J Heuristics (2014) 20:511–537
DOI 10.1007/s10732-014-9249-y

Scheduling parallel machines with single vehicle delivery

Eray Cakici · Scott J. Mason · H. Neil Geismar · John W. Fowler

Received: 27 October 2010 / Revised: 4 May 2014 / Accepted: 13 May 2014 /
Published online: 30 May 2014

Abstract We investigate the integrated production and distribution scheduling problem in a supply chain. The manufacturer's production environment is modeled as a parallel machine system. A single capacitated vehicle is employed to deliver products in batches to multiple customers. The scheduling problem can also be viewed as either parallel machines with delivery considerations or a flexible flowshop. Different inventory holding costs, job sizes (volume or storage space required in the transportation unit), and job priorities are taken into account. Efficient mathematical modeling and near-optimal heuristic approaches are presented for minimizing total weighted completion time.

Keywords Supply chain scheduling · Mathematical modeling · Greedy heuristics · Local search

1 Introduction

In a typical supply chain, raw materials or sub-products are procured, and then goods are manufactured at one or more production facilities, shipped to warehouses or

E. Cakici (✉)
IBM, Istanbul, Turkey
e-mail: eray.cakici@gmail.com

S. J. Mason
Department of Industrial Engineering, Clemson University, Clemson, SC, USA

H. N. Geismar
Department of Information and Operations Management, Mays Business School,
Texas A&M University, College Station, TX, USA

J. W. Fowler
Department of Supply Chain Management, Arizona State University, Phoenix, AZ , USA

distribution centers for intermediate storage, and finally delivered to customers or retailers. Therefore, a supply chain is the system of suppliers, manufacturers, intermediate distribution and storage units, retailers, and customers, as well as raw materials, work-in-process inventory, and finished goods. Supply chain best practices should take into account the interaction and collaboration between these different components of the supply chain in order to achieve reduced costs and increased service levels (Simchi-Levi et al. 2003).

Managing supply chain systems with effective planning and scheduling solutions is one of the most challenging issues faced by both supply chain practitioners and academic researchers. In recent years, advancements in information technology—such as warehouse management systems (WMS) and transportation management systems (TMS)—have allowed companies to use integrated decision strategies in their production and distribution operations to achieve reduced logistics costs and better customer service. Traditionally, production and distribution decisions are made by different departments of a company or by different companies in the supply chain, with each attempting to achieve its own goals. The conflicts between these stages' goals can lead to excess inventory, unnecessary capacity utilization, overtime work hours, and higher lead times. Hence, total costs generally will be higher when decisions are made independently, as compared to decisions being coordinated for the entire supply chain. Because logistics costs can be a significant percentage of a product's total cost, improving supply chain processes may provide significant savings in the overall supply chain. A well-managed system should effectively control the flow of materials and the information sharing between these stages. We mainly focus on the coordinated scheduling of production and distribution functions. In the most general form of the problem, different products are scheduled in a production facility and then delivered to various customers within a distribution network in order to satisfy demand over the planning time horizon.

The integration of production and distribution can be handled in different ways. One typical example is manufacturing the products and sending them directly without any intermediate storage to retailers or customers by dedicated fleets. Each vehicle has to return back to the production facility after deliveries. As a practical example, we refer to the distribution of catering services, in which vehicles deliver different foods from the production facility directly to the customers. In addition, companies such as Tyson Foods' Mexican Original division receive multiple orders per week from and make multiple deliveries per week to Taco Bell warehouses. Such problems also arise in the distribution of perishable products such as milk, vegetables, and bread. Integrated production and distribution production strategies have been shown to be more efficient for these types of problems (Chen and Vairaktarakis 2005 and Geismar et al. 2008).

The basic scheduling problem is to find a feasible schedule by specifying a machine and an execution interval on that machine for each of the tasks, which are also called jobs, such that all constraints are met and the given objective function is optimized (Hoogeveen 2005). In many practical situations, scheduling involves batching decisions. Batch processing (both serial and parallel as identified by Hopp and Spearman (2000)) has been widely used in production, transportation, service, and warehousing operations to provide cheaper and faster processing of the jobs. A batch is a group of

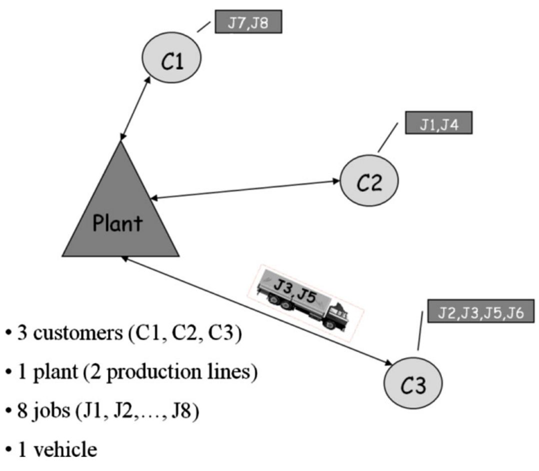

Fig. 1 Example problem instance

items (jobs) that are processed simultaneously on the same machine. There are two main decisions: allocating jobs to batches and then sequencing these batches. Since materials and products usually travel in batches throughout the supply chain, batch scheduling commonly occurs in supply chain scheduling management. Determining optimal batch formation can yield substantial savings by improving the utilization of limited resources. For example, consolidating less-than-full-truckload shipments into full-truckload shipments can reduce transportation costs significantly. However, some finished goods might be required to wait until all items of the corresponding batch finish their processing in the previous stage. Consequently, systems can be faced with increased inventory holding costs and lead times. Therefore, a solution should be found so that a compromise is achieved between the inventory holding costs, total system costs, service levels, and equipment utilizations.

In this paper, we study the problem of supply chain scheduling in an integrated production and distribution system. In a typical supply chain, the existence of multiple product types and customers leads to different inventory holding costs and job priorities. Companies generally prefer to operate with less Work-In-Process (WIP) and respond to customer orders as fast as possible. The objective of minimizing total weighted completion times can improve customer service levels while simultaneously considering the priority of different customers or product types. Completion time is defined as the delivery time of the orders to the customers. Orders are first processed by a production facility through multiple production lines and then delivered to the customers by a single capacitated vehicle. The system is modeled as a parallel machine-scheduling problem with job delivery considerations. (See Fig. 1 and Fig. 2 for an example problem instance and an example schedule representation depicting both production operations and subsequent distribution by the single vehicle.) We present a mathematical programming formulation and efficient heuristic approaches. We show how different characteristics of the problem (job weights, sizes, processing times, transportation times, etc.) can be handled effectively and also discuss which algorithms perform better depending on the system structure. One such example is the situation in which the transportation function is

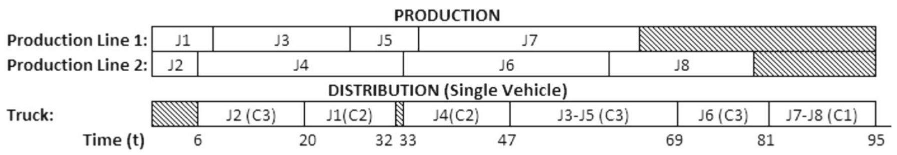

Fig. 2 Example schedule representation for motivating problem instance

the bottleneck, so it, rather than the production schedule, dominates the objective value.

The remaining sections of this paper are organized as follows. Section 2 presents a review of the literature relevant to our study. Section 3 contains the details regarding the optimization model, and Sect. 4 describes the development of heuristic solution approaches and a local search algorithm to further improve the performance of heuristics. After presenting experimental results in Sect. 5, we summarize our research findings and identify opportunities for further research in Sect. 6.

2 Literature review

Because the integrated production and distribution scheduling problem evolves from different research streams, we review the literature in both batch scheduling and supply chain scheduling.

2.1 Batch scheduling

An earlier review of the batch scheduling literature can be found in Potts and Kovalyov (2000). As related to supply chain scheduling, minimizing inventory holding costs can be achieved by minimizing total weighted completion time, where job weights are proportional to inventory holding costs. In this problem, a common setup time occurs for each scheduled batch, and each job's completion time is calculated as equal to the batch completion time. This problem has been proven to be NP-Hard in Albers and Brucker (1993).

In family scheduling models, only jobs belonging to the same family can be batched together, and a setup time is not required when the machine processes a job in the same family as its predecessor. Hence, there is only one setup per batch (Potts and Kovalyov 2000). Jobs of the same family can also be processed together simultaneously in some batching environments. One example is to batch together only jobs to be delivered to the same customer. Uzsoy (1995) considers a scheduling problem for a single batch processing machine with incompatible job families wherein only jobs of the same family can be batched together and all jobs included in a batch are processed together at the same time. Both the dynamic job arrivals case and the static case in which all jobs are available at the beginning of the planning horizon are considered. Cakici et al. (2013) study the dynamic job arrivals case on parallel batch processing machines to minimize total weighted completion time. A mathematical model and heuristics employing local search procedures under a variable neighborhood search scheme are presented. We extend this study by considering the delivery of jobs via a capacitated vehicle when the parallel machines needed for manufacturing are serial (i.e., do not process jobs in batches).

