The blocking job shop with rail-bound transportation

Reinhard Bürgy · **Heinz Gröflin**

Published in Journal of Combinatorial Optimization.
The final publication is available at Springer
via http://dx.doi.org/10.1007/s10878-014-9723-3

Abstract The Blocking Job Shop with Rail-bound Transportation (BJS-RT) considered here is a version of the job shop scheduling problem characterized by the absence of buffers and the use of a rail-bound transportation system. The jobs are processed on machines and are transported from one machine to the next by mobile devices (called robots) that move on a single rail. The robots cannot pass each other, must maintain a minimum distance from each other, but can also "move out of the way".

The objective of the BJS-RT is to determine for each machining operation its starting time and for each transport operation its assigned robot and starting time, as well as the trajectory of each robot, in order to minimize the makespan.

Building on previous work of the authors on the flexible blocking job shop and an analysis of the feasible trajectory problem, a formulation of the BJS-RT in a disjunctive graph is derived. Based on the framework of job insertion in this graph, a local search heuristic generating consistently feasible neighbor solutions is proposed. Computational results are presented, supporting the value of the approach.

Keywords job shop scheduling · blocking · rail-bound transportation · robots · disjunctive graph · job insertion · tabu search

1 Introduction

Job shop scheduling problems in practice often display features that are not addressed in the classical job shop problem. Among these features figures limited buffer capacity or even the absence of buffers. This latter case is commonly referred to as the blocking job shop. A second feature is processor flexibility allowing for an operation

R. Bürgy
Department of Informatics, University of Fribourg, Boulevard de Pérolles 90, 1700 Fribourg, Switzerland
E-mail: reinhard.buergy@unifr.ch

H. Gröflin
Department of Informatics, University of Fribourg, Boulevard de Pérolles 90, 1700 Fribourg, Switzerland,
E-mail: heinz.groeflin@unifr.ch

to be executed not only on a preassigned machine, but on any machine chosen from a specified set. A third feature is the presence of mobile devices that transport the jobs from one machine to the next. If these devices do not interfere with each other in their movements, they can be treated as ”normal” machines with sequence dependent set-up times between their operations. However, mobile devices, e.g. robots, cranes or AGV's, often interact with each other, increasing substantially the complexity of the scheduling problem. Besides decisions on machine assignment to operations and starting times of operations, trajectories of the mobile devices must also be determined.

The job shop problem addressed here displays all three features to some extent and will be called the Blocking Job Shop with Rail-bound Transportation (BJS-RT). It can be described informally as follows.

Given are jobs, machines and mobile devices which we call robots. A job is processed on machines in a sequence of machining operations and is transported from one machine to the next by a robot in transport operations. Thus a job can be seen as a sequence of alternating machining and transport operations.

There are no buffers, i.e. a job, once started and until its completion, is either on a machine or on a robot. After completing a machining operation, a job might wait on the machine - in effect blocking it - until it is picked up by a robot. Similarly, after completing a transport operation, a job must wait on the robot until being handed over to the machine for the next operation. There is processor flexibility: while a machining operation is executed on a preassigned machine, a transport operation can be executed by any robot. Finally, the robots move on a rail along which the machines are located. They cannot pass each other and must maintain a minimum distance from each other, but can move ”out of the way”. Also, a robot can move at a speed up to a limit which can be robot-dependent.

The objective is to determine the starting time of each transport and machining operation, the assigned robot of each transport operation, and the trajectory, i.e. the location at any time, of each robot, in order to minimize makespan.

Note that sequence-dependent set-up times between transport operations are needed in order to model idle moves of robots. We also allow set-up times between machining operations.

A small example is introduced here to illustrate a BJS-RT environment and a solution represented graphically. It consists of three machines m, m', m'', two robots r_1, r_2 with minimum distance δ between them and three jobs J, J', J''. The operations are numbered from 1 to 11 and the three jobs J, J', J'' comprise respectively operations $1,2,3$; $4,5,6,7,8$ and $9,10,11$. Detailed input data of the example will be given in the next section. Figure 1 sketches a possible physical layout. A solution is depicted in Figure 2 in a Gantt chart. Thick bars represent take-over, processing and hand-over steps while narrow bars represent (filled) waiting times and (hatched) minimum durations of idle moves. The numbers refer to the operations. Feasible trajectories of the robots are also displayed by two black lines. Thick line sections indicate that the robot is loaded with a job while thin sections stand for idle moves. The black dots on the lines indicate the start or completion of a hand-over or take-over step.

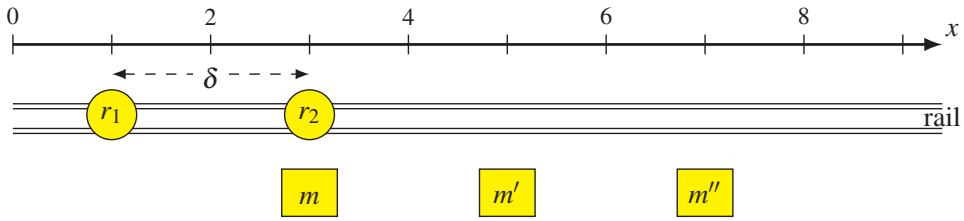

Fig. 1: Layout in the example.

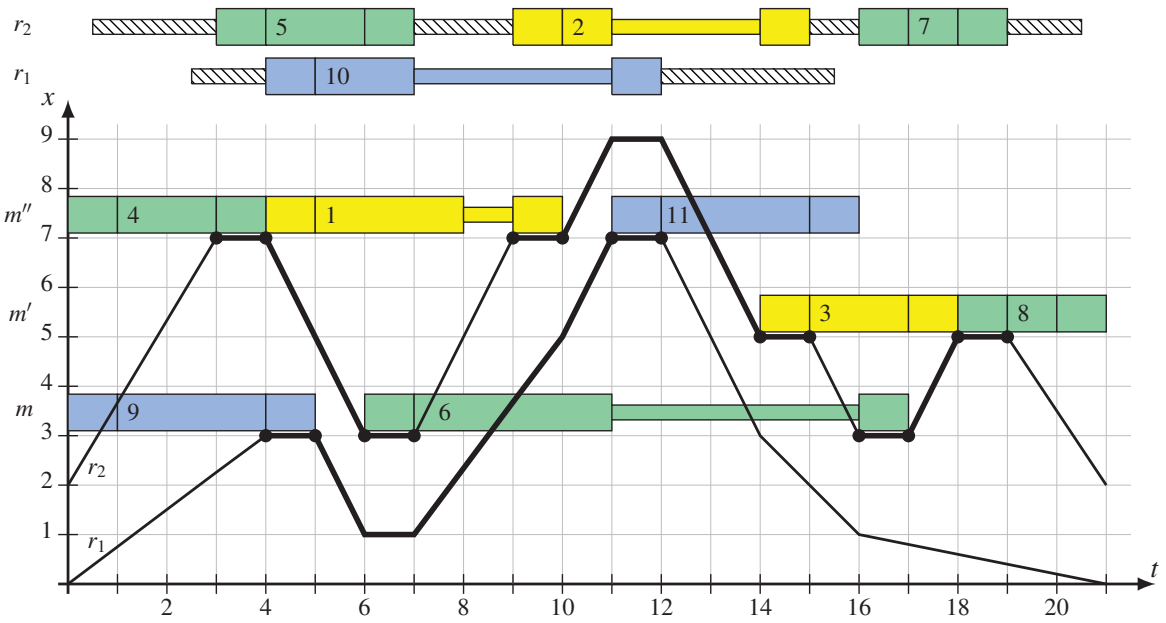

Fig. 2: A schedule with makespan 21 in the example.

To our knowledge, the BJS-RT has not yet been addressed in the literature. Nevertheless, related scheduling problems have been treated that display some of the features of the BJS-RT. They can be classified into three groups: job shop with blocking, job shop with transportation, and application-specific problems involving robot transportation. We briefly present selected papers from theses three research directions.

The Blocking Job Shop (BJS) has been tackled by several authors, e.g. Brizuela et al (2001), Mascis and Pacciarelli (2002), Brucker et al (2006) and Gröflin and Klinkert (2009). We also point out our recent extension of the BJS in (Gröflin et al, 2011) that includes machine flexibility, as it will constitute a starting basis in the sequel.

Job shop scheduling problems that include transportation have also been studied. Hurink and Knust (2005) consider a single transportation robot in a classical Job Shop (JS). Bilge and Ulusoy (1995), Brucker and Strotmann (2002), Khayat et al (2006), Deroussi et al (2008) and Lacomme et al (2010) tackle a similar problem with multiple identical robots (with no interaction between them). Only few papers address the BJS with transportation. Brucker et al (2012) recently studied the cyclic BJS with one transportation robot.

To our knowledge, no paper considers a job shop setting (JS or BJS) with multiple robots interacting with each other, although the value of incorporating interactions be-

tween robots and limited buffer capacity has been recognized by several authors. For instance Khayat et al (2006) suggested as future research to "include testing conflict avoidance and limited buffer capacities in a job shop setting".

Numerous papers deal with application-specific problems occurring in the context of factory crane scheduling, hoist scheduling and scheduling of cranes in container terminals. The following contributions are among the more closely related to the BJS-RT.

Aron et al (2010) study the problem of finding trajectories for two identical factory cranes moving on a single rail. To each crane is assigned a sequence of transportation tasks that must be executed within given time windows. The cranes are allowed to move out of the way to avoid collisions.

In electroplating facilities, panels are covered with a coat of metal by immersing them sequentially in tanks, and hoists move the panels from tank to tank. Scheduling the coating operations as well as the movements of the hoists is commonly addressed as the hoist scheduling problem, cf. Manier and Bloch (2003). Versions with multiple hoists have been studied by several authors, for example by Manier et al (2000), Leung and Zhang (2003) and by Leung et al (2004). In these applications, empty hoists can move out of the way in order to avoid collisions, whereas loaded hoists have to move directly from tank to tank.

Crane scheduling in container terminals addresses the problem of scheduling transport operations (storage, retrieval and relocation) of containers executed by yard cranes. Multiple cranes have been considered e.g. by Ng (2005) who partitions the yard into non-overlapping areas, one for each crane, to eliminate the occurrence of collisions, and by Li et al (2009) who use a time-discretized MIP formulation to enforce a minimum distance between cranes at any time period.

Several authors emphasize that the presence of multiple robots increases complexity, e.g. Leung et al (2004) write: "the scheduling problem for multi-hoist lines is significantly more difficult than for single-hoist lines because of the additional problem of hoist collision avoidance."

The paper is organized as follows. The next section describes formally the BJS-RT and gives a first problem formulation based on schedules with trajectories. In Section 3, a projection of the solution space of the first formulation is derived, yielding a disjunctive graph formulation of the BJS-RT. Based on this disjunctive graph, a local search heuristic is proposed in Section 4 and computational results are presented in Section 5. The Appendix gives an algorithm for finding feasible trajectories.

Graphs are needed for the formulations and the local search heuristic. They will be directed and the following standard notation will be used. An arc $e = (v,w)$ has a *tail* (node v), and a *head* (node w), denoted by $t(e)$ and $h(e)$ respectively. Also, given a graph $G = (V,E)$, for any $W \subseteq V$, $\gamma(W) = \{e \in E : t(e) \text{ and } h(e) \in W\}$, $\delta^-(W) = \{e \in E : t(e) \notin W \text{ and } h(e) \in W\}$, $\delta^+(W) = \{e \in E : t(e) \in W \text{ and } h(e) \notin W\}$ and $\delta(W) = \delta^-(W) \cup \delta^+(W)$. These sets are defined in G, we abstain however from a heavier notation, e.g. $\delta^+_G(W)$ for $\delta^+(W)$. It will be clear from the context which underlying graph is meant. Finally, in a graph $G = (V,E,d)$ with arc valuation $d \in \mathbb{R}^E$, a path (or cycle) in G of positive length will be called a *positive* path (or cycle) and a path of longest length a *longest* path.

2 A first problem formulation

In this section, we formulate the BJS-RT by using our problem formulation of the flexible blocking job shop in (Gröflin et al, 2011) and extending it to take into account the interactions between robots.

2.1 Notation and data

Let M and R be the sets of machines and robots respectively. The locations of the machines and robots along and on the rail are measured on an x-axis. For each machine $m \in M$, let a_m be its fixed location. Also, for each robot $r \in R$, let $x(r,t)$ denote the (variable) location of r at time t, and $a_{\sigma r}$ and $a_{\tau r}$ be prescribed initial and end location, i.e. $x(r,0) = a_{\sigma r}$ and $x(r,T) = a_{\tau r}$ must hold at makespan T (the choice of the symbols σ and τ will become clear in the sequel). Furthermore, for each robot $r \in R$, let v_r be its maximum speed, and $\delta > 0$ be the minimum distance to be maintained between two consecutive robots on the rail. Finally, L is the usable rail length: $0 \le x(r,t) \le L$ for all $r \in R$ and all t.

Let $\mathscr{J}$ be the set of all jobs and $I = I^M \cup I^R$ be the set of all operations, partitioned into set I^M of machining operations and set I^R of transport operations. Each operation $i \in I^M$ is executed on a preassigned machine $m_i \in M$ and each transport operation $i \in I^R$ can be executed by any robot $r \in R$.

A job $J \in \mathscr{J}$ is a set of operations $\{i : i \in J\}$ and $\mathscr{J} \subseteq 2^I$ forms a partition of I, i.e. any operation $i \in I$ is in exactly one job $J \in \mathscr{J}$. A job $\{i : i \in J\}$ is ordered in a sequence and sometimes denoted as the ordered set $\{J_1, J_2, ..., J_{|J|}\}$, J_q denoting the q-th operation of job J. Two operations i, j of job J are consecutive if $i = J_q$ and $j = J_{q+1}$ for some q, $1 \le q < |J|$. The operations of a job are alternately machining and transport operations, and we assume that $|J|$ is odd, $J_q \in I^M$ for q odd and $J_q \in I^R$ for q even, $1 \le q \le |J|$. Note that typically, the first operation J_1 will consist in loading job J at some storage place or device, and the last operation $J_{|J|}$ will represent the unloading of completed job J.

