^d for $h = 1, 2, \ldots, v$;
- $S_k^p = \sum_{i=1}^{k} \omega_i$: the number of jobs in the production batches from 1 to k for $k = 1, 2, \ldots, u$;
- $S_h^d = \sum_{j=1}^{h} \sigma_j$: the number of jobs in the delivery batches from 1 to h for $h = 1, 2, \ldots, v$;

The inventory cost indicates a combination of the cost of capital, the cost of physical storage and the cost of losses due to spoilage, which suggests that the inventory cost depends much on the inventory type and the stage of a supply chain, for example, the inventory holding cost per unit time of each finished product should in theory be greater than that of any intermediate product. Thus, in this paper, we consider stage-dependent inventory costs which are expected to make the coordination between production and delivery more effective. According to the difference of the time interval wherein the inventory cost incurs, the inventory cost incurred in the production stage can be divided into two parts: WIP inventory cost and finished-good inventory cost, which are defined as follows:

WIP inventory cost: Since this paper carries out batch availability assumption, so when a job is completed but not yet available for delivery, i.e., a job is finished before the production batch to which it belongs is completely finished, it has to wait until the remaining jobs in this production batch are completely finished, subsequently, will incur an inventory cost, namely WIP inventory cost, in this time interval. So the WIP inventory cost associated with a production batch should in theory depend on the number of jobs in this batch. Without loss of generality, we assume that the WIP inventory cost associated with the production batch B_k^p is $h_w(\omega_k)$ for $k = 1, 2, \ldots, u$, where h_w is a non-decreasing function of ω_k. Recall that the setup cost of the production batch B_k^p is also a non-decreasing function of ω_k, therefore, the setup cost function and WIP inventory cost function can be unified into a common expression which is given by $\theta(\omega_k)$ for $k = 1, 2, \ldots, u$, where θ is a non-decreasing function of ω_k.

Finished-good inventory cost: If a production batch is finished but delivery is not available, it will wait until delivery is available and thus will incur another inventory cost, namely finished-

good inventory cost. Finished-good inventory cost of a job should in theory depend on the time that the job spends in the time interval between the completion date of the production batch to which it belongs and the departure date of the delivery batch to which it belongs. Therefore, we assume that the finished-good inventory cost associated with job j is $h_f\left(t^d_{h(j)} - C^p_{k(j)}\right)$, where h_f is a non-decreasing function of the time interval $\left(t^d_{h(j)} - C^p_{k(j)}\right)$. Both WIP inventory cost and finished-good inventory cost constitute the total inventory cost incurred in the production stage.

Moreover, if a job arrives at the customer before its due date, it will incur an inventory cost namely customer inventory cost which should in theory depend on the time interval between the arrival time of the delivery batch to which it belongs and its due date. So we assume that the customer inventory cost associated with job j is $h_c\left(C^d_{h(j)}\right)$, where h_c is a non-decreasing function of $C^d_{h(j)}$. Therefore, with the definitions of the cost factors mentioned above, the objective function can be given by:

$$F(\psi,\varphi) = \sum_{k=1}^{u} \theta(\omega_k) + \sum_{h=1}^{v} \eta_c(\sigma_h) + \sum_{j=1}^{n} \left(h_f\left(t^d_{h(j)} - C^p_{k(j)}\right) + h_c\left(C^d_{h(j)}\right) \right) \tag{21}$$

where the three terms on the right side of the Eq.(21) represent the sum of setup cost and WIP inventory cost, the delivery cost, and the sum of finished-good inventory cost and customer inventory cost.

Subject to:

$$\sum_{k=1}^{u} \omega_k = \sum_{h=1}^{v} \sigma_h = n, \tag{22a}$$

$$1 \leqslant \sigma_h \leqslant c,\ h = 1,2,\ldots,v, \tag{22b}$$

$$C^p_{k(j)} \leqslant t^d_{h(j)},\ k = 1,2,\ldots,u,\ h = 1,2,\ldots,v, \tag{22c}$$

$$C^d_{h(j)} \leqslant d_j,\ j = 1,2,\ldots,n, \tag{22d}$$

$$C^p_{k+1} - C^p_k \geqslant s_t(\omega_{k+1}),\ k = 1,2,\ldots,u-1, \tag{22e}$$

$$C^d_{h+1} - C^d_h \geqslant \eta_t(\sigma_{h+1}),\ h = 1,2,\ldots,v-1, \tag{22f}$$

Constraint (22a) indicates that the number of processed jobs in both production and delivery stages equals to the number of orders (jobs) from the customer. Constraint (22b) indicates that

the jobs delivered by the vehicle at one time should not exceed the capacity of the vehicle. Constraint (22c) indicates that for each job, the production batch to which it belongs should be finished before the departure date of the delivery batch to which it belongs. Constraint (22d) indicates that each job should arrive at the customer before its due date. Constraint (22e) indicates that the time interval between two consecutive production batches should not be less than the setup time. Constraint (22f) indicates that the time interval between two consecutive delivery batches should not be less than a round trip delivery time.

For ease of reference, we denote this proposed problem as the Single-Product, ISPIDI problem (SP-ISPIDI).

4.3 NP-hard Complexity

We now present the NP-hard proof for the problem SP-ISPIDI by a reduction from INTEGER KNAPSACK problem, which is known to be NP-hard (see Karp (1972)).

INTEGER KNAPSACK problem: Given m items, each item is associated with a weight w_i and a value v_i, where $i = 1,2,\ldots,m$, and two positive numbers W and V which means the weight limitation of a collection and a proposed objective value, respectively. The decision version asks whether there is a set of m integer numbers $\{y_1, y_2, \ldots, y_m\}$ such that $\sum_{i=1}^{m} w_i.y_i \leqslant W$ and $\sum_{i=1}^{m} v_i.y_i \geqslant V$?

The following theorem states the computational complexity of the problem.

Theorem 6. *SP-ISPIDI problem is Np-hard in the capacity of the vehicle.*

Proof. The problem is shown to be NP-hard through a reduction from the INTEGER KNAPSACK problem, which is known to be NP-hard. We divide the proof into two steps, in which the first one is to build a special case of the problem and transform the special problem into an equivalent problem by introducing a new variable x_k, for $k = 1,2,\ldots,m$, which represent the number of batches that hold exactly c_k parts, the second step is to show that this transformed problem is equivalent to a INTEGER KNAPSACK problem. The details of this reduction are

described below.

Step 1: We construct an instance of our problem as follows:

- Customer inventory cost function: $h_c = 0$.
- Finished-good inventory cost function: $h_f \to +\infty$.
- Delivery cost function η_c is same as the common expression of WIP holding cost and setup cost θ: $\theta = \eta_c (= \alpha)$, where α is a common expression.

With this assumption, each production batch should be totally delivered upon its completion time on the machine in order to avoid the finished-good inventory cost, which indicates the production scheme is same as that of delivery, i.e., $k = h$, $\omega_k = \sigma_h$ and $u = v$. We now introduce the new of variables x_k, for $k = 1,2,\ldots,m$, which represent the number of batches that hold exactly c_k parts, i.e., $\forall k \in \{1,2,\ldots,m\}$, $x_k = |\{q \in \{1,2,\ldots,u\}/\omega_k = c_k\}|$. Then, the formulation of this special problem can be transformed into the following equivalent problem.

$$\sum_{k=1}^{m} 2\alpha(c_k)x_k \leqslant obj \quad (objective\ value) \tag{23}$$

subject to the following constraint:

$$\sum_{k=1}^{m} c_k x_k \geqslant n \quad (number\ of\ jobs) \tag{24}$$

It deserves to note that the size of the equivalent problem equals to m, rather than n.

Step 2: Let $n = V$, $obj = W$, $c^* = \{w_1, w_2, \ldots, w_m\}$, $2\alpha(w_k) = v_k$ for $\forall k \in \{1,2,\ldots,m\}$, then the formulation of the special problem can be rewrite as follows: $\sum_{k=1}^{m} v_k x_k \leqslant W$, subject to $\sum_{k=1}^{m} w_k x_k \geqslant V$, which is the formulation of a classical INTEGER KNAPSACK problem. This completes the proof. ■

The Theorem 6 indicates that SP-ISPIDI problem is NP-hard in the capacity of the vehicle. However, the number of jobs to process and deliver should also have an impact on the complexity of the problem when the customer cost function is not zero. Based on this consideration, although this problem is still open, we believe that this problem is NP-hard in the number of jobs in the general case.

We then present some straightforward optimality properties to our problem. In order to search for an optimal solution for our problem, we may confine our attention to solutions that satisfy the following properties.

(1) There should be no idle time between the first and the last processed jobs in each production batch at the factory.

(2) Once all the jobs of a production batch have finished processing and the vehicle is idle at the factory, the delivery batch should depart from the factory.

(3) When the vehicle finishes a delivery and returns to the factory, if there are still jobs that need to be transported, then the vehicle either (a) transports the next delivery batch of jobs immediately or (b) waits and starts the next delivery at the completion time of a new production batch.

By formulating the problem as a general model, we obtained the complexity of the problem in a general case and some optimality properties to the model. However, because of the nonlinear nature of the problem and general relations between the variables and the objective function, it seems to be difficult to do some simulations for this general model or to investigate it a little more deeply. Based on this consideration, in the following section, without loss of generality, we turn to study a common precise case in our practical life.

4.4 A Common Precise Model Derived From The General Model

In this section, we build a special case by making the following assumptions: (1) the setup time and setup cost are assumed to be two constants s_t and s_c, respectively; (2) the round trip delivery time and delivery cost of each delivery batch are assumed to be two constants τ and η, respectively; (3) the unit WIP inventory cost equals to the unit finished-good inventory cost, and assumed to be a constant β_1. Furthermore, the unit customer holding cost is also assumed to be a constant β_2. Based on these assumptions, we can precise the general model mentioned in Section 4.2 as follows. According to the description of WIP inventory and finished-good inventory

cost in Section 4.2, we define that the WIP inventory cost associated with the production batch B_k^p, $k = 1,2,\ldots,u$, is:

$$h_w(\omega_k) = \beta_1(\omega_k - 1)p_t + \beta_1(\omega_k - 2)p_t + ... + \beta_1 p_t = \frac{\beta_1(\omega_k)(\omega_k - 1)p_t}{2} \tag{25}$$

We define that the finished-good inventory cost associated with job j, $j = 1,...,n$, is:

$$h_f\left(t^d_{h(j)} - C^p_{k(j)}\right) = \beta_1\left(t^d_{h(j)} - C^p_{k(j)}\right) \tag{26}$$

At last, we assume that the customer holding cost associated with job j, $j = 1,...,n$, is:

$$h_c\left(C^d_{h(j)}\right) = \beta_2\left(d_j - C^d_{h(j)}\right) \tag{27}$$

So the objective function can be rewritten by

$$F(\psi,\varphi) = s_c u + \eta v + \sum_{k=1}^{u} h_w(\omega_k) + \sum_{j=1}^{n}\left(h_f\left(t^d_{h(j)} - C^p_{k(j)}\right) + h_c\left(d_j - C^d_{h(j)}\right)\right) \tag{28}$$

where the four terms on the right side of the Eq.(28) represent the setup cost, delivery cost, WIP inventory cost and the sum of finished-good inventory cost and customer inventory cost.

Subject to: constraints (22a)-(22f).

Although the complexity of this precise instance is also still open, however, we found that this proposed instance is also intractable on an empirical bases. Therefore, we try to develop a heuristic algorithm for solving this precise instance. Before proposing the heuristic, we firstly propose an optimality property about the correlation between a production scheme and a delivery scheme which will be used in the proposed heuristic algorithm.