Dobson and Nambimadom (2001) study the problem of minimizing total weighted completion time on a single batch processing machine with jobs of different sizes. The problem is proven to be NP-Hard and a number of heuristics are proposed along with a lower bound achieved through an integer programming formulation. Koh et al. (2005) extend this study by considering different objective functions including makespan, total completion time, and total weighted completion time. Azizoglu and Webster (2001) propose a branch and bound algorithm for the problem of minimizing total weighted completion time in single batch machine scheduling with incompatible job families.

Flowshop scheduling problems containing batch-processing machines are a recent trend in the literature. Gonga et al. (2010) study a two-stage flowshop scheduling problem. The first stage is modeled as a single batch-processing machine subject to a blocking constraint, while the second stage contains a discrete machine with setup times. Although the problem is shown to be NP-hard for the objective of minimizing makespan, polynomial approximation algorithms are formulated for special cases of the problem. Lei and Wang (2011) consider multiple batch-processing machines to minimize maximum lateness. An effective neighborhood search algorithm is proposed and compared with a variable neighborhood search and a genetic algorithm. Motivated by the steel industry, Tang and Liu (2009) examine transportation between two stages of a flowshop: a batch-processing machine is followed by a single machine; a transporter is employed to deliver jobs to the latter. A mixed-integer programming formulation, a heuristic, and multiple polynomial algorithms are proposed for two different cases of the problem of interest. Finally, Behnamian (2012) investigate a three-machine flowshop wherein a batch-processing machine is located between two single processing machines. There are two vehicles carrying batches of jobs between the machines. A mixed-integer programming model and a genetic algorithm are developed to minimize makespan.

The distribution phase of our problem is also a variation of a batch scheduling problem in which capacitated vehicles replace the batching machines and jobs ordered by the same customer represent jobs belonging to same family. Practical assumptions of different job sizes (volumes) and different job priorities (weights) are also considered in our study.

2.2 Supply chain scheduling

The distribution scheduling problem via batch deliveries is extended by incorporating machine scheduling decisions required for the production side. Batch delivery problems require delivery considerations when scheduling the machines, so they arise in the integrated machine and distribution scheduling in a supply chain network (Mazdeh et al. 2007). After the processing of the jobs of a batch is complete, the batch is delivered to customers or to the next stage for further processing. Batch delivery scheduling in supply chain management problems affects inventory turnover and delivery service levels. Ji et al. (2007) have shown that integrated scheduling problem of a single-machine with batch delivery considerations to a single customer is NP-Hard. The objective studied is to minimize total weighted flow times and delivery costs. A pseudo-polynomial time dynamic programming algorithm is presented for the case in which a limited number of batches are available, and two polynomially-solvable special cases of the problem are presented. Wang and Cheng (2000) study batch delivery scheduling for jobs processed on parallel machines. A dynamic programming algorithm is developed to minimize total flow time and delivery cost. In both of the above mentioned papers, delivery cost depends on number of deliveries. Different transportation modes are incorporated into the model proposed by Chen and Lee (2008). Polynomial-time algorithms are developed for special cases of the problem and an approximation algorithm is provided for the general case. Zhong et al. (2007) consider the amount of storage space each job requires during transportation. Two scheduling problems are examined in which jobs are processed on either one or two machines. The objective is to minimize the makespan, given a single capacitated vehicle and transportation times. The authors present approximation algorithms with worst-case ratio analyses.

In this paper, we focus on the distribution of products from a single source. Similarly, Van Buer et al. (1999) study the newspaper delivery problem, in which vehicles deliver newspapers from the production facility to distribution centers. The newspaper industry is modeled as a just-in-time environment involving integrated production and distribution scheduling decisions. Hall and Potts (2003) examine several batch delivery scheduling problems while considering different supply chain configurations. For each problem, they either propose a dynamic programming algorithm or prove that the problem is intractable. This study is extended in Hall and Potts (2005) by considering transporter availability constraints. Mazdeh et al. (2007) present a branch and bound algorithm with the objective of minimizing the sum of flow times and delivery costs. It is assumed that there are an unlimited number of vehicles available, the transportation time is negligible, and vehicle capacities are infinite. Jobs are processed on a single machine and delivered in batches directly to customer locations. Selvarajah and Zhang (2014) assume that both transportation times and job weights warrant consideration, propose a genetic algorithm to minimize sum of total weighted flow times and delivery costs. Lee and Chen (2001) analyze the complexity of various machine scheduling problems by considering transportation times and capacities. Chen and Vairaktarakis (2005) consider different models in which multiple customers may be visited during the same trip. Transportation costs are considered together with either average delivery time or maximum delivery time. For each problem, an exact algorithm is proposed or the problem is proved to be NP-Hard and a heuristic algorithm is presented and worst-case performance analysis provided. Conflict and cooperation issues in supply chain scheduling systems are studied by Dawande et al. (2006) and by Chen and Hall (2007).

The majority of the published literature studies supply chain scheduling problems by developing optimal algorithms for special cases. There is a need for the development of efficient and effective heuristics for the general problem. Moreover, none of the research discusses the practical assumptions of this paper simultaneously.

3 Mathematical model

3.1 Problem statement

Globalization, increased competition, short product lifecycles, growing supply chain complexities, and market volatilities are driving most companies forward from the traditional "push" systems to demand-driven "pull" systems. Customers pull products directly from manufacturers based on their needs, instead of having manufacturers push products to customers or distribution centers. Time-sensitive fashion apparel product manufacturing is an example of such a system. Because there is a high risk of excess inventory in seasonal and fashionable products, manufacturers do not start production until all orders are received.

We study a demand-driven supply chain with one manufacturer P and k customers $K = \{1, \dots, k\}$. The manufacturer receives a set of n jobs (orders) $J = \{1, \dots, n\}$ from different customers at the beginning of the planning horizon. Then, in order to serve customers as soon as possible at a low cost, the manufacturer has to schedule the production and distribution operations in a coordinated and efficient manner. The manufacturer's production environment is modeled as a parallel machine system. Each job has to be processed on any of the m parallel machines $M = \{1, \dots, m\}$ at the manufacturer and then is delivered to the requesting customers by a single capacitated vehicle. Production scheduling decisions determine the assignment of jobs to machines and the machines' processing sequences. Distribution scheduling decisions specify the batching of jobs into trips on the single vehicle in which maximum number of possible trips is denoted by β and is equal to number of jobs.

We develop an integrated scheduling model such that production and distribution decisions are made in an interconnected manner. In the distribution part, only orders destined for the same customer can be batched together for delivery. Hence, the vehicle can only visit one customer in each trip. However, each customer can be served by multiple trips during the planning horizon. Jobs become available for delivery only when all jobs in the same batch have been processed. Most of the research in the literature either considers vehicle capacity to be infinite or assumes that every job requires the same amount of storage and thus defines vehicle capacity as the maximum number of jobs that can be carried simultaneously. Since this may not be the case in many supply chains, our problem involves different job sizes q_j (space or capacity required while delivering jobs). Split deliveries and preemption are not allowed. In order to account for different job priorities or inventory holding costs, weight w_j is introduced for every job $j = 1, \dots, n$. Finally, processing and transportation times are assumed to be deterministic.

3.2 Model formulation

This problem is formulated as a mixed-integer programming problem. First, a dummy job 0 is introduced whose processing time, ready time, and weight are each set equal to 0. In the network formulation of the parallel machine scheduling problem, job 0 is required to be both the first and the last job processed on each machine in order to indicate both the starting and finishing of job processing on each machine. In other words, there are m in and out arcs for node 0, whereas all other nodes are associated with only one in and out arc. Binary decision variable x_{ij} is defined to assist with job sequencing such that $x_{ij} = 1$ if job $i \in J$ immediately precedes job $j \in J$ on the same machine; otherwise, $x_{ij} = 0$. To keep track of the vehicle assignments and batching, binary decision variable y_{ib} is introduced, where $y_{ib} = 1$ if job $i \in J$ is included in batch $b \in B$; otherwise, $y_{ib} = 0$.

Ω_i is the customer with which job $i \in J$ is associated with. The vehicle has capacity δ, and t_j is the transportation time required to deliver job $j \in J$. The time at which job $j \in J$ finishes its required processing is denoted by C_j, and the time job $j \in J$ finishes its delivery is denoted by D_j. The batches (trips) are numbered in ascending order of their delivery starting times S_b and denoted by $B = \{1, \ldots, \beta\}$. At this stage the customer served by each trip is not known and transportation times associated with trips depend on jobs assigned to them through mathematical model results. Therefore, a decision variable υ_b is defined as the time required to deliver batch $b \in B$ whereas transportation times required to deliver jobs are known parameters. The objective function is to minimize the sum of total weighted delivery time (*TWD*) of all jobs, where $TWD = \sum_{j:j\in J} w_j D_j$. Jobs are assigned to one of the m available production machines, subject to each machine starting and ending its schedule with job 0:

$$\sum_{j\in J:j\neq 0} x_{0j} \leq m \tag{1}$$

$$\sum_{j\in J:j\neq 0} x_{j0} \leq m \tag{2}$$

Each job has a unique predecessor and a unique successor:

$$\sum_{j\in J:j\neq i} x_{ji} = 1 \qquad \forall i \in J : i \neq 0 \tag{3}$$

$$\sum_{j\in J:j\neq i} x_{ij} = 1 \qquad \forall i \in J : i \neq 0 \tag{4}$$

In order to process job j immediately after job i on the same machine, job $i \in J$ completes p_j time units before job $j \in J$:

$$C_j \geq C_i + p_j - M(1 - x_{ij}) \quad \forall i \in J, \forall j \in J : j \neq 0, i \neq j \tag{5}$$

In constraint (5), $M \geq \sum_{j\in J} p_j$.

