European Journal of Operational Research 227 (2013) 152–165

Contents lists available at SciVerse ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Innovative Applications of O.R.

Integrated machine scheduling and vehicle routing with time windows

Christian A. Ullrich *

Bielefeld University, Department of Business Administration and Economics, Universitätsstraße 25, 33615 Bielefeld, Germany

ARTICLE INFO

Article history:
Received 2 May 2011
Accepted 18 November 2012
Available online 12 December 2012

Keywords:
Supply Chain Scheduling
Parallel machines
Vehicle routing
Time windows
Total tardiness
Genetic algorithm

ABSTRACT

This paper integrates production and outbound distribution scheduling in order to minimize total tardiness. The overall problem consists of two subproblems. The first addresses scheduling a set of jobs on parallel machines with machine-dependent ready times. The second focusses on the delivery of completed jobs with a fleet of vehicles which may differ in their loading capacities and ready times. Job-dependent processing times, delivery time windows, service times, and destinations are taken into account. A genetic algorithm approach is introduced to solve the integrated problem as a whole. Two main questions are examined. Are the results of integrating machine scheduling and vehicle routing significantly better than those of classic decomposition approaches which break down the overall problem, solve the two subproblems successively, and merge the subsolutions to form a solution to the overall problem? And if so, is it possible to capitalize on these potentials despite the complexity of the integrated problem? Both questions are tackled by means of a numerical study. The genetic algorithm outperforms the classic decomposition approaches in case of small-size instances and is able to generate relatively good solutions for instances with up to 50 jobs, 5 machines, and 10 vehicles.

1. Introduction

Increasing pressure on global markets forces companies as well as entire supply chains to capitalize rigorously on saving potentials. For example, just-in-time concepts are widely used to reduce logistics expenditures which often represent over 30% of the total costs of a product (Thomas and Griffin, 1996). Such approaches, however, involve a need to comply with due dates since tardy deliveries may cause the recipients to struggle with meeting their own due dates. Manufacturers of special purpose machines, for example, are not able to assemble a machine until their suppliers release the required components such as motors, cases, and control modules. A similar problem arises in modular construction. Warehouses or parking decks cannot be built until special industry doors, concrete modules, and steel components have been supplied. Tardiness of a single component can cause problems for the entire supply chain. Rearrangement or interruption of production schedules, idle time, tardiness penalties, and unnecessarily high inventories often drive up costs considerably. However, just-in-time approaches are pointless if they incur high transportation costs in an attempt to prevent such situations. In order to reduce total logistics cost, production and delivery operations need to be well coordinated, first within single companies and then within whole supply chains (Manoj et al., 2008). Efficient suppliers that rarely exceed due dates are essential for just-in-time supply chain environments and hence hold an enormous competitive advantage.

This paper deals with the coordination of a company's production and distribution operations. A given set of jobs has to be processed on parallel machines and delivered to job-dependent destinations by a fleet of vehicles, taking account of distinct delivery time windows. Such problems often occur in the automotive industry. Suppliers of injection molded plastic components, for example, usually run many parallel machines and deliver to several manufacturers like BMW, Daimler, and VW. If they deliver late, suppliers may cause the above mentioned problems, damage their reputation, and often incur contractual penalties. Hence, minimizing tardiness-based performance measures such as the number of tardy jobs, maximum tardiness, and total (weighted) tardiness is frequently encountered in practice. Because of its widespread use (Koulamas, 1994), the objective criterion here is minimizing total tardiness.

In order to reduce overall complexity, classic decomposition approaches break down the overall problem, solve the production and delivery subproblems successively, and merge the subsolutions to form a solution to the overall problem. Such approaches are especially reasonable if interdependencies between the two planning subproblems can be prevented by high buffer stocks. However, low or even zero buffer inventories are frequently encountered in practice. The popularity of the just-in-time philosophy, which serves to cut down on holding costs, is only one of several reasons for that. Make-to-order, perishable, and time-sensitive goods such as ready-mixed concrete (Garcia et al., 2002, 2004) simply cannot

* Tel.: +49 521 106 3931; fax: +49 5211066036.
E-mail addresses: cullrich@wiwi.uni-bielefeld.de, christian-a.ullrich@gmx.de

0377-2217/$ - see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.ejor.2012.11.049

be kept in stock. Especially for companies selling such products, the coordination of production and distribution is seen as one of the most important success factors (Dawande et al., 2006; Geismar et al., 2008; Akkerman et al., 2010; Huo et al., 2010; Geismar et al., 2011; Cakici et al., 2012). In order to be up to date, newspaper printing cannot be started before midnight (Hurter and Van Buer, 1996; Van Buer et al., 1999). Drop-off points, however, have to be supplied with area-dependent editions until, say, 4:00 am so home delivery carriers can pick up the newspapers early enough to prevent tardy delivery and thus dissatisfied customers. The same applies to the customized computer and food catering industries where fierce competition causes the need to produce and deliver high quality goods within very narrow time windows (Chen and Vairaktarakis, 2005; Farahani et al., 2012). Wal-Mart and Amazon venture on same-day delivery for online orders placed before noon (Bhattarai, 2012; Clifford, 2012). Customers can select a 4-h time window during which they want to receive the ordered items. The coordination of picking and delivery is a very demanding logistical task but also the key success factor for same-day delivery strategies. Effective and efficient integrated planning approaches are indispensable for managers who work with such business models.

Classic decomposition approaches that successively deal with the associated subproblems are often considered to cause poorly coordinated overall plans and therefore to result in high tardiness. Hence, there are two questions that motivate this paper. Are the results of integrating machine scheduling and vehicle routing significantly better than those of classic decomposition approaches? And if so, is it possible to capitalize on these potentials despite the complexity of the integrated problem? Both questions are addressed by means of a numerical study. A genetic algorithm that tackles the integrated problem as a whole is compared to two classic decomposition approaches.

The remainder is structured as follows. Section 2 surveys the related literature. In Section 3, the machine scheduling and vehicle routing subproblems as well as the integrated overall problem are described by mixed integer linear programs. The generation of instances, needed for the numerical study, is explained in Section 4. Section 5 introduces the decomposition approaches and the genetic algorithm. The results of the numerical study are provided in Section 6. Section 7 concludes.

2. Literature

Chen (2004) and Chen (2010) extensively review papers that deal with integrated scheduling of machines and outbound job delivery. The literature can be divided into two main areas. Some studies focus on rather 'simple' delivery considerations, like direct or shuttle shipments, while others allow for vehicle routing. Tables 1 and 2 provide an overview of the literature on these two problem types, subdividing the publications in question according to whether they consider due dates or not. Furthermore, the articles are categorized by the number of job destinations. Some papers consider a single job destination, like a distribution center or a warehouse, while others deal with multiple job destinations such as several customer locations. Since vehicle routing does not make sense in case of a single job destination, Table 2 contains only papers with multiple job destinations.

As they concern machine scheduling and vehicle routing, the publications listed in Table 2 are most closely related to this paper. Chang and Lee (2004) examine a single machine scheduling and vehicle routing problem. Only one vehicle with a limited loading capacity is available for delivery. Each job requires a job-dependent amount of loading capacity and is preassigned to one of only two possible destinations. A tailor-made heuristic is proposed to minimize the arrival time of the last job at its destination. Chen and Vairaktarakis (2005) extend the problem to more destinations and an infinite number of vehicles with a loading capacity limited to a maximum number of jobs. Single as well as parallel machine scheduling problems are addressed. The authors propose tailor-made algorithms to minimize convex combinations of the maximum arrival time (average arrival time) and the transportation costs which depend on the number of employed vehicles and the routes assigned to them. Their integrated production and delivery approaches are compared to classical decomposition approaches. Numerical experiments reveal that integration induces average relative improvements of between 12% and 40%. Li and Vairaktarakis (2007) focus on a similar problem with jobs that consist of two tasks. There is a special machine for each task. Both tasks can be done at the same time. Tailor-made heuristics are developed to minimize the sum of the weighted total arrival time and transportation costs. Garcia et al. (2004) deal with several plants, each equipped with parallel machines, and a fleet of vehicles that can deliver no more than one job at a time. The jobs can be processed only in job-dependent subsets of the plants and must be delivered immediately after completion. Furthermore, the jobs are characterized by job-dependent due dates, destinations, and service times at the destinations. The author's tailor-made heuristic aims to maximize the total weighted number of just-in-time delivered jobs minus the transportation costs. Garcia et al. (2002) apply a genetic algorithm to a similar problem which is described below.

Inspired by Thomas and Griffin (1996), who call for deterministic approaches to operational supply chain problems, Hall and Potts (2003) address integrated machine scheduling and batch

Table 1
Integrated scheduling of machines and 'simple' delivery operations.

Problems	Without due dates	With due dates
With a single job destination	Cheng and Kahlbacher (1993), Cheng and Gordon (1994), Cheng et al. (1996), Cheng et al. (1997), Wang and Cheng (2000), Lee and Chen (2001), Chang and Lee (2004), Li and Ou (2005), Soukhal et al. (2005), Chen and Pundoor (2006), Chen et al. (2007), Ji et al. (2007), Yuan et al. (2007), Wang and Cheng (2007), Zhong et al. (2007), Pan et al. (2009), Wang and Cheng (2009a), Wang and Cheng (2009b), Lee and Yoon (2010), Mazdeh et al. (2011), and Ng and Lingfa (2012)	Matsuo (1988), Herrmann and Lee (1993), Chen (1996), Yuan (1996), Yang (2000), Hall et al. (2001), Hall and Potts (2005), Qi (2008), Chen and Pundoor (2009), Fu et al. (2012), and Hamidinia et al. (2012)
with multiple job destinations	Potts (1980), Woeginger (1994), Zdrzalka (1995), Woeginger (1998), Gharbi and Haouari (2002), Liu and Cheng (2002), Hall and Potts (2003), Averbakh and Xue (2007), Li and Vairaktarakis (2007), Mazdeh et al. (2007), Chen and Lee (2008), Mazdeh et al. (2008), Li and Yuan (2009), Selvarajah and Steiner (2009), Averbakh (2010), Li et al. (2011), and Averbakh and Baysan (2012)	Hall and Potts (2003), Pundoor and Chen (2005), Wang and Lee (2005), Qi (2006), Stecke and Zhao (2007), Steiner and Zhang (2009), Huo et al. (2010), Cakici et al. (2012), and Farahani et al. (2012)

Table 2
Integrated machine scheduling and vehicle routing.

Problems	Without due dates	With due dates
	Chang and Lee (2004), Chen and Vairaktarakis (2005), and Li and Vairaktarakis (2007)	Garcia et al. (2002) and Garcia et al. (2004)

delivery problems which arise in a supply chain consisting of a supplier, several manufacturers, and several customers. In doing so they established the closely related research area of Supply Chain Scheduling. It is notable that some authors, such as Potts and Strusevich (2009) in "*Fifty years of scheduling: a survey of milestones*", already refer to the integrated scheduling of machines and outbound job delivery as Supply Chain Scheduling.

In recognition of its strong practical relevance, Pundoor and Chen (2005) and Soukhal et al. (2005) explicitly call for more research on integrated machine scheduling and vehicle routing which, as Table 2 reveals, is a rarely studied type of problem. Appeals are also made for more research including due dates (Chang and Lee, 2004) and time windows: "*While vehicle routing problems with time windows have been well studied in the vehicle routing literature, few papers have considered integrated production and outbound distribution scheduling problems with both routing decisions and time window constraints. Such problems are more challenging than any other class or subclass of integrated production and outbound distribution scheduling problems. However, due to their practicality, such problems clearly deserve more research*" (Chen, 2010). Furthermore, there is a call for research dealing with a processing company that owns a fleet of vehicles (Chen and Vairaktarakis, 2005). The number of available vehicles limits the number of concurrent shipments in such a scenario. The next section introduces a practice-oriented problem setting that combines all of these requests.