Each operation $i \in I = I^M \cup I^R$ is decomposed into three successive steps: a take-over step o_i, a processing step which is either a machining or a transport step, and a hand-over step $\overline{o}_i$. For each $i \in I$, let a_{o_i} and $a_{\overline{o}_i}$ be the locations of its hand-over and take-over steps. Note that these locations are determined by the machine locations: if $i \in I^M$, $a_{o_i} = a_{\overline{o}_i} = a_{m_i}$; if $i \in I^R$, $i \in J$ and $j,k \in J$ are the (machining) operations preceding and following i, then $a_{o_i} = a_{\overline{o}_j} = a_{m_j}$ and $a_{\overline{o}_i} = a_{o_k} = a_{m_k}$.

For each machining operation $i \in I^M$, let d_i be the duration of its machining step, and for each transport operation $i \in I^R$ and each robot $r \in R$, let $d_i^r = |a_{o_i} - a_{\overline{o}_i}|/v_r$ be the minimum duration of its transport step on r, namely the time needed by r to cover the transport distance at maximum speed.

For each pair of consecutive operations i, j of a job J, the hand-over step of i and the take-over step of j are synchronized and thus have same duration, called transfer time. This time might depend on the robot r used for the transfer, and is denoted d_{ij}^r if $i \in I^R$ is executed by r, $r \in R$ and $j \in I^M$, or $i \in I^M$ and $j \in I^R$ is executed by r, $r \in R$.

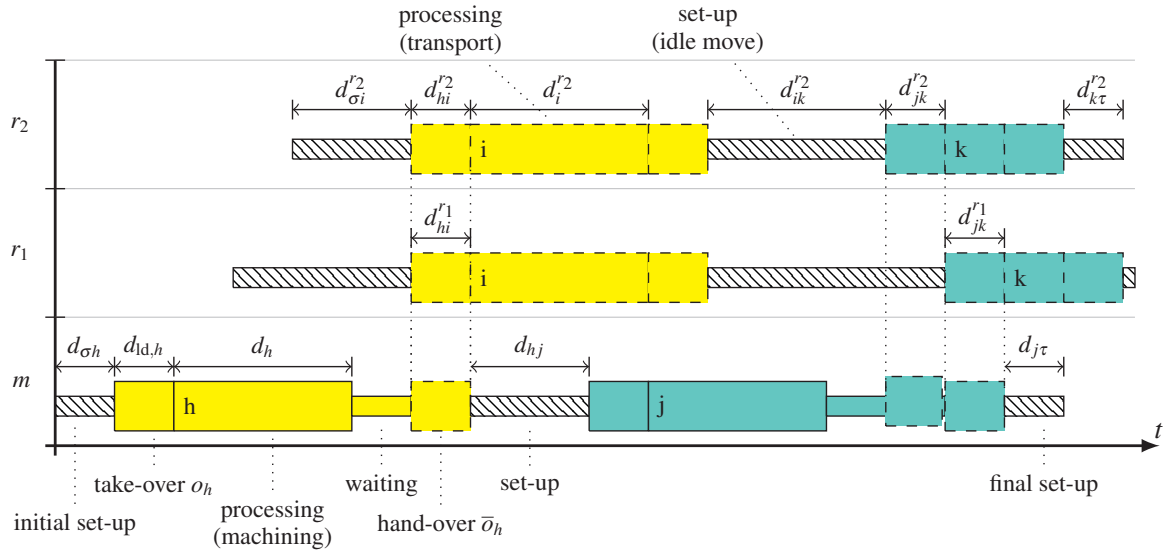

Fig. 3: Four operations h, i, j and k are illustrated in a Gantt chart. h and i are consecutive in one job, and j and k are consecutive in another job. Machining operations h, j are executed on machine m and transport operations i, k can be executed on either robot r_1 or r_2. The last two operations as well as all involved take-over and hand-over steps are depicted by dashed lines to indicate this choice. Some durations are given to illustrate our notation.

Also, if $i \in I^M$ is the first operation of a job, let $d_{ld,i}$ be the loading time of the job, and if $i \in I^M$ is the last operation of a job, let $d_{i,uld}$ be the unloading time.

Finally, there are set-up times between consecutive operations on a machine or a robot. For any two distinct operations i, $j \in I^M$ with $m = m_i = m_j$, if j immediately follows i on m, a set-up of duration d_{ij} occurs on machine m between the hand-over step $\overline{o}_i$ of i and the take-over step o_j of j. Also, for each machining operation $i \in I^M$, an initial set-up of duration $d_{\sigma i}$ (an earliest starting time) can be specified, as well as a final set-up of duration $d_{i\tau}$ (a "tail"), meaning that a time of at least $d_{i\tau}$ lapses between the completion time of i and the overall finish time (makespan).

Similarly, for any two distinct operations i, $j \in I^R$ and $r \in R$, if both i and j are executed on robot r and j immediately follows i on r, a "set-up" of duration d^r_{ij} occurs on robot r, corresponding to the minimum duration of the idle move of r from the location of the hand-over step $\overline{o}_i$ of i to the location of the take-over step o_j of j, i.e. $d^r_{ij} = |a_{\overline{o}_i} - a_{o_j}|/v_r$. Finally, for each $i \in I^R$ and $r \in R$, the initial and final set-up times are defined as $d^r_{\sigma i} = |a_{\sigma r} - a_{o_i}|/v_r$ and $d^r_{i\tau} = |a_{\overline{o}_i} - a_{\tau r}|/v_r$, the minimum time needed by r to cover the distance from its initial location to the location of the take-over o_i, respectively from the location of the hand-over $\overline{o}_i$ to its end location.

Figure 3 illustrates in a Gantt chart the described structure of the jobs and our notation.

A few standard assumptions concerning the data are made. All durations are non-negative. Durations of machining, hand-over and take-over steps, and maximum robot speeds are positive. Also, since we allow a transport operation to be executed by *any* robot, enough machine-free space on the rail left and right should be available: $(|R|-1)\delta \leq \min\{a_m : m \in M\}$ and $\max\{a_m : m \in M\} \leq L - (|R|-1)\delta$ must hold.

The data for the example are as follows. $M = \{m, m', m''\}$, $R = \{r_1, r_2\}$ and $\mathscr{J} = \{J, J', J''\}$. The jobs are identified with their ordered set of operations $J = \{1,2,3\}$, $J' = \{4,5,6,7,8\}$ and $J'' = \{9,10,11\}$. The set I^M of machining operations is $\{1,3,4,6,8,9,11\}$ and operations 6 and 9 are on machine m, 3 and 8 on m' and $1,4$ and 11 on m''. The set I^R of transport operations is $\{2,5,7,10\}$. The locations of the machines are $a_m = 3, a_{m'} = 5, a_{m''} = 7$, and the initial and final locations of the robots $a_{\sigma r_1} = a_{\tau r_1} = 0$ and $a_{\sigma r_2} = a_{\tau r_2} = 2$. The minimum distance between adjacent robots is $\delta = 2$, the maximum speed of the robots is $v_{r_1} = v_{r_2} = 2$, and the rail length is $L = 9$. For $i = 1,3,4,6,8,9,11$, the respective durations d_i of the machining step are $3,2,2,4,1,3,3$. All transfer times $d^r_{ij} = d^r_{ji}$, $r \in R$ and $i \in I^M$, $j \in I^R$, loading times $d_{\mathrm{ld},i}$ for $i = 1,4,9$ and unloading times $d_{i,\mathrm{uld}}$ for $i = 3,8,11$ are 1. Set-up times d_{ij} for any $i, j \in I^M$ with $m_i = m_j$, and initial and final set-up times $d_{\sigma i}$ and $d_{i\tau}$, $i \in I^M$ are all 0.

2.2 The flexible blocking job shop relaxation

We temporarily ignore the interactions between the robots on the rail. This relaxed BJS-RT is then a special case of the flexible blocking job shop as introduced in (Gröflin et al, 2011). Based on this work, we give a disjunctive graph formulation of the relaxed BJS-RT. Thanks to the simple type of machine flexibility, notation has been somewhat simplified.

The disjunctive graph $G = (V, A, E, \mathscr{E}, d)$ is constructed as follows, V denoting the node set, A the set of conjunctive arcs, E the set of disjunctive arcs, $\mathscr{E}$ the family of disjunctive arc pairs and $d \in \mathbb{R}^{A \cup E}$ the arc valuation.

To each operation $i \in I^M$ is associated a chain of four nodes v^1_i, v^2_i, v^3_i, v^4_i and three arcs (v^1_i, v^2_i), (v^2_i, v^3_i) and (v^3_i, v^4_i) representing the take-over step, the machining step and hand-over step of i on m_i. Let $V_i = \{v^1_i, v^2_i, v^3_i, v^4_i\}$ denote the node set of the chain associated to i. Similarly, to each operation $i \in I^R$ and each $r \in R$ is associated the chain with nodes v^1_{ir}, v^2_{ir}, v^3_{ir}, v^4_{ir} and arcs (v^1_{ir}, v^2_{ir}), (v^2_{ir}, v^3_{ir}) and (v^3_{ir}, v^4_{ir}) and let $V_{ir} = \{v^1_{ir}, v^2_{ir}, v^3_{ir}, v^4_{ir}\}$. The node set of G consists of the union of the V_i's and V_{ir}'s, together with two additional nodes σ and τ representing fictive start and end operations of duration 0 to occur before, respectively after, all other operations, i.e. $V = \cup\{V_i : i \in I^M; V_{ir} : i \in I^R, r \in R; \{\sigma, \tau\}\}$.

Denote by I^{first} and I^{last} the subsets of operations that are first and last operations of jobs, respectively. Note that $I^{\mathrm{first}} \cup I^{\mathrm{last}} \subseteq I^M$. The set A of conjunctive arcs consists of the following arcs, indicated with their weights:

1. For each $i \in I^M$, three arcs (v^1_i, v^2_i), (v^2_i, v^3_i), (v^3_i, v^4_i), with respective weights: $d_{\mathrm{ld},i}$ if $i \in I^{\mathrm{first}}$ and 0 if $i \in I^M - I^{\mathrm{first}}$; d_i; $d_{i,\mathrm{uld}}$ if $i \in I^{\mathrm{last}}$ and 0 if $i \in I^M - I^{\mathrm{last}}$. The three arcs are referred to as take-over arc, machining arc and hand-over arc.
2. For each $i \in I^R$ and $r \in R$, three arcs (v^1_{ir}, v^2_{ir}), (v^2_{ir}, v^3_{ir}) and (v^3_{ir}, v^4_{ir}), with respective weights 0, d^r_i and 0. The three arcs are referred to as take-over arc, transport arc and hand-over arc.
3. For any two consecutive operations $i = J_q$ and $j = J_{q+1}$ of a job J and $r \in R$: if $i \in I^M$ and $j \in I^R$, two pairs of synchronization arcs (v^3_i, v^1_{jr}), (v^1_{jr}, v^3_i) and (v^4_i, v^2_{jr}),

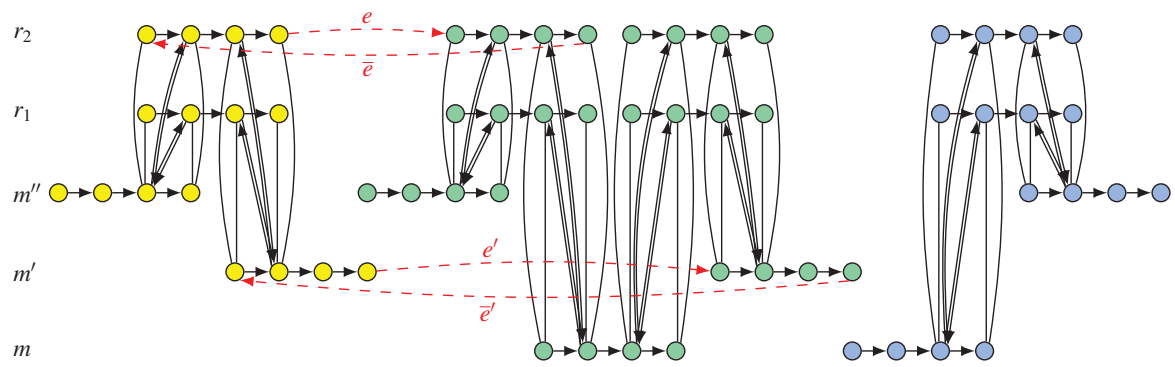

Fig. 4: Disjunctive graph of the example.

(v^2_{jr}, v^4_i) of weight 0 joining the (starts and the ends of the) hand-over step of i and take-over step of j, and a pair of transfer arcs (v^3_i, v^2_{jr}), (v^2_{jr}, v^3_i) of respective weight d^r_{ij} and $-d^r_{ij}$. If $i \in I^R$ and $j \in I^M$, two pairs of synchronization arcs (v^3_{ir}, v^1_j), (v^1_j, v^3_{ir}) and (v^4_{ir}, v^2_j), (v^2_j, v^4_{ir}) of weight 0, and a pair of transfer arcs (v^3_{ir}, v^2_j), (v^2_j, v^3_{ir}) of weight d^r_{ij} and $-d^r_{ij}$.

4. For each $i \in I^M$, an initial set-up arc (σ, v^1_i) of weight $d_{\sigma i}$ and a final set-up arc (v^4_i, τ) of weight $d_{i\tau}$. For each $i \in I^R$ and $r \in R$, an initial set-up arc (σ, v^1_{ir}) of weight $d^r_{\sigma i}$ and a final set-up arc (v^4_{ir}, τ) of weight $d^r_{i\tau}$.