Constraints (6)–(12) address the distribution part of the problem. Firstly, each job should be assigned to a batch:

$$\sum_{b\in B} y_{ib} = 1 \, \forall i \in J \tag{6}$$

The transportation time required to deliver a batch depends on the jobs included in that batch and a batch does not have a transportation time if no job is assigned to it:

$$\upsilon_b \geq t_i - M(1 - y_{ib}) \qquad \forall i \in J, \forall b \in B : i \neq 0 \tag{7}$$

D_i is the delivery start time plus the delivery time:

$$D_i \geq S_b + \upsilon_b - M(1 - y_{ib}) \qquad \forall i \in J, \forall b \in B : i \neq 0 \tag{8}$$

Batch delivery starts after the preceding batch completes its delivery and returns:

$$S_{b+1} \geq S_b + 2\upsilon_b \qquad \forall b \in B : b < \beta \tag{9}$$

A vehicle cannot start its delivery until all jobs to be delivered in the corresponding batch have finished their processing:

$$S_b \geq C_i - (1 - y_{ib})M \qquad \forall i \in J, \forall b \in B : i \neq 0 \tag{10}$$

In constraints (7) and (9), $M \geq \sum_{j \in J} p_j + \sum_{j \in J} 2t_j$.

A delivery can serve at most one customer:

$$y_{ib} + y_{jb} \leq 1 \qquad \forall i \in J, j \in J, b \in B : \Omega_i \neq \Omega_j \tag{11}$$

Vehicle capacity cannot be exceeded:

$$\sum_{i \in J} q_i y_{ib} \leq \delta \qquad \forall b \in B \tag{12}$$

In order to improve the tractability of the network formulation, constraints (13)–(16) are additional valid inequalities that can be added to the model in concert with the introduction of another binary decision variable e_{ij}, where $e_{ij} = 1$ if job $i \in J$ finishes its processing before job $j \in J$ starts its processing at the same machine; otherwise, $e_{ij} = 0$. Valid inequalities are valid for feasible regions of the problem studied and can lead faster solutions basically by cutting down the feasible region of the LP relaxation. Constraint (16) ensures that completion time of a job is at least the total processing times of the jobs scheduled earlier on the same machine and processing time of itself:

$$x_{ij} \leq e_{ij} \; \forall i \in J, \forall j \in J : j \neq 0, i \neq 0, i \neq j \tag{13}$$

$$e_{ki} \leq e_{kj} + \left(1 - x_{ij}\right) \; \forall i \in J, \forall j \in J, \forall k \in J : j \neq 0, i \neq 0, k \neq 0, i \neq j, j \neq k, k \neq i \tag{14}$$

$$e_{kj} \leq e_{ki} + \left(1 - x_{ij}\right) \; \forall i \in J, \forall j \in J, \forall k \in J : j \neq 0, i \neq 0, k \neq 0, i \neq j, j \neq k, k \neq i \tag{15}$$

$$C_i \geq p_i + \sum_{j \in J : i \neq j} p_j \left(e_{ji}\right) \; \forall i \in J : i \neq 0 \tag{16}$$

3.3 Problem complexity

By assuming the transportation time to be $t_j = 0$, $j \in J$, the problem reduces to the parallel machine problem $Pm \| \sum_{w_j C_j}$ in the scheduling notation scheme of Graham et al. (1979), which is NP hard even if preemptions are allowed (Bruno et al. 1974 and Lenstra et al. 1977). Therefore, we develop heuristics to produce near-optimal solutions in a reasonable amount of computation time. The optimization model above is used to assess our heuristics' solution quality for small problem instances.

4 Heuristics

The distribution part of the problem can be viewed as scheduling a single batch-processing machine with incompatible job families, where the objective is to minimize total weighted completion times. Dobson and Nambimadom (2001) and Koh et al. (2005) study this problem by considering the different job sizes and proposing heuristic approaches. Our problem also involves the production part, which is a parallel machine environment. The problem can be decomposed into sub-problems: machine scheduling, batch assignment and batch sequencing. Since only one vehicle exists, batch assignment and sequencing have a great impact on the production schedules. Therefore, heuristic approaches follow these steps:

1. **Batch (Trip) Assignment:** Building delivery batches for each customer
2. **Batch Sequencing:** Sequencing the delivery batches
3. **Job Sequencing:** Assigning jobs to machines and sequencing them

In such a back-to-front approach, batch assignment and scheduling provide due dates for the production scheduling. On the other side, in a front to back approach, production-scheduling decisions supply ready times for the delivery-batching problem if production decisions are made before distribution decisions.

4.1 Batch (trip) assignment

Five different heuristics are used to assign jobs to batches:

1. **Greedy:** As in greedy heuristic proposed in Dobson and Nambimadom (2001), jobs are sorted in non-increasing order of their weight-to-size ratio, then each job is assigned to the first available batch of its customer that has enough space to accommodate it. If no batches are available, a new batch is created for the corresponding customer.
2. **Knapsack:** Another method is developed by Dobson and Nambimadom (2001). Through the standard dynamic programming algorithm of Nemhauser and Wolsey (1998), a knapsack problem is solved to form each batch. The weight of the first batch is maximized and then, to maximize the weight of the next batch, the knapsack problem is solved with the remaining jobs until all jobs are assigned to a batch.
3. **BWFBWS** *(Biggest weight, first fit batching, and batch weighted shortest)*: In Koh et al. (2005), the greedy heuristic of Dobson and Nambimadom (2001) is slightly changed so that jobs are sorted in non-increasing order of their weight instead of weight-to-size ratio.
4. **BA1**: In order to examine the impact of job processing times on heuristic performances for the overall problem, jobs are sorted in non-increasing order of weight-to-processing time ratio. All other steps remain the same as the greedy heuristic.

5. **BA2**: While sorting jobs, it can be also useful to consider both processing times and job sizes concurrently. With this in mind, a new sorting index is introduced: $\frac{w_i}{\left(\frac{q_i}{\sum_{j\in A_i} q_j}+\frac{p_i}{\sum_{j\in A_i} p_j}\right)}$

In this index, $\frac{p_i}{\sum_{j\in A_i} p_j}$ is the normalized processing time of job $i \in J$, where $j \in A_i$ denotes that job $j \in J$ and job $i \in J$ are ordered by the same customer. Note that the total normalized processing time of jobs ordered by each customer is 1. Likewise, $\frac{q_i}{\sum_{j\in A_i} q_j}$ is the normalized size of job $i \in J$.

4.2 Batch sequencing

First, notation is introduced for the heuristic approaches of this phase. For convenience, let

ω_k	Total weight of the jobs delivered in batch (trip) k
T_k	Transportation time to deliver batch k and return to the plant
P_k	Sum of the processing times of the jobs in batch k
Γ_k	Longest processing time of the jobs in batch k
η	Number of identical parallel machines in the production environment

Batches are sequenced in four different ways:

1. **BS1**: Both Dobson and Nambimadom (2001) and Koh et al. (2005) sequence the batches by weighted shortest processing time of the batch in their studies. This is adapted to our problem as non-increasing order of ω_k/T_k

The following three heuristics also consider processing times to order batches.

2. **BS2**: Sequence batches in non-increasing order of $\omega_k/(\mathrm{T}_k + \mathrm{P}_k)$
3. **BS3**: Sequence batches in non-increasing order of $\omega_k/\left(\mathrm{T}_k + \frac{\mathrm{P}_k}{\eta}\right)$
4. **BS4**: Sequence batches in non-increasing order of $\omega_k/(\mathrm{T}_k + \Gamma_k)$

4.3 Job sequencing

Once batches are formed and sequenced, all jobs should be processed as soon as possible in order to make batches ready for delivery. Since batches become available for delivery only if all jobs in the same batch have been processed, one reasonable choice is to apply the longest processing time (LPT) algorithm, which yields good solutions quickly for the problem of minimizing makespan on parallel machines (Akyol and Bayhan 2006):

LPT Steps:

1. Sequence jobs of the next batch in non-increasing order of their processing time.
2. Assign the first job in the list to the first available machine and remove the job from the list.

4.4 Improving batch assignment using an accept/reject rule

Batch assignment approaches are developed with the idea of continuously adding jobs to the first available batch (trip) with enough capacity and to make the batches as large as possible in order to minimize the number of batches. However, in some cases this may cause unnecessary delays in job deliveries. A not-fully-loaded batch can be delivered earlier by not waiting a long time for the next job to complete its processing, and thus the truck can also be available earlier for the delivery of the next batch. Therefore, an accept/reject rule is presented to evaluate the quality of the batch in terms of the transportation and processing times required to process and deliver the jobs included. This rule is similar in purpose to the extended greedy ratio (EXGR) heuristic of Uzsoy (1994).

Before adding each job to the batch, it is ensured that doing so would improve batch quality, which is defined by the index used to sort batches in the Batch Sequencing phase. For example, adding a job can slightly increase the total weight of the batch, whereas the total waiting times can be increased significantly. Therefore, the batch-sequencing index based on the ratio of the total weight to the sum of transportation and processing times is not improved. A batch assignment approach can be modified as follows by applying an accept/reject check:

Batch Assignment with Accept/Reject Rule Steps:

1. Sort jobs in non-increasing order of a given index.
2. Assign the job to the first available batch of its customer only if the batch-sequencing index is improved by adding the job to the batch; otherwise, reject to be evaluated for a later batch and evaluate the next job in the list.

