Scheduling of Multi-Robot Job Shop Systems in Dynamic Environments: Mixed-Integer Linear Programming and Constraint Programming Approaches

Soroush Fatemi-Anaraki, Reza Tavakkoli-Moghaddam, Mehdi Foumani, Behdin Vahedi-Nouri

PII: S0305-0483(22)00177-3
DOI: https://doi.org/10.1016/j.omega.2022.102770
Reference: OME 102770

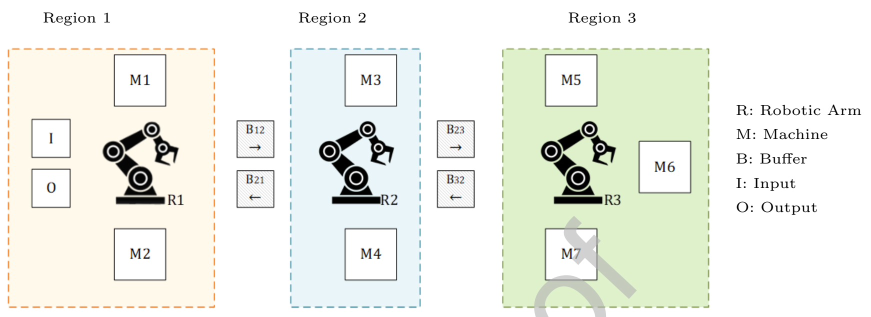
To appear in: *Omega*

Received date: 7 December 2020
Accepted date: 17 September 2022

Highlights

- Considering a dynamic scheduling problem in a U-shaped job shop robotic cell.
- Devising a mixed-integer linear programming model and three speed-up constraints.
- Proposing a novel Constraint Programming model for the considered problem.
- Analyzing the impact of critical parameters' values and model presumption.

Graphical abstract



Sample of the studied U-shaped robotic cell with multiple robotic arms

Scheduling of Multi-Robot Job Shop Systems in Dynamic Environments: Mixed-Integer Linear Programming and Constraint Programming Approaches

Soroush Fatemi-Anaraki[a,b], Reza Tavakkoli-Moghaddam[a], Mehdi Foumani[c,*], Behdin Vahedi-Nouri[a]

[a]*School of Industrial Engineering, College of Engineering, University of Tehran, Tehran, Iran*

[b]*TUM School of Management, Technical University of Munich, Arcisstr. 21, Munich 80333, Germany*

[c]*Rio Tinto Centre for Mine Automation, Australian Centre for Field Robotics, University of Sydney, NSW, Australia*

Abstract

This paper investigates a dynamic scheduling problem within a job shop robotic cell, wherein multiple robotic arms are responsible for material handling in a U-shaped arrangement. Each robotic arm has access to specific workstations based on their distance in the cell layout. Therefore, a part may need to be exchanged between several robots according to its process plan. For this purpose, intermediate buffers are positioned between each pair of consecutive robots. Due to the dynamic nature of the problem, new jobs arrive at unpredictable times, which in turn necessitates rescheduling taking the system's current state into account. To tackle this problem, firstly, a Mixed-Integer Linear Programming (MILP) model is devised. Secondly, three distinct Speed-up Constraints (SCs) derived from the problem's inherent characteristics are designed and implemented to accelerate the MILP model's solving procedure. Afterward, the problem is formulated using Constraint Programming (CP) approach. The performance of the CP model and the MILP model in presence of all possible combinations of the SCs are evaluated and compared through solving various random instances. Next, an analysis is performed on the buffers' pick-up criterion and how it is affected by the problem's size. Besides, the impact of changes in the robots' speed on the productivity of the cell is assessed. Finally, the extent to which the rescheduling priority affects the output of the model is studied.

keywords: Job shop robotic cell; Material handling; Dynamic scheduling; Mixed-integer linear programming; Constraint programming

1. Introduction

Material handling within manufacturing cells plays a pivotal role in their efficiency. Robotic cells incorporate robots to meet this requirement since they are safe, accurate, fast, and capable of repeatedly performing tasks without fatigue. A robotic cell comprises several workstations, robots to carry

[*]Corresponding author.