Since genetic algorithms are the method of choice to approach this paper's integrated problem, the second part of the literature overview surveys publications that apply genetic algorithms to relevant problems. For example, genetic algorithms have been successfully used to solve related machine scheduling problems (Aytug et al., 2003; Reeves, 2010). Min and Cheng (1999) aim to minimize the makespan on parallel machines. Their genetic algorithm outperforms a simulated annealing approach and a priority rule based heuristic. Sivrikaya-Şerifoğlu and Ulusoy (1999) address a parallel machine scheduling problem with job-dependent due dates. The goal is to minimize the sum of earliness and tardiness penalties. For larger-size instances, a heuristic based on neighborhood exchange turns out to be worse than a genetic algorithm. Moreover, Valente and Gonçalves (2009) develop a well-performing genetic algorithm for a single machine scheduling problem with linear earliness and quadratic tardiness penalties. Genetic algorithms have been also successfully applied to vehicle routing problems. Numerical studies with common benchmark instances for capacitated vehicle routing problems with total travel distance (time) and total travel cost objectives reveal that the hybrid genetic algorithms by Berger and Barkaoui (2003) and Prins (2004) can compete with the best known tailor-made heuristics. Ting and Huang (2005), Alvarenga et al. (2007), Vidal et al. (2013) as well as some earlier papers that are surveyed by Bräysy and Gendreau (2005) obtain similar results for vehicle routing problems with time windows.

Despite these successful applications of genetic algorithms to problems similar to the subproblems introduced in the next section, there are only very few genetic algorithm and other meta-heuristic approaches to integrated production and delivery scheduling problems. Hamidinia et al. (2012) apply a genetic algorithm to a single machine scheduling and batch delivery problem with the aim to minimize the sum of transportation, inventory holding, earliness, and tardiness costs. Cakici et al. (2012) combine genetic algorithms with simple dispatching rules to solve multi-objective single machine scheduling and batch delivery problems. Garcia et al. (2002) simultaneously select jobs, assign them to one of several plants equipped with parallel machines, and schedule the production and delivery operations. A fleet of vehicles is available to deliver the jobs immediately after processing in single-job shipments to their respective customers. The objective is to maximize the profit associated with the selected jobs minus penalty payments incurred for deviations from job-dependent due dates. A comparison with an exact graph-based method reveals that the author's genetic algorithm results in near-optimal solutions. To the best of our knowledge, this is the sole application of genetic algorithms to an integrated machine scheduling and vehicle routing problem so far.

3. Problem formulation

The integrated problem of this paper consists of two subproblems which are described and modeled in Sections 3.1 and 3.2, respectively. Section 3.3 shows how the models of the subproblems can be linked to describe the integrated machine scheduling and vehicle routing problem. The following parameters and variables are employed.

Parameters:

c_v	Loading capacity of vehicle v ($v = 1, \ldots, V$)
d_j	Processing due date of job j ($j = 1, \ldots, J$)
u_j	Size of job j($j = 1, \ldots, J$)
p_j	Processing time of job j ($j = 1, \ldots, J$)
q	Sufficiently large number
r_m	Ready time of machine m ($m = 1, \ldots, M$)
$\hat{r}_v$	Ready time of vehicle v ($v = 1, \ldots, V$)
s_j	Service time at the destination of job j ($j = 0, \ldots, J$; the index 0 denotes the processing site)
t_{ij}	Travel time from the destination of job i to the destination of job j ($i,j = 0, \ldots, J$). Note that two or more jobs may belong to the same customer and destination. In this case $t_{ij} = 0$ holds.
$\underline{w}_j, \overline{w}_j$	Lower and upper bound of the delivery time window of job j ($j = 1, \ldots, J$)

Variables:

C_j	Completion time of job j ($j = 1, \ldots, J$; parameter in the vehicle routing model of Section 3.2)
D_j	Delivery time of job j ($j = 1, \ldots, J$)
S_{vt}	Start time of the tth tour of vehicle v ($v = 1, \ldots, V$; $t = 1, \ldots, J$). Note that it may be optimal to employ only one vehicle that ships each job separately. Hence, the tour index t must go up to J
T_j	Tardiness of job j ($j = 1, \ldots, J$)
g_{jvt}	Binary variable which takes the value 1 if job j is delivered on the tth tour of vehicle v ($j = 1, \ldots, J$; $v = 1, \ldots, V$; $t = 1, \ldots, J$)
x_{ij}	Binary variable which takes the value 1 if job i is processed before job j and no other job is processed in between on that machine ($i, j = 1, \ldots, J$; $i \neq j$)
$x_{j,J+1}$	Binary variable which takes the value 1 if job j is the last job processed on a machine ($j = 1, \ldots, J$). Job $J + 1$ is an artificial last job
y_{mj}	Binary variable which takes the value 1 if job j is the first job processed on machine m ($m = 1, \ldots, M$; $j = 1, \ldots, J$)
z_{ijvt}	Binary variable which takes the value 1 if job i is delivered before job j on the tth tour of vehicle v ($i, j = 0, \ldots, J$; $v = 1, \ldots, V$; $t = 1, \ldots, J$)

3.1. Parallel machine scheduling

Consider a set of J jobs with job-dependent processing times p_j and due dates d_j ($j = 1, \ldots, J$). How the due dates for processing are imposed by delivery planning is explained in Section 3.2. Each job j has to be assigned to one of M identical parallel machines and scheduled to fit the timetable of that machine. Preemption is not allowed. The point in time a machine finishes processing its last job is equal to the moment the machine is ready to process the next set of jobs. As these points in time generally differ between machines, each machine m has its own ready time r_m ($m = 1, \ldots, M$).

According to the Three-Field-Notation by Graham et al. (1979), this problem can be described by $IP, r_m//\sum_j T_j$ where IP in conjunction with r_m is an abbreviation for identical parallel machines with machine ready times. $\sum_j T_j$ denotes the objective of minimizing total tardiness.

Theorem 1. *The problem $IP, r_m//\sum_j T_j$ is NP-hard.*

Proof. Du and Leung (1990) proved that minimizing total tardiness on a single machine ($1//\sum_j T_j$) is NP-hard (in the ordinary sense). Since $IP, r_m//\sum_j T_j$ contains $1//\sum_j T_j$ as a special case, it must be NP-hard, too. □

In the following, the deterministic and static machine scheduling subproblem is formalized by a mixed integer linear program based on the model by Biskup et al. (2008). Much fewer binary variables are needed than in classical approaches (Bowman, 1959; Wagner, 1959; Manne, 1960), which affects the computation time of commercial optimization software packages like CPLEX or LINDO.

$$\min_{T_j; j=1,\ldots,J} \left\{ \sum_{j=1}^{J} T_j \right\} \tag{1}$$

subject to

$$1 \geqslant \sum_{j=1}^{J} y_{mj} \quad \forall\, m = 1, \ldots, M \tag{2}$$

$$y_{mj} \in \{0, 1\} \quad \forall\, m = 1, \ldots, M;\ j = 1, \ldots, J \tag{3}$$

$$1 = \sum_{i=1, i\neq j}^{J+1} x_{ji} \quad \forall\, j = 1, \ldots, J \tag{4}$$

$$x_{ij} \in \{0, 1\} \quad \forall\, i = 1, \ldots, J;\ j = 1, \ldots, J+1;\ i \neq j \tag{5}$$

$$1 = \sum_{m=1}^{M} y_{mj} + \sum_{i=1, i\neq j}^{J} x_{ij} \quad \forall\, j = 1, \ldots, J \tag{6}$$

$$C_j \geqslant y_{mj}(r_m + p_j) \quad \forall\, j = 1, \ldots, J;\ m = 1, \ldots, M \tag{7}$$

$$C_j \geqslant C_i + p_j - q(1 - x_{ij}) \quad \forall\, i, j = 1, \ldots, J;\ i \neq j \tag{8}$$

$$T_j \geqslant 0 \quad \forall\, j = 1, \ldots, J \tag{9}$$

$$T_j \geqslant C_j - d_j \quad \forall\, j = 1, \ldots, J \tag{10}$$

Explanation of the constraints. (2) and (3): There is at most one job that is the first to be processed on machine m. (4) and (5): Each job either precedes another job or is the last to be processed on a machine. (6): Each job is either the first to be processed on a machine or succeeds another one. (7): The completion time of the first job in the sequence of a machine must be equal to or greater than the ready time of the machine plus the processing time. (8): The completion time of a job that is not the first to be processed on a machine is equal to or greater than the completion time of its predecessor plus its processing time. (9) and (10): The tardiness of a job is calculated as the maximum of 0 and the difference between its completion time and its due date.

A second machine scheduling subproblem is needed to explain the decomposition approaches. The sole difference from the first is its objective of minimizing total completion time instead of total tardiness so it can be described by $IP, r_m//\sum_j C_j$. Due dates and tardiness are not of interest here, so Restrictions (9) and (10) can be omitted.

$$\min_{C_j\, j=1,\ldots,J} \left\{ \sum_{j=1}^{J} C_j \right\} \tag{11}$$

subject to

(2)–(8)

If all machines are ready at the same time, a simple algorithm based on shortest-processing-time priority leads to optimality (Baker, 1998). This algorithm has to be slightly modified to apply also to the case of different machine ready times. The job with the shortest processing time has to be assigned to the first machine ready. If there are several eligible jobs and/or machines, selection can be made arbitrarily. The job is scheduled without idle time on that machine's timetable. The new ready time of the machine is given by the completion time of the job. The remaining jobs are successively scheduled in the same way.

3.2. Vehicle routing with time windows

The literature on machine scheduling usually defines a due date as the latest point in time a job should be completed or delivered. The literature on vehicle routing[1] additionally refers to earliest delivery dates, which are given by the opening time of the recipient's incoming goods department, for example. Earliest and latest delivery dates form the bounds of delivery time windows. A distinction is made between soft and hard time windows. Hard time window bounds may not be violated while soft allow for early or tardy deliveries but generally involve a penalty for any violation. In order to avoid reputational damage or contractual penalties, a supplier should endeavor to ensure that all jobs are delivered within their time windows. At the same time, he should accept enough jobs to maximize his capacity utilization. The trade-off between these two competing aims, plus other factors like small time windows and widely scattered customer locations, often means that even the best possible solution will include at least one tardy delivery. To achieve a feasible solution in the first place, upper time window bounds must be seen as soft in many real world applications. Hence, the job-dependent time windows in this paper are confined by hard lower bounds $\underline{w}_j$ and soft upper bounds $\overline{w}_j$. Such time windows can be encountered in the automotive industry where the strong bargaining power of large manufacturers enables them to impose small and restrictive time windows on their suppliers. If a job arrives early, it has to remain on the vehicle until its delivery time window opens. Tardiness occurs but is generally penalized.

The jobs have to be delivered by a fleet of V heterogenous vehicles which may differ in their loading capacities c_v and ready times $\hat{r}_v$ ($v = 1, \ldots, V$). Each job must be assigned to a certain tour t ($t = 1, \ldots, J$) of one of the vehicles. A job-dependent physical size u_j, measured in the number of pallets or in weight, for example, determines the loading capacity that job j requires. Jobs cannot be split across several vehicles or different tours. Service times s_j, which correspond to the time needed for waiting, registering, quality inspection, unloading, and signing documents, need to be taken

[1] For a brief and recent introduction to the entire research area of vehicle routing and scheduling see Grunow and Stefánsdóttir (2012).

into account. The time needed to service the vehicles at the processing site is denoted by s_0.