The set E of disjunctive arcs is given as follows. For any two distinct operations $i, j \in I^M$ with $m_i = m_j$, there are two disjunctive arcs (v^4_i, v^1_j), (v^4_j, v^1_i) with respective weights d_{ij}, d_{ji}. For each $r \in R$ and distinct $i, j \in I^R$, there are two disjunctive arcs (v^4_{ir}, v^1_{jr}), (v^4_{jr}, v^1_{ir}) with respective weights d^r_{ij}, d^r_{ji}.

$\mathscr{E}$ is the family of all pairs $\{(v^4_i, v^1_j), (v^4_j, v^1_i)\}$, $i, j \in I^M$ and $\{(v^4_{ir}, v^1_{jr}), (v^4_{jr}, v^1_{ir})\}$, $r \in R$, $i, j \in I^R$ of disjunctive arcs introduced above. A generic element of $\mathscr{E}$, i.e. a pair of disjunctive arcs, will sometimes be denoted by $\{e, \bar{e}\}$ and arc $\bar{e}$ will be said to be the mate of e and vice-versa.

Figure 4 depicts the disjunctive graph of the example. For clarity however, nodes σ and τ, as well as all disjunctive arcs, except two pairs denoted by $e, \bar{e}$ and $e', \bar{e}'$, have been omitted. A pair of synchronization arcs is represented by an undirected edge.

Definition 1 A mode is a tuple $\pi = (\pi(i) : i \in I^R) \in \Pi = R \times \ldots \times R$ assigning to each transport operation $i \in I^R$ a robot $\pi(i) \in R$.

A mode π selects a node-induced subgraph G^π in G defined as follows. Let $V^\pi = \cup\{V_i : i \in I^M; V_{i,\pi(i)} : i \in I^R; \{\sigma, \tau\}\}$, $A^\pi = A \cap \gamma(V^\pi)$, $E^\pi = E \cap \gamma(V^\pi)$ and $\mathscr{E}^\pi = \{\{e, \bar{e}\} \in \mathscr{E} : e \cup \bar{e} \subseteq E^\pi\}$. The resulting graph $G^\pi = (V^\pi, A^\pi, E^\pi, \mathscr{E}^\pi, d)$ is the disjunctive graph *associated to the mode* π. We take the liberty of denoting the restriction of d to $A^\pi \cup E^\pi$ again by d.

Definition 2 For any mode π and $S \subseteq E^\pi$, (π, S) is called a selection. Selection (π, S) is said to be positive acyclic if the graph $(V^\pi, A^\pi \cup S, d)$ contains no positive cycle.

(π, S) is said to be complete if $S \cap \{e, \bar{e}\} \neq \emptyset$ for all $\{e, \bar{e}\} \in \mathscr{E}^\pi$, and to be feasible if it is positive acyclic and complete.

A feasible selection (π, S) specifies with π the assignment of robots to the transport operations and with S the sequencing of the operations on the machines and robots. For any mode π, let $\mathscr{F}^\pi = \{S \subseteq E^\pi : (\pi, S) \text{ is feasible}\}$.

Given $\pi \in \Pi$ and $S \in \mathscr{F}^\pi$, any $\alpha = (\alpha_v : v \in V^\pi)$ satisfying:

$$\alpha_w - \alpha_v \geq d(v, w) \text{ for all arcs } (v, w) \in A^\pi \cup S \tag{1}$$

$$\alpha_\sigma = 0 \tag{2}$$

specifies starting times for the events corresponding to the nodes of V^π. α_τ is the makespan. For any machining operation $i \in I^M$, α_v for $v = v_i^1, v_i^2, v_i^3, v_i^4$ is the starting and completion time of its take-over step o_i and the starting and completion time of its hand-over step $\bar{o}_i$. The same holds for any transport operation $i \in I^R$ and α_v for $v = v_{i,\pi(i)}^1, v_{i,\pi(i)}^2, v_{i,\pi(i)}^3, v_{i,\pi(i)}^4$. Note that the lag between the starting time and the completion time of all these steps is exactly the transfer time. Let

$$\Omega(\pi, S) = \{\alpha \in \mathbb{R}^{V^\pi} : \alpha \text{ satisfies } (1) - (2)\}$$
$$\Omega(\pi) = \cup_{S \in \mathscr{F}^\pi} \Omega(\pi, S).$$

Definition 3 The solution space of the relaxed BJS-RT is $\Omega = \{(\pi, \alpha) : \pi \in \Pi, \alpha \in \Omega(\pi)\}$. Any $(\pi, \alpha) \in \Omega$ is called a schedule.

The relaxed BJS-RT is the problem of finding a schedule $(\pi, \alpha) \in \Omega$ minimizing α_τ. Note that, given $\pi \in \Pi$ and $S \in \mathscr{F}^\pi$, finding a schedule α minimizing the makespan is finding $\alpha \in \Omega(\pi, S)$ minimizing α_τ. As is well-known, this is easily done by longest path computation in $(V^\pi, A^\pi \cup S, d)$ and letting α_v be the length of a longest path from σ to v for all $v \in V^\pi$. The relaxed BJS-RT can therefore also be formulated as: "among all feasible selections, find a selection (π, S) minimizing the length of a longest path from σ to τ in $(V^\pi, A^\pi \cup S, d)$".

A remark on the machine set-up times is in order. They should satisfy the so-called weak triangle inequality (cf. Brucker and Knust (2011), p. 11) for the disjunctive graph formulation to be valid. Otherwise, arcs between non-consecutive operations on a machine may become active when computing longest paths in $(V^\pi, A^\pi \cup S, d)$, yielding a wrong makespan since set-ups take place only between consecutive operations on a machine. However, the disjunctive graph model and the solution method proposed in Section 4 can easily be adapted to handle arbitrary set-up times by ignoring all disjunctive arcs that link non-consecutive operations on a machine when computing longest paths in $(V^\pi, A^\pi \cup S, d)$.

2.3 Schedules with trajectories

Not every schedule $(\pi, \alpha) \in \Omega$ is feasible in the BJS-RT. Indeed, due to interference of the robots with each other, there might not exist feasible trajectories $x(r, .)$, $r \in R$, that "meet" the schedule. Given $(\pi, \alpha) \in \Omega$, we examine now which constraints the trajectories must satisfy in order to be feasible.

First, since the robots $r \in R$ cannot pass each other on the rail, it is convenient to index them $r_1, r_2, \ldots, r_K, K = |R|$ according to their natural ordering on the rail, with their locations at any time t satisfying $x(r_1,t) < x(r_2,t) < \ldots < x(r_K,t)$. From now on, for ease of notation, reference to robot r_k will be made simply through its index k, e.g. $x(r_k,t)$ is denoted by $x(k,t)$ and the maximum speed v_{r_k} by v_k.

Second, main input data for the trajectories are the locations, starting times and durations of the take-over and hand-over steps of all transport operations. For brevity, we refer in the sequel to a take-over or a hand-over step simply as a *transfer step*, when distinction is not necessary. For $k = 1, \ldots, K$, let $O_k = \{o_i, \overline{o}_i : i \in I^R \text{ with } \pi(i) = k\}$ be the set of transfer steps executed by robot k. The location of a transfer step $o \in O_k$ is denoted by a_o, its starting time by $\alpha(o)$ and its duration by $d(o)$. Note that these data are all determined by the schedule (π, α), e.g. if $o = o_i \in O_k$ for some $i \in I^R$, then $a_o = a_{o_i}$, $\alpha(o) = \alpha_{v^1_{ik}}$ and $d(o) = d^k_{ji}$, where i is in some job J and j is the machining operation preceding i in J. It is convenient to add to O_k a fictive initial and final transfer step σ_k and τ_k, both of duration 0, and respective locations the prescribed initial and final locations $a_{\sigma k}$ and $a_{\tau k}$, and starting times 0 and α_τ. Denote again by O_k the so extended set and let $O = \bigcup_k O_k$.

Feasible trajectories $x(k,.)$, $k = 1, \ldots, K$, must satisfy the following constraints (3) to (6):

$$|x(k,t') - x(k,t)| \leq (t' - t)v_k \text{ for all } k = 1, \ldots, K \text{ and } t' > t \geq 0 \quad (3)$$

$$x(k,t) = a_o \text{ for all } k = 1, .., K,\ o \in O_k \text{ and } t \text{ with } \alpha(o) \leq t \leq \alpha(o) + d(o) \quad (4)$$

$$x(k,t) + \delta \leq x(k+1,t) \text{ for all } k = 1, \ldots, K-1 \text{ and } t \geq 0 \quad (5)$$

$$0 \leq x(1,t) \text{ and } x(K,t) \leq L \text{ for all } t \geq 0 \quad (6)$$

(3) expresses that a robot cannot cover a greater distance than allowed by its maximum speed. (4) enforces that a robot is at a_o while it executes the transfer step o. (5) maintains a minimum distance δ between two adjacent robots, while (6) restricts the moves of the robots to the interval $[0, L]$.

Given a schedule $(\pi, \alpha) \in \Omega$, let

$$X(\pi, \alpha) = \{\mathbf{x} = (x(k,.), k = 1, \ldots, K) : \mathbf{x} \text{ satisfies (3) to (6)}\}$$

Definition 4 The solution space of the BJS-RT is $\Gamma = \{(\pi, \alpha, \mathbf{x}) : (\pi, \alpha) \in \Omega \text{ and } \mathbf{x} \in X(\pi, \alpha)\}$. Any $(\pi, \alpha, \mathbf{x}) \in \Gamma$ is called a *schedule with trajectories*. The BJS-RT is the problem of finding a schedule with trajectories minimizing α_τ.

3 A compact formulation

The objective in this section is to transform the BJS-RT into a "pure" scheduling problem, i.e. we derive a formulation whose decision variables involve only starting times and robot assignments and whose constraints ensure the existence of feasible trajectories. The following development formalizes this approach.

Since the objective function in the BJS-RT depends only on α, a more compact formulation is obtained in principle by projecting Γ onto the space of the schedules. Letting

$$\begin{aligned}\Gamma_{\text{proj}} &= \{(\pi,\alpha) : \exists\ (\pi,\alpha,\mathbf{x}) \in \Gamma\} \\ &= \{(\pi,\alpha) : (\pi,\alpha) \in \Omega \text{ and } X(\pi,\alpha) \neq \emptyset\},\end{aligned}$$

the BJS-RT is then the problem of finding a schedule $(\pi,\alpha) \in \Gamma_{\text{proj}}$ minimizing α_τ.

The usefulness of this formulation depends on how the condition $X(\pi,\alpha) \neq \emptyset$ can be expressed more adequately and trajectories $\mathbf{x} = (x(k,.), k = 1,\ldots,K) \in X(\pi,\alpha)$ can be determined efficiently.

3.1 The feasible trajectory problem

Definition 5 Given a schedule $(\pi,\alpha) \in \Omega$, the *feasible trajectory problem (FTP) at* (π,α) is the problem of determining trajectories $\mathbf{x} = (x(k,.), k = 1,\ldots,K) \in X(\pi,\alpha)$ or establishing $X(\pi,\alpha) = \emptyset$.

We characterize when the FTP at (π,α) has a feasible solution, i.e. $X(\pi,\alpha) \neq \emptyset$, and define for this purpose the following *discrete version* of the FTP at (π,α).

Consider the set $\mathscr{Q} = \{\alpha(o), \alpha(o)+d(o) : o \in O\}$ of distinct starting and completion times of all transfer steps, and order $\mathscr{Q}$ such that $\mathscr{Q} = \{t_1,\ldots,t_Q\}$ with $Q \leq 2|O|$ and $t_1 < \ldots < t_Q$. Also, for any $0 \leq t \leq t'$, let $\mathscr{P}[t,t'] = \{p : 1 \leq p \leq Q \text{ and } t \leq t_p \leq t'\}$. For all $k = 1,\ldots,K$, $p = 1,\ldots,Q$, denote by $x_{kp} = x(k,t_p)$ the location of the robot k at t_p and consider the system:

$$|x_{k,p+1} - x_{kp}| \leq (t_{p+1} - t_p)v_k \text{ for all } k = 1,\ldots,K,\ p = 1,\ldots,Q-1, \tag{7}$$

$$x_{kp} = a_o \text{ for all } k = 1,\ldots,K,\ o \in O_k \text{ and } p \in \mathscr{P}[\alpha(o), \alpha(o)+d(o)], \tag{8}$$

$$x_{kp} + \delta \leq x_{k+1,p} \text{ for all } k = 1,\ldots,K-1,\ p = 1,\ldots,Q, \tag{9}$$

$$0 \leq x_{1p} \text{ and } x_{Kp} \leq L \text{ for all } p = 1,\ldots,Q. \tag{10}$$

(7) to (10) give a discrete version of the FTP at (π,α) as the following holds.

Proposition 1 *i) For any* $(\widehat{x}(k,.)$*:* $k = 1,\ldots,K)$ *satisfying (3) to (6),* $\widehat{x}_{kp} = \widehat{x}(k,t_p)$*,* $k = 1,\ldots,K$*,* $p = 1,\ldots,Q$ *satisfies (7) to (10).*
ii) For any $\widehat{x}_{kp}$*,* $k = 1,\ldots,K$*,* $p = 1,\ldots,Q$*, satisfying (7) to (10),*
$\widehat{x}(k,.)$*,* $k = 1,\ldots,K$*, defined by:*

$$\widehat{x}(k,t_p) = \widehat{x}_{kp},\ p = 1,\ldots,Q \text{ and} \tag{11}$$

$$\widehat{x}(k,t) = \frac{t_{p+1}-t}{t_{p+1}-t_p}\widehat{x}_{kp} + \frac{t-t_p}{t_{p+1}-t_p}\widehat{x}_{k,p+1},\ t_p < t < t_{p+1},\ p = 1,\ldots,Q-1 \tag{12}$$

satisfies (3) to (6).