Due to the structure of the dynamic programming algorithm of the knapsack approach, the accept/reject rule cannot be applied to batch assignments using this method. On the other hand, job index BS1 (ω_k/T_k) is always improved when a new job is added to the batch. Therefore, any job will be accepted if there is enough capacity in the batch, and applying the accept/reject check will not have any impact on the performance. Preliminary experiments were conducted to assess the impact of including this accept-reject rule where applicable. On average a 5.1% improvement in objective function value is achieved by applying the rule before assigning jobs to batches with available capacity. As a result, we include the rule where applicable in the 20 heuristics approaches to be examined in Table 1.

4.5 Post-processing for improved solution quality via local search

An efficient $O(n^2)$ local search heuristic is developed to search for optimal/near-optimal solutions within a pre-specified neighborhood. The neighborhood is defined as adjacent pairwise interchanges of the batch sequencing orders. The search is started with a batch located in a later position of the schedule and aims to move that batch forward as long as an improvement is achieved in terms of *TWD*. Since batches are already sequenced effectively in the initial phase and considering the transitivity of the outcomes for the interchanges with batches ordered earlier, the local search is terminated for the associated batch when a move does not improve the solution. If a

Table 1 Heuristic descriptions

Heuristic	Batch (Trip) Assignment	Accept/reject rule?	Batch sequencing	Job sequencing
H1	Greedy	No	BS1	LPT
H2	Greedy	Yes	BS2	LPT
H3	Greedy	Yes	BS3	LPT
H4	Greedy	Yes	BS4	LPT
H5	Knapsack	No	BS1	LPT
H6	Knapsack	No	BS2	LPT
H7	Knapsack	No	BS3	LPT
H8	Knapsack	No	BS4	LPT
H9	BWFBWS	No	BS1	LPT
H10	BWFBWS	Yes	BS2	LPT
H11	BWFBWS	Yes	BS3	LPT
H12	BWFBWS	Yes	BS4	LPT
H13	BA1	No	BS1	LPT
H14	BA1	Yes	BS2	LPT
H15	BA1	Yes	BS3	LPT
H16	BA1	Yes	BS4	LPT
H17	BA2	No	BS1	LPT
H18	BA2	Yes	BS2	LPT
H19	BA2	Yes	BS3	LPT
H20	BA2	Yes	BS4	LPT

move reduces *TWD*, the new order is kept and the search procedure continues with the same individual for the next preceding position.

Local Search Procedure

1. *Initialization.* Let k denote the position of the candidate batch and Ψ denote the total number of batches (trips) to be considered. Select an initial sequence using any heuristic described earlier, and set $k = 2$ and $\rho = k$. ρ is used to keep track of every iteration where the local search is started. Thus, if the search terminates, it will restart from the next candidate.
2. *Batch selection.* Select the batches in the positions k and $k - 1$ for interchange.
3. *Move.* Swap the position of the selected batches, leaving all other batches‘ positions alone.
4. *Objective function assessment.* If the objective function value resulting from the proposed swap is better than the best objective value found so far, then update the best schedule accordingly and go to Step 5. Otherwise, go to Step 6.
5. *Position update.* If $k \neq 2$, then set $k = k - 1$, else $k = \rho + 1$. If $k > \Psi$, then STOP. Otherwise, go to Step 2.
6. *Candidate update.* Set $k = \rho + 1$. If $k > \Psi$, then STOP. Else, go to Step 2.

Given this local search procedure, each of the 20 heuristics in Table 1 can be modified to include local search as a post-processing procedure; we designate this by appending

Table 2 Experimental design

Factors	Levels	Level description
Number of machines	2	2, 3
Number of customers	2	3, 5
Job weights	2	DU[1,5], DU[1,10]
Job sizes	2	DU[1,25], DU[1,50]
Job processing times	2	DU[1,5] * Job Size, DU[1,10] * Job Size
Size of delivery area	2	50×50, 100×100 square
Number of jobs	4	10, 25, 50, and 100
Total	256	

the heuristic name with "_L". This procedure results in 40 total heuristics for evaluation (i.e., H1, H1_L, H2, H2_L, etc.). In the results section that follows, we report solution quality for the proposed heuristics both with and without the benefit of the post-processing local search procedure.

5 Computational study

5.1 Experimental design

Hall and Posner (2007) highlight the importance of understanding the relationship between problem characteristics and the performance of solution procedures. Therefore, the heuristics' performances are tested on an extensive range of problem instances (Table 2). First, we consider two different levels for the number of customers (3 and 5). The customer locations for each data set are randomly generated from a two-dimensional discrete uniform distribution of equal length and width (50×50 or 100×100). The plant is located in the center of this square. Similar to Dobson and Nambimadom (2001), batch capacity is set as 50 and two different levels of job sizes are generated from a discrete uniform distribution by using different ranges. Ranges for the two levels are [1, 25] and [1, 50]. Job processing times are generated proportional to job sizes. A random integer is generated on the interval $[1, v]$, $v \in \{5, 10\}$, and multiplied by the job sizes in order to generate job-processing times. Experiments also are performed for both two and three machines operating in parallel. Furthermore, two different levels of job weights are randomly assigned as discrete uniform integer values on [1, 5] and [1, 10]. Finally, four different numbers of job levels are examined in the experimental design: 10 jobs, 25 jobs, 50 jobs, and 100 jobs.

A total of 10 problem instances are generated for each of the $2^6(4) = 256$ experimental combinations, thereby resulting in a total of 2,560 problem instances that will be evaluated by each of the 40 proposed heuristics.

5.2 Experimental results

Each heuristic's objective value is compared with the best-known solution for each instance. All algorithms are implemented in Visual Basic for Applications and run on

Table 3 10-Job case heuristic performance ratios as compared to optimal

Level	No local search				With local search			
	Best PR	Avg PR	Worst PR	Best heur	Best PR	Avg PR	Worst PR	Best heur
# of Customers								
3	1.104	1.205	1.388	H14	1.056	1.130	1.236	H16_L
5	1.100	1.200	1.403	H18	1.051	1.107	1.198	H16_L
# of Machines								
2	1.090	1.192	1.387	H18	1.053	1.117	1.220	H16_L
3	1.111	1.213	1.403	H16	1.054	1.120	1.215	H16_L
Plant Locations								
50 × 50	1.103	1.229	1.466	H18	1.061	1.143	1.270	H16_L
100 × 100	1.097	1.177	1.324	H16	1.046	1.094	1.164	H16_L
Job Sizes (***q***)								
DU[1,25]	1.105	1.220	1.406	H14	1.059	1.140	1.246	H16_L
DU[1,50]	1.098	1.185	1.387	H16	1.048	1.097	1.190	H16_L
Job Proc Times								
q*DU[1,5]	1.102	1.171	1.307	H16	1.052	1.095	1.165	H16_L
q*DU[1,10]	1.099	1.234	1.483	H14	1.055	1.142	1.270	H16_L
Job Weights								
DU[1,5]	1.106	1.207	1.405	H16	1.054	1.120	1.220	H16_L
DU[1,10]	1.098	1.198	1.385	H14	1.053	1.117	1.215	H16_L
Overall Averages	1.101	1.203	1.395		1.054	1.119	1.217	

a PC with an Intel Pentium Dual-Core Processor (2.93 GHz CPU speed) and 2GB RAM. Every instance is solved by the heuristics within a couple of seconds.

5.2.1 Comparing heuristics to optimal solutions: 10 job problem instances

The optimization model of Sect. 3.2 is implemented in AMPL and solved by CPLEX 11.1 in order to evaluate the best solution obtained from any of the heuristics in small sized (10-jobs) problem instances. Let $TWD(O)$ be the optimal objective value obtained via our mathematical programming formulation and $TWD(B^*)$ be the best value obtained over all of the heuristics. The performance ratio, $PR(B^*) = \frac{TWD(B^*)}{TWD(O)}$, is computed to assess the solution quality of the best heuristic in small-sized instances. In Table 3, we summarize the results for 10-job problem instances, presenting both the without and with local search summary statistics.

The best non-local search-aided heuristic's individual performance across all instances was by heuristic H14, whose solutions averaged 10.2% above optimal for the 10 job cases. When each of the 20 heuristics' solutions are then post-processed via the proposed local search procedure, the best heuristic becomes H16_L with an average solution performance of 5.4% above optimal for the 10 job instances. It is interesting to note that the only significant difference between these two heuristics lies in the batch-sequencing rule followed—they both use the same batch assignment rule and both employ the proposed accept/reject rule. This strong performance of heuristic H16_L is further verified when one considers that if the best heuristic's solution for each 10-job problem instance was chosen, the resulting average performance ratio is only a 2% improvement over using H16_L alone. Finally, additional experiments with 15 job cases for which the optimal solution is attainable suggest that the performance of these heuristics scales with problem size, as the optimal solutions were also obtained by a number of the heuristics.