The start time of a tour S_{vt} minus the service time at the processing site denotes the latest point in time a job must be completed in order to be delivered on that tour. Hence, this moment constitutes the due date for the parallel machine scheduling subproblem, if the vehicle routing subproblem is solved first. On the other hand, if the machine scheduling subproblem is solved first, the completion times of the jobs impose release dates on the vehicle routing subproblem.

The jobs have to be assigned to certain tours. Simultaneously, delivery routes of the tours must be defined and scheduled on the basis of the travel times between each pair of locations t_{ij} ($\forall\ i,j = 0,\ldots,J$). The delivery time D_j denotes the point in time a job arrives at its destination and begins to be serviced.

According to Desrochers et al. (1990), this capacitated vehicle routing problem with time windows can be described by $1, s_j, twhs_j, C_j/V, c_v, \hat{r}_v, mT//\sum_j T_j$. A single depot, service times, time windows with hard lower and soft upper bounds, and job release dates need to be considered ($1, s_j, twhs_j, C_j$). Furthermore, V vehicles, which may differ in their loading capacities and ready times, are available for multiple tours ($V, c_v, \hat{r}_v, mT$). The objective is to minimize total tardiness ($\sum_j T_j$).

Theorem 2. *The problem* $1, s_j, twhs_j, C_j/V, c_v, \hat{r}_v, mT//\sum_j T_j$ *is NP-hard.*

Proof. If there are solutions without tardy jobs in case of a single non-capacitated vehicle, it is an NP-complete problem to find at least one of them (Savelsbergh, 1985). The more general problem $1, s_j, twhs_j, C_j/V, c_v, \hat{r}_v, mT//\sum_j T_j$ must thus be NP-hard. □

The deterministic and dynamic vehicle routing subproblem is formalized by the following mixed integer linear program.

$$\min_{T_j; j=1,\ldots J}\left\{\sum_{j=1}^{J} T_j\right\} \tag{12}$$

subject to

$$1 = \sum_{v=1}^{V}\sum_{t=1}^{J} g_{jvt} \quad \forall j = 1,\ldots J \tag{13}$$

$$g_{jvt} \in \{0,1\} \quad \forall j = 0,\ldots J;\ v = 1,\ldots,V;\ t = 1,\ldots J \tag{14}$$

$$g_{0vt} \geqslant g_{jvt} \quad \forall j = 1,\ldots J;\ v = 1,\ldots,V;\ t = 1,\ldots J \tag{15}$$

$$q\sum_{j=1}^{J} g_{jvt} \geqslant \sum_{j=1}^{J} g_{jv,t+1} \quad \forall\ v = 1,\ldots,V;\ t = 1,\ldots J-1 \tag{16}$$

$$g_{jvt} = \sum_{i=0, i\neq j}^{J} z_{ijvt} \quad \forall j = 0,\ldots J;\ v = 1,\ldots,V;\ t = 1,\ldots J \tag{17}$$

$$g_{jvt} = \sum_{i=0, i\neq j}^{J} z_{jivt} \quad \forall j = 0,\ldots J;\ v = 1,\ldots,V;\ t = 1,\ldots J \tag{18}$$

$$z_{ijvt} \in \{0,1\} \quad \forall i,j = 0,\ldots J;\ v = 1,\ldots,V;\ t = 1,\ldots J \tag{19}$$

$$c_v \geqslant \sum_{j=1}^{J} u_j\, g_{jvt} \quad \forall\ v = 1,\ldots,V;\ t = 1,\ldots J \tag{20}$$

$$S_{v1} \geqslant \hat{r}_v + s_0 \quad \forall\ v = 1,\ldots,V \tag{21}$$

$$S_{vt} \geqslant C_j + s_0 - q(1-g_{jvt}) \quad \forall j = 1,\ldots J;\ v = 1,\ldots,V;\ t = 1,\ldots J \tag{22}$$

$$S_{vt+1} \geqslant D_j + s_j + t_{j0} + s_0 - q(1-g_{jvt}) \quad \forall j = 1,\ldots J;\ v = 1,\ldots V;\ t = 1,\ldots J-1 \tag{23}$$

$$D_j \geqslant \underline{w}_j \quad \forall j = 1,\ldots J \tag{24}$$

$$D_j \geqslant S_{vt} + t_{0j} - q(1-g_{jvt}) \quad \forall j = 1,\ldots J;\ v = 1,\ldots,V;\ t = 1,\ldots J \tag{25}$$

$$D_j \geqslant D_i + s_i + t_{ij} - q(1-z_{ijvt}) \quad \forall i,j = 1,\ldots J;\ i\neq j;\ v = 1,\ldots,V;\ t = 1,\ldots J \tag{26}$$

$$T_j \geqslant 0 \quad \forall j = 1,\ldots J \tag{27}$$

$$T_j \geqslant D_j - \overline{w}_j \quad \forall j = 1,\ldots J \tag{28}$$

Explanation of the constraints. (13) and (14): Each job must be assigned to exactly one tour of one vehicle. (15): A tour is denoted 'empty' if there is no job assigned to it. Tours with at least one job are referred to as 'active' tours. The processing site has to be included in each active tour. (16): For each vehicle it has to be ensured that there is no empty before an active tour. (17)–(19): The vehicle, delivering job j on its tth tour, travels either from another customer or from the processing site to the destination of job j. After being serviced, the vehicle returns to the processing site or delivers another job. (20): The cumulated size of all jobs delivered on the same tour may not exceed the loading capacity of the vehicle. (21): The start time of the first tour of each vehicle is equal to or greater than its ready time plus the service time needed at the processing site. (22): The start time of a tour is equal to or greater than the latest completion time of the jobs assigned to it plus the service time needed at the processing site. (23): If it is not the first tour of a vehicle, the start time is equal to or greater than the delivery time of the last delivered job on the preceding tour plus the time needed for the service at the destination of that job, for returning to the processing site, and for servicing at the processing site. (24) and (25): The delivery time of the first job on a tour is equal to or greater than the lower time window bound and the start time of the tour plus the time needed to reach the destination. (26): If a job is not the first to be delivered on a certain tour, its delivery time is equal to or greater than the sum of the delivery time of the preceding job, the service time at the previous destination, and the travel time between the two destinations. (27) and (28): The tardiness of a job is calculated as the maximum of 0 and the difference between its delivery time and its upper time window bound.

3.3. Integrated machine scheduling and vehicle routing with time windows

The two subproblems are linked and integrated by the completion times of the jobs. Due dates and thus Restrictions (9) and (10) can be omitted.

$$\min_{T_j; j=1,\ldots J}\left\{\sum_{j=1}^{J} T_j\right\} \tag{29}$$

subject to

(2)–(8) and (13)–(28).

Theorem 3. *The integrated problem is NP-hard.*

Proof. As the integrated problem is a generalization of the subproblems, the integrated problem must be NP-hard, too. □

Unless $P = NP$, it is unlikely that an algorithm exists that ensures finding global optima in polynomial bounded computation time (Garey and Johnson, 1979). Hence, heuristics must be developed and applied to obtain acceptable solutions to larger-size instances within reasonable computation time. After Section 4 introduces the instance generator, Section 5 presents two decomposition heuristics and a genetic algorithm.

4. Generation of instances

The instance generator and the genetic algorithm make use of the linear congruential pseudo random numbers generator introduced by Lehmer (1951). Biskup and Feldmann (2001) provide a brief description.

The processing times p_j are generated as follows:

$$p_j \sim RD[1,\rho] \quad \forall\ j = 1,\ldots,J \tag{30}$$

RD is an abbreviation for rectangular distribution, which means that all integers of the interval $[1,\rho]$ are equally probable.

The job sizes u_j are assumed to depend on the processing times:

$$u_j \sim RD[1, p_j] \quad \forall\, j = 1, \ldots, J \tag{31}$$

The longer the processing time, the greater the probability that the job needs a lot of space. This relationship frequently holds true in practice, such as in beverage bottling, newspaper printing, and injection molding, for example.

To allow for the existence of a feasible solution, it is necessary that at least one vehicle has enough capacity to deliver the job with the largest size: $\max_{j;j=1,\ldots,J}\{u_j\}$. Companies need to comply with this constraint. The capacities of the vehicles c_v are therefore matched with the job sizes, which, in practice, should be alike:

$$c_v = \max_{j;j=1,\ldots,J}\{u_j\} + \tau_v \text{ with } \tau_v \sim RD\left[0, \mu \max_{j;j=1,\ldots,J}\{u_j\}\right] \quad \forall\, v = 1, \ldots, V \tag{32}$$

The smaller the adjustment parameter μ, the smaller the capacities of the vehicles in relation to the job sizes.

In order to generate the matrix of travel times, the job destinations, one destination for each job, are first randomly allocated in a two-dimensional area around the processing site. As is common practice, the rounded integer values of the Euclidian distances can also be interpreted as travel times t_{ij} ($\forall\, i,j = 0, \ldots, J$) with $t_{ij} = t_{ji}$ and $t_{ij} = 0$ for $i = j$. When allocating the destinations around the processing site, it has to be ensured that the maximum travel time does not exceed $\lfloor \rho\,(V/M) \rfloor$. The floor function $\lfloor \cdot \rfloor$ denotes the largest integer value lower than or equal to $\rho\,(V/M)$. Integrated planning is easy if the production and transportation capacities, suffice to immediately process and deliver all jobs. However, since competition generally causes the need for efficient management, the numbers of vehicles and machines are mostly reduced to a minimum and well-matched to each other in practice. Here, the capacities are balanced out by reasonably including the vehicles (V), the machines (M), and the processing times (ρ) in the generation of the travel times.

The service times s_j are also matched to the processing times:

$$s_j \sim RD[1, \lfloor \lambda\ \rho \rfloor] \quad \forall\, j = 0, \ldots, J \tag{33}$$

The machine that is the first ready to process is labeled machine 1. Its ready time is scaled to $r_1 = 0$. The ready times of the other machines are:

$$r_m \sim RD[0, \rho] \quad \forall\, m = 2, \ldots, M \tag{34}$$

Since a machine could start processing its last job of the previous set of jobs at the moment machine 1 is ready for the new set, the upper limit of the interval is the maximum processing time ρ.

The latest machine ready time plus the service time at the processing site denotes the earliest moment at which the last job of the previous set can start to be delivered. The vehicle delivering that job is labeled vehicle 1. Its ready time is:

$$\hat{r}_1 = \max_{m;m=1,\ldots,M}\{r_m\} + s_0 + \gamma \text{ with } \gamma \sim RD[3, \lfloor \rho\ (V/M) \rfloor + \lfloor \lambda\ \rho \rfloor] \tag{35}$$

Since the last job of the previous set is assumed to be delivered in a one-job-shipment, the vehicle needs at least three and at most $\lfloor \rho\ (V/M) \rfloor + \lfloor \lambda\ \rho \rfloor$ time units to travel to the destination, service the job, and return to the processing site.[2] The ready times of the other vehicles are drawn from an interval with the maximum ready time of vehicle 1 as limit:

$$\hat{r}_v \sim RD[0, \max_{m;m=1,\ldots,M}\{r_m\} + s_0 + \lfloor \rho\ (V/M) \rfloor + \lfloor \lambda\ \rho \rfloor] \ \forall\ v = 2, \ldots, V \tag{36}$$

Finally, the lower and upper time window bounds need to be generated. To avoid unsatisfiable cases, the earliest moment a job can be delivered, given by the processing time plus the service time at the processing site and the travel time from the processing site to the destination, is the minimum lower bound. These minimum lower bounds are enlarged by a random integer:

$$\underline{w}_j = p_j + s_0 + t_{0j} + \pi_j \quad \text{with} \quad \pi_j \sim RD[0, \lfloor \delta_1\ \rho\, J/(M+V) \rfloor] \ \forall\, j = 1, \ldots, J \tag{37}$$

The smaller the adjustment parameter δ_1, the tighter the lower bounds. Small values of δ_1 produce time windows close to the planning moment and to each other, so that there is little time to produce and deliver the jobs. By considering the factor $J/(M+V)$, the number of jobs, machines, and vehicles can be varied without making major changes to the relative tightness of the instances. The sizes of the time windows, which are defined by the lower and upper bounds, are adjusted to ρ and thus also to the processing times:

$$\overline{w}_j = \underline{w}_j + \kappa_j \text{ with } \kappa_j \sim RD[0, \lfloor \delta_2\ \rho \rfloor] \quad \forall\, j = 1, \ldots, J \tag{38}$$

The following small-size instance with seven jobs, two machines, and two vehicles ($J = 7, M = 2, V = 2$) was obtained from the instance generator using the parameter settings $\rho = 100$, $\mu = 1$, $\lambda = 0.2$, $\delta_1 = 0.5$, and $\delta_2 = 0.5$. The vector notation is explained by the processing times: $p = (p_1, p_2, \ldots, p_J)$.