Email addresses: `soroush.fatemianaraki@tum.de` (Soroush Fatemi-Anaraki), `tavakoli@ut.ac.ir` (Reza Tavakkoli-Moghaddam), `mehdi.foumani@sydney.edu.au` (Mehdi Foumani), `b.vahedi@ut.ac.ir` (Behdin Vahedi-Nouri)

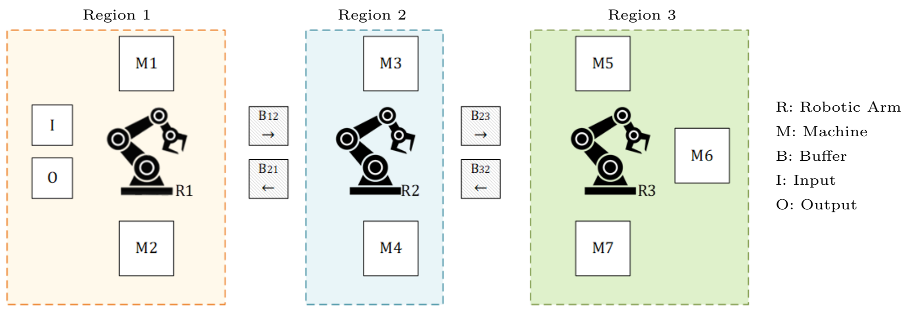


Figure 1: A sample of the studied U-shaped robotic cell with multiple robotic arms

out material handling in an automated manner, and an input/output station. It is undeniable that robotic cells enjoy widespread use in a myriad of industries, including semiconductor manufacturing, electroplating, glass manufacturing, and cosmetics (Dawande et al., 2007).

Robots are hence an inseparable component of smart factories (Watanabe and Inada, 2020), performing a variety of tasks, including but not limited to material handling, parts assembling, and quality inspection. Shifting towards such automations is remarkably capital-intensive (Bock and Bruhn, 2021). Furthermore, the extensive implementation of robots in manufacturing has experienced significant growth as 435,000 new industrial robots were incorporated in 2021, and 500,000 new units are expected to be installed worldwide per year until 2024 (IFR, 2021). This demonstrates the prominence of casting light on this area of research and problems arising from it. When it comes to utilizing robots for material handling in a manufacturing cell, a vital consideration is the sequencing and scheduling of their movements in accordance with workstations to improve cell productivity.

This research investigates a no-wait job shop scheduling problem in a U-shaped manufacturing cell. An instance of the problem at hand is depicted in Figure 1. The cell consists of several workstations, multiple robotic arms, intermediate buffers, and an input/output station. For each robotic arm, the area in which the robot executes the material handling process is called a *region*. In other words, each region comprises several machines that a robot can visit. Since each machine is visited solely by one robot, a part may require to be exchanged between consecutive robots multiple times based on the process plan to reach a specific machine. The process plan represents the sequence of operations each job (or part) should go through to become a finished job. Therefore, intermediate buffers are designed to pass parts from one robot to another. It should be noted that as the initial schedule is being executed, new jobs may arrive at unpredictable times. As a result, rescheduling based on the system's state at the moment when the new schedule is expected to initiate may also be required. To further understand the problem at hand, an *illustrative example* is devised based on Figure 1 and thoroughly explained in Section 3.

To tackle the problem, firstly, a Mixed-Integer Linear Programming (MILP) model is devised. Next, three Speed-up Constraints (SCs) are designed to facilitate solving the MILP model. Further, a Constraint Programming (CP) model is developed to provide a solution for large-size instances in a timely manner. The results obtained from the CP model are compared with that of the MILP model considering all combinations of the proposed SCs. Finally, a sensitivity analysis is performed to assess different pick-up criteria for buffers, the impact of robot speed on overall cell performance, and the changes in the model's outputs regarding the rescheduling importance.

The primary contributions of this research are as follows:

- This study is the first attempt to consider the dynamic scheduling problem of multiple robotic arms in a job shop manufacturing cell while restricting robots' access to specific workstations. The single-robot variant of the problem was considered by Yan et al. (2018).
- Three SCs are extracted based on the inherent characteristics of the problem and all combinations of their presence in the MILP model are evaluated using extensive numerical experiments to provide further insight into their performance.
- To provide quality solutions for large-scale instances promptly, a CP model is developed. The existing literature on using CP approach to solve different variants of the robotic cell scheduling problem is scarce, as the method is relatively new.
- A numerical analysis is carried out to compare two pick-up criteria, namely no-wait and free pick-up for intermediate buffers. Additionally, the impact of changes in robots' speed on cell productivity is assessed. The insight obtained from the mentioned analysis extends our understanding of resource management in the studied robotic cell. Finally, a sensitivity analysis is performed to better understand the rescheduling importance and the extent to which the results of the model are affected by this approach.