5.2.2 Comparing heuristic solutions for larger problem instances

Now define heuristic ratio $HR(H) = \frac{TWD(H)}{TWD(B^*)}$; this is computed to assess the solution quality of each of the heuristics listed in Table 1 across all instances, where $TWD(H)$ is the total weighted delivery time value achieved by heuristic H. The average performance ratio value computed with respect to the best heuristic for each experimental factor combination (see Table 2) is given in Table 4. Heuristics incorporating job weights with processing times while assigning jobs to a trip improve substantially upon the heuristics suggested in the literature that do not consider the processing times of the jobs for the batch assignment phase.

RuleBA1 (jobs are sorted in non-increasing order of weight-to-processing time ratio) is the best performing batch assignment rule. Among the batch sequencing rules, BS2 (batches are sequenced by $\omega_k/(\mathrm{T}_k + \mathrm{P}_k)$) yields better results than the other approaches overall. As the production environment causes fewer bottlenecks by an increased number of machines or decreased processing times, the heuristic considering longest processing times instead of total processing times in the batch sequencing phase performs better (H16). Similarly, when the distribution environment can be the bottleneck due to large transportation times, the best performance is achieved by giving more emphasis to transportation times by batch sequencing in which the total of the processing times is divided by the number of machines (H15). All the heuristics generally perform better when there are more consolidation opportunities due to a larger number of jobs, fewer customers, or smaller job sizes. On the other side, scheduling becomes more challenging when processing times are drawn from a narrow range and more machines are employed (see Appendix 1 for more details). Overall, heuristic H14 produces schedules that are better than all other heuristics in terms of *TWD*. In H14, BA1 forms the batches and then BS2 sequences them with an accept-reject evaluation.

When the local search procedure is seeded with the schedule from the initial heuristics, all solutions are achieved within a minute, even for large instances. For example, the average computational time for 100-job instances increases from 0.21 seconds for H14 to 22.88 seconds for H14_L.For an added slight computational expense, the average *TWD* performance ratio improves from 13.1 % above best to 7.9 % above best. Heuristic H14 with local search is still the best performing heuristic; it found the best solution in 994 instances out of 2,560 problem instances. The local search procedure improves the performance ratio of H14 from 5.2 to 2.2 % for H14_L.

Table 5 illustrates the 95 % confidence intervals for $HR(H)$ for all 40 possible heuristic solution approaches, sorted according to non-decreasing lower confidence

Table 4 Heuristic results across entire experimental design region ($HR(H)$)

Heur	Jobs				Customers		Machines		Plant location		Job size (q)		Job proc time		Job weight		Overall
	10	25	50	100	3	5	2	3	50 × 50	100 × 100	DU[1,25]	DU[1,50]	q*DU[1,5]	q*DU[1,10]	DU[1,5]	DU[1,10]	
H1	1.321	1.266	1.225	1.214	1.243	1.269	1.261	1.251	1.300	1.213	1.264	1.249	1.202	1.311	1.257	1.256	1.256
H2	1.071	1.076	1.064	1.061	1.066	1.069	1.056	1.080	1.053	1.083	1.072	1.064	1.076	1.059	1.069	1.066	1.068
H3	1.108	1.097	1.079	1.073	1.092	1.086	1.078	1.101	1.086	1.092	1.103	1.076	1.088	1.090	1.091	1.087	1.089
H4	1.076	1.088	1.079	1.078	1.079	1.081	1.078	1.083	1.075	1.085	1.088	1.072	1.082	1.079	1.080	1.080	1.080
H5	1.341	1.283	1.229	1.210	1.252	1.279	1.272	1.260	1.315	1.217	1.273	1.259	1.204	1.328	1.268	1.264	1.266
H6	1.187	1.166	1.114	1.089	1.140	1.138	1.138	1.140	1.158	1.120	1.171	1.107	1.115	1.163	1.139	1.139	1.139
H7	1.212	1.178	1.123	1.095	1.153	1.152	1.149	1.155	1.175	1.130	1.184	1.121	1.123	1.181	1.151	1.153	1.152
H8	1.192	1.174	1.125	1.102	1.150	1.147	1.152	1.145	1.170	1.126	1.183	1.114	1.119	1.177	1.148	1.148	1.148
H9	1.348	1.301	1.281	1.275	1.295	1.307	1.305	1.297	1.352	1.250	1.316	1.287	1.240	1.362	1.308	1.294	1.301
H10	1.094	1.094	1.091	1.094	1.100	1.087	1.082	1.105	1.089	1.098	1.100	1.087	1.095	1.092	1.096	1.091	1.093
H11	1.125	1.116	1.119	1.121	1.128	1.112	1.106	1.134	1.124	1.117	1.136	1.104	1.115	1.126	1.124	1.117	1.120
H12	1.107	1.108	1.110	1.116	1.119	1.102	1.110	1.111	1.118	1.103	1.124	1.096	1.103	1.118	1.110	1.110	1.110
H13	1.299	1.215	1.171	1.158	1.193	1.229	1.214	1.208	1.244	1.178	1.206	1.216	1.167	1.255	1.211	1.211	1.211
H14	1.066	1.061	1.044	1.037	1.047	1.058	1.039	1.065	1.035	1.069	1.050	1.055	1.066	1.039	1.053	1.052	$\underline{1.052}$
H15	1.098	1.067	1.045	1.036	1.059	1.065	1.051	1.072	1.056	1.067	1.065	1.058	1.066	1.057	1.063	1.060	1.062
H16	1.069	1.068	1.052	1.045	1.053	1.065	1.054	1.064	1.050	1.068	1.060	1.057	1.065	1.053	1.059	1.059	1.059
H17	1.306	1.228	1.184	1.168	1.204	1.239	1.225	1.217	1.258	1.184	1.218	1.225	1.173	1.269	1.220	1.222	1.221
H18	1.067	1.066	1.051	1.043	1.053	1.061	1.044	1.069	1.041	1.072	1.055	1.058	1.067	1.047	1.059	1.055	1.057
H19	1.100	1.076	1.056	1.046	1.068	1.071	1.059	1.080	1.067	1.072	1.075	1.064	1.070	1.069	1.072	1.067	1.070
H20	1.071	1.073	1.062	1.054	1.062	1.069	1.062	1.069	1.059	1.071	1.067	1.064	1.068	1.062	1.065	1.065	1.065
Avg	1.163	1.140	1.115	1.106	1.128	1.134	1.127	1.135	1.141	1.121	1.140	1.122	1.115	1.147	1.132	1.130	1.131

Table 5 95 % Confidence intervals of performance ratios for heuristics ($HR(H)$)

Heuristic	10 Jobs		Heuristic	25 Jobs		Heuristic	50 Jobs		Heuristic	100 Jobs	
	LCL	UCL		LCL	UCL		LCL	UCL		LCL	UCL
H16_L	**1.016**	**1.022**	**H14_L**	**1.025**	**1.031**	**H14_L**	**1.017**	**1.021**	**H14_L**	**1.013**	**1.016**
H20_L	**1.018**	**1.023**	**H16_L**	**1.026**	**1.031**	H16_L	1.022	1.025	H18_L	1.018	1.021
H4_L	**1.021**	**1.027**	**H18_L**	**1.028**	**1.034**	H15_L	1.022	1.026	H15_L	1.019	1.022
H14_L	**1.022**	**1.029**	**H20_L**	**1.029**	**1.034**	H18_L	1.023	1.027	H16_L	1.020	1.024
H18_L	**1.023**	**1.030**	**H15_L**	**1.032**	**1.037**	H20_L	1.029	1.033	H20_L	1.028	1.030
H2_L	**1.026**	**1.033**	H2_L	1.037	1.043	H19_L	1.032	1.036	H19_L	1.028	1.031
H15_L	1.034	1.043	H19_L	1.038	1.045	H2_L	1.033	1.037	H14	1.034	1.041
H19_L	1.035	1.043	H4_L	1.039	1.045	H14	1.041	1.048	H2_L	1.034	1.038
H3_L	1.039	1.049	H10_L	1.052	1.060	H4_L	1.042	1.046	H15	1.034	1.038
H12_L	1.042	1.052	H3_L	1.055	1.063	H15	1.043	1.047	H18	1.040	1.046
H10_L	1.043	1.053	H14	1.057	1.065	H18	1.048	1.055	H16	1.043	1.048
H11_L	1.053	1.066	H12_L	1.060	1.068	H16	1.050	1.055	H19	1.044	1.047
H14	1.061	1.071	H18	1.061	1.070	H3_L	1.051	1.057	H4_L	1.047	1.051

limit (LCL)in order to evaluate the variance of performance as discussed in Hall and Posner (2007). The bolded cells in Table 5 denote the best heuristic performance and those that are statistically indistinguishable from the best performance (i.e. those with overlapping of the confidence intervals). Heuristic H14_L is the best, and its overlap with other heuristics' confidence intervals is quite small. Furthermore, H14 (without local search) performs better than some heuristics that use local search. In 10-job instances, H16_L outperforms H14_L and produces promising solutions for other problem instances as well. Both heuristics use the same batch assignment rule (BA1—jobs sorted in non-increasing order of weight-to-processing time ratio, assigned to first available trip) with an accept-reject evaluation and consider production times together with transportation times in the batch-sequencing phase.