$p = (48, 30, 54, 84, 95, 23, 46)$ $\quad u = (32, 15, 7, 42, 54, 18, 25)$

$\underline{w} = (133, 120, 142, 197, 194, 122, 94)$ $\quad \overline{w} = (152, 143, 176, 214, 225, 168, 131)$

$s = (1, 1, 7, 1, 11, 15, 5, 13)$ $\quad c = (77, 57)$

$r = (0, 28)$ $\quad \hat{r} = (58, 123)$

$$(t) = \begin{pmatrix} t_{11} & t_{12} & \ldots & t_{1J} \\ t_{21} & & & \\ \vdots & & \ddots & \vdots \\ t_{J1} & & \ldots & t_{JJ} \end{pmatrix} \qquad (t) = \begin{pmatrix} 0 & 19 & 32 & 45 & 43 & 34 & 41 & 5 \\ 19 & 0 & 15 & 37 & 24 & 21 & 43 & 22 \\ 32 & 15 & 0 & 28 & 19 & 8 & 41 & 34 \\ 45 & 37 & 28 & 0 & 46 & 19 & 22 & 43 \\ 43 & 24 & 19 & 46 & 0 & 27 & 61 & 47 \\ 34 & 21 & 8 & 19 & 27 & 0 & 34 & 35 \\ 41 & 43 & 41 & 22 & 61 & 34 & 0 & 36 \\ 5 & 22 & 34 & 43 & 47 & 35 & 36 & 0 \end{pmatrix}$$

Fig. 1 visualizes the optimal solution to the example, computed by a commercial optimization software. Dark gray rectangles illustrate the intervals during which machine 2 (M2) and the vehicles (V1 and V2) are not available. Both vehicles are employed for two tours whose start times are indicated by dotted lines. The capacity utilization of a vehicle is quoted to the left of the dotted lines. Moreover, service times are scheduled to the left of the dotted lines and between the arrows that illustrate travel times. The time windows (TW) of the jobs are depicted by pale gray rectangles. Jobs 2, 4, and 5 are tardy and cause a total tardiness of 70. This example shows that, using the proposed settings, the generator is able to provide reasonable instances.

5. Heuristics

5.1. Decomposition approaches

This subsection introduces two decomposition approaches that tackle the subproblems of Section 3 one after the other and merge the resulting solutions into an overall solution to the integrated problem. Note that even solving the subproblems optimally does not guarantee an overall optimum.

[2] Remember that the job destinations are randomly allocated around the processing site and that the maximum travel time between two destinations $\max_{i,j;i,j=1,\ldots,J}\{t_{ij}\}$ does not exceed $\lfloor \rho\ (V/M) \rfloor$. Consequently, the maximum travel time from the processing site to a destination $\max_{j;j=1,\ldots,J}\{t_{0j}\}$ cannot exceed $1/2 \cdot \lfloor \rho\ (V/M) \rfloor$. Hence, at least $2 \cdot 1 = 2$ and at most $2 \cdot 1/2 \cdot \lfloor \rho\ (V/M) \rfloor = \lfloor \rho\ (V/M) \rfloor$ periods are needed to shuttle between the processing site and a destination. Furthermore, the service at the destination needs at least one and at most $\lfloor \lambda\ \rho \rfloor$ periods.

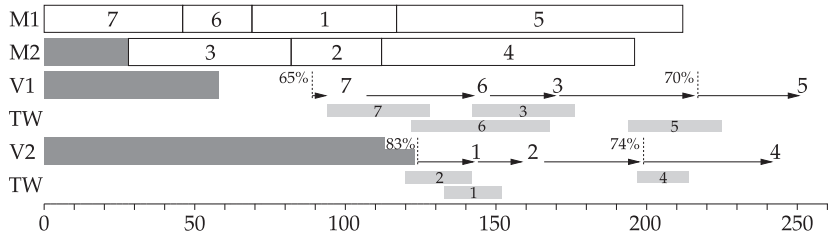

Fig. 1. The optimal solution to the example illustrated by a Gantt chart.

5.1.1. Approach 1: VMV

In brief, this approach consists of firstly vehicle routing, secondly machine scheduling, and thirdly vehicle routing, so it is referred to as VMV. It proceeds as follows:

1. Set: $C_j^{\text{Step1}} := p_j \ (\forall j = 1, \ldots, J)$
   Since the first moment at which processing begins is 0, no job can be completed earlier than its processing time.
2. Solve: $1, s_j, twhs_j, C_j^{\text{Step1}} / V, c_v, \hat{r}_v, mT // \sum_j T_j$ (Section 3.2)
   To obtain the best possible delivery schedule, solve the vehicle routing subproblem with respect to the completion times resulting from step 1.
3. Set: $d_j := S_{vt} - s_0$ for $g_{jvt} = 1 \ (\forall\, j = 1, \ldots, J)$
   Calculate the due dates for machine scheduling from the solution of step 2.
4. Solve: $IP, r_m // \sum_j T_j$ (Section 3.1, machine subProblem 1)
   Search for that machine schedule which is best in meeting the due dates calculated in step 3. If there is a feasible machine schedule that complies with all due dates, the optimum of the integrated problem is found and can be composed of the solutions produced in the second and fourth step. Otherwise → step 5.
5. Solve: $1, s_j, twhs_j, C_j^{\text{Step4}} / V, c_v, \hat{r}_v, mT // \sum_j T_j$ (Section 3.2)
   Solve the vehicle routing subproblem again, but now take account of the feasible completion times resulting from step 4.
6. Aggregate the solutions of step 4 and 5 to form a feasible solution to the integrated problem.

Computations have revealed that more iterations (VMVMV, VMVMVMV, etc.) do not significantly improve the results. Applying this approach to the instance given in Section 4 leads to a total tardiness of 86.

5.1.2. Approach 2: MV

In brief, this approach consists of firstly machine scheduling and secondly vehicle routing, so it is referred to as MV. It proceeds as follows:

1. Solve: $IP, r_m // \sum_j C_j$ (Section 3.1, machine subproblem 2)
   The aim of minimizing total completion time tends to cause that at each point in time as many jobs as possible are completed (Conway et al., 1967). The intention behind it is to provide the subsequent vehicle routing subproblem with great degrees of freedom.
2. Solve: $1, s_j, twhs_j, C_j^{\text{Step1}} / V, c_v, \hat{r}_v, mT // \sum_j T_j$ (Section 3.2)
   Solve the vehicle routing subproblem with respect to the feasible completion times resulting from step 1.
3. Aggregate the solutions of steps 1 and 2 to produce a solution to the integrated problem. Feasibility is ensured.

Applying this approach to the example also results in a total tardiness of 86.

5.2. Genetic algorithm approach

Genetic algorithms (Holland, 1975; Goldberg, 1989; Davis, 1991; Gendreau and Potvin, 2010) are stochastic search methods which are part of the evolution-motivated meta-heuristics. A population of individuals, given by encoded solutions to the considered problem, is iteratively altered. Each new individual emerges from crossover, mutation, or reproduction of the current individuals. Genetic algorithms simulate the Darwinian principle of survival of the fittest. The lower an individual's objective value (in the case of a minimization problem), the better its fitness and in turn, the higher the probability it will influence the next generation of individuals. In this way, the population is intended to gradually converge towards the optimum.

The great advantage of genetic algorithms is that they have no need for continuity, differentiability, and convexity of the objective function. Furthermore, genetic algorithms can be relatively easily adjusted to almost every linear and non-linear problem. Genetic algorithms with an elitist strategy have been proven to find the global optimum even if the restrictions or the objective function include non-smooth operators such as *IF*, *MIN*, *MAX*, and *ABS* functions (Suzuki, 1995; Lozano et al., 1999). However, the sun could explode before this happens (Albright and Winston, 2012) and, unless the optimum is verified by some other means, one never knows whether optimality is already reached. Hence, genetic algorithms are rather applied as heuristics, reporting the best solution found after a pre-defined runtime or a certain number of iterations without improving the solution, for example.

Justified by the above mentioned advantages and the successful applications summarized in the literature review, this paper's integrated problem is tackled by a genetic algorithm. Fig. 2 illustrates the proposed genetic algorithm on a meta-level. First, a predefined number of arbitrary feasible solutions is generated in order to build the initial population which is the first current population. How to generate arbitrary feasible solutions is later explained in conjunction with the mutation procedure. The fitness of each individual has to be calculated in the next step. Aiming to minimize total tardiness, an individual with a low objective value obj_{ind} should have a high fitness. This is achieved by defining the fitness fit_{ind} of an individual *ind* by $fit_{ind} = 1/obj_{ind}$.[3] The fittest individual of a generation is compared to the best individual found before and saved if it is better. After a predefined number of new generations has been built, the pale gray aborting criterion is answered with 'yes' and the genetic algorithm stops. The best individual found provides the solution. If the pale gray aborting criterion is answered with 'no', a new generation is created.

The genetic algorithm applies an elitist strategy which means that the currently best individual is assigned to each new generation in order for it to change for the better. All other new

[3] Ensure that the genetic algorithm stops if an individual with an objective value of 0 is found.

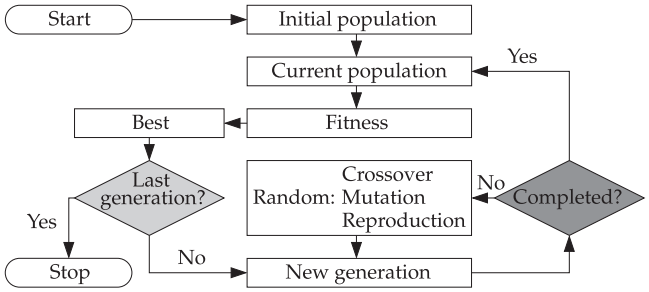

Fig. 2. The major cycle of the genetic algorithm.

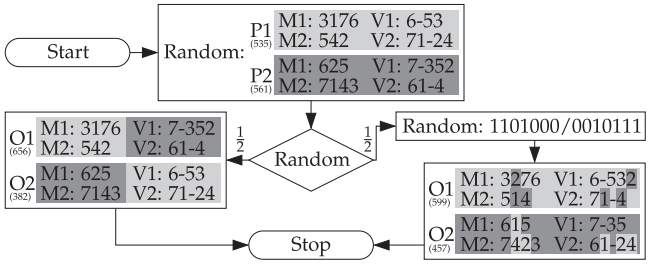

Fig. 3. The crossover procedure visualized by an example.

individuals are successively created by either crossover, mutation, or reproduction of the current population's individuals. Predefined probabilities, summing up to 100%, influence the random choice between those three procedures. The dark gray aborting criterion is answered with 'yes' if both populations have the same number of individuals. At this moment, the new generation replaces the current population, a process referred to as general replacement. Now, the next major cycle of the genetic algorithm begins again with the calculation of the individuals' fitness values.