Theorem 7. *In an optimal schedule, for any $k \in [1,\ldots,u]$, there exist $h \in [1,\ldots,v]$ such that $S^p_k = S^d_h$.*

Proof. (*By contradiction*) Suppose that the property is not satisfied by an optimal schedule π. Then, in this schedule there must be at least one pair of integers, say (k_0,h_0) with $1 \leqslant k_0 \leqslant u$ and $1 \leqslant h_0 \leqslant v$, such that $S^d_{h_0} < S^p_{k_0} < S^d_{h_0+1}$. Let $x = S^p_{k_0} - S^d_{h_0}$ and $t = (C^p_{k_0+1} - C^p_{k_0} - p_t.\omega_{k_0+1})$, see Fig. 13. So these x jobs will wait for a time of $(C_{k_0+1} - C_{k_0})$ for delivery and finally be

delivered in the delivery batch $B^d_{h_0+1}$ which is composed of these x jobs and a certain number of jobs processed in the production batch $B^p_{k_0+1}$. Consequently, these x jobs will incur a finished-good inventory cost in the time interval of $(C^p_{k_0+1} - C^p_{k_0})$.

Now, we need to verify whether there exist a schedule denoted by π' which is better than π. If we can determine such a schedule π', then the property holds. We build a new schedule π' based on the following transform, see the transformation process presented in Fig. 13 and Fig. 14.

$$\pi' = \begin{cases} \omega'_k = \omega_k, \forall k \in \{1,\ldots,k_0-1\} \cup \{k_0+2,\ldots,u\} \\ \omega'_{k_0} = \omega_{k_0} - x \\ \omega'_{k_0+1} = \omega_{k_0+1} + x \\ T'_{k_0} = T_{k_0} + min\{xp_t, (t - p_t\omega_{k_0+1} - s_t)\} \\ T'_{k_0+1} = T_{k_0+1} - xp_t \\ \sigma'_h = \sigma_h, \ \forall h \in \{1,\ldots,v\} \end{cases}$$

where T_{k_0} is the starting time of $B^p_{k_0}$. Depending on the size of t, the proof can be divided into

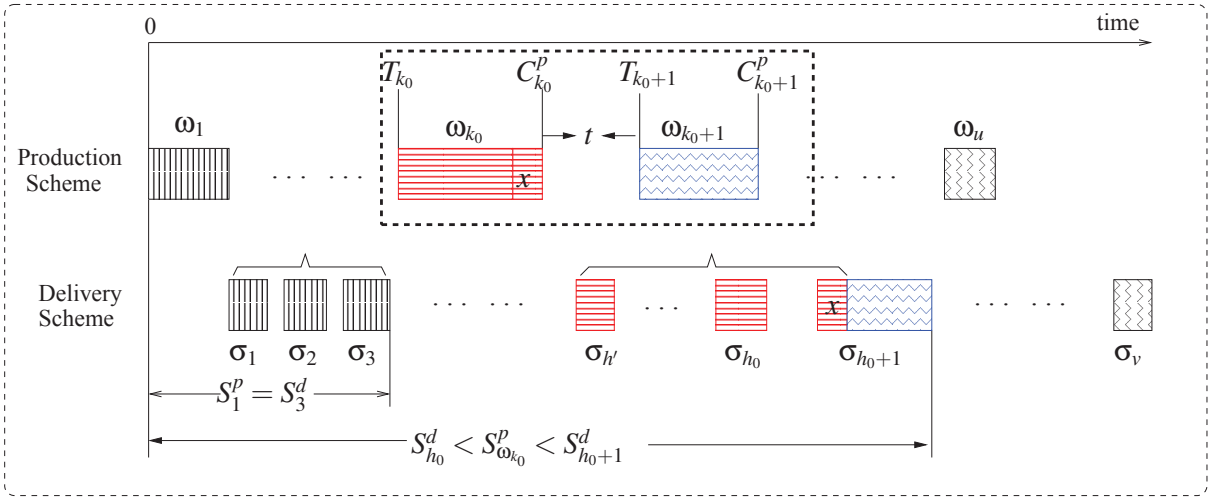

Fig. 13. *A feasible joint scheme* π

the following three cases:

(1) $t \geqslant (s_t + xp_t)$;

(2) $t < (s_t + xp_t)$;

(3) $t = s_t$.

According to the method of building schedule π', it is obvious that the third case is the worst case, this is so because in the third case, after the transformation from π to π', the jobs in the production batch $B_{k_0}^{'p}$, i.e. $(\omega'_{k_0} = \omega_{k_0} - x)$, incur the most finished-good inventory cost. Therefore, we only need to prove this property for the third case, then the other two cases

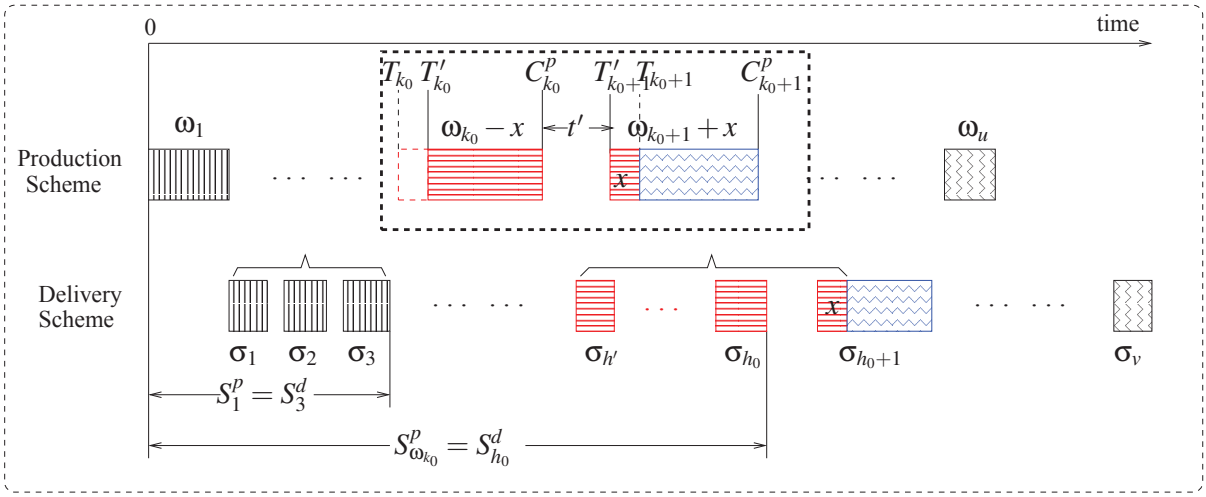

Fig. 14. *A feasible joint scheme π' derived from π*

are automatically proved. In the third case, after the transformation from π to π', all other jobs remain in their original position except these x jobs. Moreover, the number of production batches is not changed. Thus, the total setup, delivery, customer inventory cost is not changed. Consequently, the difference in the values of the objectives under schedule π and π' is due only to the WIP inventory cost and finished-good inventory cost incurred by these x jobs. By Eq.(25) and Eq.(26), under π the total WIP inventory and finished-good inventory cost is:

$$f(\pi) = h_w(\omega_{k_0}) + h_w(\omega_{k_0+1}) + x\beta_1(s_t + p_t\omega_{k_0+1}) \tag{29}$$

$$= \frac{\omega_{k_0}(\omega_{k_0}-1)}{2}\beta_1.p_t + \frac{\omega_{k_0+1}(\omega_{k_0+1}-1)}{2}\beta_1.p_t + x\beta_1(s_t + p_t\omega_{k_0+1}) \tag{30}$$

while under $\pi^{'}$ it is:

$$f(\pi') = h_w(\omega_{k_0} - x) + h_w(\omega_{k_0+1} + x) + (\omega_{k_0} - x)p_t x\beta_1 \tag{31}$$

$$= \frac{(\omega_{k_0} - x)(\omega_{k_0} - x - 1)}{2}\beta_1 p_t + \frac{(\omega_{k_0+1} + x)(\omega_{k_0+1} + x - 1)}{2}\beta_1 p_t + x\omega_{k_0}\beta_1 p_t - x^2\beta_1 p_t \tag{32}$$

Thus, it is easily verified that: $f(\pi) - f(\pi') = xs_t\beta_1 > 0$. Consequently, we have the sum of the WIP and finished-good inventory cost under $\pi^{'}$ is strictly less than that under π. This contradicts the optimality of π and completes the proof. ■

Corollary 1. *In an optimal schedule, jobs processed in different production batches can not be transported in one delivery batch.*

Proof. Assume a delivery batch contains a number of ζ jobs in which x jobs came from the kth production batch and the other y jobs came from the (k+1)th production batch. Then we can always move the x jobs from the kth production batch into the (k+1)th production batch for processing without increasing the objective function. ■

According to theorem 7 and Corollary 1, for a given delivery solution φ, we can construct the possible production batches of a potential production solution ψ by merging some delivery batches together, i.e.

$$\varphi = \{\sigma_1, \sigma_2, \ldots, \sigma_v\} \Rightarrow \begin{cases} \psi_1 = \{\sigma_1 + \cdots + \sigma_v\} \\ \psi_2 = \{\sigma_1, \sigma_2, \ldots, \sigma_{v-1} + \sigma_v\} \\ \cdots \\ \psi_{(2^{v-1})} = \{\sigma_1, \sigma_2, \ldots, \sigma_v\} \end{cases}$$

For ease of reference, we denote this method as "Merging Method" (M-Method). Consequently, we obtain an approach to generate the possible production solutions with a given delivery solution. For example, for a given delivery solution $\varphi = \{2, 1, 6\}$, according to M-Method, we

obtain the following possible production solutions:

$$\varphi = \{2,1,6\} \Rightarrow \begin{cases} \psi_1 = \{2+1+6\} = \{9\} \\ \psi_2 = \{2,1+6\} = \{2,7\} \\ \psi_3 = \{2+1,6\} = \{3,6\} \\ \psi_4 = \{2,1,6\} \end{cases}$$

Therefore, we obtain four possible joint schedules which are: $\pi_1 = \{\psi_1, \varphi\} = \{9|2,1,6\}$, $\pi_2 = \{\psi_2, \varphi\} = \{2,7|2,1,6\}$, $\pi_3 = \{\psi_3, \varphi\} = \{3,6|2,1,6\}$ and $\pi_4 = \{\psi_4, \varphi\} = \{2,1,6|2,1,6\}$. Moreover, it is obviously that, for a given delivery solution, to obtain an optimal schedule we only need to check at most 2^{v-1} possible production solutions.

After generating a joint schedule, we need to further calculate the date factors involving job completion time, job departure time, job arrival time, etc. In this paper, we calculate the date factors for a given joint scheme as follows. For a given schedule scheme, the dates calculation starts from the last production batch and goes backwards to the first. The completion date calculation for each production batch is divided into two stages in which the first is to determine the departure dates and arrival dates of the delivery batches which are produced in the given production batch starting from the last delivered batch and going backwards to the first; the second stage is to set the completion date of the production batch as the earliest departure date of these delivery batches. This strategy enables us to obtain the latest dates of the batches for the whole system, that incurs the minimal cost for a given joint sequence of batches (see Elmahi et al. (2006)).

As an example of the above date calculation method, we can consider the following joint schedule of a problem with five jobs, (2,3 | 1,1,3). For ease of reference, we denote the two production batches by respectively B_1^p and B_2^p, and the three delivery batches by respectively B_1^d, B_2^d and B_3^d. The due dates associated with these jobs are 100, 102, 115, 116, 117. The unit production time is 1. The setup time and delivery time are 15 and 10, respectively. The data calculation process is described in detail as follows.