We further evaluate heuristic performances by employing multiple heuristics simultaneously (see Table 6). Substantial improvements can be achieved when a small number of the best heuristics are selected either with local search or not. The top four heuristics with no local search produce solutions 3.1 % above the best. When all heuristics without local search are selected, a performance ratio similar to the best heuristic with local search can be achieved, but with less computational effort. However, as more heuristics with local search are selected substantial performance ratio improvements are also achieved. Simultaneously employing the two best heuristics with local search improves the performance ratio from 2.2 to 1.3 %. An average performance ratio within 1 % of the best solution can be achieved by the top three heuristics with local search. However, Table 6 suggests that while good solutions can be achieved quite quickly by applying multiple (if not all) heuristics without local search, the extra time spent performing local search can prove beneficial (see Fig. 3 for a summary plot in objective space). A decision maker can evaluate how to make the tradeoff between computational effort and quality of the actual schedule.

Table 6 Performance ratios of simultaneously selected heuristics

Levels	H14	H14 or H18	H14 or H18 or H15	H14 or H18 or H15 or H16	All Heuristics, No Local Search	H14_L	H14_L or H16_L	H14_L or H16_L or H18_L	H14_L or H16_L or H18_L or H15_L
Number of jobs									
10	1.066	1.063	1.050	1.039	1.032	1.025	1.009	1.008	1.005
25	1.061	1.055	1.040	1.035	1.023	1.028	1.017	1.013	1.009
50	1.044	1.038	1.028	1.027	1.017	1.019	1.013	1.009	1.008
100	1.037	1.032	1.024	1.023	1.014	1.015	1.012	1.008	1.007
Number of customers									
3	1.047	1.041	1.033	1.028	1.019	1.021	1.011	1.008	1.007
5	1.058	1.053	1.038	1.034	1.024	1.023	1.014	1.011	1.008
Number of machines									
2	1.039	1.035	1.029	1.025	1.019	1.016	1.009	1.007	1.005
3	1.065	1.059	1.042	1.037	1.025	1.028	1.016	1.013	1.009
Plant locations									
50 × 50	1.035	1.032	1.027	1.022	1.019	1.012	1.005	1.003	1.002
100 × 100	1.069	1.062	1.044	1.039	1.024	1.032	1.021	1.016	1.012
Job sizes(q)									
25	1.050	1.043	1.034	1.028	1.018	1.023	1.013	1.009	1.006
50	1.055	1.051	1.037	1.034	1.025	1.020	1.012	1.011	1.008
Job processing times									
5	1.066	1.058	1.043	1.039	1.025	1.029	1.019	1.014	1.011
10	1.039	1.035	1.028	1.023	1.019	1.014	1.007	1.005	1.003
Job weights									
5	1.053	1.048	1.036	1.032	1.022	1.023	1.013	1.010	1.008
10	1.052	1.046	1.035	1.030	1.021	1.021	1.012	1.009	1.007
Avg performance ratio	1.052	1.047	1.036	1.031	1.022	1.022	1.013	1.010	1.007
Avg comp time (in s)	0.130	0.261	0.391	0.521	2.865	8.532	16.839	25.509	33.144

5.2.3 Worst performance ratios: empirical observations

In order to see how badly heuristics can perform for the general case, we summarize the worst performance ratios achieved by each heuristic across all problem instances (Appendix-2). As the number of jobs increases, the worst performance ratios decrease. H16_L and H20_L have the smallest worst performances (1.239). Each heuristic with local search generates solutions with a worst performance ratio that is less than 2. In addition, if we assume that an assignment of jobs to batches has been made, it follows that the worst possible result when using BS2 (weighted sum of processing and transportation times) to schedule the batches is bounded by m times the optimal, where m is the number of machines at the factory (Kyparisis and Koulamas

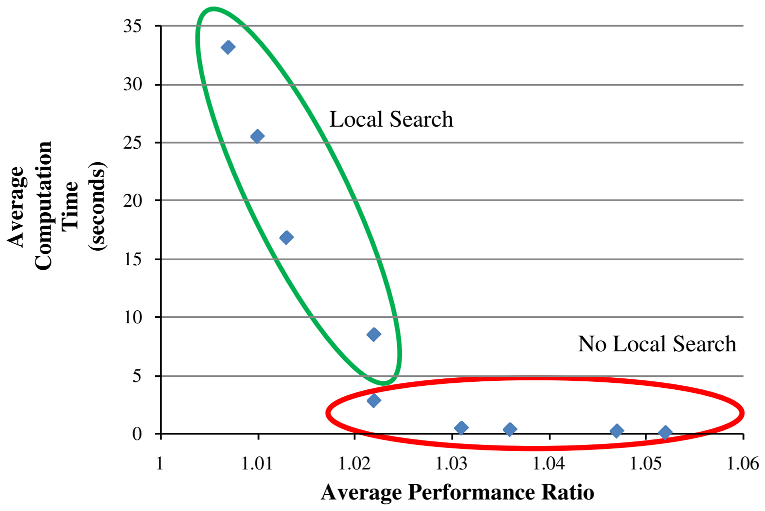

Fig. 3 Objective space comparison of heuristic solution approaches from Table 6

2001). Such poor performance may occur when a batch has a single big job contained in it.

6 Conclusions and future research

This paper is practically motivated by the critically important problem of supply chain scheduling systems in which the production facility is modeled as a parallel machine environment and a single capacitated vehicle is used for delivery of products to different customer locations. Such a system can be found in many industries. To the best of our knowledge, no previous research has studied this problem with the practical assumptions discussed here. Because most previous studies in the area of supply chain scheduling involve polynomial or pseudo-polynomial algorithms developed for special cases of the problem, there is a need for research that develops efficient heuristics to handle more general large-sized and practically motivated problems (Chang and Lee 2004; Qi 2005; Ji et al. 2007; Hall and Potts 2005; Li et al. 2005). This study expands on existing assumptions described in the literature and develops mathematical programming and heuristic solution approaches.

We have shown via the accept/reject rule that it is not necessarily the best choice to fill trucks as much as possible in an integrated production and distribution environment. Since there is only one truck employed, that truck can be more efficiently utilized by eliminating unnecessary waiting times for loading. The accept-reject rule improves the quality of the job batches that are loaded in terms of weight, size, and required production times, by considering the impact of adding a new job to the next delivery. Heuristics incorporating job weights with processing times while assigning jobs to a trip improve substantially upon the heuristics suggested in the literature that do not consider the processing times of the jobs for the batch assignment phase. Results also suggest that after forming the job batches for delivery, both transportation times and required production times should be considered while sequencing the trips. Schedules produced by heuristics are further improved through the use of a local search procedure. The combination of BA1 (jobs sorted in non-increasing order of weight-to-processing time ratio, assigned to first available trip) and BS2 (batches sequenced by weighted shortest sum of required transportation and processing time), modified with an accept-reject rule and local search, yields results that are only 2.2 % (on average) above the best for the problem of interest. We have also presented performance ratio improvements through selecting multiple heuristics simultaneously. A user can decide on employing several heuristics at the same time in order to improve solution qualities with the expense of increased computational effort. Different batch forming and sequencing rules can also be preferred, depending on the problem characteristics (i.e., giving more emphasis to transportation times when the distribution part is the bottleneck).

The solution approaches proposed in this study are fast, intuitive, practical, and easy to understand and implement. They attempt to optimize the overall system by considering all components of the problem simultaneously in an interconnected manner. For example, the accept-reject check employed in the batch assignment rule evaluate show adding a job to a batch will affect the batch's position that was determined in batch sequencing phase. Because starting solutions generally have significant impact on the solution quality of algorithms (Naderi et al. 2008),good initial solutions are found via simple dispatching rules are used for assigning jobs to batches and for sequencing these batches. With the adaptation of accept-reject rule, high quality batches are built, and local search is effectively applied to re-sequence batches that are already well sequenced based on the different properties of the jobs included.

There is still a vast area of research opportunities for studying supply chain scheduling. Specifically, effective heuristic solution procedures are needed because of the complexity of the problems and the prevalence of practical applications requiring fast solutions. The problem studied in this paper can be extended by considering multiple vehicles and routing decisions allowing multiple customer visits in the same trip. Different objective functions and multi-objective supply chain scheduling problems can be investigated. Another important extension would include the raw material supply to the manufacturer in order to study a three-stage supply chain.

Appendix 1

See Tables 7 and 8.