The individuals are implemented by permutation coding with path representation which is generally assumed to be the method of choice in case of sequencing problems. This type of coding and the way an individual's fitness can be determined from its code is explained by parent 1 (P1) in Fig. 3. The instance introduced in Section 4 provides the data. Jobs 3, 1, 7, and 6 (5, 4, and 2) are processed in sequence on M1 (M2). It is obvious that only active machine schedules without idle time need to be considered when minimizing total tardiness. Taking account of the machine ready times and the processing times of the jobs, the following completion times can be easily obtained from the code of P1: $C = (102,237,54,207,123,171,148)$. The part of the code that relates to the vehicles is similarly interpretable. V1 (V2) delivers job 6 (7 and 1) on its first tour and jobs 5 and 3 (2 and 4) in sequence on its second tour. The earliest possible delivery dates, which are of interest when minimizing total tardiness, unambiguously result from the vehicle ready times, the completion times of the jobs, the travel times, the lower time window bounds, and the service times: $D = (189,270,328,296,294,213,154)$. The tardiness of a job is determined by the maximum of 0 and the difference between its earliest possible delivery date and its upper time window bound: $T = (37,127,152,82,69,45,23)$. Accordingly, P1 has a total tardiness of 535 and a fitness of 1/535.

In Fig. 2, the rectangle that encloses the procedures 'crossover', 'mutation', and 'reproduction' represents the core of the genetic algorithm. Fig. 3 is used to describe the crossover procedure. Initially, two parents have to be chosen from the current population using the commonly applied roulette wheel selection method (Iba et al., 2009). The probability that an individual is a parent is given by its relative fitness: fit_{ind}/Fit. Fit denotes the cumulated fitness of the current population: $Fit = \sum_{ind} fit_{ind}$. These parents are altered using one of two different crossover methods. Both methods are equally probable. As visualized by the pale and the dark gray rectangles, the machine and vehicle codes of the two parents are rearranged en bloc if the left-hand method is chosen. In doing so, promising machine codes are given the chance of being matched up with better fitting vehicle codes and vice versa. Offspring 1 (O1) receives the machine code of P1 and the vehicle code of P2, leading to a greater total tardiness (656) than either of its parents (535 and 561). However, O2 (382), which receives the remaining codes, is better than both parents. Note that crossover is the only one of the three core procedures that generates two offspring at once.[4]

The right-hand crossover method starts with randomly generating a sequence of J equally probable ones and zeros. The resulting binary mask (Reeves, 2010) serves as an instruction for combining O1 from the parents' codes. For example, in Fig. 3 the third value of the left-hand binary mask 1101000 is zero which means that job 3 is assigned to the machine and vehicle position of P1. Hence, O1 inherits from P1 that job 3 is the first job processed on M1 and the second job delivered on the second tour of V1. If the third value was 1, job 3 would be assigned to the machine and vehicle position of P2. The machine position comprises the number of the machine processing the job and the position of the job in the sequence of that machine. The vehicle position reflects the number of the vehicle the job is assigned to, the number of the tour, and the position of the job in the delivery sequence of that tour. The jobs are placed successively on their new positions in ascending order according to their indices. The right-hand binary mask 0010111, which is the 'inverse' of the left, serves as an instruction for combining O2. The pale gray (P1) and dark gray (P2) background colors illustrate which elements of an offspring originate from which parent. O2 (457) outperforms both parents. Jobs with good matching machine and vehicle positions have a chance to be combined with other promising ones by the right-hand crossover method. However, this method could cause two jobs to be assigned to the same machine or vehicle position, that machine or vehicle positions remain free although the following positions are occupied, and that jobs batched to one tour exceed the loading capacity of the vehicle. In these cases, fixing mechanisms take effect:

- If the machine position of a job is already occupied, the job is passed onto the next free position in front. If all positions in front are occupied, it is assigned to the next free position following.
- If the remaining capacity of a vehicle is insufficient for accommodating the next job assigned to a certain tour, the job is passed onto the next possible tour in front. The position in the delivery sequence remains unchanged. If there is no tour in front with enough remaining capacity, it is assigned to the next possible tour following.
- If the vehicle position of a job is already occupied, the job is passed onto the tour's next free position in front. If all positions in front are occupied, it is assigned to the tour's next free position following.
- If there are free machine and vehicle positions in front of occupied ones after all jobs are placed, the free positions are deleted.

Fig. 4 exemplifies the mutation procedure. Again, two different methods take effect. The left-hand mutation method, which randomly creates completely new individuals, is also used to generate the initial population. Its purpose is to bring new chromosomes

[4] If only one individual is needed to complete a new generation and if this individual is chosen to be generated by crossover, only the first of the two offspring is assigned to the new generation.

into the population and thus to avoid a premature convergence. Since the right-hand method, comprising mutation in its proper meaning, is more important, the left-hand method is applied with a probability of only $\frac{1}{J}$. The left-hand mutation method works as follows. Choose a job randomly and place that job in the queue of a machine that is also randomly chosen. This is repeated until all jobs occupy a certain machine position. A similar approach is applied to generate the vehicle code. First, chose a job and a vehicle randomly, but ensure that the capacity of the vehicle suffices to carry out the job. Then, assign the job randomly to one of the vehicle's already active tours with enough remaining capacity or an empty last tour. The job is placed in the queue of that tour. Note that, if no job was assigned to this vehicle before, the empty last tour is equal to the first tour. This is repeated until all jobs occupy a certain vehicle position. In Fig. 4, the individual generated by the left-hand mutation method gains a total tardiness of 611. The dark gray background indicates that each machine and vehicle position is newly generated.

The right-hand mutation method starts with the choice of a parent from the current population. In the example, the chosen parent is the second offspring obtained from the left-hand crossover method in Fig. 3. Then, it must be decided whether to mutate the machine or the vehicle code of a randomly selected job (here, job 4). The probabilities $\frac{M}{M+V}$ and $\frac{V}{M+V}$ depend on the numbers of machines and vehicles. If there are many machines in relation to vehicles, meaning that there is a relatively large number of possible machine-job permutations, the machine code is expected to be more often mutated. The same consideration, but vice versa, holds for the case of many vehicles in relation to machines.

In order to mutate the machine code, the selected job first needs to be deleted from the machine code. Here, job 4 and the resulting empty position have been deleted from the sequence of M2. Then, re-allocate that job randomly to a possible position in the sequence of an also randomly chosen machine. In the example, job 4 now occupies the fourth of four possible positions in the sequence of M1. The offspring achieves a total tardiness of 280. If job 4 was assigned to the second position, it would displace job 2 to the third and job 5 to the fourth position.

The mutation of the vehicle code proceeds in a similar manner. The selected job is deleted from the vehicle code, a vehicle is randomly chosen taking account of the size of the job, then the job is randomly assigned to a possible position in one of the active tours with enough remaining capacity or an empty last tour. Here, job 4 has been re-allocated to the first and sole possible position in the sequence of the empty third tour, leading to a total tardiness of 395.

Reproduction is the third and last procedure of the genetic algorithm. Reproduction means that a parent is chosen and without modification passed onto the new generation. Note that an individual can be assigned twice or more often to the new generation. Since this is expected to happen more to individuals with high fitness values, data redundancy is accepted as part of convergence.

6. Computational results

Although there has been a lot of research on genetic algorithms, no standardized way of adjusting the parameters has been developed so far (Aytug et al., 2003). Suitable parameter adjustments for certain applications must still be found by trial and error (Pongcharoen et al., 2002). For this reason, a commercial optimization software is used to calculated optimal solutions to small-size instances. By comparing the results of the optimization software with the solutions obtained for varying parameter adjustments, an appropriate parameter adjustment for the genetic algorithm is identified. Section 6.1 describes this approach in detail and furthermore compares the genetic algorithm to the decomposition approaches introduced in Section 5. Section 6.2 investigates the performance and computation time of the genetic algorithm for large-size instances.

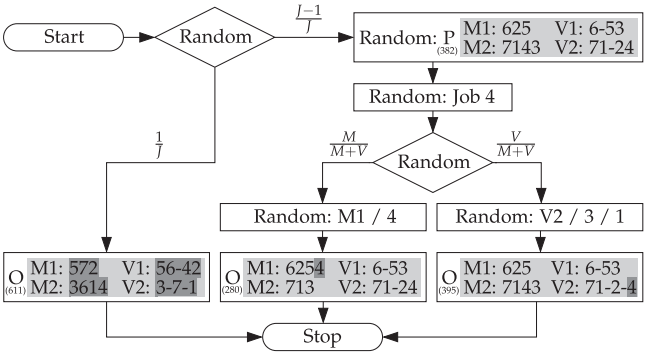

Fig. 4. The mutation procedure visualized by an example.

6.1. Small-size instances

90 instances with seven jobs ($J = 7$), two machines ($M = 2$), and two vehicles ($V = 2$) are examined ($\rho = 100$, $\mu = 1, \lambda = 0.2, \delta_1 = 0.5, \delta_2 = 0.5$). The mean computation time of the optimization software is 28.306 hours whereas the genetic algorithm, implemented in Turbo Delphi, needs only seconds for 10^5 generations. All computations were performed on one core of an Intel Core 2 Duo CPU with 2.33 Gigahertz and 2 Gigabyte RAM. Commercial optimization software is rarely able to find optima for more jobs, machines, or vehicles within an acceptable time. The study on small-size instances is hence limited to these relatively low numbers.

Fig. 5 illustrates the relative deviations from the optimum for the decomposition approaches (VMV and MV) and for different parameter adjustments of the genetic algorithm. The first entry in the parentheses below the boxplots shows the probability of crossover, the second of mutation, and the third of reproduction. The four probability constellations (45,45,10), (60,30,10), (30,60,10), and (0,90,10) are tested with 10^2, 10^3, 10^4, and 10^5 generations, each generation consisting of 100 individuals. Since the subproblems arising in case of the decomposition approaches are optimally solved, deviations from the overall optimum can be unambiguously ascribed to decomposition. There are no distorting influences caused by the application of heuristics.

The boxplots suggest that VMV outperforms MV. Both the median and the standard deviation are lower. However, compared to the genetic algorithm, both decomposition approaches produce poor results. The probability constellation (30,60,10) seems to perform best, especially for large numbers of generations where around half of the 90 instances are optimally solved and the standard deviation is only 4.7. Remember that in contrast to mutation and reproduction, each time crossover is chosen, not only one but two offspring are generated. So the constellation (30,60,10) is expected to generate as many offspring by crossover as by mutation. To deflect any criticism that this constellation may only perform well due to its high mutation-probability (60%), the results of the constellation (0,90,10), which lead to the poorest results by far, are also presented. Accordingly, mutation as well as crossover need to be applied to achieve a good performance. Note that (30,60,10) is not necessarily the best constellation at all, but the best of all those investigated. For many types of problem, the genetic algorithm implemented in Microsoft Excel 2010, for example, is also

found to perform best with a high probability of mutation (Albright and Winston, 2012).

Table 3 presents the average results for three other small-size combinations of the number of machines and vehicles: $(M = 1, V = 1)$, $(M = 1, V = 2)$, and $(M = 2, V = 1)$. Using the above mentioned generator setting, 30 instances are created and solved for each of them. The column headed by 'Genetic algorithm' contains the average deviations from the optimum after 10^5 generations with the probability constellation (30,60,10). Note that this constellation turns out to perform best during the whole numerical study. For the sake of clarity and to save space, the following section discusses only the solutions after 10^5 generations with the probability constellation (30,60,10). Comparing the results for $(M = 1, V = 1)$ to $(M = 2, V = 1)$ and for $(M = 1, V = 2)$ to $(M = 2, V = 2)$ suggests that an increasing number of machines and thus a larger search space ceteris paribus causes a deterioration in the solutions obtained by the decomposition approaches and in the solutions of the genetic algorithm. The same generally applies to the number of vehicles. An attempt was made to find other factors that influence the hardness of the instances. Except for the parameter δ_1, which indicates the tightness of the instances, all investigated factors turned out to be non-significant. The investigation of large-size instances therefore mainly focuses on δ_1 and the number of machines, vehicles, and jobs.