Proof i) is obvious. ii) is easily proven by observing that with (11) and (12), the trajectory $\widehat{x}(k,.)$, $k \in \{1,\ldots,K\}$, is simply obtained by joining in the time-location space each pair of consecutive points $(t_p,\widehat{x}_{kp})$, $(t_{p+1},\widehat{x}_{k,p+1})$ by a line segment. □

Definition 6 For any k, k' with $1 \leq k < k' \leq K$, and any $o \in O_k$, $o' \in O_{k'}$, let

$$\Delta_{oo'}^{kk'} = [(k'-k)\delta + a_o - a_{o'}] / \min\{v_l : k \leq l \leq k'\}. \tag{13}$$

Assume $\Delta_{oo'}^{kk'} > 0$. Then obviously o and o' cannot occur simultaneously. Suppose o' is completed at time t' and o begins at time $t > t'$. In the interval $[t',t]$, robot k' needs to cover at least distance $(k'-k)\delta + a_o - a_{o'}$, and so do all robots l with $k \leq l \leq k'$. Similarly if $t < t'$, all robots l with $k \leq l \leq k'$ have to travel at least distance $(k'-k)\delta + a_o - a_{o'}$ in the interval $[t,t']$, hence

$$\alpha(o) + d(o) + \Delta_{oo'}^{kk'} \leq \alpha(o') \text{ or } \alpha(o') + d(o') + \Delta_{oo'}^{kk'} \leq \alpha(o).$$

Following prior work, we call such a disjunctive constraint a collision avoidance constraint. Indeed, constraints of this type have been introduced e.g. by Manier et al (2000), Leung and Zhang (2003) and Leung et al (2004) in the hoist scheduling problem. They establish necessity and sufficiency of these constraints by a case-by-case analysis of the various ways collisions between two hoists can occur. We show here necessity and sufficiency in the following Lemma by identifying the discrete FTP as a network problem in a graph H and showing the equivalence of the collision avoidance constraints with the absence of negative cycles in H.

Lemma 1 *(7) to (10) admits a solution if and only if for all $o \in O_k$, $o' \in O_{k'}$ with $k < k'$ and $\Delta_{oo'}^{kk'} > 0$:*

$$\alpha(o) + d(o) + \Delta_{oo'}^{kk'} \leq \alpha(o') \text{ or } \alpha(o') + d(o') + \Delta_{oo'}^{kk'} \leq \alpha(o) \tag{14}$$

Proof Let $H = (W, B, c)$ be the following graph. Node set W consists of node w_* and $K \times Q$ nodes w_{kp}, $k = 1,...,K$, $p = 1,...,Q$. The arc set B and the valuation $c \in \mathbb{R}^B$ are given in the table below:

arcs of B	*weights c*	
$(w_{k+1,p}, w_{kp})$	$-\delta$	$k = 1,...,K-1$, $p = 1,...,Q$,
$(w_{kp}, w_{k,p+1})$ $(w_{k,p+1}, w_{kp})$	$(t_{p+1} - t_p)v_k$	$k = 1,...,K$, $p = 1,...,Q-1$,
(w_*, w_{kp}) (w_{kp}, w_*)	a_o $-a_o$	for all $k = 1,...,K$, $o \in O_k$ and $p \in \mathscr{P}[\alpha(o), \alpha(o) + d(o)]$,
$(w_{1p}, w_*)^0$ $(w_*, w_{Kp})^L$	0 L	$p = 1,...,Q$.

Note that parallel arcs are present, explaining the indexing 0 and L in $(w_{1p}, w_*)^0$ and $(w_*, w_{Kp})^L$. Graph H is depicted in Figure 5.

It is easy to see that the system (7) to (10) is equivalent to the following system of inequalities in graph $H = (W, B, c)$:

$$x_w - x_v \leq c_{vw} \text{ for all } (v,w) \in B, \tag{15}$$

$$x_{w_*} = 0, \tag{16}$$

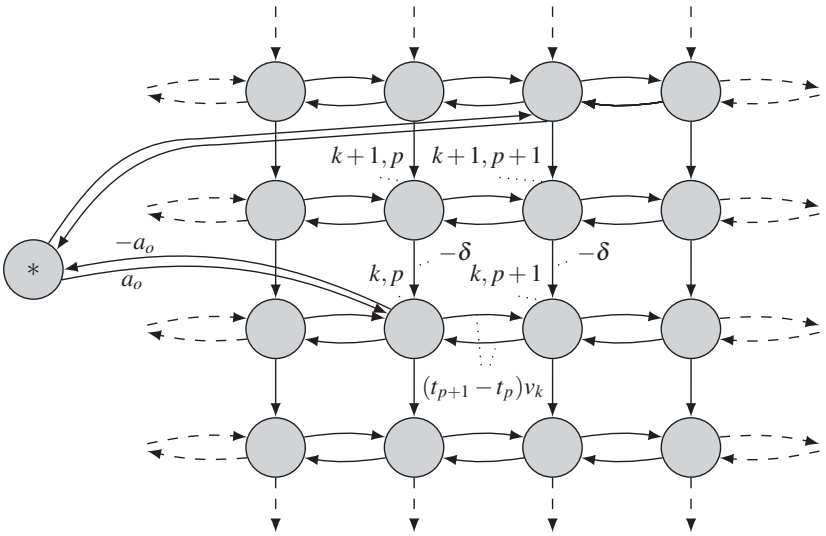

Fig. 5: Graph H. (Not all arcs are shown.)

i.e. x satisfying (15) and (16) is a *feasible potential function*. By a well-known result of combinatorial optimization (see e.g. Cook et al (1997), p. 25), $H=(W,B,c)$ admits a feasible potential function - and hence (7) to (10) admits a feasible solution - if and only if there exists no cycle of negative length in H.

We prove therefore that constraints (14) hold if and only if H has no negative cycle.

First, the following observations are useful. Consider the graph H^- obtained from H by deleting node w_*. H^- contains no cycle of negative length. Also, there exists a path in H^- from a node $w_{k'p'}$ to a node w_{kp} if and only if $k \le k'$. Finally, it is easy to see that a shortest path in H^- from $w_{k'p'}$ to w_{kp} has length $|t_{p'}-t_p| \cdot \min\{v_l : k \le l \le k'\} - (k'-k)\delta$.

i) Suppose now that (14) does not hold: there are $o \in O_k$, $o' \in O_{k'}$, with $k' > k$ and $\Delta_{oo'}^{kk'} > 0$, such that $\Delta_{oo'}^{kk'} > \alpha(o') - \alpha(o) - d(o)$ and $\Delta_{oo'}^{kk'} > \alpha(o) - \alpha(o') - d(o')$. If o and o' are both in execution at some time $t_p, p \in \{1,\dots,Q\}$, then the cycle Z in H consisting of arc $(w_*, w_{k'p})$, a shortest path in H^- from $w_{k'p}$ to w_{kp} and arc (w_{kp}, w_*) has length

$$c(Z) = a_{o'} - (k'-k)\delta - a_o = -\Delta_{oo'}^{kk'} \cdot \min\{v_l : k \le l \le k'\},$$

hence $c(Z) < 0$. If o is executed before o', i.e. $\alpha(o) + d(o) \le \alpha(o')$, let p and p' be such that $t_p = \alpha(o) + d(o)$ and $t_{p'} = \alpha(o')$. Then the cycle Z in H consisting of arc $(w_*, w_{k'p'})$, a shortest path in H^- from $w_{k'p'}$ to w_{kp} and arc (w_{kp}, w_*) has length

$$\begin{aligned} c(Z) &= a_{o'} + |t_{p'} - t_p| \cdot \min\{v_l : k \le l \le k'\} - (k'-k)\delta - a_o \\ &= [\alpha(o') - \alpha(o) - d(o) - \Delta_{oo'}^{kk'}] \cdot \min\{v_l : k \le l \le k'\} < 0. \end{aligned}$$

Finally, if o' is executed before o, i.e. $\alpha(o') + d(o') \le \alpha(o)$, the existence of a negative cycle is shown similarly.

ii) Conversely, suppose H has a cycle Z of negative length. By the preceding observations, Z must pass through node w_*, leaving w_* by an arc b' with head $w_{k'p'}$

and entering w_* by an arc b with tail w_{kp} for some $k \leq k'$ and p, p'. Also, we may assume that Z takes a shortest path in H^- from $w_{k'p'}$ to w_{kp}. Hence Z has length

$$c(Z) = c_{b'} + c_b + |t_{p'} - t_p| \cdot \min\{v_l : k \leq l \leq k'\} - (k' - k)\delta < 0.$$

First, we exclude the following three cases for b' and b. Case a): $b' = (w_*, w_{Kp'})^L$ and $b = (w_{1p}, w_*)^0$. We may assume $p' = p$ since $(w_{1p}, w_*)^0 \in B$ with same weight 0, so that $c(Z) = L + 0 - (K-1)\delta < 0$, violating the standard assumption $\max\{a_m : m \in M\} \leq L - (K-1)\delta$. Case b): $b' = (w_*, w_{Kp'})^L$ and $b = (w_{kp}, w_*)$ with weight $-a_o$. We may assume $p' = p$ since $(w_*, w_{Kp'})^L \in B$ with same weight L, so that $c(Z) = L - a_o - (K-k)\delta < 0$, in contradiction to $\max\{a_m : m \in M\} \leq L - (K-1)\delta$. Case c) $b' = (w_*, w_{k'p'})$ with weight $a_{o'}$ and $b = (w_{1p}, w_*)^0$. We may assume $p = p'$, so that $c(Z) = a_{o'} - (k'-1)\delta < 0$, contradicting the assumption $(K-1)\delta \leq \min\{a_m : m \in M\}$.

Therefore arc b' is $(w_*, w_{k'p'})$ for some $o' \in O_{k'}$ and $\alpha(o') \leq t_{p'} \leq \alpha(o') + d(o')$, and arc b is (w_{kp}, w_*) for some $o \in O_k$ and $\alpha(o) \leq t_p \leq \alpha(o) + d(o)$. The case $k = k'$ can be excluded, using the fact that for any two $o, o' \in O_k$, with say, $\alpha(o) \leq \alpha(o')$, $\alpha(o') - (\alpha(o) + d(o)) \geq |a_{o'} - a_o|$ holds. Therefore $k < k'$ and the length of Z is

$$\begin{aligned} c(Z) &= a_{o'} - a_o + |t_{p'} - t_p| \cdot \min\{v_l : k \leq l \leq k'\} - (k' - k)\delta \\ &= [|t_{p'} - t_p| - \Delta_{oo'}^{kk'}] \cdot \min\{v_l : k \leq l \leq k'\} < 0 \end{aligned}$$

Therefore $|t_{p'} - t_p| - \Delta_{oo'}^{kk'} < 0$, so that $\Delta_{oo'}^{kk'} > 0$, and both $\Delta_{oo'}^{kk'} > t_{p'} - t_p$ and $\Delta_{oo'}^{kk'} > t_p - t_{p'}$ hold. Then $\Delta_{oo'}^{kk'} > t_{p'} - t_p \geq \alpha(o') - \alpha(o) - d(o)$ and $\Delta_{oo'}^{kk'} > t_p - t_{p'} \geq \alpha(o) - \alpha(o') - d(o')$, so that (14) is violated for this pair o, o'. □

Assuming that the FTP at (π, α) has a feasible solution, trajectories $\mathbf{x} = (x(k,.), k = 1, \ldots, K)$ can be determined by finding a potential function in H - an elementary task in network flows - and applying Proposition 1. Furthermore, a natural objective is to find trajectories minimizing the total distance traveled by the robots. It is easy to define a network problem in an adapted graph H that finds a potential optimizing this objective and then apply Proposition 1. However, optimal trajectories can also be determined more efficiently with an algorithm based on geometric arguments, as shown in the Appendix.

3.2 Projection onto the space of schedules

A compact disjunctive graph formulation of the BJR-RT is now readily obtained by introducing in $G = (V, A, E, \mathscr{E}, d)$ additional conjunctive and disjunctive arcs to take into account constraints (14) for any mode.

First, observe that each transfer step of a transport operation on a given robot is represented by a specific arc in G. Indeed, a transfer step o on robot k is either o_i or $\overline{o}_i$ for some $i \in I^R$ executed by k; o_i on k is represented in G by the arc (v_{ik}^1, v_{ik}^2) and $\overline{o}_i$ on k by (v_{ik}^3, v_{ik}^4).

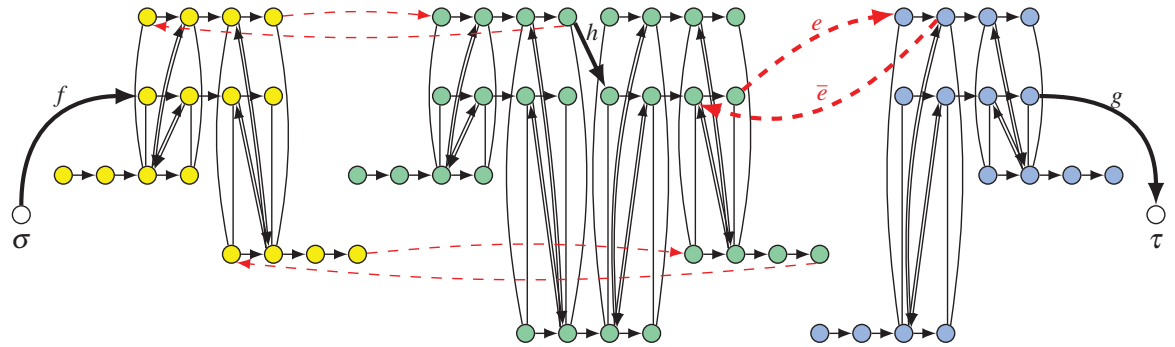

Fig. 6: Some collision avoidance arcs (e, $\bar{e}$, f, g and h) in the example.