Table 7 95 % Confidence intervals of performance ratios for batch assignment rules (in heuristics without local search)

		Batch assignment				
		Greedy	Knapsack	BWFBWS	BA1	BA2
Number of jobs	10	[1.137,1.151]	[1.225,1.241]	[1.162,1.176]	[1.127,1.139]	[1.130,1.142]
	25	[1.127,1.136]	[1.195,1.206]	[1.150,1.160]	[1.099,1.107]	[1.106,1.115]
	50	[1.108,1.115]	[1.144,1.151]	[1.146,1.155]	[1.075,1.081]	[1.085,1.092]
	100	[1.103,1.110]	[1.121,1.127]	[1.147,1.156]	[1.066,1.072]	[1.075,1.081]
Number of customers	3	[1.117,1.124]	[1.170,1.178]	[1.157,1.164]	[1.085,1.091]	[1.094,1.100]
	5	[1.123,1.130]	[1.175,1.183]	[1.148,1.156]	[1.101,1.107]	[1.106,1.113]
Number of machines	2	[1.115,1.122]	[1.174,1.182]	[1.147,1.155]	[1.086,1.093]	[1.094,1.101]
	3	[1.125,1.132]	[1.171,1.179]	[1.158,1.166]	[1.099,1.106]	[1.105,1.112]
Plant Locations	50 × 50	[1.125,1.132]	[1.200,1.209]	[1.166,1.175]	[1.093,1.100]	[1.103,1.110]
	100 × 100	[1.115,1.121]	[1.144,1.152]	[1.139,1.145]	[1.093,1.098]	[1.097,1.103]
Job sizes (q)	DU[1,25]	[1.128,1.135]	[1.198,1.207]	[1.165,1.173]	[1.092,1.098]	[1.100,1.107]
	DU[1,50]	[1.112,1.118]	[1.146,1.154]	[1.140,1.147]	[1.094,1.100]	[1.100,1.106]
Job processing times	q*DU[1,5]	[1.109,1.115]	[1.137,1.144]	[1.135,1.141]	[1.088,1.093]	[1.092,1.097]
	q*DU[1,10]	[1.131,1.139]	[1.208,1.217]	[1.170,1.179]	[1.097,1.104]	[1.108,1.116]
Job weights	DU[1,5]	[1.121,1.128]	[1.172,1.181]	[1.156,1.163]	[1.093,1.100]	[1.101,1.107]
	DU[1,10]	[1.119,1.126]	[1.172,1.180]	[1.149,1.157]	[1.092,1.099]	[1.099,1.105]
Overall		[1.121,1.126]	[1.173,1.179]	[1.154,1.159]	[1.094,1.098]	[1.101,1.105]

Table 8 95 % Confidence intervals of performance ratios for batch sequencing rules (in heuristics without local search)

	Batch sequencing			
	BS1	BS2	BS3	BS4
Number of jobs				
10	[1.315,1.331]	[1.093,1.101]	[1.124,1.133]	[1.099,1.107]
25	[1.253,1.264]	[1.090,1.095]	[1.104,1.110]	[1.100,1.105]
50	[1.214,1.222]	[1.071,1.075]	[1.082,1.087]	[1.084,1.087]
100	[1.201,1.209]	[1.063,1.067]	[1.072,1.076]	[1.077,1.081]
Number of customers				
3	[1.233,1.242]	[1.079,1.083]	[1.098,1.102]	[1.090,1.094]
5	[1.260,1.269]	[1.081,1.084]	[1.095,1.099]	[1.091,1.095]
Number of machines				
2	[1.251,1.259]	[1.070,1.074]	[1.087,1.091]	[1.089,1.093]
3	[1.242,1.251]	[1.090,1.094]	[1.106,1.111]	[1.092,1.096]
Plant locations				
50×50	[1.290,1.298]	[1.073,1.077]	[1.099,1.104]	[1.092,1.096]
100×100	[1.204,1.212]	[1.087,1.090]	[1.094,1.098]	[1.089,1.093]
Job sizes (q)				
DU[1,25]	[1.251,1.260]	[1.088,1.092]	[1.110,1.115]	[1.102,1.107]
DU[1,50]	[1.243,1.251]	[1.072,1.076]	[1.083,1.086]	[1.079,1.082]
Job processing times				
q*DU[1,5]	[1.194,1.201]	[1.082,1.086]	[1.090,1.094]	[1.086,1.089]
q*DU[1,10]	[1.301,1.309]	[1.078,1.082]	[1.102,1.107]	[1.095,1.100]
Job weights				
DU[1,5]	[1.248,1.257]	[1.081,1.085]	[1.098,1.102]	[1.091,1.095]
DU[1,10]	[1.245,1.254]	[1.079,1.083]	[1.095,1.099]	[1.090,1.094]
Overall	[1.248,1.254]	[1.080,1.083]	[1.097,1.100]	[1.091,1.094]

Appendix 2

See Tables 9 and 10

Table 9 Heuristic worst performance ratios without local search

Heur	Jobs				Customers		Machines		Plant location		Job size (q)		Job proc time		Job weight		Overall
	10	25	50	100	3	5	2	3	50 × 50	100 × 100	DU[1,25]	DU[1,50]	q*DU[1,5]	q*DU[1,10]	DU[1,5]	DU[1,10]	
H1	2.340	1.989	1.649	1.674	2.092	2.340	2.238	2.340	2.340	2.053	2.340	2.053	2.091	2.340	2.238	2.340	2.340
H2	1.389	1.362	1.266	1.205	1.389	1.339	1.330	1.389	1.339	1.389	1.389	1.362	1.362	1.389	1.389	1.330	<u>1.389</u>
H3	1.576	1.380	1.257	1.210	1.576	1.536	1.511	1.576	1.501	1.576	1.576	1.413	1.417	1.576	1.536	1.576	1.576
H4	1.498	1.366	1.250	1.187	1.498	1.416	1.433	1.498	1.498	1.416	1.498	1.416	1.416	1.498	1.416	1.498	1.498
H5	2.395	1.989	1.631	1.687	2.395	2.340	2.238	2.395	2.395	2.053	2.340	2.395	2.091	2.395	2.319	2.395	2.395
H6	2.091	1.616	1.341	1.223	2.091	1.994	1.902	2.091	2.091	1.978	2.091	1.978	2.091	1.994	2.091	1.994	2.091
H7	2.091	1.599	1.338	1.238	2.091	1.994	1.918	2.091	2.091	1.990	2.091	1.990	2.091	1.994	2.091	1.994	2.091
H8	1.994	1.698	1.344	1.249	1.806	1.994	1.902	1.994	1.994	1.789	1.994	1.789	1.806	1.994	1.813	1.994	1.994
H9	2.395	1.989	1.698	1.727	2.395	2.340	2.238	2.395	2.395	2.053	2.340	2.395	2.091	2.395	2.319	2.395	2.395
H10	1.503	1.376	1.342	1.263	1.456	1.503	1.458	1.503	1.456	1.503	1.448	1.503	1.503	1.456	1.503	1.456	1.503
H11	1.576	1.461	1.409	1.312	1.576	1.574	1 511	1.576	1.574	1.576	1.576	1.562	1 562	1.576	1 536	1.576	1.576
H12	1.546	1.367	1.367	1.289	1.546	1.503	1.529	1.546	1.546	1.503	1.546	1.503	1.546	1.498	1.503	1.546	1.546
H13	2.340	1.947	1.621	1.654	2.092	2.340	2.207	2.340	2.340	2.053	2.340	2.053	2.091	2.340	2.091	2.340	2.340
H14	1.430	1.342	1.283	1.201	1.430	1.342	1.430	1.342	1.307	1.430	1.430	1.342	1.342	1.430	1.318	1.430	1.430
H15	1.576	1.345	1.196	1.122	1.576	1.536	1.430	1.576	1.510	1.576	1.576	1.413	1.417	1.576	1.536	1.576	1.576
H16	1.498	1.340	1.228	1.174	1.498	1.350	1.433	1.498	1.498	1.354	1.498	1.350	1.354	1.498	1.340	1.498	1.498
H17	2.340	1.929	1.614	1.664	2.092	2.340	2.207	2.340	2.340	2.053	2.340	2.053	2.091	2.340	2.091	2.340	2.340
H18	1.430	1.342	1.275	1.206	1.430	1.342	1.430	1.342	1.307	1.430	1.430	1.342	1.342	1.430	1.326	1.430	1.430
H19	1.576	1.317	1.259	1.120	1.576	1.536	1.430	1.576	1.510	1.576	1.576	1.413	1.417	1.576	1.536	1.576	1.576
H20	1.498	1.340	1.222	1.176	1.498	1.350	1.433	1.498	1.498	1.354	1.498	1.350	1.354	1.498	1.340	1.498	1.498
Max	2.395	1.989	1.698	1.727	2.395	2.340	2.238	2.395	2.395	2.053	2.340	2.395	2.091	2.395	2.319	2.395	2.395