6.2. Large-size instances

Due to the NP-hard nature of the integrated problem, the optimization software is not able to solve large-size instances within a reasonable time. Moreover, since the integrated problem has not been investigated so far, there is no benchmark heuristic that can be compared with the genetic algorithm. For lack of a suitable heuristic approach to the vehicle routing subproblem, the decomposition approaches cannot be applied either. Instead, the integrated problem is reduced to two simpler integrated problems which already have been successfully tackled by heuristics. For a given instance, these problems and the respective heuristics can be used to approximate a lower and an upper bound for the optimum. With the help of these bounds, it becomes possible to assess the accuracy of the genetic algorithm for large-size instances.

- *The lower bound problem and heuristic:* Suppose that there is a sufficiently large number of vehicles so that each job can be delivered to its destination immediately after it is completed. Furthermore, let the due date of a job be given by its upper time window bound minus the service time at the processing site and the travel time to its destination: $d_j := \overline{w}_j - s_0 - t_{0j}$ $(\forall\, j = 1, \ldots, J)$. If the completion time of a job is equal to or lower than that due date, it is ensured that the job meets its time window. The integrated problem is thereby reduced to the first machine scheduling problem of Section 3.1: $IP, r_m // \sum_j T_j$. It is obvious that an optimal solution to this problem provides a lower bound for the optimal solution to the original problem.
  Since this and the following upper bound problem are still NP-hard, heuristics must be applied to solve large-size instances. Biskup et al. (2008) provide the currently best heuristic for the problem $IP // \sum_j T_j$ which is $IP, r_m // \sum_j T_j$ with $r_m = 0$ $(m = 1, \ldots, M)$. Their heuristic is used to approximate the lower bounds.
- *The upper bound problem and heuristic:* Now suppose that, as in the original problem, there are V vehicles, however, the vehicles again deliver only one job at a time. This assumption leads to shuttle shipments with job-dependent durations comprising the service time at the processing site, the travel time to the destination, the service time at the destination, and the travel time back to the processing site: $s_0 + 2\ t_{0j} + s_j$ $(\forall\ j = 1, \ldots, J)$. The reduced problem can be interpreted as a hybrid flow-shop with M identical parallel machines at the first and V identical parallel machines at the second stage. Each machine m and v has its own ready time r_m and $\hat{r}_v$, respectively $(m = 1, \ldots, M, v = 1, \ldots, V)$. The processing times at the first

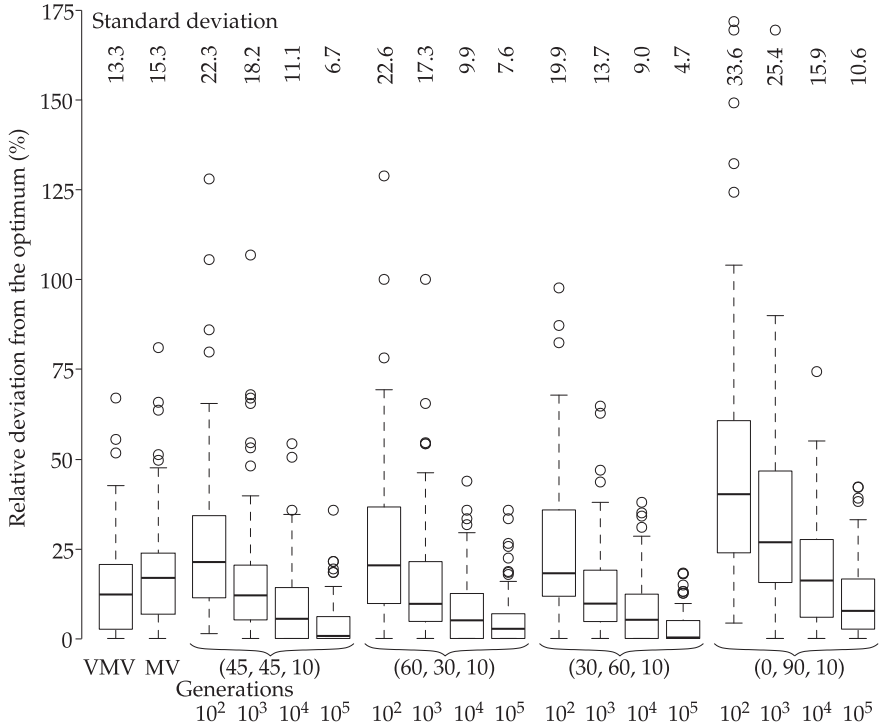

Fig. 5. The performance of the heuristics for small-size instances.

Table 3
Performance results for varying numbers of machines and vehicles.

Number of machines and number of vehicles	Mean relative deviation from the optimum (%)		
	VMV	MV	Genetic algorithm
$M = 1, V = 1$	7.9	18	0.7
$M = 1, V = 2$	13	13.8	2.2
$M = 2, V = 1$	8.9	17.1	2.2
$M = 2, V = 2$	13.8	18	3.2

and at the second stage are $p_{1j} := p_j$ and $p_{2j} := s_0 + 2\ t_{0j} + s_j$, respectively ($\forall\ j = 1, \ldots, J$). The completion time at the second stage C_{2j}, which describes the moment a vehicle arrives at the processing site after having delivered a job, must be greater than or equal to the lower time window bound plus the service time at the destination and the travel time back to the processing site: $\underline{w}_j + s_j + t_{0j} \leqslant C_{2j}$ ($j = 1, \ldots, J$). This restriction ensures that the delivery time of job j, $D_j = C_{2j} - s_j - t_{0j}$, is equal to or greater than the lower time window bound: $\underline{w}_j \leqslant D_j$ ($j = 1, \ldots, J$). Let the tardiness of a job be $T_j = \max\{0, D_j - \bar{w}_j\}$ ($\forall\ j = 1, \ldots, J$) then the problem can be described by $HFS, IP^{1,2}, r_m, \hat{r}_v / \underline{w}_j \leqslant D_j / \sum_j T_j$ where *HFS* is an abbreviation for hybrid flow-shop scheduling. $IP^{1,2}$ means that there are identical parallel machines at stage 1 as well as at stage 2. Optimal solutions to this problem provide upper bounds for the optima of the original problem.

Brah (1996) compares some dispatching rules for a multiple stage hybrid flow shop problem with due dates. It turns out that a rule based on modified due dates performs very well. Hence, the two-stage hybrid flow-shop upper bound problem is tackled by the following dispatching rule. Schedule the job with the currently smallest modified due date in such a way that the job is delivered as early as possible. The modified due date is calculated as the maximum of the earliest possible delivery date and the upper time window bound. When scheduling a job, ensure that the availability of the machines, which changes with each job that is added to the schedule, and the restriction $\underline{w}_j \leqslant D_j$ are taken into account. It may be necessary to insert idle time at the second stage. If two or more machines are eligible, choose one arbitrarily. The same holds if two or more jobs have the same modified due date. Apply this rule successively until all jobs are scheduled. Note that, unlike the lower bound solutions, the upper bound solutions are feasible to the original problem.

The lower bound, the upper bound, and the genetic algorithm's solution can be aggregated to the following performance ratio:

$$\frac{\text{solution of the genetic algorithm} - \text{lower bound}}{\text{upper bound} - \text{lower bound}} \tag{39}$$

A ratio of 0.50, for example, means that the solution of the genetic algorithm is exactly in the middle of the interval defined by the lower and the upper bound. Note that the upper bound solutions are henceforth assigned to the initial population of the genetic algorithm so the genetic algorithm is at least as good as the upper bound. Consequently, the worst possible performance ratio is 1.

The 90 small-size instances of the previous subsection ($J = 7$, $M = 2$, and $V = 2$) are again investigated in order to calculate benchmark performance ratios. Without benchmark ratios it would be impossible to assess whether a performance ratio of, say, 0.75 is good or bad. Remember that the optimization software is able to optimally solve small-size instances, so the lower and upper bound problems can be tackled by the heuristics described in this subsection as well as the optimization software. This provides the additional opportunity to validate the accuracy of the lower and upper bound heuristics. Both the solutions of the genetic algorithm and the overall optimum obtained by the optimization software are inserted in Formula (39). Table 4 presents the mean ratios for each of the four possible combinations. The optimal solutions to the original problem together with the optimal solutions to the lower and upper bound problems result in an exact mean performance ratio of 0.43. Note that calculating the ratio with the solutions of the genetic algorithm and the bounds obtained by optimization software lead to an average ratio of 0.45 which comes very close to the exact ratio. The row 'Heuristic bounds' reveals that the mean performance ratios are generally around 0.05 higher if the lower and upper bound problems are solved with the above proposed lower and upper bound heuristics, respectively. This may be partly due to inaccuracy of the heuristics, but also arise from the fact that the lower bound heuristic does not consider the machine ready times. The optimization software is able to take the ready times into account which causes higher lower bounds and thus lower performance ratios. To sum up, the results for small-size instances recommend a performance ratio of 0.50 as benchmark for the following investigations.

Table 5 presents the mean performance ratios for various values of the number of machines M (3,4,5), the number of vehicles V (4,6,8,10), the number of jobs J (30,50), and the tightness parameter δ_1 (1,0.5). 100 instances are investigated for each combination, resulting in 4800 large-size instances in total. The main observations are summarized below.

- Just about acceptable, the worst average ratio in Table 5 is 0.71 ($J = 50, M = 5, V = 10, \delta_1 = 0.5$), which suggests that the genetic algorithm provides relatively good solutions for instances with up to 50 jobs, 5 machines, and 10 vehicles.

Table 4
Mean performance ratios calculated for small-size instances.

	Optimal solution	Genetic algorithm
Optimal bounds:	0.43	0.45
Heuristic bounds:	0.48	0.50

Table 5
The genetic algorithm's performance for large-size instances.

V	δ_1	$M = 3$		$M = 4$		$M = 5$	
		$J = 30$	$J = 50$	$J = 30$	$J = 50$	$J = 30$	$J = 50$
4	1	0.51	0.57	0.51	0.57	0.50	0.58
	0.5	0.58	0.57	0.59	0.59	0.61	0.60
6	1	0.54	0.58	0.55	0.58	0.54	0.59
	0.5	0.63	0.62	0.64	0.63	0.64	0.64
8	1	0.55	0.61	0.57	0.60	0.58	0.61
	0.5	0.65	0.65	0.66	0.65	0.68	0.67
10	1	0.58	0.61	0.57	0.63	0.59	0.64
	0.5	0.64	0.67	0.68	0.69	0.69	0.71

Table 6
Aggregated average performance ratios for large-size instances.

Vehicles	∅ ratio	δ_1	∅ ratio	Machines	∅ ratio	Jobs	∅ ratio
4	0.565	1	0.573	3	0.598	30	0.595
6	0.598	0.5	0.641	4	0.607	50	0.619
8	0.623			5	0.617		
10	0.642						

Table 7
The impact of the number of job destinations on the performance ratio.