Second, conflicts of a transfer step o of a transport operation executed by a robot k with the fictive initial and final transfer steps $\sigma_{k'}$ and $\tau_{k'}$ of the robots $k' \neq k$, simply result in an initial set-up time and a final set up time for o on k. Also, a conflict between two transfer steps o, o' (of distinct transport operations) of a *same* job simply results in a precedence constraint.

Conjunctive arcs are now added to A and disjunctive arcs are added to E, respectively arc pairs to $\mathscr{E}$, as specified in the three steps below, convening that arcs *are added only if their weight is positive*:

1. For all $o \in \{o_i, \overline{o}_i : i \in I^R\}$ and all k, $1 \leq k \leq K$, if (v,w) represents o on k, add to A the arc (σ, v) with weight $\Delta^k_{\sigma o} = \max\{0; \Delta^{kk'}_{o\sigma_{k'}} : k < k'; \Delta^{k'k}_{\sigma_{k'} o} : k > k'\}$, and (w,τ) with weight $\Delta^k_{o\tau} = \max\{0; \Delta^{kk'}_{o\tau_{k'}} : k < k'; \Delta^{k'k}_{\tau_{k'} o} : k > k'\}$. (If an arc is added that is parallel to an arc already present, retain only the arc with largest weight.)
2. For each $o, o' \in \{o_i, \overline{o}_i : i \in I^R\}$ where o and o' are transfer steps of *distinct* transport operations of a *same* job, assuming without loss of generality that o precedes o', for each $k \neq k'$, if (v,w) and (v',w') represent o on k and o' on k', add to A the arc (w,v') with weight $\Delta^{kk'}_{oo'}$ if $k' > k$ and $\Delta^{k'k}_{o'o}$ if $k' < k$.
3. For all $o, o' \in \{o_i, \overline{o}_i : i \in I^R\}$ where o and o' are transfer steps of *distinct* jobs, and all k, k' with $1 \leq k < k' \leq K$, if (v,w) and (v',w') represent o on k and o' on k', add to E, respectively to $\mathscr{E}$, the pair of arcs $(w,v'), (w',v)$, both of weight $\Delta^{kk'}_{oo'}$.

Denote by $G' = (V, A', E', \mathscr{E}', d')$ the disjunctive graph thus obtained. Figure 6 depicts G' in the example, obtained by adding in G of Figure 4 conjunctive and disjunctive arcs as described above. For sake of clarity, only two additional arcs f and g from step 1, arc h from step 2 and disjunctive arc pair $e, \bar{e}$ from step 3 are displayed. The weights of e, $\bar{e}$, f, g and h are 2, 2, 3.5, 3.5 and 1. In the example, G contains altogether 15 disjunctive arcs pairs, and 32 arcs in step 1, 5 arcs in step 2 and 26 disjunctive arc pairs in step 3 are added to obtain G'.

Define in G' modes, (complete, acyclic, feasible) selections, and $\mathscr{F}'^{\pi}$, $\Omega'(\pi, S')$, $\Omega'(\pi)$ and Ω' similarly to the corresponding definitions in G given in Section 2.2.

Theorem 1 *The projection Γ_{proj} of the set of schedules with trajectories (defined in G) is precisely the set of schedules Ω' defined in G'.*

Proof i) The FTP at (π,α) in G admits a feasible solution, i.e. $X(\pi,\alpha)\neq\emptyset$, if and only the constraints (14) hold. Indeed, by Proposition 1 the FTP at (π,α) admits a feasible solution if and only if its discrete version (7) to (10) admits a feasible solution, hence by Lemma 1, if and only if (14) holds. Therefore $\Gamma_{\text{proj}}=\{(\pi,\alpha):\pi\in\Pi,\ \alpha\in\Omega(\pi)\text{ and (14) holds}\}$.

ii) Let $(\pi,\alpha)\in\Omega'=\{(\pi,\alpha):\pi\in\Pi,\ \alpha\in\Omega'(\pi)\}$, hence $\alpha\in\Omega'(\pi,S')$ for some $S'\in\mathscr{F}'^{\pi}$. Then for $S=S'\cap E\in\mathscr{F}^{\pi}$ and, since $\Omega'(\pi,S')\subseteq\Omega(\pi,S)$, $\alpha\in\Omega(\pi,S)$. Therefore $\alpha\in\Omega(\pi)$. Also, the constraints (1)' for $(v,w)\in A'\cup S'-A\cup S$ ensure that (14) is satisfied by α. Hence $\Omega'\subseteq\Gamma_{\text{proj}}$. Conversely, let $(\pi,\alpha)\in\Gamma_{\text{proj}}$. There exists $S\in\mathscr{F}^{\pi}$ such that $\alpha\in\Omega(\pi,S)$ and α satisfies (14). Then $S'=S\cup\{(v,w)\in E'-E:\alpha_v\leq\alpha_w\}\in\mathscr{F}'^{\pi}$ and $\alpha\in\Omega'(\pi,S')$, hence $\alpha\in\Omega'(\pi)$ and $\Gamma_{\text{proj}}\subseteq\Omega'$. □

The BJS-RT can therefore be formulated as: "among all feasible selections in G', find a selection (π,S') minimizing the length of a longest path from σ to τ in $(V^{\pi},A'^{\pi}\cup S',d')$".

4 A local search for the BJS-RT

Since the relaxed BJS-RT of Section 2.2 is a Flexible Blocking Job Shop (FBJS), the local search heuristic proposed below is inspired from our approach for the FBJS in (Gröflin et al, 2011). It operates in the disjunctive graph G' and its main ingredient is the construction of a feasible neighborhood.

4.1 A feasible neighborhood

Given a current feasible selection, a feasible neighbor selection will be generated by rescheduling an operation, keeping the machine or robot it is currently assigned to, or - if it is a transport operation - also allowing it to change its assigned robot. Moves of other operations might be "implied" to maintain feasibility.

The framework for the construction of a feasible neighbor selection is the so-called job insertion similarly to (Gröflin et al, 2011), although slightly complicated by the presence of the disjunctive collision avoidance arcs.

Given a feasible selection (π,S) in G' and a (machining or transport) operation $i\in I^M\cup I^R$, let J be the job to which i belongs, and consider the following insertion problem: extract J and (re-)insert J, keeping the same mode π for all operations $j\in I^R-i$ and allowing any robot k for i if $i\in I^R$. The ways to insert J are described by the feasible selections, called *insertions*, in the disjunctive subgraph $G_J=(V_J,A_J,E_J,\mathscr{E}_J,d)$ of G', called *insertion graph* and obtained as follows.

Delete from V all node sets V_{jk}, $j\in I^R-i$ and $k\neq\pi(j)$, obtaining V_J. Then, letting $W_J=\cup\{V_j:j\in J;V_{j,\pi(j)}:j\in J-i;V_{ik}:k=1,\ldots K\}$ be the subset of nodes of G_J associated to J, add to A the set

$$R_J=S-\delta(W_J) \tag{17}$$

obtaining A_J, and delete all disjunctive arcs not incident to J, i.e. obtaining $E_J = E' - \delta(W_J)$ and $\mathscr{E}_J$ accordingly.

Note that if $i \in I^R$, the set of possible modes in G_J is $\Pi' = \{\pi^k : \ k = 1,...K\}$ where π^k is given by $\pi^k(j) = \pi(j)$ for $j \in I^R - i$ and $\pi^k(i) = k$, and if $i \in I^M$, the mode remains unchanged, i.e. $\Pi' = \{\pi\}$. In both cases, $\pi \in \Pi'$.

Let $G_J^{\pi'} = (V_J^{\pi'}, A_J^{\pi'}, E_J^{\pi'}, \mathscr{E}_J^{\pi'}, d)$ be the disjunctive subgraph of G_J associated to π', $\pi' \in \Pi'$. For any $T \subseteq E_J^{\pi'}$, (π', T) is an *insertion*; (π', T) is i) *positive acyclic* if $(V_J^{\pi'}, A_J^{\pi'} \cup T, d)$ contains no positive cycle; ii) *complete* if $\{e, \overline{e}\} \cap T \neq \emptyset$ for all $(e, \overline{e}) \in \mathscr{E}_J^{\pi'}$ and iii) *feasible* if positive acyclic and complete.

Let $T^S = S \cap \delta(W_J)$. Clearly, $\left(\pi, T^S\right)$ is a feasible insertion in G_J. Generating a feasible neighbor selection (π', S') will be done by generating in G_J a feasible neighbor insertion (π', T') of $\left(\pi, T^S\right)$ and setting $S' = R_J \cup T'$ where R_J is given by (17).

To derive feasible neighbor insertions, we use the concepts of *closure* and *span* in graph $G_J^{\pi'} = (V_J^{\pi'}, A_J^{\pi'}, E_J^{\pi'}, \mathscr{E}_J^{\pi'}, d)$. They are introduced in (Gröflin and Klinkert, 2007; Gröflin et al, 2011) and recalled here for completeness. For any $T \subseteq E_J^{\pi'}$, let $\varphi(T) = T \cup \{e \in E_J^{\pi'} - T : (V_J^{\pi'}, A_J^{\pi'} \cup T \cup \overline{e}, d)$ contains a positive cycle Z with $Z \ni \overline{e}\}$. Clearly, any feasible insertion (π', T') with $T \subseteq T'$ must contain $\varphi(T)$, hence arcs $\varphi(T)$ are implied by T. We call T *closed* if $\varphi(T) = T$. It is easy to see that if T and $\widehat{T}$ are closed then $T \cap \widehat{T}$ is closed, so that the following closure operator Φ is well-defined for all $T \subseteq E_J^{\pi'}$:

$$\Phi(T) = \cap\{\widehat{T} \subseteq E_J^{\pi'} : T \subseteq \widehat{T}, \widehat{T} \text{ closed}\}.$$

$\Phi(T)$ is the unique smallest closed set containing T and can be computed by repeatedly applying φ, defining $\varphi^r(T) = T$ for $r = 0$ and computing $\varphi^r(T) = \varphi(\varphi^{r-1}(T))$ for $r = 1, 2, ...$, until $\varphi^{r+1}(T) = \varphi^r(T)$.

The span of an insertion comprises all its arcs together with their mates. Specifically, for any $T \subseteq E_J^{\pi'}$, the span of T is the set

$$[T] = \{e \in E_J^{\pi'} : e \text{ or } \overline{e} \in T\}.$$

Feasible neighbors (π', T') of $\left(\pi, T^S\right)$ are now constructed as follows. If the mode remains unchanged, i.e. $\pi' = \pi$, a neighbor $\left(\pi, T^f\right)$ is generated by choosing some arc $e \in T^S$ incident to i and enforcing its mate. Specifically, choose $e \in T^S \cap \delta(V_i)$ if $i \in I^M$, or $e \in T^S \cap \delta(V_{i,\pi(i)})$ if $i \in I^R$, and enforce its mate $f = \overline{e}$:

$$T^f = \Phi(f) \cup (T^S - [\Phi(f)]). \tag{18}$$

The neighbor insertion $\left(\pi, T^f\right)$ keeps the operation i on the same machine or robot, enforces f and all arcs implied by f (set $\Phi(f)$) and keeps T^S on the remaining part (set $T^S - [\Phi(f)]$).

If the mode changes, i is a transport operation which is moved to another robot $k \neq \pi(i)$, i.e. the new mode is $\pi^k \neq \pi$, and i needs to be inserted in the sequence of the other operations on k. However, also the positioning of the hand-over step o_i and take-over step $\overline{o}_i$ of i with respect to those transfer steps on a robot l in conflict

with o_i, respectively $\overline{o}_i$, needs to be specified for all $l \neq k$. A feasible neighbor insertion $\left(\pi^k, T^F\right)$ is thus generated in the graph $G_J^{\pi^k} = (V_J^{\pi^k}, A_J^{\pi^k}, E_J^{\pi^k}, \mathscr{E}_J^{\pi^k}, d)$ by the following three successive steps. i) Choose some arc set F that enforces i to be before some operation on robot k and puts the transfer steps of i before some conflicting transfer steps. Take all arcs implied by F, say $P = \Phi(F)$. ii) Place operation i after all operations on robot k that have not already been sequenced with respect to i, and similarly, place the transfer steps of i after all conflicting transfer steps that have not been sequenced with respect to the transfer steps of i, by choosing $E_i^- - [P]$ and taking all arcs implied by it, say $Q = \Phi(E_i^- - [P])$. iii) Keep T^S on the remaining part. Specifically,

$$\begin{aligned} T^F = P \cup Q \cup (T^S - [P \cup Q]) \text{ where} \\ P = \Phi(F) \text{ and } Q = \Phi(E_i^- - [P]). \end{aligned} \tag{19}$$

Technically, set F is build as follows. Let $F_i = \{e \in E_J^{\pi^k} : t(e) \in V_{ik} \text{ and } h(e) \in V_{jk} \text{ for some } j \notin J\}$, and for all $l \neq k$, $F_{o_i}^l = \{e \in E_J^{\pi^k} : t(e) = v_{ik}^2 \text{ and } h(e) \in V_{jl} \text{ for some } j \notin J\}$ and $F_{\overline{o}_i}^l = \{e \in E_J^{\pi^k} : t(e) = v_{ik}^4 \text{ and } h(e) \in V_{jl} \text{ for some } j \notin J\}$. Note that some of these sets might be empty. At most one arc from each of these sets will be chosen to "position" i with respect to the other operations on k, and o_i and $\overline{o}_i$ with respect to transfer steps in conflict with o_i, respectively $\overline{o}_i$, on a robot l, $l \neq k$. In order to generate a "close" neighbor insertion, these choices are made so that i is likely to be scheduled at a time not too far from its time in the current schedule. Let α_v, $v \in V$, be the earliest starting times computed in $(V_J^{\pi}, A_J^{\pi} \cup T^S, d)$. Determine the arc $f_i \in F_i$ with $\alpha_{h(f_i)} = \min\{\alpha_{h(e)} \geq \alpha_{v_{i\pi(i)}^1} : e \in F_i\}$ convening $f_i = \emptyset$ if it does not exist. Similarly, for all $l \neq k$, determine the arcs $f_{o_i}^l \in F_{o_i}^l$ with $\alpha_{h(f_{o_i}^l)} = \min\{\alpha_{h(e)} \geq \alpha_{v_{i\pi(i)}^1} : e \in F_{o_i}^l\}$ and $f_{\overline{o}_i}^l \in F_{\overline{o}_i}^l$ with $\alpha_{h(f_{\overline{o}_i}^l)} = \min\{\alpha_{h(e)} \geq \alpha_{v_{i\pi(i)}^3} : e \in F_{\overline{o}_i}^l\}$, with the same convention if an arc does not exist. Then, $F = \{f_i, f_{o_i}^l, f_{\overline{o}_i}^l : 1 \leq l \leq K, l \neq k\}$ and $E_i^- = \{e \in E_J^{\pi^k} : h(e) \in V_{ik}\}$.