Table 10 Heuristic worst performance ratios with local search

Heur	Jobs				Customers		Machines		Plant location		Job size (q)		Job proc time		Job weight		Overall
	10	25	50	100	3	5	2	3	50 × 50	100 × 100	DU[1,25]	DU[1,50]	q*DU[1,5]	q*DU[1,10]	DU[1,5]	DU[1,10]	
H1	1.952	1.572	1.541	1.625	1.837	1.952	1.952	1.952	1.952	1.952	1.952	1.810	1.774	1.952	1.952	1.952	1.952
H2	1.365	1.209	1.172	1.114	1.365	1.255	1.239	1.365	1.255	1.365	1.365	1.255	1.255	1.365	1.365	1.209	1.365
H3	1.380	1.260	1.200	1.161	1.380	1.255	1.290	1.380	1.364	1.380	1.380	1.364	1.364	1.380	1.380	1.359	1.380
H4	1.255	1.253	1.200	1.129	1.245	1.255	1.245	1.255	1.255	1.245	1.255	1.209	1.255	1.253	1.255	1.253	1.255
H5	1.952	1.589	1.544	1.644	1.789	1.952	1.952	1.952	1.952	1.952	1.952	1.810	1.774	1.952	1.952	1.952	1.952
H6	1.902	1.588	1.308	1.219	1.789	1.902	1.902	1.788	1.902	1.789	1.902	1.789	1.702	1.902	1.707	1.902	1.902
H7	1.902	1.589	1.308	1.222	1.789	1.902	1.902	1.788	1.902	1.789	1.902	1.789	1.702	1.902	1.707	1.902	1.902
H8	1.902	1.589	1.306	1.224	1.789	1.902	1.902	1.788	1.902	1.789	1.902	1.789	1.739	1.902	1.707	1.902	1.902
H9	1.952	1.696	1.551	1.710	1.789	1.952	1.952	1.952	1.952	1.952	1.952	1.810	1.774	1.952	1.952	1.952	1.952
H10	1.389	1.251	1.314	1.244	1.389	1.319	1.389	1.361	1.314	1.389	1.389	1.361	1.389	1.361	1.300	1.389	1.389
H11	1.479	1.293	1.318	1.286	1.479	1.319	1 423	1.479	1.409	1.479	1.422	1.479	1 479	1.398	1 479	1.409	1.479
H12	1.398	1.306	1.318	1.260	1.398	1.319	1.347	1.398	1.398	1.370	1.398	1.370	1.308	1.398	1.398	1.370	1.398
H13	1.952	1.435	1.515	1.617	1.789	1.952	1.952	1.952	1.952	1.952	1.952	1.810	1.627	1.952	1.952	1.952	1.952
H14	1.430	1.216	1.196	1.136	1.430	1.214	1.430	1.332	1.185	1.430	1.430	1.255	1.255	1.430	1.262	1.430	1.430
H15	1.430	1.213	1.117	1.099	1.430	1.251	1.430	1.335	1.335	1.430	1.430	1.251	1.251	1.430	1.335	1.430	1.430
H16	1.187	1.239	1.191	1.108	1.187	1.239	1.239	1.238	1.238	1.239	1.239	1.191	1.238	1.239	1.239	1.238	1.239
H17	1.952	1.485	1.483	1.615	1.789	1.952	1.952	1.952	1.952	1.952	1.952	1.810	1.627	1.952	1.952	1.952	1.952
H18	1.430	1.241	1.164	1.108	1.430	1.214	1.430	1.332	1.228	1.430	1.430	1.255	1.255	1.430	1.305	1.430	1.430
H19	1.430	1.261	1.138	1.072	1.430	1.251	1.430	1.335	1.335	1.430	1.430	1.261	1.261	1.430	1.335	1.430	1.430
H20	1.187	1.239	1.147	1.080	1.187	1.239	1.239	1.204	1.178	1.239	1.239	1.185	1.185	1.239	1.239	1.204	1.239
Max	1.952	1.696	1.551	1.710	1.837	1.952	1.952	1.952	1.952	1.952	1.952	1.810	1.774	1.952	1.952	1.952	1.952

References

Akyol, D.E., Bayhan, G.M.: Minimizing makespan on identical parallel machines using neural networks. Lect. Notes Comput. Sci. **4234**, 553–562 (2006)

Albers, S., Brucker, P.: The complexity of one-machine batching problems. Discrete. Appl. Math. **47**, 87–107 (1993)

Azizoglu, M., Webster, S.: Scheduling a batch processing machine with incompatible job families. Comput. Ind. Eng. **39**, 325–335 (2001)

Behnamian, J., Fatemi Ghomi, S.M.T.: Minimizing makespan on a three-machine flowshop batch scheduling problem with transportation using genetic algorithm. Appl. Soft Comput. **12**, 768–777 (2012)

Bruno, L., Coffman, E.G., Sethi, J.R.: Scheduling independent tasks to reduce mean finishing time. Commun. ACM. **17**, 382–387 (1974)

Cakici, E., Mason, S.J., Fowler, J.W., Geismar, H.N.: Batch scheduling on parallel machines with dynamic job arrivals and incompatible job families. Int. J. Prod. Res. **51**, 2462–2477 (2013)

Chang, Y.C., Lee, C.Y.: Machine scheduling with job delivery coordination. Eur. J. Oper. Res. **158**, 470–487 (2004)

Chen, B., Lee, C.: Logistics scheduling with batching and transportation. Eur. J. Oper. Res. **187**, 871–876 (2008)

Chen, Z.L., Hall, N.G.: Supply chain scheduling: conflict and cooperation in assembly systems. Oper. Res. **55**, 1072–1089 (2007)

Chen, Z.L., Vairaktarakis, G.L.: Integrated scheduling of production and distribution operations. Manag. Sci. **51**, 614–628 (2005)

Dawande, M., Geismar, H.N., Hall, N.G., Sriskandarajah, C.: Supply chain scheduling: distribution systems. Prod. Oper. Manag. **15**, 243–261 (2006)

Dobson, G., Nambimadom, R.S.: The batch loading and scheduling problem. Oper. Res. **49**, 52–65 (2001)

Geismar, H.N., Laporte, G., Lei, L., Sriskandarajah, C.: The integrated production and transportation scheduling problem for a product with a short lifespan. INFORMS J. Comput. **20**, 21–33 (2008)

Gonga, H., Tanga, L., Duinb, C.W.: A two stage flowshop schedulingproblem on a batching machine and a discrete machine with blockingand shared setup times. Comput. Oper. Res. 37, 960–969 (2010)

Graham, R.L., Lawler, E.L., Lenstra, J.K.: Optimization and approximation in deterministic machine scheduling: a survey. Ann. Discrete Math. **5**, 287–326 (1979)

Hall, N.G., Posner, M.E.: Performance prediction and pre-selection for optimization and heuristic solution procedures. Oper. Res. **55**, 703–716 (2007)

Hall, N.G., Potts, C.N.: Supply chain scheduling: batching and delivery. Oper. Res. **51**, 566–584 (2003)

Hall, N.G., Potts, C.N.: The coordination of scheduling and batch deliveries. Ann. Oper. Res. **135**, 41–64 (2005)

Hoogeveen, H.: Multi-criteria scheduling — invited review. Eur. J. Oper. **167**, 592–623 (2005)

Hopp, W., Spearman, M.: Factory Physics. McGraw Hill, New York (2000)

Ji, M., He, Y., Cheng, T.C.E.: Batch delivery scheduling with batch delivery cost on a single machine. Eur. J. Oper. Res. **176**, 745–755 (2007)

Kyparisis, G.J., Koulamas, C.: A note on weighted completion time minimization in a flexible flow shop. Oper. Res. Lett. **29**, 5–11 (2001)

Koh, S.G., Koo, P.H., Kim, D.C., Hur, W.S.: Scheduling a single batch processing machine with arbitrary job sizes and incompatible job families. Int. J. Prod. Econ. **98**, 81–96 (2005)

Lee, C.Y., Chen, Z.L.: Machine scheduling with transportation considerations. J. sched. **4**, 3–24 (2001)

Lei, D., Wang, T.: An effective neighborhood search algorithm for scheduling a flowshop of batch processing machines. Comput. Ind. Eng. **61**, 739–743 (2011)

Lenstra, J.K., Rinnooy Kan, A.H.G., Brucker, P.: Complexity of machine scheduling problems. Ann. Discrete Math. **1**, 343–362 (1977)

Li, C.L., Vairaktarakis, G.L., Lee, C.Y.: Machine scheduling with deliveries to multiple customer locations. Eur. J. Oper. Res. **164**, 39–51 (2005)

Mazdeh, M.M., Sarhadi, M., Hindi, K.S.: A branch-and-bound algorithm for single-machine scheduling with batch delivery minimizing flow times and delivery costs. Eur. J. Oper. Res. **183**, 74–86 (2007)

Nemhauser, G.L., Wolsey, L.A.: Integer and Combinatorial Optimization. Wiley, New York (1998)

Naderi, B., Khalili, M., Taghavifard, M.T., Roshanaei, V.: A variable neighborhood search for hybrid flexible flowshops with setup times minimizing total completion time. J. Appl. Sci. **8**, 2843–2850 (2008)

Potts, C.N., Kovalyov, M.Y.: Scheduling with batching: A review. Eur. J. Oper. Res. **120**, 228–249 (2000)

Qi, X.: A logistics scheduling model: inventory cost reduction by batching. Naval Res. Logist. **52**, 312–320 (2005)

Selvarajah, E., Zhang, R.: Supply chain scheduling at the manufacturer to minimize inventory holding and delivery costs. Int. J. Prod. Econ. **147**(Part A), 17–124 (2014)

Simchi-Levi, D., Kaminski, P., Simchi-Levi, E.: Designing and Managing the Supply Chain. McGraw Hill, New York (2003)

Tang, L., Liu, P.: Flowshop scheduling problems with transportation or deterioration between the batching and single machines. Comput. Ind. Eng. **56**, 1289–1295 (2009)

Uzsoy, R.: Scheduling a single batch processing machine with non-identical job sizes. Int. J. Prod. Res. **32**, 1615–1635 (1994)

Uzsoy, R.: Scheduling batch processing machines with incompatible job families. Int. J. Prod. Res. **33**, 2685–2708 (1995)

Van Buer, M.G., Woodruff, D.L., Olson, R.T.: Solving the mediumnewspaper production/distribution problem. Eur. J. Oper. Res. 115,237–253 (1999)

Wang, G., Cheng, T.C.E.: Parallel machine scheduling with batch delivery costs. Int. J. Prod. Econ. **68**, 177–183 (2000)

Wolsey, L.A.: Integer Programming. Wiley, New York (1998)

Zhong, W., Dosa, G., Tan, Z.: On the machine scheduling problem with job delivery coordination. Eur. J. Oper. Res. **182**, 1057–1072 (2007)