Number of job destinations	1	2	5	10	25	50
Genetic algorithm	0.34	0.45	0.56	0.64	0.68	0.71

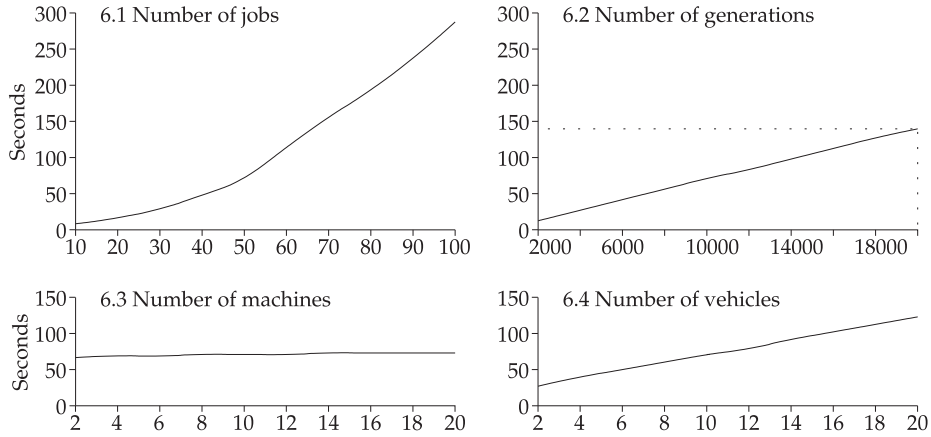

Fig. 6. Computation time of the genetic algorithm.

- Comparing the columns for $J = 30$ and $J = 50$ indicates that an increasing number of jobs generally has a negative impact on performance. Note that the performance ratios for 70 job-instances, which are not reported here for the sake of clarity and to save space, are often close to 1, meaning that the genetic algorithm is no longer able to yield good solutions. The average ratio for the hardest examined problem setting with 70 jobs, 5 machines, 10 vehicles, and $\delta_1 = 0.5$, for example, is even 0.99.

Table 6 aggregates the results of Table 5. For example, half of the 4800 instances deal with 30 jobs. The average performance ratio of these 2400 instances is 0.595 which is recorded in the last column. As a second example, the mean ratio of the 1200 instances with 8 vehicles is 0.623.

- Checking the mean ratio of the 30 job-instances against the mean ratio of the 50 job-instances confirms that an increasing number of jobs negatively affects performance. The same generally holds for the number of vehicles and machines.
- Moreover, comparing the aggregated results for $\delta_1 = 0.5$ and $\delta_1 = 1$ suggests that the genetic algorithm produces better results if the time windows are less close to the planning moment and to each other.

In the default setting, the instance generator creates an individual destination for each job. Table 7 deals with the special case that several jobs need to be shipped to the same destination. Therefore, 50 jobs are assigned to either 1, 2, 5, 10, 25, or 50 different job destinations. For example, in the scenario dealing with 25 destinations, each destination belonged to two jobs (50/25 = 2). The other settings are held constant on $\delta_1 = 0.5$, $M = 5$, and $V = 10$. For each scenario, 100 instances are investigated. Note that the results for 50 job destinations are taken from Table 5 since this is the default scenario where each job has its own destination.

- Table 7 reveals that an increasing number of job destinations has a negative impact on the performance of the genetic algorithm.
- A further investigation of 100 instances with $J = 70$, $M = 5$, $V = 10$, $\delta_1 = 0.5$, and only one destination results in a mean performance ratio of 0.72, showing that the genetic algorithm can handle instances with even up to 70 jobs in this special case.

Finally, Fig. 6 examines the influence of the number of jobs, generations, machines, and vehicles, on the computation time of the genetic algorithm. While one of these factors is varied, the others are held constant on $J = 50$, 10,000 generations, $M = 5$, and $V = 10$. All mean computation times depicted in Fig. 6 hold for the probability constellation (30,60,10) that, as already mentioned, performs best throughout the whole numerical study. For instances with 50 jobs, 5 machines, and 10 vehicles, the algorithm on average needs about 140 seconds to carry out 20,000 generations, for example, which is indicated by the dotted lines in Fig. 6.2.

- Fig. 6.1 reveals that the number of jobs is the major driver of computation time.
- A growing number of generations causes an approximatively linear increase of computation time (see Fig. 6.2). The same holds for the number of vehicles (see Fig. 6.4).
- The number of machines has almost no impact on computation time (see Fig. 6.3).
- Further investigations show that the number of different job destinations, the parameter δ_1, and other parameters of the instance generator do not affect computation time.

While the probabilities of reproduction and mutation do not have a significant influence on computation time, the probability of crossover does. The second crossover method needs $O(n^2)$ time in the worst case which is at the same time the upper bound for the computation time of the genetic algorithm if crossover is applied.

7. Conclusion

This paper tackles a practice-oriented integrated machine scheduling and vehicle routing problem by a genetic algorithm and two classic decomposition approaches. The decomposition approaches, which break down the overall problem, successively solve the subproblems, and aggregate the solutions to the subproblems, yet yield relatively poor results. The mean relative deviations from the optimum for 90 small-size-instances is 13.8% and 18%, respectively. Hence, the first question that motivates this paper is answered. For many instances, integrating machine scheduling and vehicle routing holds a significant potential for performance improvements. The second and more interesting question about the possibility of capitalizing on these potentials despite the complexity of the integrated problem is also answered. Around half of the 90 small-size instances are optimally solved by the proposed genetic algorithm. The mean relative deviation from the optimum of 3.2% is close to zero. Moreover, an investigation of large-size instances shows that the genetic algorithm is able to provide relatively good results for instances with up to 50 jobs, 5 machines, and 10 vehicles within an acceptable computation time. In the special case of only one job destination, the performance for

instances with even up to 70 jobs, 5 machines, and 10 vehicles is acceptable.

As motivated in the introduction, especially companies selling time-sensitive goods and companies embedded in just-in-time supply chain environments should seriously consider integrating production and distribution planning to improve their competitiveness. Unfortunately, as seen in the previous sections, the resulting planning problems can be very demanding. When handling integrated problems, managers are dependent on effective and efficient decision support tools and on the progress in research. Indeed, Tables 1 and 2 reveal that this topic has already attracted a lot of attention. Nevertheless, there are still many problems of practical relevance that have not yet been addressed. This applies particularly to problems involving vehicle routing decisions. For example, Miele, a German manufacturer of high quality domestic appliances and commercial equipment, picks up a great part of the required components and materials from regional suppliers using its own vehicles or a subcontracted logistics provider. Taking account of deadlines imposed by production schedules, routes and schedules have to be determined for inbound milk-runs. Due to low buffer inventories and the danger of production breakdowns, this is especially important for materials sourced just-in-time. Focusing explicitly on logistic expenditures, future research should also endeavor to trade-off inventory holding, transportation, earliness, and tardiness costs so as to find cost-minimal overall schedules. Furthermore, besides genetic algorithms, there are many other promising meta-heuristic approaches, such as simulated annealing, ant colony optimization, and tabu search, which could be examined as to their ability to solve integrated problems. However, besides developing effective and efficient planning approaches to intra-organizational planning problems, the coordination of inter-company operations within supply chains, referred to as Supply Chain Scheduling, clearly demands more research.

To sum up, this paper shows that it is possible to improve overall performance by integrating machine scheduling and vehicle routing. Future research to bridge the gap between theory and practical application would be highly worthwhile.

References

Akkerman, R., Farahani, P., Grunow, M., 2010. Quality, safety and sustainability in food distribution: a review of quantitative operations management approaches and challenges. OR Spectrum 32 (4), 863–904.

Albright, S.C., Winston, W.L., 2012. Management Science Modeling, fourth ed. Cengage Learning, South-Western.

Alvarenga, G.B., Mateus, G.R., de Tomi, G., 2007. A genetic and set partitioning two-phase approach for the vehicle routing problem with time windows. Computers & Operations Research 34 (6), 1561–1584.

Averbakh, I., 2010. On-line integrated production–distribution scheduling problems with capacitated deliveries. European Journal of Operational Research 200 (2), 377–384.

Averbakh, I., Baysan, M., 2012. Semi-online two-level supply chain scheduling problems. Journal of Scheduling 15 (3), 381–390.

Averbakh, I., Xue, Z., 2007. On-line supply chain scheduling problems with preemption. European Journal of Operational Research 181 (1), 500–504.

Aytug, H., Khouja, M., Vergara, F.E., 2003. Use of genetic algorithms to solve production and operations management problems: a review. International Journal of Production Research 41 (17), 3955–4009.

Baker, K.R., 1998. Elements of Sequencing and Scheduling. Dartmouth College, Hanover.

Berger, J., Barkaoui, M., 2003. A new hybrid genetic algorithm for the capacitated vehicle routing problem. Journal of the Operational Research Society 54 (12), 1254–1262.

Bhattarai, A., 2012. Wal-Mart tests Same-Day Delivery in Northern Virginia. The Washington Post (October 10).

Biskup, D., Feldmann, M., 2001. Benchmarks for scheduling on a single machine against restrictive und unrestrictive common due dates. Computers & Operations Research 28 (8), 787–801.

Biskup, D., Herrmann, J., Gupta, J.N.D., 2008. Scheduling identical parallel machines to minimize total tardiness. International Journal of Production Economics 115 (1), 134–142.

Bowman, E.H., 1959. The schedule-sequencing problem. Operations Research 7 (5), 621–624.

Brah, S.A., 1996. A comparative analysis of due date based sequencing rules in a flow shop with multiple processors. Production Planning & Control 7 (4), 362–373.

Bräysy, O., Gendreau, M., 2005. Vehicle routing problem with time windows, Part II: metaheuristics. Transportation Science 39 (1), 119–139.

Cakici, E., Mason, S.J., Kurz, M.E., 2012. Multi-objective analysis of an integrated supply chain scheduling problem. International Journal of Production Research 50 (10), 2624–2638.

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

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

Chen, J.-S., Liu, H.-S., Nien, H.-Y., 2007. Minimizing makespan in single machine scheduling with job deliveries to one customer area. International Journal of Industrial Engineering 14 (2), 203–211.

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., 2004. Integrated production and distribution operations – taxonomy, models, and review. In: Simchi-Levi, D., Wu, S.D., Shen, Z.-J. (Eds.), Handbook of Auantitative Supply Chain Analysis. Kluwer Academic Publishers., Boston, Dordrecht, London, pp. 711–740.

Chen, Z.-L., 2010. Integrated production and outbound distribution scheduling: review and extension. Operations Research 58 (1), 130–148.

Chen, Z.-L., Pundoor, G., 2006. Order assignment and scheduling in a supply chain. Operations Research 54 (3), 555–572.

Chen, Z.-L., Pundoor, G., 2009. Integrated order scheduling and packing. Production and Operations Management 18 (6), 672–692.

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

Cheng, T.C.E., Gordon, V.S., 1994. Batch delivery scheduling on a single machine. Journal of the Operational Research Society 45 (10), 1211–1215.

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

Cheng, T.C.E., Kahlbacher, H.G., 1993. Scheduling with delivery and earliness penalties. Asia-Pacific Journal of Operational Research 10 (2), 145–152.

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

Clifford, S., 2012. Same-Day Delivery Test at Wal-Mart. The New York Times (October 9).

Conway, R.W., Maxwell, W.L., Miller, L.W., 1967. Theory of Scheduling. Dover Publications, Mineola.

Davis, L., 1991. Handbook of Genetic Algorithms. Van Nostrand Reinhold, New York.

Dawande, M., Geismar, H.N., Hall, N.G., 2006. Supply chain scheduling: distribution systems. Production and Operations Management 15 (2), 243–261.

Desrochers, M., Lenstra, J.K., Savelsbergh, M.W.P., 1990. A classification scheme for vehicle routing and scheduling problems. European Journal of Operational Research 46 (3), 322–332.

Du, J., Leung, J.Y.-T., 1990. Minimizing total tardiness on one machine is NP-hard. Mathematics of Operations Research 15 (3), 483–495.

Farahani, P., Grunow, M., Guenther, H.-O., 2012. Integrated production and distribution planning for perishable food products. Flexible Services and Manufacturing Journal 24 (1), 28–51.

Fu, B., Huo, Y., Zhao, H., 2012. Coordinated scheduling of production and delivery with production window and delivery capacity constraints. Theoretical Computer Science 422 (9), 39–51.