The feasibility of the insertions (π, T^f) and (π^k, T^F) determined by (18) and (19) can be shown similarly to corresponding proofs in (Gröflin et al, 2011), after establishing the short cycle property of the disjunctive graphs G_J^{π} and $G_J^{\pi^k}$.

4.2 A tabu search

Based on the neighborhood described above, the following tabu search has been developed. First, only *critical* operations are considered for rescheduling. Given a feasible selection (π, S) in G', let $\mathscr{L}$ be the arc set of an arbitrary longest path from σ to τ in $(V'^{\pi}, A'^{\pi} \cup S, d)$. The arcs of $S \cap \mathscr{L}$ are usually called *critical arcs*. For any $e \in S$, call i the *tail operation of* e if $t(e) \in V_i$ or $V_{i,\pi(i)}$ and the *head operation of* e if $h(e) \in V_i$ or $V_{i,\pi(i)}$. Critical operations are head or tail operations of critical arcs.

For each $e \in S \cap \mathscr{L}$, two neighbors (π, T^f) are generated according to (18), with i being the head and tail operation of e respectively, and $f = \overline{e}$. Additionally, if i is the head or tail operation of e and is a transport operation, then i is moved to an adjacent

robot $k = \pi(i)+1$ or $\pi(i)-1$, provided $1 \le k \le K$, and the neighbor insertion (π^k, T^F) is constructed according to (19). Between two and six neighbors are thus generated for each critical arc.

A tabu list of maximum length *maxt* is maintained, containing entries of the *maxt* last iterations. Let $L_{(\pi,S)}$ be the list associated to the selection (π,S). The list associated to the initial selection is empty. In an iteration, i.e. after moving from the selection (π,S) to a neighbor (π',S'), $L_{(\pi',S')}$ is obtained from $L_{(\pi,S)}$ by dropping the oldest entry if $|L_{(\pi,S)}| = maxt$, and placing in the first position either the entry $e = \overline{f}$ if the neighbor is generated with an insertion (π, T^f), or the entry $(i,\pi(i))$ if operation i has been moved from robot $\pi(i)$ to robot k with an insertion (π^k, T^F). Hence, the tabu list $L_{(\pi,S)}$ contains arcs that were reversed and old robot assignments that were changed. A neighbor (π',S') of (π,S) is *tabu* if it contains an arc or robot assignment that is in the tabu list, i.e. $S' \cap L_{(\pi,S)} \neq \emptyset$ or $\pi'(i) = l$ for some $(i,l) \in L_{(\pi,S)}$.

For further details, e.g. on the choice of the move to be executed, the termination of a search path after a maximum number *maxiter* of iterations without improving the best makespan and the maintenance of a list of bounded length *maxl* of elite solutions, a feature that has proven useful also in the "classical" job shop (Nowicki and Smutnicki, 1996), we refer the reader to (Gröflin et al, 2011).

5 Computational results

The tabu search has been implemented (single-threaded) in Java and run on a PC with 3.1 GHz Intel Core i5-2400 processor (4 threads) and 4 GB memory. Since the BJS-RT has not been addressed in the literature, we created a test set of 160 instances starting from the standard job shop instances la01 to la20 introduced by Lawrence (1984) and adding data to describe the transportation system.

For each Lawrence-instance la*pq* and number of robots $K = 1,\ldots,4$, two BJS-RT instances la*pq*-E and la*pq*-V were generated as follows. In both instances, the location of machine $m_i, i = 0,\ldots,|M|-1$ is $a_{m_i} = 120+50i$, where $|M|$ is the number of machines and i is the number attributed by Lawrence. The initial and final locations of robot $k = 1,\ldots,K$ are $a_{\sigma k} = a_{\tau k} = 40(k-1)$. The minimum distance between adjacent robots is $\delta = 40$, and the rail length is $L = 120+50(|M|-1)+\delta(K-1)$. The loading times are $d_{\mathrm{ld},i} = 10$ for all $i \in I^{\mathrm{first}}$, the unloading times $d_{i,\mathrm{uld}} = 10$ for all $i \in I^{\mathrm{last}}$, and the initial and final set-up times are $d_{\sigma i} = d_{i\tau} = 0$ for all $i \in I^M$. In instance la*pq*-E, the maximum speeds and transfer times are equal, and there are no machine set-ups. Specifically, the maximum speed is $v_k = 10$ for all robots $k = 1,\ldots,K$, the transfer times are $d^r_{ij} = d^r_{ji} = 10$ for all $r \in R$, $i \in I^M$, $j \in I^R$, and the set-up times are $d_{ij} = 0$ for distinct $i,j \in I^M$ with $m_i = m_j$. In instance la*pq*-V, the maximum speeds and transfer times vary, and machine set-ups are present. The maximum speeds v_k and transfer times d^r_{ij} are robot-dependent and generated randomly based on a uniform distribution in the interval [5,10] and [5,20], respectively, and machine set-up times are specified based on a format introduced by Brucker and Thiele (1996) in the interval [0,40].

With la01 to la20, the problem sizes in the test set are 10×5 (10 jobs on 5 machines), 15×5, 20×5 and 10×10. These sizes might appear modest at first glance,

but should be used with caution when comparing the BJS-RT for example with the classical job shop problem for which la01 to la20 were originally introduced. In a BJS-RT, an $m \times n$ instance contains nearly twice the number of operations since for each job with m (machining) operations, $m-1$ transport operations are introduced. Moreover, there are typically many collision avoidance constraints between the $2(m-1)n$ transfer steps. Finally, flexibility in choosing a robot further increases complexity.

The computation settings were chosen similarly to (Gröflin et al, 2011). An initial solution is a permutation schedule generated by choosing randomly a job permutation and a mode. For each instance, five independent runs with different initial solutions were performed. The computation time of a run was limited to 1800 seconds. Parameter tuning was performed for the tabu search in extensive experiments, carefully observing not only the makespans of the obtained solutions, but also characteristics such as cycling, usage of the elite solutions and makespan improvement behavior. Detailed numerical data as well as graphical outputs were helpful in this regard. The following parameter values were chosen: $maxt = 12$, $maxl = 300$ and $maxiter = 3000$, and turned out to be quite robust.

Tables 1 and 2 provide detailed results for instances lapq-E and lapq-V, respectively. The first line splits the tables into four groups according to the number of robots. Columns "best" and "mean" refer to the best and mean results, respectively, of the five runs. The tables are subdivided horizontally according to the size of the instances, e.g. the first block reports on instances 10×5 with 10 jobs and 5 machines. We discuss now the results, evaluating solution quality, convergence behavior of the tabu search and impact of increasing the number of robots.

Since the BJS-RT has not yet been addressed in publications, a comparison of our results with benchmarks was not possible. For this reason, we tried to assess the quality of the tabu search with results obtained via a MIP model that we derived in a straightforward manner from the disjunctive graph formulation. Instances la01-E to la05-E with 1 robot have been solved to optimality with the MIP model, using the solver Gurobi 5.0 and a time limit of five hours. However, with 2 robots, no feasible solution could even be found for la01-E to la05-E. We reduced the size of these instances by keeping only the first six jobs (out of ten). These instances, called la01-E* to la05-E*, were solved by Gurobi, after providing the best solution found by the tabu search as an initial solution and allowing more computation time. Table 3 shows the results obtained for the instances with 1 robot (left) and 2 robots (right). Columns "result" give the optimal values or the upper and lower bounds $(ub; lb)$ if optimality could not be established. Columns "time" give the computation time in seconds used by Gurobi, and columns "best" and "mean" the results of the tabu search. The following observations can be made. As is the case in other complex scheduling problems, only small instances could be solved to optimality with a MIP approach and even finding a feasible solution appears to be a challenge in multiple robot instances. Comparing now the MIP and tabu search results, in all 10 instances, the best of the five runs reached the MIP optimum or upper bound, and all five runs yield results that are as good or very close. Albeit limited, these results suggest that the tabu search performs adequately.

# robots	1 robot		2 robots		3 robots		4 robots	
instance	best	mean	best	mean	best	mean	best	mean
10×5								
la01-E	1736	1746	1315	1356	1155	1196	1108	1136
la02-E	1727	1727	1329	1353	1203	1222	1155	1171
la03-E	1695	1695	1262	1284	1089	1124	1044	1104
la04-E	1748	1749	1280	1299	1140	1165	1044	1089
la05-E	1654	1655	1251	1270	1111	1130	1057	1099
15×5								
la06-E	2465	2478	1899	1974	1726	1760	1655	1693
la07-E	2473	2496	1964	1990	1707	1764	1638	1659
la08-E	2483	2502	1912	1949	1748	1790	1675	1716
la09-E	2501	2520	2018	2056	1747	1813	1696	1718
la10-E	2529	2550	1968	2014	1777	1818	1692	1748
20×5								
la11-E	3381	3399	2640	2749	2349	2478	2424	2446
la12-E	3296	3326	2541	2696	2188	2251	2128	2262
la13-E	3335	3373	2624	2655	2364	2402	2192	2338
la14-E	3391	3419	2690	2823	2499	2604	2323	2433
la15-E	3353	3384	2723	2807	2385	2514	2216	2407
10×10								
la16-E	4664	4967	2907	3216	2652	2853	2392	2666
la17-E	4608	4776	3079	3340	2774	2968	2539	2826
la18-E	4655	4827	3304	3438	2699	2857	2476	2881
la19-E	4562	4683	3051	3299	2500	2757	2333	2662
la20-E	4710	4786	3019	3362	2736	2986	2360	2761

Table 1: Best and mean results over five runs (time limit: 1800 seconds per run) in the instances la*pq*-E with equal maximum speeds and transfer times, and without machine setup times.

Further support is found by examining the evolution of attained solution quality over computation time. For this purpose, the best makespan ω at the beginning (initial solution) and during the execution of the tabu search were recorded for each instance and run, and its (relative) deviation from the final solution $(\omega - \omega_{final})/\omega_{final}$ determined. Figure 7 illustrates these deviations for all instances with 3 robots in an aggregated way, depicting average deviations (in %) over runs and instances of the same size. The following can be observed. Initial solutions are far from the obtained final solutions, with makespans twice to three times as large. Also, most of the improvements are found within minutes. Consider for example the 10×10 instances with 3 robots. While the deviation is initially 166.4%, it drops to 9.9% and 5.8% after 300 and 600 seconds.

Furthermore, we investigated the impact of adding a robot to the transportation system. Information of this type may be of interest at the design stage, when capacity is calibrated or the benefit of installing additional equipment is assessed. We compared each instance la*pq*-E with K robots with the same instance with $K-1$ robots by determining the relative change in the makespan $(mean_K - mean_{K-1})/mean_{K-1}$, where $mean_i, i = 1, \ldots, 4$, can be found in Table 1, column "mean" of group "i robots". Table 4 (left) reports these changes in % in an aggregated way. As expected, adding a robot reduces the makespan, and this return diminishes with the number of robots.

# robots	*1 robot*		*2 robots*		*3 robots*		*4 robots*	
instance	*best*	*mean*	*best*	*mean*	*best*	*mean*	*best*	*mean*
10×5								
la01-V	3075	3076	2044	2085	1479	1509	1327	1401
la02-V	1673	1673	1460	1508	1386	1423	1355	1425
la03-V	2023	2023	1552	1591	1352	1430	1315	1407
la04-V	2016	2018	1544	1571	1352	1373	1310	1340
la05-V	2034	2044	1254	1291	1221	1233	1199	1241
15×5								
la06-V	2974	2997	2257	2297	2002	2139	2072	2159
la07-V	3067	3078	2604	2637	2298	2354	2135	2170
la08-V	3695	3710	2547	2593	2195	2273	2026	2070
la09-V	2394	2430	2199	2272	2022	2129	1928	2050
la10-V	3529	3558	2240	2321	2015	2048	1906	1988
20×5								
la11-V	3258	3283	2722	2808	2595	2669	2558	2660
la12-V	3686	3742	3260	3352	2905	2988	2818	2938
la13-V	3152	3201	2905	3013	2732	2799	2557	2626
la14-V	3349	3378	3121	3288	2870	2970	2865	2966
la15-V	4616	4636	3843	3900	3016	3112	2813	2988
10×10								
la16-V	5836	6383	3946	4111	3096	3347	2864	3051
la17-V	4352	4584	3836	3959	3168	3311	2828	3111
la18-V	6324	6497	4651	4835	3475	3839	3541	3754
la19-V	6809	7056	4605	4959	3699	3889	3653	4248
la20-V	5980	6252	4376	4739	3459	3825	3294	3484

Table 2: Best and mean results over five runs (time limit: 1800 seconds per run) in the la*pq*-V instances with robot-dependent maximum speeds and transfer times, and with machine setup times.