The last (second) production batch (B_2^p) involving three jobs is associated with only one delivery batch (B_3^d). According to constraint (22c), we obtain that the arrival time of the last delivery

batch (C_3^d) is min{115,116,117}=115, and the departure time is 115-10=105. Consequently, the completion of the last production batch (C_2^p) is 105.

The first production batch (B_1^p) involving two jobs is associated with two delivery batches (B_1^d and B_2^d). According to constraint (22d) and (22f), we obtain that the arrival date of the second delivery batch is min{115-10-10, 102}=95, and the departure time of delivery batch B_2^d is 95-10=85. In this same way, we obtain that the arrival date of the first delivery batch (B_1^d) is min{95-10-10, 100}=75. Consequently, according to constraint (22c) and (22e), the completion time of the first production batch (C_1^p) should be equal to min{105-3-15, 75-10}=65.

4.5 Dominance Related Heuristic Approach

In the following part, we propose a heuristic algorithm for solving the proposed problem. Note that our heuristic sequentially, rather than simultaneously, considers the delivery and production stages. The main idea of the proposed heuristic is described as follows. First, determine the delivery solution. Then, build the possible production solutions in terms of the Corollary 1. Finally, generate the joint solution by matching them together. However, in the process of generating delivery solutions, the number of delivery solutions increases drastically when the number of jobs increases, so we need to prune the search space to reduce the running time. With this consideration, we provide a prune rule in the following section. Before proposing the prune rule, we define the following notations which will be used in the following parts.

We define a partial solution as two subsequences of batches which include the same number (less than n) of jobs. In this partial solution, the first subsequence of batches represents the production scheme, and the second one represents the delivery scheme. Moreover, we define that if a solution $\pi_e = \{\psi_e, \varphi_e\} = (B_1^p, B_2^p, \ldots, B_u^p | B_1^d, B_2^d, \ldots, B_v^d)$ includes another partial solution $\pi = \{\psi, \varphi\} = (B_k^p, \ldots, B_u^p | B_h^d, \ldots, B_v^d)$ with $0 < k < u$ and $0 < h < v$, then we say that the solution π_e is derived from the partial solution π. For example, consider a problem with 10 jobs, any feasible two subsequences of batches that includes the same number (less than 10) of jobs can be treated as a partial solution, e.g., $\pi = \{\varphi_1, \psi_1\} = (3,1,2|1,2,1,2)$. Any feasible subsequences of batches that starts from the partial solution π and includes π is an extended

solution of π, e.g. $\pi_e = \{\varphi_e, \psi_e\} = \{2,2,3,1,2|1,1,2,1,2,1,2\}$.

The Pruning rule: If two partial solutions $\pi_1 (= (B^p_{k'}, \ldots, B^p_{u'} | B^d_{h'}, \ldots, B^d_{v'}))$ and $\pi_2 (= (B^p_k, \ldots, B^p_u | B^d_h, \ldots, B^d_v))$ satisfy the following constraints:

$$f(\pi_1) \leqslant f(\pi_2) \tag{33a}$$

$$t^d_{h'} \geqslant t^d_h \tag{33b}$$

$$T_{k'} \geqslant T_k \tag{33c}$$

, then we say π_1 is a 'good solution' and π_2 is a 'bad solution' and subsequently delete the bad one. Otherwise, we keep the good one.

In the pruning rule, the constraint (33a) indicates the objective value of the partial schedule π_1 is less than that of the partial schedule π_2, Constraint (33b) indicates that the departure date of the partial schedule π_1 is later than that of π_1, this constraint will result in a lower customer inventory cost. Constraint (33c) indicates that the starting processing date of the partial schedule π_1 is later than that of π_2; this constraint may result in a lower finished-good inventory cost and customer inventory cost.

With the three above constraints, we can intuitively observe that the partial solution π_1 can generate a good solution. We now describe further on this point. Let π'_1 and π'_2 be the complementary solutions of π_1 and π_2, respectively. Moreover, we denote π_{1-best} and π_{2-best} as the best solutions derived from π_1 and π_2, respectively, i.e. $\pi_{1-best} = \{\pi'_1, \pi_1\}$ and $\pi_{2-best} = \{\pi'_2, \pi_2\}$. Assume that π_1 and π_2 satisfy the above constraints, but $f(\pi_{1-best}) > f(\pi_{2-best})$. Because of constraints (33b) and (33c), we can always modify the solution π_{1-best} by replacing the complementary solution π'_1 using π'_2, i.e. $\pi'_{1-best} = \{\pi'_2, \pi_1\}$. Moreover, because of the constraint (33a) and the assumption that $f(\pi_{1-best}) > f(\pi_{2-best})$, we obtain that $f(\pi'_2) < f(\pi'_1)$. Thus, we obtain $f(\pi'_{1-best}) < f(\pi_{2-best})$, which contradicts the optimality of π_{2-best}. Therefore, we conclude that π_1 can generate a good solution. However, it should be noted that because of the existence of the intermediate inventory, this pruning rule does not work for generating joint solution. Thus, in this paper we take it as a heuristic knowledge in the generation process of the delivery solution to reduce the running time.

Based on the prune rule, we proposed the following heuristic: for level 0, there is no jobs; For the first level (includes only one job), there is only one possible joint solution which is (1|1). For level k (includes k jobs), all the "good solutions" for a number of k jobs will be kept. The process to build the "good solutions" for level k is described as follows: (1) build delivery solutions of level k by considering all the delivery solutions in the retained "good solutions" of all the previous levels from 1 to $k-1$. For each retained solution of level $k' \leq k$, a new delivery solution of level k is built by simply adding a batch of $k-k'$ jobs, if this is possible. Then, repeat the procedure mentioned above until the level n is considered. As an example of the proposed heuristic, we can consider a scheduling issue with three jobs in which the due date associated with each job is 100,150,151, respectively. In this example, the setup time and setup cost are set to be 5 and 10, respectively. The delivery time and delivery cost are set to be 5 and 10, respectively. The unit holding costs in the supplier and customer area are set to be 1 and 2, respectively. We now present the details of the proposed algorithm by considering this example. For level 1, there is only one job, therefore there is only one possible joint solution, which is (1|1).

For level 2 (includes 2 jobs), we firstly build the possible delivery schemes by considering the delivery scheme generated in level 1, which is (1). We build the delivery solution of level 2 by simply adding a batch of (2-1) jobs to the previous delivery solution, i.e. (1,1) and (2). According to M-Method (see Fig. 15), we build the potential production schemes for the delivery schemes (1,1) and (2). For delivery scheme (1,1), we obtain the potential production schemes (1,1) and (2). For delivery scheme (2), we obtain the potential production scheme (2). Then, generate the joint solutions by matching them together, i.e. (1,1|1,1), (2|1,1) and (2|2). Finally, compare the three potential joint solutions and keep the good one, which is (2|1,1) in this example.

For level 3 (includes 3 jobs), build the delivery schemes in the same way as described for level 2. First, based on the solution of level 0, we can build the delivery scheme (3); based on the delivery scheme of the first level, which is (1), we obtain the delivery scheme (1,2); based on the delivery scheme generated in level 2, we obtain the delivery scheme (1,1,1). Then generate the potential production schemes for each given delivery scheme according to the M-Method. For

the delivery scheme (3), the potential production scheme is (3); For the delivery scheme (1,2), the potential production schemes are (1,2) and (3); For the delivery scheme (1,1,1), the potential production schemes are (1,1,1), (2,1), (1,2) and (3). Then match them together to generate the potential joint solutions, which are (3|3), (1,2|1,2), (3|1,2), (1,1,1|1,1,1), (2,1|1,1,1), (3|1,1,1) and (1,2|1,1,1). Finally, compare the three potential joint solutions and choose the best one, which is (3|1,2), as the final solution.

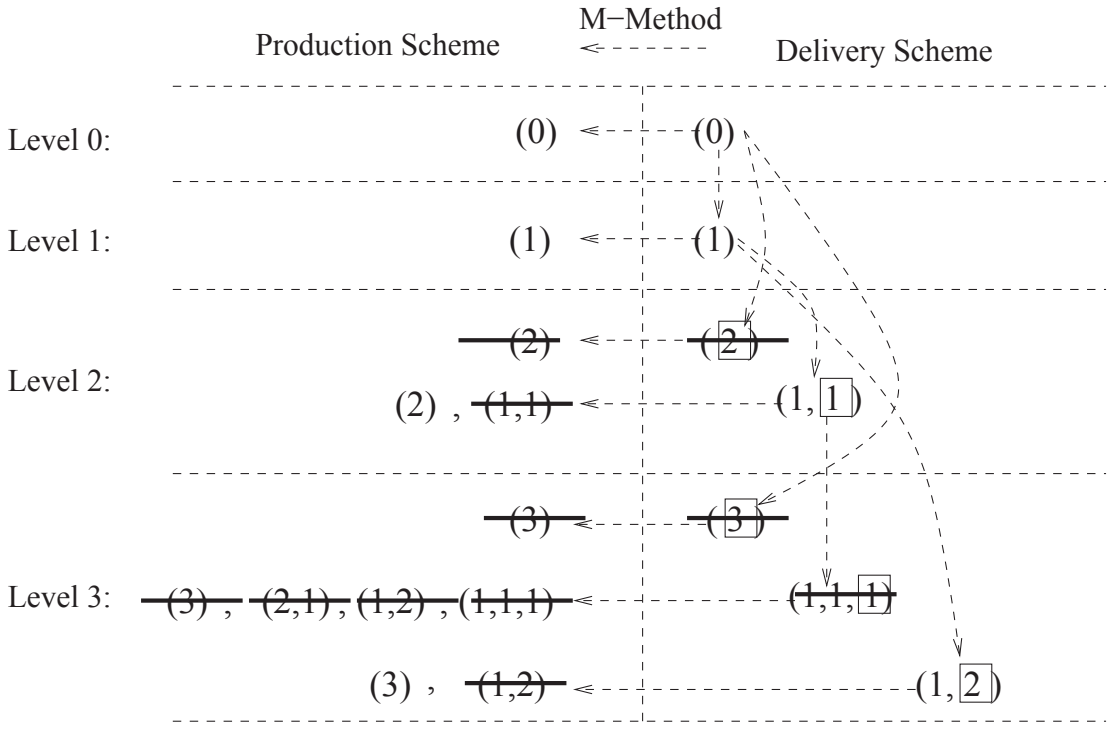

Fig. 15. *Description of the proposed heuristic algorithm*

The formal algorithm is described in the Algorithm 6.

Let $D(i)$ be the set of delivery schemes in the kept 'good solution' set of level i, and $v(i)$ the set of number of batches in the corresponding delivery schemes, then the complexity of the algorithm is the sum of the number of operations to construct $D(i)$ from $i = 1$ to $i = n$ and the production schemes for each given delivery scheme. In the worst case, i.e. the pruning rule is not applicable, the number of delivery schemes of a given level k will be 2^{k-1}, and the number of potential production schemes for each delivery scheme that has a number of v batches is 2^{v-1}. Consequently, the complexity of the proposed algorithm is $O(2^n)$ in the worst case. However,

Algorithm 6 Steps of the proposed heuristic algorithm.

```
1: minsol ← build a coordinated solution ;
2: evalmin ← getEvaluation (minsol) ;
3: for j ← 1 to n do
4:    for b ∈ c* and b ⩽ j do
5:       for each delivscheme in nonDominated(j − b) do
6:          delivscheme ← addNewBatch(delivscheme, b) ;
7:          setOfProdscheme ← generateProdschemeByM-Method(delivscheme) ;
8:          for each prodscheme in setOfProdscheme do
9:             sol = (prodscheme, delivscheme) ;
10:            eval =getEvaluation(sol) ;
11:            dominated ← false ;
12:            for each dom in nonDominated(j) do
13:               if dominates(dom, sol) then
14:                  dominated ← true ;
15:               else
16:                  if dominates(sol, dom) then
17:                     remove dom from nonDominated(j);
18:                  end if
19:               end if
20:               if !dominated then
21:                  add sol to nonDominated(j) ;
22:               end if
23:               if j == n and eval < evalmin then
24:                  minsol ← sol ;
25:                  evalmin ← eval ;
26:               end if
27:            end for
28:         end for
29:      end for
30:   end for
31: end for
32: bestSolution ← minsol ;
```

experimental results show that this case is rarely met.