Garcia, J.M., Lozano, S., Canca, D., 2004. Coordinated scheduling of production and delivery from multiple plants. Robotics and Computer-Integrated Manufacturing 20 (3), 191–198.

Garcia, J.M., Lozano, S., Smith, K., Kwok, T., Villa, G., 2002. Coordinated scheduling of production and delivery from multiple plants and with time windows unsing genetic algorithms. In: Proceedings of the 9th International Conference on Neuronal Information Processing (ICONP'02), vol. 3, pp. 1153–1158.

Garey, M.R., Johnson, D.S., 1979. Computers and Intractability. W.H. Freemann and Company, New York.

Geismar, H.N., Dawande, M., Sriskandarajah, C., 2011. Pool-point distribution of zero-inventory products. Production and Operations Management 20 (5), 737–753.

Geismar, H.N., Laporte, G., Lei, L., Sriskandarajah, C., 2008. The integrated production and transportation scheduling problem for a product with a short lifespan. Journal on Computing 20 (1), 21–33.

Gendreau, M., Potvin, J.-Y., 2010. Handbook of Metaheuristics,, .. Operations Research & Management Science, second ed. Springer, New York Dordrecht, Heidelberg, London.

Gharbi, A., Haouari, M., 2002. Minimizing makespan on parallel machines subject to release dates and delivery times. Journal of Scheduling 5 (4), 329–355.

Goldberg, D.E., 1989. Genetic Algorithms in Search. Optimization, and Machine Learning. Addison-Wesley, Reading.

Graham, R.L., Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G., 1979. Optimization and approximation in deterministic sequencing and scheduling: a survey. Annals of Discrete Mathematics 5, 278–326.

Grunow, M., Stefánsdóttir, B., 2012. Transportation planning/vehicle scheduling (TP/VS). In: Stadtler, H., Fleischmann, B., Grunow, M., Meyr, H., Snrie, C. (Eds.), Advanced Planning in Supply Chains: Illustrating the Concepts Using an SAP® APO Case Study. Springer, Berlin, Heidelberg, pp. 249–285 (Chapter 9).

Hall, N.G., Lesaoana, M., Potts, C.N., 2001. Scheduling with fixed delivery dates. Operations Research 49 (1), 134–144.
Hall, N.G., Potts, C.N., 2003. Supply chain scheduling: batching and delivery. Operations Research 51 (4), 566–584.
Hall, N.G., Potts, C.N., 2005. The coordination of scheduling and batch deliveries. Annals of Operations Research 135 (1), 41–64.
Hamidinia, A., Khakabimamaghani, S., Mazdeh, M.M., Jafari, M., 2012. A genetic algorithm for minimizing total tardiness/earliness of weighted jobs in a batched delivery system. Computers and Industrial Engineering 62 (1), 29–38.
Herrmann, J.W., 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.
Holland, J.H., 1975. Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor.
Huo, Y., Leung, Y.-T., Wang, X., 2010. Integrated production and delivery scheduling with disjoint windows. Discrete Applied Mathematics 158 (8), 921–931.
Hurter, A.P., Van Buer, M.G., 1996. The newspaper production/distribution problem. Journal of Business Logistics 17 (1), 85–107.
Iba, H., Paul, T.K., Hasegawa, Y., 2009. Applied Genetic Programming and Machine Learning. CRC Press, Boca Raton.
Ji, M., He, Y., 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.
Koulamas, C., 1994. The total tardiness problem: review and extensions. Operations Research 42 (6), 1025–1041.
Lee, C.-Y., Chen, Z.-L., 2001. Machine scheduling with transportation considerations. Journal of Scheduling 4 (1), 3–24.
Lee, I.S., Yoon, S.H., 2010. Coordinated scheduling of production and delivery stages with stage-dependent inventory costs. Omega 38 (6), 509–521.
Lehmer, D.H., 1951. Mathematical methods in large-scale computing units. In: Proceedings of the Second Symposium on Large-Scale Digital Calculating Machinery. Harvard University Press, pp. 141–146.
Li, C.-L., Ou, J., 2005. Machine scheduling with pickup and delivery. Naval Research Logistics 52 (7), 617–630.
Li, C.-L., Vairaktarakis, G.L., 2007. Coordinating production and distribution of jobs with bundling operations. IIE Transactions 39 (2), 203–215.
Li, S., Yuan, J., 2009. Scheduling with families of jobs and delivery coordination under job availability. Theoretical Computer Science 410 (47–49), 4856–4863.
Li, S., Yuan, J., Fan, B., 2011. Unbounded parallel-batch scheduling with family jobs and delivery coordination. Information Processing Letters 111 (12), 575–582.
Liu, Z., Cheng, T.C.E., 2002. Scheduling with job release dates, delivery times and preemption penalties. Information Processing Letters 82 (2), 107–111.
Lozano, J.A., Larrañaga, P., Graña, M., Albizuri, F.X., 1999. Genetic algorithms: bridging the convergence gap. Theoretical Computer Science 229 (1–2), 11–22.
Manne, A.S., 1960. On the job-shop scheduling problem. Operations Research 8 (2), 219–223.
Manoj, U.V., Gupta, J.N.D., Gupta, S.K., Sriskandarajah, C., 2008. Supply chain scheduling: just-in-time environment. Annals of Operational Research 161 (1), 53–86.
Matsuo, H., 1988. The weighted total tardiness problem with fixed shipping times and overtime utilization. Operations Research 36 (2), 293–307.
Mazdeh, M.M., Sarhadi, M., 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., Sarhadi, M., Hindi, K.S., 2008. A branch-and-bound algorithm for single-machine scheduling with batch delivery and job release times. Computers & Operations Research 35 (4), 1099–1111.
Mazdeh, M.M., Shashaani, S., Ashouri, A., Hindi, K.S., 2011. Single-machine batch scheduling minimizing weighted flow times and delivery costs. Applied Mathematical Modelling 35 (2), 563–570.
Min, L., Cheng, W., 1999. A genetic algorithm for minimizing the makespan in the case of scheduling identical parallel machines. Artificial Intelligence in Engineering 13 (4), 399–403.
Ng, C.T., Lingfa, L., 2012. On-line integrated production and outbound distribution scheduling to minimize the maximum delivery completion time. Journal of Scheduling 15 (3), 391–398.
Pan, J.C.-H., Wu, C.-L., Huang, H.-C., Su, C.-S., 2009. Coordinating scheduling with batch deliveries in a two-machine flow shop. International Journal of Advanced Manufacturing Technologie 40 (5–6), 607–616.
Pongcharoen, P., Hicks, C., Braiden, P.M., Stewardson, D.J., 2002. Determining optimum genetic algorithm parameters for scheduling the manufacturing and assembly of complex products. International Journal of Production Economics 78 (3), 311–322.
Potts, C., Strusevich, V., 2009. Fifty years of scheduling: a survey of milestones. Journal of the Operational Research Society 60 (1), 41–68.
Potts, C.N., 1980. Analysis of a heuristic for one machine sequencing with release dates and delivery times. Operations Research 28 (6), 1436–1441.
Prins, C., 2004. A simple and effective evolutionary algorithm for the vehicle routing problem. Computers & Operations Research 31 (12), 1985–2002.
Pundoor, G., Chen, Z.-L., 2005. Scheduling a production–distribution system to optimize the tradeoff between delivery tardiness and distribution cost. Naval Research Logistics 52 (6), 571–589.
Qi, X., 2006. A logistics scheduling model: scheduling and transshipment for two processing centers. IIE Transactions 38 (7), 537–546.
Qi, X., 2008. Coordinated logistics scheduling for in-house production and outsourcing. IEEE Transactions on Automation Science and Engineering 5 (1), 188–192.
Reeves, C.R., 2010. Genetic algorithms. In: Gendreau, M., Potvin, J.-Y. (Eds.), Handbook of Metaheuristics, second ed. Springer, New York, Dordrecht, Heidelberg, London, pp. 109–139.
Savelsbergh, M.W.P., 1985. Local search in routing problems with time windows. Annals of Operations Research 4 (1), 285–305.
Selvarajah, E., Steiner, G., 2009. Approximation algorithms for the supplier's supply chain scheduling problem to minimize delivery and inventory holding costs. Operations Research 57 (2), 426–438.
Sivrikaya-Şerifoğlu, F., Ulusoy, G., 1999. Parallel machine scheduling with earliness and tardiness penalties. Computers & Operations Research 26 (8), 773–787.
Soukhal, A., Oulamara, A., Martineau, P., 2005. Complexity of flow shop scheduling problems with transportation constraints. European Journal of Operational Research 161 (1), 32–41.
Stecke, K., Zhao, X., 2007. Production and transportation integration for a make-to-delivery business mode. Manufacturing & Service Operations Management 9 (2), 206–224.
Steiner, G., Zhang, R., 2009. Approximation algorithms for minimizing the total weighted number of late jobs with late deliveries in two-level supply chains. Journal of Scheduling 12 (6), 565–574.
Suzuki, J., 1995. A Markov chain analysis on simple genetic algorithms. IEEE Transactions on Systems, Man, and Cybernetics 25 (4), 655–659.
Thomas, D.J., Griffin, P.M., 1996. Coordinated supply chain management. European Journal of Operational Research 94 (1), 1–15.
Ting, C.-J., Huang, C.-H., 2005. An improved genetic algorithm for vehicle routing problem with time windows. International Journal of Industrial Engineering 12 (3), 218–228.
Valente, J.M.S., Gonçalves, J.F., 2009. A genetic algorithm approach for the single machine scheduling problem with linear earliness and quadratic tardiness penalties. Computers & Operations Research 36 (10), 2707–2715.
Van Buer, M.G., Woodruff, D.L., Olson, R.T., 1999. Solving the medium newspaper production/distribution problem. European Journal of Operational Research 115 (2), 237–253.
Vidal, T., Crainic, T.G., Gendreau, M., Prins, C., 2013. A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time windows. Computers & Operations Research 40 (1), 475–489.
Wagner, H.M., 1959. An integer linear-programming model for machine scheduling. Naval Research Logistics 6 (2), 131–140.
Wang, G., Cheng, T.C.E., 2000. Parallel machine scheduling with batch delivery costs. International Journal of Production Economics 68 (2), 177–183.
Wang, H., Lee, C.-Y., 2005. Production and transport logistics scheduling with two transport mode choices. Naval Research Logistics 52 (8), 796–809.
Wang, X., Cheng, T.C.E., 2007. Machine scheduling with an availability constraint and job delivery coordination. Naval Research Logistics 54 (1), 11–20.
Wang, X., Cheng, T.C.E., 2009a. Logistics scheduling to minimize inventory and transport costs. International Journal of Production Economics 121 (1), 266–273.
Wang, X., Cheng, T.C.E., 2009b. Production scheduling with supply and delivery considerations to minimize the makespan. European Journal of Operational Research 194 (3), 743–752.
Woeginger, G.J., 1994. Heuristics for parallel machine scheduling with delivery times. Acta Informatica 31 (6), 503–512.
Woeginger, G.J., 1998. A polynomial-time approximation scheme for single-machine sequencing with delivery times and sequence-indepent batch set-up times. Journal of Scheduling 1 (1), 79–87.
Yang, X., 2000. Scheduling with generalized batch delivery dates and earliness penalties. IIE Transactions 32 (8), 735–741.
Yuan, 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.
Yuan, J., Soukhal, A., Chen, Y., Lu, L., 2007. A note on the complexity of flow shop scheduling with transportation constraints. European Journal of Operational Research 178 (3), 918–925.
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.
Zhong, W., Dósa, G., Tan, Z., 2007. On the machine scheduling problem with job delivery coordination. European Journal of Operational Research 182 (3), 1057–1072.