1 robot	*MIP*		tabu search	
instance	*result*	*time*	*best*	*mean*
la01-E	1736	3000	1736	1746
la02-E	(1727;1556)	18000	1727	1727
la03-E	1695	1638	1695	1695
la04-E	1748	7016	1748	1749
la05-E	(1654;1347)	18000	1654	1655

2 robots	*MIP*		tabu search	
instance	*result*	*time*	*best*	*mean*
*la01-E**	832	8840	832	835
*la02-E**	864	51892	864	864
*la03-E**	833	15636	833	833
*la04-E**	823	13225	823	823
*la05-E**	765	230228	765	765

Table 3: MIP results and computation times compared to best and mean results over five runs (time limit: 1800 seconds per run) of the tabu search.

Going from 1 to 2 robots (column "2 robots") reduces the makespan significantly, the range of the decrease being 18% to 31%. Adding a third robot yields a decrease of 10% to 14%, and adding a fourth robot, a decrease of 3% to 5%.

Finally, Table 4 (right) shows the number of tabu search iterations averaged over instances of the same size. With increasing number of robots and problem size, the number of iterations drops drastically reflecting the increasing computation time per iteration. This is due to an increase of the neighborhood size and to the fact that the

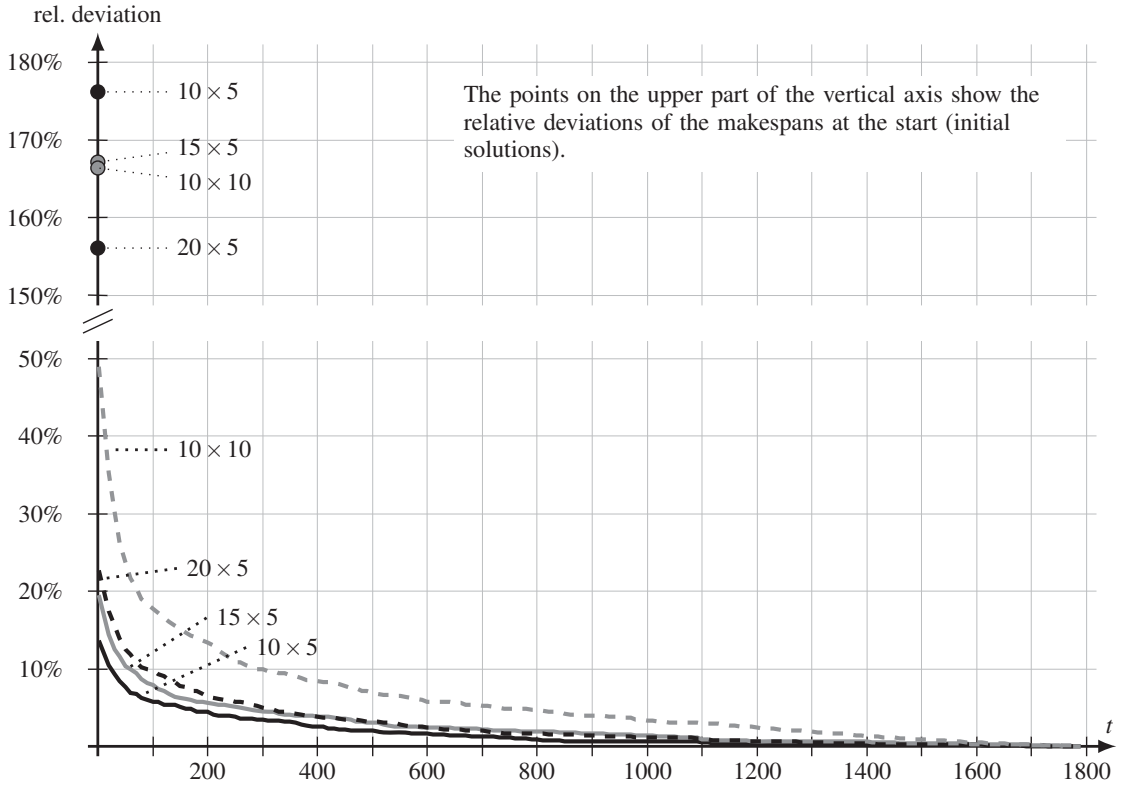

Fig. 7: Relative deviations of the makespan from the final makespan during run time.

size	*2 robots*	*3 robots*	*4 robots*
10×5	-23.4%	-11.1%	-4.0%
15×5	-20.4%	-10.4%	-4.6%
20×5	-18.8%	-10.8%	-2.9%
10×10	-30.7%	-13.4%	-4.3%

size	1 robot	*2 robots*	*3 robots*	*4 robots*
10×5	883'036	248'778	197'289	141'109
15×5	729'273	131'593	106'908	72'005
20×5	466'302	74'963	60'093	43'113
10×10	515'292	53'927	41'466	32'683

Table 4: (Left) Relative changes in the makespan when adding a robot. (Right) Number of tabu search iterations per run.

effort for generating a neighbor of type (19) is larger than for a neighbor of type (18) (about twice as large in our implementation).

The computational results are concluded with Figure 8 which displays a schedule with makespan 1221 for instance la05-V with 3 robots. Thick bars represent take-over, processing and hand-over steps while narrow bars represent waiting times. The numbers refer to the operations, e.g. "2.3" refers to the third operation of job 2. Transport operations are not shown, but can be inferred from the trajectories of the robots. Thick line sections indicate that the robot is loaded with a job, executing a transport operation, while thin sections correspond to idle moves.

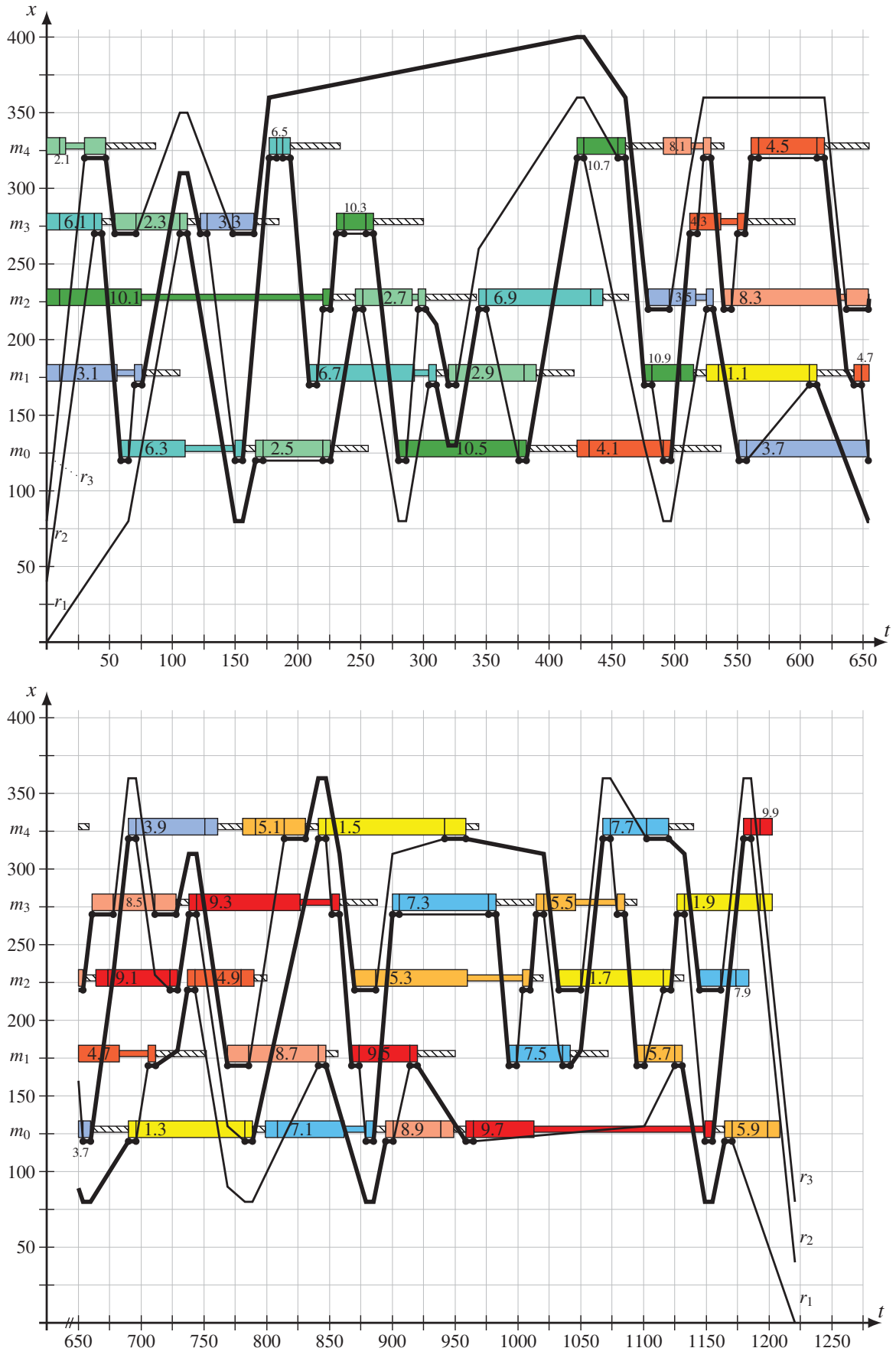

Fig. 8: A schedule with makespan 1221 of instance la05-V with 3 robots.

6 Concluding remarks

The disjunctive graph formulation of the BJS-RT involves a family of disjunctive arc pairs that, due to the collision avoidance constraints, is not only substantially larger, but also of a more complex structure than in scheduling problems where disjunctive pairs are between operations on a same machine. Nevertheless, the insertion theory (Gröflin and Klinkert, 2007), which has been applied to the flexible blocking job shop (Gröflin et al, 2011) and the no-wait job shop (Bürgy and Gröflin, 2013), proved also valuable in devising a local search for the BJS-RT.

The approach taken for the derivation of the disjunctive graph formulation of the BJS-RT can be used in other cases by changing the relaxed scheduling problem - e.g. substituting the flexible blocking job shop by the flexible job shop if sufficient buffer space is available in the system - or changing the characteristics of the transportation system and accordingly, the corresponding feasible trajectory problem. The second change raises interesting opportunities from both a research and application perspective.

Acknowledgements We thank an anonymous referee for her or his lucid and constructive remarks which led to several improvements in the exposition of the paper.

7 Appendix

7.1 Finding Trajectories

Finding feasible trajectories with minimum total travel distance is the problem of minimizing $\sum_{k=1}^{K}\sum_{p=1}^{Q-1}|x_{k,p+1}-x_{kp}|$ subject to constraints (7) to (10). It can be expressed by the linear program *LP Trajectory*

$$\text{Minimize } \sum_{k=1}^{K}\sum_{p=1}^{Q-1}(y_{kp}^{+}+y_{kp}^{-})$$

subject to

$$x_{k,p+1}-x_{kp}\le(t_{p+1}-t_p)v_k \text{ for all } k=1,\dots,K,\ p=1,\dots,Q-1,$$
$$x_{kp}-x_{k,p+1}\le(t_{p+1}-t_p)v_k \text{ for all } k=1,\dots,K,\ p=1,\dots,Q-1,$$
$$x_{kp}-x_{*}\le a_p \text{ for all } k=1,\dots,K,\ o\in O_k \text{ and } p\in\mathscr{P}[\alpha(o),\alpha(o)+d(o)],$$
$$x_{*}-x_{kp}\le -a_p \text{ for all } k=1,\dots,K,\ o\in O_k \text{ and } p\in\mathscr{P}[\alpha(o),\alpha(o)+d(o)],$$
$$x_{kp}-x_{k+1,p}\le-\delta \text{ for all } k=1,\dots,K-1,\ p=1,\dots,Q,$$
$$x_{*}-x_{1p}\le 0 \text{ for all } p=1,\dots,Q,$$
$$x_{Kp}-x_{*}\le L \text{ for all } p=1,\dots,Q,$$
$$x_{*}=0,$$
$$x_{k,p+1}-x_{kp}-y_{kp}^{+}\le 0 \text{ for all } k=1,\dots,K,\ p=1,\dots,Q-1,$$
$$x_{kp}-x_{k,p+1}-y_{kp}^{-}\le 0 \text{ for all } k=1,\dots,K,\ p=1,\dots,Q-1,$$
$$y_{kp}^{+}\ge 0,\ y_{kp}^{-}\ge 0 \text{ for all } k=1,\dots,K,\ p=1,\dots,Q-1.$$

The above linear program is the dual of a min-cost circulation problem and can be solved by a min-cost flow algorithm. However, optimal trajectories can be determined more efficiently with the following algorithm based on geometric arguments.

Assume that the FTP at (π,α) has a feasible solution, and consider $\mathscr{Q}=\{t_1,\ldots,t_Q\}$ (see Section 3.1). At each t_p, $p=1,\ldots,Q$, a given robot k is either required to be at a fixed location, say a_p^k, or there is no such requirement. For $p=1,\ldots,Q$, let $\{a_p^{k(1)},\ldots,a_p^{k(q_p)}\}$ be the set of fixed locations at t_p over all robots (it is non-empty as at least one robot is at a fixed location at t_p), and for $k=1,\ldots,K$, let $\{a^k_{p(1)},\ldots,a^k_{p(q^k)}\}$ be the set of all fixed locations of k over all times.

Given k and t_p, $p\in\{1,\ldots,Q\}$, the following *lower and upper bounds* l_p^k and u_p^k hold for the location $x(k,t_p)$ at t_p:

$$l_p^k=\max\{(k-1)\delta;a_p^{k(q)}+(k-k(q))\delta : k(q)\le k,\ 1\le q\le q_p\}$$
$$u_p^k=\min\{L-(K-k)\delta;a_p^{k(q)}-(k(q)-k)\delta : k(q)\ge k,\ 1\le q\le q_p\}$$

Note that if k is at a fixed location a_p^k, $l_p^k=a_p^k=u_p^k$.