Due to the associated complex objective cost function (See Eq.(28)), it is difficult to develop a worst-case analysis for the proposed heuristic. Therefore, in order to evaluate the performance of the proposed heuristic, we compare its solution value to the lower bound, which is described in the following section.

4.6 MIP Model and Lagrangian Lower Bound Derivation

In this section, we firstly formulate the precise instance of the general model into a MIP model, then propose a lower bound based on the MIP model using the Lagrangian relaxation method.

4.6.1 MIP Model

In the following, we formulate our problem as a mixed integer programming model. Before the model is presented, the parameters and variables used in the model are firstly described below.

Parameters:

1. d_j: the due date of job j, $j = 1,2,\ldots,n$;
2. c: the capacity of the vehicle;
3. β_1, β_2: the unit holding cost in production and customer area, respectively;
4. τ, η: the round trip time and round trip cost of the vehicle, respectively;
5. p_t:the unit processing time;
6. s_t, s_c: the setup time and setup cost in factory, respectively;
7. k,h: the index of the kth production batch and hth delivery batch, respectively, $k = 1,2,\ldots,n$, $h = 1,2,\ldots,n$;

8. M: a sufficiently large positive constant;

Decision variables:

1. δ_{jk}^{p}: 1, if job j belongs to the kth production batch; 0, otherwise;
2. δ_{jh}^{d}: 1, if job j belongs to the hth delivery batch; 0, otherwise;
3. C_j^p: the completion time of job j on the single machine;
4. C_j^d: the arrival time at customer of job j;

With the notations mentioned above, we build the mixed integer programming model as follows:

$$MinZ = s_c u + \eta v + \sum_{j=1}^{n} \left(\beta_1 (C_j^d - C_j^p - \tau) + \beta_2 (d_j - C_j^d) \right) \tag{34}$$

$$\sum_{h=1}^{n} \delta_{jh}^{d} = \sum_{k=1}^{n} \delta_{jk}^{p} = 1,\ j = 1,\ldots,n, \tag{35a}$$

$$u = \sum_{k=1}^{n} k \delta_{nk}^{p}, \tag{35b}$$

$$v = \sum_{h=1}^{n} h \delta_{nh}^{d}, \tag{35c}$$

$$\sum_{j=1}^{n} \delta_{jh}^{d} \leqslant c,\ h = 1,\ldots,n, \tag{35d}$$

$$C_j^d \leqslant d_j,\ j = 1,\ldots,n, \tag{35e}$$

$$C_j^d - C_j^p \geqslant \tau,\ j = 1,\ldots,n, \tag{35f}$$

$$C_{j+1}^p - C_j^p \geqslant p_t,\ j = 1,\ldots,n, \tag{35g}$$

$$C_{j+1}^p - C_j^p \leqslant p_t + (2 - (\delta_{jk}^{p} + \delta_{j+1,k}^{p}))M, j = 1,\ldots,n-1, k = 1,\ldots,n, \tag{35h}$$

$$C_{j+1}^p - C_j^p \geqslant (p_t + s_t)(\delta_{jk}^{p} + \delta_{j+1,k+1}^{p} - 1),\ j,k = 1,\ldots,n-1, \tag{35i}$$

$$C_{j+1}^d - C_j^d \geqslant 0,\ j = 1,2,\ldots,n-1, \tag{35j}$$

$$C_{j+1}^d - C_j^d \leqslant (2 - (\delta_{jh}^{d} + \delta_{j+1,h}^{d}))M,\ j = 1,\ldots,n-1, h = 1,\ldots,n, \tag{35k}$$

$$C_{j+1}^d - C_j^d \geqslant 2\tau(\delta_{jh}^{d} + \delta_{j+1,h+1}^{d} - 1),\ j,h = 1,\ldots,n-1, \tag{35l}$$

$$\delta_{jh}^{d} \leqslant \delta_{j+1,h}^{d} + \delta_{j+1,h+1}^{d},\ j,h = 1,\ldots,n-1, \tag{35m}$$

$$\delta_{jk}^{p} \leqslant \delta_{j+1,k}^{p} + \delta_{j+1,k+1}^{p},\ j,k = 1,\ldots,n-1, \tag{35n}$$

$$C_j^p, C_j^d \geqslant 0,\ j = 1,2,\ldots,n, \tag{35o}$$

$$\delta_{jk}^p, \delta_{jh}^d \in \{0,1\},\ j = 1,\ldots,n,\ k,h = 1,\ldots,n, \tag{35p}$$

The objective function (34) minimizes the sum of setup, delivery and inventory costs. Constraint (35a) ensures that, in both production and delivery stages, each job must be scheduled exactly once. Constraint (35b) and (35c) define the number of production batches and delivery batches, respectively. Constraint (35d) guarantees that the number of jobs scheduled in one delivery batch cannot exceed the capacity of the vehicle. Constraint (35e) guarantees that each job must arrive at the customer on time. Constraint (35f) indicates that the each job j should be completed before its departure date. Constraint (35g) and (35h) define the property of the completion time of the two consecutive jobs that are in one production batch. They indicate that the single machine may start to process one job of a batch (apart from the first job in this batch) only after its previous job in this batch has been completed. Constraint (35i) indicates that the single machine may start to process one production batch only after the jobs of the previous batch have been completed. Constraint (35j) and (35k) indicate the jobs of the same delivery batch will have the same arrival time in the customer area. Constraint (35l) indicates that the transporter may start to deliver one delivery batch only after its previous delivery batch has been completely delivered. Constraint (35m) and (35n) indicates that, in both production and delivery stages, consecutive jobs j and $j+1$ will either be in the same batch or be in consecutive batches. Constraint (35o) and (35p) define the range of the variables.

4.6.2 Lagrangian decomposition

The Lagrangian relaxation (LR) approach have been successfully applied to many industrial problems (See Mouret et al. (2011)). Lee and Yoon (2010) has applied lagrangian relaxation and decomposition techniques to the integrated scheduling problems in order to generate an efficient lower bound. Pirkul and Jayaraman (1998) studied the capacitated plant and warehouse supply chain management problem, they formulated the problem as a mixed integer programming model and then solved the model by the Lagrangian relaxation method. For more details about the Lagrangian relaxation method, see Fisher (1981), Frangioni (2005) and Neiro (2006),

among others. The LR approach presented here is on a basis of stage decomposition. From the mixed integer programming model presented in Section 4.6.1, we can see that only Constraint (35f) couple different stages. Thus, we form the following LR problem by introducing the constraint (35f) into the objective function (Eq.(34)) through Lagrangian multipliers λ_j.

Let λ_j be non-negative Lagrangian multipliers that are associated with Constraint (35f). The associated Lagrangian problem can be expressed as follows:

$$Z_{LR}(\lambda) = \min_{\{C_j^p, C_j^d\}} \left\{ s_c u + \sum_{j=1}^{n} (\lambda_j - \beta_1) C_j^p + \eta v + \sum_{j=1}^{n} (\beta_1 - \beta_2 - \lambda_j) C_j^d \right. \tag{36}$$

$$\left. + \sum_{j=1}^{n} (\beta_2 d_j - \beta_1 \tau + \lambda_j \tau) \right\} \tag{37}$$

subject to constraints (35a)-(35e) and constraints (35g)-(35p). Here λ is a vector of non-negative Lagrangian multipliers with elements $\{\lambda_j\}$, where $j = 1, 2, \ldots, n$. The Lagrangian dual problem is described as follows:

$$Z_{LD} = \max_{\{\lambda_j\}} \left\{ \sum_{j=1}^{n} (\beta_2 d_j - \beta_1 \tau) + \sum_{j=1}^{n} \lambda_j \tau + \min_{\{C_j^p, C_j^d\}} \left\{ s_c u + \sum_{j=1}^{n} (\lambda_j - \beta_1) C_j^p \right. \right. \tag{38}$$

$$\left. \left. + \eta v + \sum_{j=1}^{n} (\beta_1 - \beta_2 - \lambda_j) C_j^d \right\} \right\} \tag{39}$$

subject to constraints (35a)-(35e), constraints (35g)-(35p).

For given values of $\{\lambda_j\}$, the relaxed problem (LR) can be decomposed into two smaller subproblems, Z_{LR}^P and Z_{LR}^D, which are for the production and delivery stage, respectively. The production problem is given as follows :

$$Z_{LR}^P(\lambda) = \min_{\{C_j^p\}} \left\{ s_c u + \sum_{j=1}^{n} (\lambda_j - \beta_1) C_j^p \right\} \tag{40}$$

subject to $\sum_{k=1}^{n} \delta_{jk}^p = 1$, $C_j^p \geqslant 0$, $\delta_{jk}^p \in \{0, 1\}$, constraint (35b), constraints (35g)-(35i) and constraint (35n). After decomposition, the production problem will be unbounded on the production completion time C_j^p, therefore we introduce the following bound for C_j^p: $C_j^p + \tau - d_j \leqslant M - M|\delta_{j+1,h}^p - \delta_{j,h}^p|$, where M is a sufficiently large positive constant.

The transportation problem is given as follows :

$$Z_{LR}^D(\lambda) = \min_{\{C_j^d\}} \left\{ \eta v + \sum_{j=1}^{n} (\beta_1 - \beta_2 - \lambda_j) C_j^d \right\} \tag{41}$$

subject to $\sum_{k=1}^{n} \delta_{jh}^{d} = 1$, $C_j^d \geqslant 0$, $\delta_{jh}^{d} \in \{0,1\}$, constraint (35c), constraints (35d)-(35e) and constraints (35j)-(35m).

Because of the similarity of the two subproblems, we apply the following general dynamic programming approach (DP) to solve the production problem (Z_{LR}^{P}) and the transportation problem (Z_{LR}^{P}). The proposed DP approach is similar with the heuristic algorithm, but is operated on the single stage (production stage or delivery stage). We take the production subproblem as an example to describe the dynamic algorithm. In the dynamic algorithm, for level 1 (includes only one job), there is only one production scheme which is (1). For level k, we build the possible production schemes by adding a batch with $k-k'$ jobs, $1 \leqslant k' < k$, before the previous partial production schemes. We define $F(k)$ as the minimum solution value of a problem with k jobs. We note $f(h)$ as the cost of the first batch (includes h jobs) in a feasible solution, $1 \leqslant h \leqslant c$. Further, we denote by $v(l)$ the increasing amount of evaluation after the batch with l jobs is added to an existing partial solution. A formal description of the algorithm DP can be given as follows. From the process of the DP algorithm, it is readily seen that the DP algorithm can obtain an optimal solution for the corresponding subproblem.

Step 1. (Initialization) For a given set of λ_j, calculate the evaluation of the first batch with k jobs of a feasible schedule, where $k = 0,1,\ldots,c$, i.e. $f(0)$, $f(1)$, $f(2)$, …, $f(c)$.

Step 2. (Recursion) $F(k) = \min\{F(k-l)+v(l)\}$, $k = 1,2,\ldots,n,\ \ l = 1,2,\ldots,k$

Step 3. (Optimal Solution) The optimal solution can easily be obtained using backtracking method.

An example: We now take the production subproblem as an example to illustrate how the algorithm DP works. Consider one production subproblem where a set of three jobs $\{1, 2, 3\}$ is to be processed on the single machine. The unit inventory cost incurred in the production stage is set to be 0.5. Unit processing time is set to be 1 and the due date of the job is set to be $\{11, 41, 43\}$. The given set of lagrangian multipliers are 0. Setup time and setup cost are set to be 2 and 5, respectively. The transporter capacity is set to be 3. The round-trip delivery time of a batch is set to be 3.

1. (Initialization) Index all jobs according to the non-increasing order of the due dates → (43, 41, 11). For the given set of lagrangian multipliers $\lambda = 0$, calculate the cost of the first batch, which may contain one, two up to three jobs.

$$
\begin{aligned}
f(1) &= 5 - 0.5 \times (d_1 - \tau) = 5 - 0.5 \times (43 - 3) = -15 \\
f(2) &= 5 - 0.5 \times \min\{d_1 - \tau, d_2 - \tau\} \times 2 = 5 - 0.5 \times \min\{43 - 3, 41 - 3\} \times 2 = -33 \\
f(3) &= 5 - 0.5 \times (d_1 - \tau, d_2 - \tau, d_3 - \tau) \times 3 \\
&= 5 - 0.5 \times \min\{43 - 3, 41 - 3, 11 - 3\} \times 3 = -7
\end{aligned}
$$

2. (Recursion)

$$
\begin{aligned}
F(2) &= \min\{f(1) + v(1), f(2)\} \\
&= \min\{-15 + 5 - 0.5 \times \min\{d_1 - \tau - p_t - s_t, d_2 - \tau\}, -33\} \\
&= \min\{-15 + 5 - 0.5 \times \min\{43 - 3 - 1 - 2, 41 - 3\}, -33\} \\
&= -33 \\
F(3) &= \min\{f(1) + v(2), F(2) + v(1), f(3)\} \\
&= \min\{f(1) + v(2), f(1) + v(1) + v(1), f(2) + v(1), f(3)\}
\end{aligned}
$$

Since

$$
\begin{aligned}
f(1) + v(2) &= -10 - 0.5 \times \min\{d_1 - \tau - p_t - s_t, \min\{d_2 - \tau, d_3 - \tau\}\} \times 2 \\
&= -18 \\
f(2) + v(1) &= -33 + 5 - 0.5 \times \min\{d_2 - \tau - 2p_t - s_t, d_3 - \tau\} = -32 \\
f(1) + v(1) + v(1) &= -28.5 + 5 - 0.5 \times \min\{\min\{d_1 - \tau - p_t - s_t, d_2 - \tau\} - p_t - s_t, d_3 - \tau\} \\
&= -27.5
\end{aligned}
$$

Therefore, $F(3) = \min\{-18, -27.5, -32, -7\} = -32$

3. (Optimal Solution) Rearrange the jobs according to the non-decreasing order of due dates. The optimal schedule is (1,2).

4.6.3 Solving the lagrangian dual problem

In order to solve the dual problem Z_{LD}, the subgradient method is adopted for updating the Lagrangian multipliers. As we know the algorithm is commonly used to solve the types of Lagrangian dual problems that require optimally solving all the subproblems so that a subgradient direction is obtained. In this way the vector of multipliers, λ, is updated by

$$\lambda^{m+1} = \lambda^m + t^m g^m \tag{42}$$

where t^m is the step size at the mth iteration and g^m is the subgradient of Z_{LR}. The subgradient component can be obtained by the relaxed constraint (35f), i.e. $g^m = C_j^p + \tau - C_j^d$. The step size t^m is given by

$$t^m = \alpha \frac{Z^U - Z^m}{\sum_{j=1}^{n} (C_j^p - C_j^d + \tau)^2} \qquad 0 \leq \alpha \leq 2, \tag{43}$$

where α is the parameter of the step size t^m and it is assumed such as $\varepsilon_1 \leqslant \alpha \leqslant 2\varepsilon_2$ with $\varepsilon_1, \varepsilon_2 > 0$. Z^U is an estimate of the optimal value (an upper bound) and is derived from the following heuristic algorithm. The algorithmic steps of the heuristic are described as follows.

Step 1. Derive the delivery schedule that is obtained from the transportation subproblem (Z_{LR}^P) for the transportation stage.

Step 2. Find the possible production schedules according to the Corollary 1 for the delivery schedule that is obtained in *Step* 1.

Step 3. Get the joint schedules by combining the delivery schedule obtained in *Step* 1 with the possible production schedules obtained in *Step* 2. Evaluate each possible joint schedules and choose best evaluation as Z^U.

It deserves to note that, in the algorithm mentioned above, the possible production schedules are generated according to the given delivery schedule by the Corollary 1. Therefore, the joint schedules, which are obtained by combining the delivery schedule with the possible production schedules, are always feasible.

Moreover, Z^m in Eq.(43) is the value of Z_{LR} at the mth iteration. The parameter α is initially set to a value greater than 1 and is multiplied by a factor if the value of Z^m remains approximately the same over several consecutive iterations. The algorithm terminates when a given iteration number has been executed or the improvement between two consecutive iterations is equal to or less than a given value, which is set to be 0.001 in our study. The algorithm of updating the Lagrangian multiplier is as follows:

Step 1. Initialize the Lagrangian multiplier $\lambda = 0$, iteration index $m = 0$, iteration counter $m = 0$, parameter of step size $\alpha = 0.8$, $Z^0 = 0$ and $Z^U = +\infty$.

Step 2. Solve the production subproblem $Z^P_{LR}(\lambda^m)$ and the transportation subproblem $Z^D_{LR}(\lambda^m)$ by the algorithm DP described in Subsection 4.6.2.

Step 3. Calculating the value of Z^m such as $Z^m = Z_{LR}(\lambda^m) = Z^P_{LR}(\lambda^m) + Z^D_{LR}(\lambda^m)$. If the stop criteria is satisfied, then stop and return Z^m. Else calculating the subgradient and step size based on the Eq.(43). Then updating the Lagrangian multiplier λ according to Eq.(42).

Step 4. update the value of Z^m such as $Z^m = Z_{LR}(\lambda^{m+1})$ and construct the feasible solution associated with a value of Z^l according to heuristic mentioned in Subsection 4.6.3. If $Z^l < Z^u$, then let $Z^u = Z^l$. Go to *Step* 2.

4.7 Experiment and Computational Results

In this section, the computational experiments are carried out to test the performance of the proposed heuristic. The heuristic is coded in JAVA language and implemented with two Intel core 2 processors operating at 2.80 GHz clock speed and 4Gb RAM. As a comparison, we propose a simple branch and bound approach (B&B) for the small size problems, and a lower bound based on the Lagrangian relaxation method for the large size problems.

4.7.1 Random instances with small sizes

We construct 5 random instances with small sizes based on the following parameter settings. The processing time of each job on the single machine was 1. The setup costs and setup times were randomly generated from the uniform distributions over the intervals [800,1000] and [5,10], respectively. The round trip delivery costs and delivery times were randomly generated from uniform distributions over the intervals [800,1000] and [100,200], respectively. The vehicle's capacity is 5. The unit inventory cost in the production and customer area are respectively 0.5 and 0.8. The due date associated with job j, $j = 1,2,\ldots,n$, was generated using $d_j = 1000 + \sum_{i=0}^{j}(rand(0,1]) \times 19 + 11$.

For each combination, we randomly generated 50 problem instances and took the average value (Avg.Value) and the average cpu time (Avg.Cpu), which are defined in the Subsection 4.7.2, for the performance test of the heuristic.

As a comparison, we proposed a basic B&B approach for exactly solving the small size problems. The main idea of the B&B approach can be divided into the following two steps: (1) explore the possible delivery solutions for each level k, $1 \leqslant k \leqslant n$; and (2) generate all the potential production schemes according to M-Method. In the proposed B&B, we define the initial upper bound as the evaluation of the current solution and it will be updated whenever a feasible solution with a lower evaluation than the current upper bound is obtained. The exploration of the current solution is stopped if its partial evaluation is larger than the evaluation of the best solution found so far, and subsequently another branch of the solution space is explored.

We run the B&B solver and the heuristic using the five instances and the results are shown in Table 12. We observe that our heuristic runs much faster than the B&B solver. Although the B&B solver finds the optimal solution, the computational time of B&B solver increases exponentially as the instance size increases. The computational time of the proposed heuristic is very short, and the heuristic can obtain optimal or near-optimal solutions for all the small size problem instances.

Table 12. *Results of random instances with small sizes.*

Instance	Size	B&B		Heu	
No.	(n)	Avg.Value	Avg.Cpu (s)	Avg.Value	Avg.Cpu (s)
1	5	5618.01	0.01	5618.01	0.02
2	10	6848.23	0.16	6848.23	0.20
3	15	7914.46	2.36	7918.17	0.23
4	20	8704.73	61.72	8704.73	0.28
5	25	8834.11	397.34	8834.11	8.10

4.7.2 Random instances with large sizes

To test the performance of the heuristic algorithm thoroughly, in this section we conducted experiments using random instances with large problem sizes.

We considered three scenarios where the transporter capacities were respectively n, $n/2$ and $n/5$. For each scenario, we consider three situations where setup costs were randomly generated from the uniform distributions over the intervals [800,1000], [1000,1200], [1200,1500], respectively. For each situation, we considered three cases with small, middle and large round trip delivery costs, which were randomly generated from the uniform distributions over the intervals [800,1000], [1000,1200] and [1200,1500], respectively.

Moreover, for each case, we set the number of jobs as 30, 50, 70 and 100. The job processing time, setup time and round trip delivery time were randomly generated from the uniform distributions over the intervals [1,10], [5,10] and [100,200], respectively. The capacity of the vehicle is n, $n/2$ and $n/5$. The unit inventory costs in the production stage β_1 and customer area β_2 are generated from a discrete uniform distribution over the interval [0.5,1.0] and [1,1.5], respectively. Here, we set the unit holding cost in the customer area a little higher than that in the production area in order to truly reflect the real case. Since the inventory cost represents a combination of the cost of capital, the cost of physical storage and the cost of losses due to spoilage, it greatly depends on the inventory type. Moreover, the value of the production is added according to the supply chain from the original material supplier to the last customer in

the supply chain. Therefore, it is logical to assume that the unit customer inventory cost is a little higher than the unit inventory cost incurred in the production area. The due date associated with job j, $j = 1,2,\ldots,n$, was generated by $d_j = 1000 + \vartheta \times \sum_{i=0}^{j}(rand(0,1]) \times 80 + 80$, where ϑ is a parameter used to control the variation of the due date. Here, we set the parameter ϑ such as $\vartheta = n/50$.

Considering the different setup costs, delivery costs and number of jobs, we tested 36 situations for each scenario of the problem. For each situation, we randomly generated 50 problem instances. Since it is difficult to obtain an optimal solution in reasonable computing time even for the situation with 30 jobs, we evaluated the proposed heuristic using the lower bounds generated by the Lagrangian relaxation method mentioned in Section 4.6. The error ratio of a solution is defined as $ER = (Heu - LB)/LB$, where Heu denotes the evaluation of the solution generated by the proposed heuristic, LB denotes the lower bound. The average error ratio is defined as Avg.ER=$(\sum ER)/number\ of\ instances\ tested\ for\ a\ parameter\ combination$. The average running time is defined by Avg.Cpu=$\sum(running\ time\ of\ an\ instance)/number\ of\ instances\ tested\ for\ a\ parameter\ combination$.