It is helpful to consider trajectories in the two-dimensional time-location space with horizontal axis t and vertical axis x. In this space, a point will be denoted by $P=(t(P),x(P))$, $t(P)$, $x(P)$ denoting the t- and x-coordinate of P. For any two points P,P' where $t(P)<t(P')$, $[P,P']$ denotes the (line) segment joining P to P'. Its slope $(x(P')-x(P))/(t(P')-t(P))$ is denoted $\theta[P,P']$. A point P^* is *above* (*below*) the segment $[P,P']$ if there is $P''\in[P,P']$ with $t(P'')=t(P^*)$ and $x(P'')<x(P^*)$, $(x(P^*)<x(P''))$. For any k, let F_q^k, $q=1,\ldots,q^k$, be the *fixed points*, and L_p^k and U_p^k, $p=1,\ldots,Q$, the *lower* and *upper points* for the trajectory of k, i.e. $t(F_q^k)=t_{p(q)}$, $x(F_q^k)=a^k_{p(q)}$, $t(L_p^k)=t(U_p^k)=t_p$, $x(L_p^k)=l_p^k$ and $x(U_p^k)=u_p^k$. The following algorithm constructs for each robot k a piecewise linear trajectory $\mathscr{T}^k$.

Algorithm Trajectory
for $k=1,\ldots,K$ **do** *Trajectory*$(k,\mathscr{T}^k)$ **end**

Subroutine *Trajectory*$(k,\mathscr{T}^k)$
$\mathscr{T}:=\cup\{[F_q^k,F_{q+1}^k],1\le q\le q^k-1\}$. No segment of $\mathscr{T}$ is scanned.
while not all segments of $\mathscr{T}$ are scanned **do**
  choose an unscanned $[P,P']\in\mathscr{T}$, and scan $[P,P']$ as follows:
  if there exists some L_p^k above $[P,P']$ **then do**
    determine $\theta^*=\max\{\theta[P,L_p^k] : L_p^k \text{ above } [P,P']\}$ and
    $L^k_{p^*}$ such that p^* is the largest p with $\theta[P,L_p^k]=\theta^*$.
    $\mathscr{T}:=\mathscr{T}\cup\{[P,L^k_{p^*}]\cup[L^k_{p^*},P']\}-[P,P']$.
  else
    if there exists some U_p^k below $[P,P']$ **then do**
    determine $\theta^*=\min\{\theta[P,U_p^k] : U_p^k \text{ below } [P,P']\}$ and
    $U^k_{p^*}$ such that p^* is the largest p with $\theta[P,U_p^k]=\theta^*$.
    $\mathscr{T}:=\mathscr{T}\cup\{[P,U^k_{p^*}]\cup[U^k_{p^*},P']\}-[P,P']$.

end if
end while
$\mathscr{T}^k := \mathscr{T}$.

Theorem 2 *The trajectories $\mathscr{T}^k$, $k = 1,\dots,K$, are feasible and each $\mathscr{T}^k$ is a minimum travel distance trajectory for k.*

Proof For any consecutive segments $[P,P'],[P',P''] \in \mathscr{T}^k$, call $\mathscr{T}^k$ *concave at* P' if P' is above $[P,P'']$ and *convex at* P' if P' is below $[P,P'']$. We first show that at any points that are not fixed points of k, $\mathscr{T}^k$ is concave at a lower point and convex at an upper point.

Indeed, suppose L' is a lower point of $\mathscr{T}^k$ (and not a fixed point). L' became part of the trajectory $\mathscr{T}$ in the subroutine *Trajectory*$(k,\mathscr{T}^k)$ as a lower point $L' = L^k_{p*}$ above a segment $[P,P']$ previously in $\mathscr{T}$, with the property that any L^k_p with $t(P) \le t(L^k_p) < t(L^k_{p*})$ is not above $[P,L^k_{p^*}]$, and any L^k_p with $t(L^k_{p*}) < t(L^k_p) \le t(P')$ is below the line containing $[P,L^k_{p^*}]$. As a result, from then on and until completion of the subroutine, for any t with $t(P) \le t < t(L')$, $\mathscr{T}$ is not above this line, $\mathscr{T}$ is on the line at $t(L')$, and at any t with $t(L') < t \le t(P')$, $\mathscr{T}$ is below the line. Hence $\mathscr{T}^k$ is concave at L'. Similarly, one shows that $\mathscr{T}^k$ is convex at any upper point that is not fixed.

Examining the constraints (3) to (6), we show now the feasibility of the trajectories $\mathscr{T}^k$, $k = 1,\dots,K$.

Suppose (3) does not hold. Then, letting $[P,P'] \in \mathscr{T}^k$ be a steepest segment of $\mathscr{T}^k$ (with maximum $|\theta[P,P']|$),

$$|\theta[P,P']| = |x(P') - x(P)|/(t(P') - t(P)) > v_k. \tag{20}$$

Assume $\theta[P,P'] > 0$. P cannot be lower and not fixed since $\mathscr{T}^k$ is concave at such a point and P' cannot be upper and not fixed since $\mathscr{T}^k$ is convex at such a point. Hence P is upper or fixed and P' is lower or fixed, and $P = (t_p, u^k_p)$ and $P' = (t_{p'}, l^k_{p'})$ for some t_p , $t_{p'}$, $1 \le p < p' \le Q$. By (20), $l^k_{p'} - u^k_p > (t_{p'} - t_p)v_k$, contradicting the feasibility of the FTP, since $l^k_{p'} \le x(k,t_{p'})$, $x(k,t_p) \le u^k_p$ and $x(k,t_{p'}) - x(k,t_p) \le (t_{p'} - t_p)v_k$ hold for any feasible trajectory for k. The case $\theta[P,P'] < 0$ is similar.

Constraints (4) hold since for any transfer step of k with starting and completion time, say, $t_{p'}$ and $t_{p''}$, at any t_p with $p' \le p \le p''$, the trajectory of k is fixed (at the location of the transfer step). Constraints (5) also hold. Indeed, for any k, $1 \le k < K$, let trajectories $\mathscr{T}^k$ and $\mathscr{T}^{k+1}$ be at some t at a minimal (x-) distance from each other. By the concavity and convexity properties described above, we may assume that at time t, $\mathscr{T}^k$ is at a lower or fixed point, or $\mathscr{T}^{k+1}$ at an upper or fixed point. In the first case, for p such that $t_p = t$, $\mathscr{T}^k$ is at point $L^k_p = (t_p, l^k_p)$. By definition of the lower bounds and feasibility of the FTP, $l^{k+1}_p - l^k_p \ge \delta$, and by construction of $\mathscr{T}^{k+1}$, l^{k+1}_p is below or on $\mathscr{T}^{k+1}$, hence $\mathscr{T}^k$ and $\mathscr{T}^{k+1}$ are at least at an (x-) distance δ from each other. The second case is similar. Finally, (6) obviously holds.

To show that each $\mathscr{T}^k$ is a trajectory of minimum travel distance, it is enough to observe that at any step of the subroutine *Trajectory*$(k,\mathscr{T}^k)$, the total distance traveled by $\mathscr{T}$ is a lower bound on the total travel distance of any feasible trajectory for k. □

Proposition 2 *The trajectory algorithm runs in time* $\mathscr{O}(K|I^R|^2)$.

Proof Lower and upper bounds l_p^k and u_p^k can be computed in $\mathscr{O}(KQ)$ if one takes into account that for any robot k and event p the closest fixed robot at p below (above) k determines the lower bound l_p^k (upper bound u_p^k).

Consider now the run time of the trajectory subroutine by estimating the number of scanned segments and the effort spent for one scan.

As any segment is scanned exactly once, we estimate the number of segments considered in a run. At the start, the number of segments is bounded by $Q-1$. In any scan of a segment, two new segments are added if some point L_{p*}^k or U_{p*}^k is found. However, any point P of robot k can be selected at most in one scan as L_{p*}^k or U_{p*}^k. Therefore, at most $2Q$ new segments can be added to trajectory $\mathscr{T}$, hence there are at most $3Q$ segments considered in a run. The time spend for scanning a segment $[P,P']$ is bounded by the number of events in $[P,P']$, hence the effort of a scan is $\mathscr{O}(Q)$ and the effort of the subroutine is then $\mathscr{O}(Q^2)$.

The subroutine is executed for all robots $k = 1,\ldots,K$, hence the overall effort is $\mathscr{O}(KQ^2)$, or $\mathscr{O}(K|I^R|^2)$, since $Q \leq 4|I^R|$. This effort is smaller than that of a generic min-cost flow algorithm. $KQ+1$ is the number of nodes in the network of *LP Trajectory*. Solving a min-cost flow problem in a network with v nodes and e arcs requires an effort of $\mathscr{O}(e\log e(e+v\log v))$ with currently best algorithms (Orlin, 1993; Vygen, 2002). □

We also observe that in the "classical" case where a job is assumed to have at most one machining operation on a given machine, this complexity can be related to the number m of machines and the number n of jobs. Then $|I^R| \leq (m-1)n$, and the trajectory algorithm runs in time $\mathscr{O}(Km^2n^2)$.

Remark 1 The trajectories $\mathscr{T}^k$, $k = 1,\ldots,K$, make use of the variable speed (up to its maximum v_k) of each robot k. If all v_k's are equal, it can be shown that the $\mathscr{T}^k$'s, $k = 1,\ldots,K$, can be modified such that each robot is moving at maximum speed or stands still (the travel distance remaining the same and the number of the switches stand-move being at most Q).

References

Aron I, Genç-Kaya L, Harjunkoski I, Hoda S, Hooker J (2010) Factory crane scheduling by dynamic programming. In: Wood RK, Dell RF (eds) Operations Research, Computing and Homeland Defense (ICS 2011 Proceedings), INFORMS, pp 93–107

Bilge U, Ulusoy G (1995) A time window approach to simultaneous scheduling of machines and material handling system in an FMS. Operations Research 43(6):1058–1070

Brizuela CA, Zhao Y, Sannomiya N (2001) No-wait and blocking job-shops: challenging problems for GA's. In: IEEE international conference on systems, man, and cybernetics, pp 2349–2354

Brucker P, Knust S (2011) Complex Scheduling, 2nd edn. Springer

Brucker P, Strotmann C (2002) Local search procedures for job-shop problems with identical transport robots. In: Eight International Workshop on Project Management and Scheduling, Valencia (Spain)

Brucker P, Thiele O (1996) A branch & bound method for the general-shop problem with sequence dependent setup-times. OR Spectrum 18(3):145–161

Brucker P, Heitmann S, Hurink J, Nieberg T (2006) Job-shop scheduling with limited capacity buffers. OR Spectrum 28(2):151–176

Brucker P, Burke EK, Groenemeyer S (2012) A branch and bound algorithm for the cyclic job-shop problem with transportation. Computers & Operations Research 39(12):3200–3214

Bürgy R, Gröflin H (2013) Optimal job insertion in the no-wait job shop. Journal of Combinatorial Optimization 26(2):345–371

Cook WJ, Cunningham WH, Pulleyblank WR, Schrijver A (1997) Combinatorial Optimization. Wiley-Interscience

Deroussi L, Gourgand M, Tchernev N (2008) A simple metaheuristic approach to the simultaneous scheduling of machines and automated guided vehicles. International Journal of Production Research 46(8):2143–2164

Gröflin H, Klinkert A (2007) Feasible insertions in job shop scheduling, short cycles and stable sets. European Journal Of Operational Research 177(2):763–785

Gröflin H, Klinkert A (2009) A new neighborhood and tabu search for the blocking job shop. Discrete Applied Mathematics 157(17):3643–3655

Gröflin H, Pham DN, Bürgy R (2011) The flexible blocking job shop with transfer and set-up times. Journal of Combinatorial Optimization 22(2):121–144

Hurink J, Knust S (2005) Tabu search algorithms for job-shop problems with a single transport robot. European Journal of Operational Research 162(1):99–111

Khayat GE, Langevin A, Riopel D (2006) Integrated production and material handling scheduling using mathematical programming and constraint programming. European Journal of Operational Research 175(3):1818–1832

Lacomme P, Larabi M, Tchernev N (2010) Job-shop based framework for simultaneous scheduling of machines and automated guided vehicles. International Journal of Production Economics

Lawrence S (1984) Supplement to resource constrained project scheduling: an experimental investigation of heuristic scheduling techniques. GSIA, Carnegie Mellon University, Pittsburgh, PA

Leung JMY, Zhang G (2003) Optimal cyclic scheduling for printed circuit board production lines with multiple hoists and general processing sequence. IEEE Transactions on Robotics and Automation 19(3):480–484

Leung JMY, Zhang G, Yang X, Mak R, Lam K (2004) Optimal cyclic multihoist scheduling: a mixed integer programming approach. Operations Research 52(6):965–976

Li W, Wu Y, Petering M, Goh M, de Souza R (2009) Discrete time model and algorithms for container yard crane scheduling. European Journal of Operational Research 198(1):165–172

Manier MA, Bloch C (2003) A classification for hoist scheduling problems. International Journal of Flexible Manufacturing Systems 15(1):37–55

Manier MA, Varnier C, Baptiste P (2000) Constraint-based model for the cyclic multi-hoists scheduling problem. Production Planning & Control 11(3):244–257

Mascis A, Pacciarelli D (2002) Job-shop scheduling with blocking and no-wait constraints. European Journal of Operational Research 143(3):498–517

Ng W (2005) Crane scheduling in container yards with inter-crane interference. European Journal of Operational Research 164(1):64–78

Nowicki E, Smutnicki C (1996) A fast taboo search algorithm for the job shop problem. Management Science 42(6):797–813

Orlin J (1993) A faster strongly polynomial minimum cost flow algorithm. Operations Research 41(2):338–350

Vygen J (2002) On dual minimum cost flow algorithms. Mathematical Methods of Operations Research 56(1):101–126