The computational results are shown in Tables 13-15. The results reveal that, for each parameter combination, the average error ratios appear in an increasing trend as the number of jobs increases. They also indicate that when the delivery cost is fixed in a certain variation range and the variation range of setup cost increases, the average error ratios of the solutions for the problems with same jobs appear in a decreasing trend. When the setup cost and delivery cost are generated from respectively [1200,1500] and [800,1000], the average error ratio of each case is smaller than that of other corresponding cases (with same number of jobs). This can be interpreted as follows: when the setup costs are relatively larger than delivery cost, the heuristic works like the Lagrangian relaxation method in that both the two methods build the production schedules according to the same principle, which is the number of production batches should be as small as possible in order to reduce the setup cost; this may result in small gaps between the lower bounds and heuristic solutions. Furthermore, when the variation range of the setup cost is fixed in a certain variation range, we observe that average error ratios fluctuate slightly as the variation range of delivery cost increases. This indicates that the error ratios are not sensitive to

the changes in the delivery cost.

Table 13. *Results of random instances with large sizes for $c = n$*

s_c	Size	$\eta_c \in [800, 1000]$		$\eta_c \in [1000, 1200]$		$\eta_c \in [1200, 1500]$	
	(n)	Avg.ER(%)	Avg.Cpu(s)	Avg.ER(%)	Avg.Cpu(s)	Avg.ER(%)	Avg.Cpu(s)
[800, 1000]	30	1.41	1.17	2.61	0.95	1.81	0.76
	50	4.26	14.98	3.91	7.41	3.64	3.87
	70	5.81	72.47	5.43	21.83	6.50	18.96
	100	11.71	101.05	10.52	147.18	9.62	213.25
[1000, 1200]	30	1.40	1.27	1.59	0.89	1.77	2.24
	50	3.87	7.19	3.88	19.10	3.44	3.16
	70	3.89	21.52	3.69	55.73	5.57	16.86
	100	9.08	118.64	8.54	94.02	9.54	273.51
[1200, 1500]	30	1.17	1.09	1.35	1.94	1.22	0.71
	50	2.23	12.48	2.67	9.12	2.89	3.18
	70	3.39	43.67	3.40	93.16	4.29	37.08
	100	6.92	139.57	7.44	258.82	7.35	106.29

With comparison of the results in Tables 13 and 14, we observe that when the transporter capacity decreases from n to $n/2$, the average error ratios of 26 instances increase. Similarly, by comparing the results in Tables 14 and 15, we observe that when the transporter capacity decreases from $n/2$ to $n/5$, the average error ratios of 27 instances increase. This indicates that when we modify the transporter capacity but fix the other parameters, even through most of the average error ratios appear in a general increasing trend as the transporter capacity decrease, there are also some counterexamples. One of the reasons for this may be explained as follows. The production subproblem derived by the lagrangian relaxation method is not affected by modification of the transporter capacity. However, the heuristic algorithm performance is influenced by the transporter capacity. Thus, when the transporter capacity decreases, the sub production scheme generated by solving the production subproblem becomes worse. Consequently, the lower bound becomes also worse, which may be the reason why the average ratios increases as

the transporter capacity decreases.

Table 14. *Results of random instances with large sizes for $c = n/2$*

s_c	Size	$\eta_c \in [800, 1000]$		$\eta_c \in [1000, 1200]$		$\eta_c \in [1200, 1500]$	
	(n)	Avg.ER(%)	Avg.Cpu(s)	Avg.ER(%)	Avg.Cpu(s)	Avg.ER(%)	Avg.Cpu(s)
[800, 1000]	30	1.58	1.94	1.76	1.85	1.61	4.34
	50	3.70	9.59	3.11	4.72	3.67	6.05
	70	6.14	10.14	5.82	19.16	7.38	18.24
	100	8.96	98.34	10.55	93.83	14.57	218.45
[1000, 1200]	30	1.57	2.34	1.58	2.27	1.84	2.60
	50	2.53	8.11	3.10	67.81	3.37	10.33
	70	5.83	113.40	5.03	158.18	6.01	21.82
	100	7.41	512.11	10.54	218.76	10.55	108.44
[1200, 1500]	30	1.33	34.61	1.61	3.29	1.37	1.49
	50	1.86	44.87	2.77	5.59	2.48	24.05
	70	3.43	26.58	4.13	47.17	4.70	78.25
	100	7.38	153.43	7.10	184.50	8.48	199.35

The computational results in Tables 13-15 also reveal that the running time of the proposed heuristic appears in an increasing trend as the number of jobs increases, and the average running time of the proposed heuristic for all the situations were no longer than 15 minutes; this indicates the advantage of the heuristic in the practical application. Moreover, the average error ratios of the heuristic for all the situations were no more than 15%, which indicates that the performance of the heuristic is good for the randomly generated problems. Thus, the computational results in the Tables 12-15 show that the proposed heuristic is able to obtain near-optimal and optimal solutions in a reasonable running time.

It should be noted that the behavior of the proposed heuristic is correlated with the test instance generation scheme. However, the test instance generation scheme could well reflect the real cases in some industries (e.g., iron industry, automotive components industry) in which the delivery cost is relatively larger than the inventory cost; thus the proposed heuristic is useful

Table 15. *Results of random instances with large sizes for $c = n/5$*

s_c	Size	$\eta_c \in [800, 1000]$		$\eta_c \in [1000, 1200]$		$\eta_c \in [1200, 1500]$	
	(n)	Avg.ER(%)	Avg.Cpu(s)	Avg.ER(%)	Avg.Cpu(s)	Avg.ER(%)	Avg.Cpu(s)
[800, 1000]	30	2.99	0.84	2.69	1.12	2.25	0.95
	50	4.85	20.45	5.18	3.91	5.46	3.36
	70	7.15	135.72	7.87	9.44	12.18	9.31
	100	8.91	222.65	11.59	85.70	12.42	89.32
[1000, 1200]	30	2.46	0.65	3.29	2.17	2.45	0.73
	50	4.19	18.54	3.60	10.06	3.05	4.81
	70	5.64	18.44	6.14	22.31	7.32	7.28
	100	7.56	93.54	8.82	111.56	10.14	198.43
[1200, 1500]	30	2.57	1.65	2.86	1.67	2.65	0.65
	50	2.98	25.19	3.63	4.80	3.55	4.61
	70	5.05	19.68	4.93	148.25	5.03	22.88
	100	5.51	178.46	6.69	147.12	8.17	624.48

for the practical applications with respect to its efficiency in computational time and solution quality.

4.8 Summary

This chapter studies a coordinated scheduling problem for a single-item, make-to-order supply chain system consisting of one manufacturer, one capacitated transporter and one customer. In particular, we assume the existence in the production stage of an inventory that functions as a buffer to balance the production rate and the transportation speed such that the production batch size will not be limited by the capacity of the vehicle. Moreover, it is assumed that a job which is finished before its departure date or arrives at the customer before its due date will incur a stage-dependent inventory cost (WIP inventory, finished-good inventory or customer inventory cost). Our objective is to find a joint schedule such that the total cost involving setup, inventory and

delivery costs is minimized. We first formulate the problem in a general way, and show some straightforward optimality properties for this general model. Then we derive a precise instance from the general model, and propose a heuristic for solving this precise instance. Finally, we analyze the ability of the proposed heuristic for finding good solutions within a reasonable time. For small size problems, we compare the heuristic with an exact algorithm (B&B), and for large size problems, we establish a lower bound on the objective value using the Lagrangian relaxation method as a comparison. The results indicate the efficiency of the proposed heuristic in terms of both running time and solution quality.

The research has a number of limitations, however. The results of this paper can be only applied to single-product production situations and the finished products can be delivered to only one site by only one vehicle. Nevertheless, to the best of our knowledge, this paper represents the first attempt that allows differential schemes in production and delivery stages in the integrated scheduling research area. Therefore, the results may provide the basis for further studies of on this new integrated scheduling area.

CONCLUSIONS AND FUTURE RESEARCH DIRECTIONS

In this thesis, we have considered two categories of integrated scheduling problems. One is Integrated Scheduling of Production-Distribution-Inventory problems (ISPDI problems) and the other is Integrated Scheduling of Production-Inventory-Distribution-Inventory problems (ISPIDI problems). In the first category of problem, the production and distribution are very closely connected and no finished product inventory is held between them; in the second category, the production and distribution are indirectly linked through an intermediate stage of finished product inventory which works as a buffer to balance the production rate and distribution speed. For each of the two categories of problems, we have estimated both the total logistics cost and the customer service level. The logistics cost is measured by actual expenses of operations, e.g. setup cost, WIP inventory holding cost, finished-product inventory cost, distribution cost and customer inventory cost. The customer service performance is expressed in terms of the due date or deadline of each job. In this thesis, the chapters 2 and 3 fall into the first category of problem while chapter 4 falls into the second category.

In the second chapter, we analyzed the integrated scheduling of production and distribution with arbitrary job volumes and distinct job due dates considerations. This problem (P1) has been shown to be NP-hard, and formulated as a mixed integer programming model. Then, an improved genetic algorithm has been proposed for solving this model. In order to evaluate the performance of the proposed genetic algorithm, a lower bound based on the classical bin-packing problem has also been proposed. Finally, we have analyzed the average-case and worst-case performances of the proposed genetic algorithm in terms of both solution quality and computational time. Based on the consideration that the inventory cost depends much on the product itself, the proposed model has been then extended to the model where each job is associated with a distinct unit inventory cost. We have formulated this extended problem (P2)

as a non-linear model, and proposed a Tabu-based method for solving it. Based on the lower bound generation method proposed for problem P2, we have analyzed the average-case and worst-case performances of the proposed Tabu-based method.

In the third chapter, we selected a supply chain environment which is composed of multiple supply links as the studied object. Particularly, we assumed that the production start dates of jobs in one supply link equal to the due dates of the jobs in its previous supply link. In each link of the supply chain, we studied an integrated scheduling problem of production and distribution. We have provided the NP-hardness proof for the problem through a reduction from the knapsack problem. Then a genetic algorithm and a dominance related dynamic programming approach have been developed for solving this model. Finally, by comparing with a lower bound, we have tested the performances of the two proposed algorithms.

In the fourth chapter, we studied the second category of problem where the production and distribution are indirectly linked through an intermediate stage of finished product inventory. In specific, we assumed that the intermediate stage worked as a buffer to balance the production rate and the distribution speed. The existence of the intermediate inventory allowed the jobs to be rescheduled for transportation process after completion on the machine. The proposed problem has been proved to be NP-hard by a reduction from the knapsack problem. We formulated the problem as a non-linear model in a general way and provided some properties. Based on the general model, we derived a special instance and provided an efficient property between the production and transportation schedules. Then, we developed a heuristic algorithm based on the property proposed above for solving the special instance. In order to evaluate the performance of proposed heuristic algorithm, we developed a basic branch and bound approach and a lower bound based on the lagrangian decomposition method. Finally, we have analyzed the average-case performance of the proposed heuristic algorithm in terms of both solution quality and computational time.

Contributions made in the thesis are threefold. Firstly, we have proposed various integrated production, inventory and transportation scheduling models that closely mirror practical supply chain operations in some environments. All the models studied in this thesis took account of the different stage-dependent inventory costs considerations. Particulary, to the best of our

knowledge, the integrated model studied in Chapter 4 represents the first attempt that allowed the existence, between the production and distribution stage, of an intermediate inventory which worked as a buffer for resequencing and rebatching the jobs after completion on the machine for transportation process. Secondly, we provided some optimal properties for these models, and the NP-hardness proofs for the problems studied in chapters 3 and 4. Thirdly, we have developed various computationally effective heuristic algorithms for solving these models. Our solution approaches can be used as decision tools by practitioners in the real-world applications.

Limitations and Future Research Directions

The thesis has a number of limitations however. Because of the difficulties in obtaining real-situation data, in the experiment part of each problem under study, we evaluated the performance the proposed models and corresponding algorithms using the randomly generated problem instances. Even through the proposed models and corresponding algorithms have been proved to be efficient by these randomly generated test instances, their performances should still be evaluated by the corresponding real situations before applying the models and corresponding algorithms to practice. This study only considered one customer in the proposed models, i.e., the results can be only applied to the situations with single customer, which narrows their range of application.

There are many interesting extensions to this work worthy of studying. We have not explored routing options in models that involve more than one supplier or more than one customer. In all the models studied in this thesis, it has been assumed that only one transportation mode is available for distribution of product. Therefore, it will be interesting to introducing routing options and transportation mode decisions into the models. For some models, such as models studied in chapters 3, 4, we only studied the single product situation, it is interesting to extend the single product models to multiple product models. In all models studied in this thesis, the customer service is expressed in terms of the deadline of each job, i.e., each job must be delivered to customer before its deadline. It is worth introducing the customer service measurement into the objective function. For example, the objective function could be the sum of total joint cost and

the makespan. Finally, it is also worth extending the models studied in the chapters 2 and 4 for a supply chain environment which is composed of multiple supply links.

BIBLIOGRAPHY

Agnetis, A., Hall, N. G., and Pacciarelli, D. (2006). "Supply chain scheduling: Sequence coordination". *Discrete Applied Mathematics*, 154(15):2044–2063.

Allahverdi, A., Gupta, J. N. D., and Aldowaisan, T. (1999). "A review of scheduling research involving setup considerations". *Omega*, 27(2):219–239.

Allahverdi, A., Ng, C. T., Cheng, T. C. E., and Kovalyov, M. Y. (2008). "A survey of scheduling problems with setup times or costs". *European Journal of Operational Research*, 187(3):985–1032.

Baker, K. R. and Scudder, G. D. (1990). "Sequencing with earliness and tardiness penalties: a review". *Operations Research*, 38(1):22–36.

Baptiste, P. (2000). "Batching identical jobs". *Mathematical Methods of Operation Research*, 52(1):355–367.

Behnamian, J., Ghomi, S. M. T. F., Jolai, F., and Amirtaheri, O. (2012). "Minimizing makespan on a three-machine flowshop batch scheduling problem with transportation using genetic algorithm". *Applied Soft Computing*, 12(2):768–777.

Bhatnagar, R. and Chandra, P. (1993). "Models for multi-plant coordination". *European Journal of Operational Research*, 67(2):141–160.

Buer, M. G. V., Woodruff, D. L., and Olson, R. T. (1999). "Solving the medium newspaper production/distribution problem". *European Journal of Operational Research*, 115(2):237–253.

Chandra, P. and Fisher, M. L. (1994). "Coordination of production and distribution planning". *European Journal of Operational Research*, 72(3):503–517.

Chang, Y. C. and Lee, C. Y. (2004). "Machine scheduling with job delivery coordination". *European Journal of Operational Research*, 158(2):470–487.

Chen, B. and Lee, C. Y. (2008). "Logistics scheduling with batching and transportation". *European Journal of Operational Research*, 189(3):871–876.

Chen, L., Bostel, N., DEJAX, P., Cai, J. G., and Xi, L. F. (2007). "A tabu search algorithm for the integrated scheduling problem of container handing systems in a maritime terminal". *European Journal of Operational Research*, 181(1):40–58.

Chen, Z. L. (1996). "Scheduling and common due date assignment with earliness-tardiness penalties and batch delivery costs". *European Journal of Operational Research*, 93(1):49–60.

Chen, Z. L. (1997). "Scheduling with batch setup times and earliness-tardiness penalties". *European Journal of Operational Research*, 96(3):518–537.

Chen, Z. L. (2004). *Handbook of Quantitative Supply Chain Analysis: Modeling in the E-Business Era*. Kluwer Academic Publishers, Norwell, MA, USA.

Chen, Z. L. (2010). "Integrated production and outbound distribution scheduling: Review and extensions". *Operations Research*, 58(1):130–148.

Chen, Z. L. and Powell, W. B. (1999). "A column generation based decomposition algorithm for a parallel machine just-in-time scheduling problem". *European Journal of Operational Research*, 116(1):220–232.

Chen, Z. L. and Vairaktarakis, G. L. (2005). "Integrated scheduling of production and distribution operations". *Management Science*, 51(4):614–628.

Cheng, T. C. E., Gordon, V. S., and Kovalyov, M. Y. (1996). "Single machine scheduling with batch deliveries". *European Journal of Operational Research*, 94(2):277–283.

Cheng, T. C. E. and Gupta, M. C. (1989). "Theory and methodology survey of scheduling research involving due date determination decision". *European Journal of Operational Research*, 38(2):156–166.

Cheng, T. C. E., Kovalyov, M. Y., and Lin, B. M. T. (1997). "Single machine scheduling to minimize batch delivery and job earliness penalties". *SIAM Journal on Optimization*, 7(2):547–559.

Cheng, T. C. E. and Sin, C. C. S. (1990). "A state-of-the-art review of parallel-machine scheduling research". *European Journal of Operational Research*, 47(3):271–292.

Cheng, T. C. E. and Wang, X. L. (2010). "Machine scheduling with job class setup and delivery considerations". *Computers and Operations Research*, 37(6):1123–1128.

De, P., Ghosh, J. B., and Wells, C. E. (1990). "Scheduling about a common due date with earliness and tardiness penalties". *Computers and Operations Research*, 17(2):231–241.

Devapriya, P., Ferrell, W., and Geismar, N. (2006). *"Optimal fleet size of an integrated production and distribution scheduling problem for a perishable product"*. Working Paper, Clemson University, Clemson, SC.

Dobson, G. and Yano, C. A. (1994). "Cyclic scheduling to minimize inventory in a batch flow line". *European Journal of Operational Research*, 75(2):441–461.

Drexl, A. and Kimms, A. (1997). "Lot sizing and scheduling-Survey and extensions". *European Journal of Operational Research*, 99(2):221–235.

Eksioglu, S. D. (2002). *"Optimizing integrated production, inventory and distribution problems in supply chains"*. P.H.D. Thesis, University of Florida, Florida, USA.

Elmahi, I., Grunder, O., and Moudni, A. E. (2006). "A modelling-optimization approach for discrete event systems using the (max+) algebra and genetic algorithms". *International Journal of Innovative Computing, Information and Control*, 2(4):771–788.

Fandel, G. and Hegene, C. S. (2006). "Simultaneous lot sizing and scheduling for multi-product multi-level production". *International Journal of Production Economics*, 104(2):308–316.

Ferretti, I., Zanoni, S., and Zavanella, L. (2006). "Production-inventory scheduling using ant system metaheuristic". *International Journal of Production Economics*, 104(2):317–326.

Fisher, M. L. (1981). "The lagrangian relaxation method for solving integer programming problems". *Management Science*, 27(1):1–18.

Fleischmann, B. (1990). "The discrete lot-sizing and scheduling problem". *European Journal of Operational Research*, 44(3):337–348.

Fleischmann, B. (1994). "The discrete lot-sizing and scheduling problem with sequence-dependent setup costs". *European Journal of Operational Research*, 75(2):395–404.

Frangioni, A. (2005). "About lagrangian methods in integer optimization". *Annals of Operations Research*, 139(1):163–193.

Fumero, F. and Vercellis, C. (1999). "Synchronized development of production, inventory, and distribution schedules". *Transportation Science*, 33(3):330–340.

Gen, M. and Cheng, R. (1997). *"Genetic algorithms and engineering design"*. Wesley-Interscience, New York.

Glover, F. (1989). "Tabu search, Part 1". *ORSA Journal on Computing*, 1(3):190–206.

Glover, F. (1990). "Tabu search, Part 2". *ORSA Journal on Computing*, 2(1):4–32.

Glover, F., Jones, G., Karney, D., Klingman, D., and Mote, J. (1979). "An integrated production, distribution, and inventory planning system". *Interfaces*, 9(5):21–35.

Goldberg, D. H. (1989). *"Genetic algorithms in search, optimization and machine learning"*. MA: Addison-Wesley, Boston.

Gong, H. and Tang, L. X. (2011). "Two-machine flowshop scheduling with intermediate transportation under job physical space consideration". *Computers and Operations Research*, 38(9):1267–1274.

Gordon, V., Proth, J. M., and Chu, C. B. (2002). "A survey of the state-of-the-art of common due date assignment and scheduling research". *European Journal of Operational Research*, 139(1):1–25.

Grunder, O. (2010). "Lot sizing, delivery and scheduling of identical jobs in a single-stage supply chain". *International Journal of Innovative Computing, Information and Control*, 6(8):3657–3668.

Hahm, J. and Yano, C. A. (1992). "The economic lot and delivery scheduling problem: the single item case". *International Journal of Production Economics*, 28(2):235–252.

Hall, N. G., Kubiak, W., and Sethi, S. P. (1991). "Earliness-tardiness scheduling problems, 2: Deviation of completion times about a restrictive common due date". *Computers and Operations Research*, 39(5):847–856.

Hall, N. G., Lesaoana, M. A., and Potts, C. N. (2000). "Scheduling with fixed delivery dates". *Operations Research*, 49(1):134–144.

Hall, N. G. and Potts, C. N. (2003). "Supply chain scheduling: Batching and delivery". *Operations Research*, 51(4):566–584.

Hanczar, P. (2010). "An inventory-distribution system with LTL deliveries-Mixed Integer Approach". *Computers and Chemical Engineering*, 34(10):1705–1718.

Hassin, R. and Shani, M. (2005). "Machine scheduling with earliness, tardiness and non-execution penalties". *Computer and Operations Research*, 32(3):683–705.

Herrmann, J. W. and Lee, C. Y. (1993). "On scheduling to minimize earliness-tardiness and batch delivery costs with a common due date". *European Journal of Operational Research*, 70(3):272–288.

Hertz, A. and Widmer, M. (1996). "An improved tabu search approach for solving the job shop scheduling problem with tooling constraints". *Discrete Applied Mathematics*, 65(1-3):319–345.

Holland, J. H. (1975). *"Adaptation in natural and artificial systems"*. Ann Arbor, The University of Michigan Press.

Iyer, S. K. and Saxena, B. (2004). "Improved genetic algorithm for the permutation flow shop scheduling problem". *Computers and Operations Research*, 31(4):593–606.

Kang, J. H. and Kim, Y. D. (2010). "Coordination of inventory and transportation managements in a two-level supply chain". *International Journal of Production Economics*, 123(1):137–145.

Karp, R. M. (1972). *Reducibility among combinatorial problems, complexity of computer computations*. Plenum Press, USA.

Koulamas, C. (2010). "The single-machine total tardiness scheduling problem: Review and extensions". *European Journal of Operational Research*, 202(1):1–7.

Lauff, V. and Werner, F. (2004). "Scheduling with common due date, earliness and tardiness penalties for multi-machine problems: A survey". *Mathematical and Computer Modelling*, 40(5-6):637–655.

Lee, C. Y. and Chen, Z. L. (2001). "Machine scheduling with transportation considerations". *Journal of Scheduling*, 4(1):3–24.

Lee, I. (2001). "Artificial intelligence search methods for multi-machine two-stage scheduling with due date penalty, inventory, and machining costs". *Computers and Operations Research*, 28(9):835–852.

Lee, I. S. and Yoon, S. H. (2010). "Coordinated scheduling of production and delivery stages with stage-dependent inventory holding costs". *Omega*, 38(6):509–521.

Lejeune, M. A. (2006). "A variable neighborhood decomposition search method for supply chain management planning problems". *European Journal of Operational Research*, 175(2):959–976.

Levner, E., Kats, V., Pablo, D. A. L. D., and Cheng, T. C. E. (2010). "Complexity of cyclic scheduling problems: A state-of-the-art survey". *Computers and Industrial Engineering*, 59(2):352–361.

Li, C. L. and Ou, J. W. (2005). "Machine scheduling with pickup and delivery". *Naval Research Logistics*, 52(7):617–630.

Li, C. L., Vairaktarakis, G., and Lee, C. Y. (2005). "Machine scheduling with deliveries to multiple customer locations". *European Journal of Operational Research*, 164(1):39–51.

Li, S. S. and Yuan, J. J. (2009). "Scheduling with families of jobs and delivery coordination under job availability". *Theoretical Computer Science*, 410(47-49):4856–4863.

Li, S. S., Yuan, J. J., and Fan, B. Q. (2011). "Unbounded parallel-batch scheduling with family jobs and delivery coordination". *Information Processing Letters*, 111(12):575–582.

Liu, C. H. (2011). "Using genetic algorithms for the coordinated scheduling problem of a batching machine and two-stage transportation". *Applied Mathematics and Computation*, 217(24):10095–10104.

Liu, P. H. and Lu, X. W. (2011). "An improved approximation algorithm for single machine scheduling with job delivery". *Theoretical Computer Science*, 412(3):270–274.

Liu, S. G. (2003). *"On the integrated production, inventory and distribution routing problem"*. P.H.D. Thesis, The State University of New Jersey, New Jersey, USA.

Lu, L. F., Yuan, J. J., and Zhang, L. Q. (2008). "Single machine scheduling with release dates and job delivery to minimize the makespan". *Theoretical Computer Science*, 393(1-3):102–108.

Martello, S. and Toth, P. (1990). "Lower bounds and reduction procedures for the bin-packing problem". *Discrete Applied Mathematics*, 28(1):59–70.

Mazdeh, M. M., Sarhadi, M., and Hindi, K. S. (2007). "A branch-and-bound algorithm for single-machine scheduling with batch delivery minimizing flow times and delivery costs". *European Journal of Operational Research*, 183(1):74–86.

Mazdeh, M. M., Shashaani, S., Ashouri, A., and Hindi, K. S. (2011). "Single-machine batch scheduling minimizing weighted flow times and delivery costs". *Applied Mathematical Modelling*, 35(1):563–570.

Min, J., He, Y., and Cheng, T. C. E. (2007). "Batch delivery scheduling with batch delivery cost on a single machine". *European Journal of Operational Research*, 176(2):745–755.

Mouret, S., Grossmann, I. E., and Pestiaux, P. (2011). "A new lagrangian decomposition approach applied to the integration of refinery planning and crude-oil scheduling". *Computers and Chemical Engineering*, 35(12):2750–2766.

Mndez, C. A., Cerda, J., Grossmann, I. E., Harjunkoski, I., and Fahl, M. (2006). "State-of-the-art review of optimization methods for short-term scheduling of batch processes". *Computers and Chemical Engineering*, 30(6-7):913–946.

Neiro, S. M. S. (2006). "Lagrangean decomposition applied to multiperiod planning of petroleum refineries under uncertainty". *Latin American Applied Research*, 36(4):213–220.

Ouenniche, J. and Boctor, F. F. (2001). "The two-group heuristic to solve the multi-product, economic lot sizing and scheduling problem in flow shops". *European Journal of Operational Research*, 129(3):539–554.

Pan, J. C. H., Chen, J. S., and Cheng, H. L. (2001). "A heuristic approach for single-machine scheduling with due dates and class setups". *Computers and Operations Research*, 28(11):1111–1130.

Panwalkar, S. S., Smith, M. L., and Seidmann, A. (1982). "Common due date assignment to minimize total penalty for the one machine scheduling problem". *Operations Research*, 30(2):391–399.

Pirkul, H. and Jayaraman, V. (1998). "A multi-commodity, multi-plant, capacitated facility location problem: formulation and efficient heuristic solution". *Computers and Operations Research*, 25(10):868–878.

Poon, P. W. and Carter, J. N. (1995). "Genetic algorithm crossover operators for ordering application". *Computers and Operations Research*, 22(1):135–147.

Potts, C. N. (1980). "Analysis of a heuristic for one machine sequencing with release dates and delivery times". *Operations Research*, 28(6):1436–1441.

Potts, C. N. and Kovalyov, M. Y. (2000). "Scheduling with batching: A review". *European Journal of Operational Research*, 120(2):228–249.

Potts, C. N. and Wassenhove, L. N. V. (1992). "Integrating scheduling with batching and lot-sizing: A review of algorithms and complexity". *Journal of the Operational Research Society*, 43(5):395–406.

Pundoor, G. and Chen, Z. L. (2005). "Scheduling a production-distribution system to optimize the tradeoff between delivery tardiness and total distribution cost". *Naval Research Logistics*, 52(6):571–589.

Pundoor, G. and Chen, Z. L. (2009). "Joint cyclic production and delivery scheduling in a two-stage supply chain". *International Journal of Production Economics*, 119(1):55–74.

Qi, X. (2005). "A logistics scheduling model: inventory cost reduction by batching". *Naval Research Logistics*, 52(4):312–320.

Qi, X. (2009). "Production scheduling with supply and delivery considerations to minimize the makespan". *European Journal of Operational Research*, 194(3):743–752.

Rabadi, G., Mollaghasemi, M., and Anagnostopoulos, G. C. (2004). "A branch-and-bound algorithm for the early/tardy machine scheduling problem with a common due-date and sequence-dependent setup time". *Computers and Operations Research*, 31(10):1727–1751.

Rodriguez, M. A. and Vecchietti, A. (2010). "Inventory and delivery optimization under seasonal demand in the supply chain". *Computers and Chemical Engineering*, 34(10):1705–1718.

Sarmiento, A. M. and Nagi, R. (1999). "A review of integrated analysis of production-distribution systems". *IIE Transactions*, 31(11):1061–1074.

Sawik, T. (2009). "Coordinated supply chain scheduling". *International Journal of Production Economics*, 120(2):437–451.

Seidmann, A., Panwalkar, S. S., and Smith, M. L. (1982). "Optimal assignment of due dates for a single processor scheduling problem". *International Journal of Production Research*, 19(4):393–399.

Selvarajah, E. and Steiner, G. (2009). "Approximation algorithms for the suppliers supply chain scheduling problem to minimize delivery and inventory holding costs". *Operations Research*, 57(2):426–438.

Soukhal, A., Oulamara, A., and Martineau, P. (2005). "Complexity of flow shop scheduling problems with transportation constraints". *European Journal of Operational Research*, 161(1):32–41.

Sung, C. S. and Kim, Y. H. (2002). "Minimizing makespan in a two-machine flowshop with dynamic arrivals allowed". *Computers and Operations Research*, 29(3):275–294.

Supithak, W., Liman, S. D., and Montes, E. J. (2010). "Lot-sizing and scheduling problem with earliness tardiness and setup penalties". *Computer and Industrial Engineering*, 58(3):363–372.

Tang, L. X. and Gong, H. (2008). "A hybrid two-stage transportation and batch scheduling problem". *Applied Mathematical Modelling*, 32(12):2467–2479.

Tang, L. X. and Gong, H. (2009). "The coordination of transportation and batching scheduling". *Applied Mathematical Modelling*, 33(10):3854–3862.

Tang, L. X., Liu, J. Y., Rong, A. Y., and Yang, Z. H. (2001). "A review of planning and scheduling systems and methods for integrated steel production". *European Journal of Operational Research*, 133(1):1–20.

Tang, L. X. and Liu, P. (2009a). "Flowshop scheduling problems with transportation or deterioration between the batching and single machines". *Computers and Industrial Engineering*, 56(4):1289–1295.

Tang, L. X. and Liu, P. (2009b). "Two-machine flowshop scheduling problems involving a batching machine with transportation or deterioration consideration". *Applied Mathematical Modelling*, 33(2):1187–1199.

Thomas, D. J. and Griffin, P. M. (1996). "Coordinated supply chain management". *European Journal of Operational Research*, 94(1):1–15.

Torabi, S. A., Ghomi, S. M. T. F., and Karimi, B. (2006). "A hybrid genetic algorithm for the finite horizon economic lot and delivery scheduling in supply chains". *European Journal of Operational Research*, 173(1):173–189.

Wang, G. Q. and Cheng, T. C. E. (2000). "Parallel machine scheduling with batch delivery costs". *International Journal of Production Economics*, 68(2):177–183.

Wang, H. F. and Wu, K. Y. (2004). "Hybrid genetic algorithm for optimization problems with permutation property". *Computers and Operations Research*, 31(14):2453–2471.

Wang, L. and Wang, G. Q. (2010). "Supply chain scheduling with deadlines". *International Conference on Advanced Management Science (ICAMS)*, Chengdu, China:662–655.

Wang, Q., Batta, R., and Szczerba, R. J. (2005). "Sequencing the processing of incoming mail to match an outbound truck delivery schedule". *Computers and Operations Research*, 32(7):1777–1791.

Wang, X. L. and Cheng, T. C. E. (2009a). "Heuristics for parallel-machine scheduling with job class setups and delivery to multiple customers". *International Journal of Production Economics*, 119(1):199–206.

Wang, X. L. and Cheng, T. C. E. (2009b). "Logistics scheduling to minimize inventory and transport costs". *International Journal of Production Economics*, 121(1):266–273.

Wang, X. L. and Cheng, T. C. E. (2009c). "Production scheduling with supply and delivery considerations to minimize the makespan". *European Journal of Operational Research*, 194(3):743–752.

Webster, S. T. and Baker, K. R. (1995). "Scheduling groups of jobs on a single machine". *Operations Research*, 43(4):692–703.

Xu, K. L., Feng, Z. R., and Jun, K. L. (2010). "A tabu-search algorithm for scheduling jobs with controllable processing times on a single machine to meet due-dates". *Computers and Operations Research*, 37(11):1924–1938.

Yang, X. (2000). "Scheduling with generalized batch delivery dates and earliness penalties". *IIE Transactions*, 32(8):735–741.

Yeung, W. K., Choi, T. M., and Cheng, T. C. E. (2011). "Supply chain scheduling and coordination with dual delivery models and inventory storage cost". *International Journal Production Economics*, 132(2):223–229.

Yuan, J. J. (1996). "A note on the complexity of single-machine scheduling with a common due date, earliness-tardiness, and batch delivery costs". *European Journal of Operational Research*, 94(1):203–205.

Zdrzalka, S. (1995). "Analysis of approximation algorithms for single-machine scheduling with delivery times and sequence independent batch setup times". *European Journal of Operational Research*, 80(2):371–380.

Zhao, Q. H., Chen, S., Leung, S. C. H., and Lai, K. K. (2010). "Integration of inventory and transportation decisions in a logistics system". *Transportation Research Part E: Logistics and Transportation Review*, 46(6):913–925.

Zhong, W. Y., Chen, Z. L., and Chen, M. (2010). "Integrated production and distribution scheduling with committed delivery dates". *Operations Research Letters*, 38(2):133–138.

Zhong, W. Y., Dosa, G., and Tan, Z. Y. (2007). "On the machine scheduling problem with job delivery coordination". *European Journal of Operational Research*, 182(3):1057–1072.