The remainder of this research is organized as follows. A comprehensive review of the literature is provided in Section 2. The studied problem is thoroughly elaborated in Section 3. The MILP model and the three SCs are presented in Section 4. Section 5 comprises the devised CP model for the proposed robotic cell scheduling problem. Numerical results are presented in Section 6. Section 7 provides a comprehensive sensitivity analysis regarding resources in the studied robotic cell and the rescheduling approach. The conclusion and future areas of study are presented in Section 8.

2. Literature Review

Robotic cells are composed of a wide variety of characteristics, altering from one cell to another. Distinction in the application, as well as innate dissimilarity among different industries, are the underlying driving forces behind the diversity in robotic cells. In this section, our focus is two-fold. First, several prominent aspects of robotic cells are selected and investigated to illuminate the background of the research at hand. Second, a comprehensive overview of CP and its applications in related problem settings is provided.

A significant characteristic of robotic cells is their behavior regarding the pick-up criterion. Three pick-up criteria exist in the literature: no-wait, free, and time window (Geismar et al., 2008). In a no-wait case, a part must be unloaded from its current position as soon as the machine ends its ongoing process. This is the case in industries where maintaining heat, viscosity, or freshness of the product is significant, such as in the manufacturing processes of plastics, steel, chemicals, pharmaceuticals, and the food industry (Hall and Sriskandarajah, 1996). This is also the case in printed circuit boards, electroplating, and chemical processes involving galvanization (Che et al., 2003). In contrast, the free pick-up criterion addresses a situation where parts are allowed to remain on a machine as long as needed (Geismar et al., 2008). Finally, the time window criterion specifies a bounded interval for the duration of time that a part is permitted to stay on each machine (Yan et al., 2018).

Two main research streams exist in the literature of robotic cell scheduling. Some studies assume a single robot to be responsible for material handling (Carlier et al., 2010; Kharbeche et al., 2011; Brucker et al., 2012; Che et al., 2017; Liu and Kozan, 2017; Gultekin et al., 2018; Zhou and Li, 2018). However, more than one robot can be incorporated into a cell. The vast majority of studies incorporating multiple robots consider the number of robots working in the cell as an input parameter (Elmi and Topaloglu, 2016; Nouri et al., 2016; Yang et al., 2016; Elmi and Topaloglu, 2017; Dang et al., 2019). The layout of the manufacturing cell, budget limitations, the type of robot, the number of workstations, and technology levels are all contributing factors that can restrict the number of robots used in a cell. In this study, the number of robots is established as an input parameter.

Another primary aspect of studies focusing on the robotic cell scheduling problem is whether they are cyclic or dynamic. In a cyclic situation, a similar pattern is repeated in a duration, called cycle time. Several studies considered this variant of the robotic cell scheduling problem (Che and Chu, 2009; Che et al., 2011; Jolai et al., 2012; Li et al., 2015; Gultekin et al., 2018). The cornerstone of cyclic studies is the existence of a steady-state for the production system. However, in reality, the system state can be subject to alterations, including machine breakdowns, unscheduled maintenance activities, and the arrival of new jobs to the system at unpredictable times. Consequently, the dynamic variant of the robotic cell scheduling problem has gained more prominence recently. The majority of dynamic studies focus on the arrival of new jobs at unpredictable times with two distinct approaches for tackling the problem: complete rescheduling, or preserving the current schedule while planning new jobs in empty intervals of the current schedule. Considering studies adopting a complete rescheduling approach to cope with the problem's dynamic nature, Yan et al. (2014) considered a job shop robotic cell with a single robot. In their research, only one job could arrive at the cell each time. They proposed a metric to measure the deviation of the new schedule. Feng et al. (2015) then developed a mathematical model deeming the current state of workstations and jobs while rescheduling. da Silva et al. (2019) further examined several methods for inserting arriving jobs into the current schedule and compared them with rescheduling through a mathematical model, concluding that the latter produces more favorable results. Zhao et al. (2013) adopted a rescheduling approach for a dynamic hoist scheduling problem and devised a mechanism for calculating the time when the new schedule should initiate. Another attitude in this regard is placing arriving jobs into empty time intervals of robots and workstations as opposed to complete rescheduling (Yan et al., 2018).

Compared to the single-robot variant of the problem, utilizing multiple robots remarkably increases the complexity in two ways: the exchange of parts between robots in case they directly collaborate and the likely collision of robots. To the best of our knowledge, the majority of studies considering multiple robots assume that all robots have access to every workstation. Thus, robots do not exchange parts. Some studies also refer to moving resources wherein workers processing jobs can walk along the line (Hashemi-Petroodi et al., 2022). Such production systems enjoy considerable flexibility. However, this presumption does not hold in many real-world cases, e.g. when using robotic arms, since not only do they have physical limitations, but also inevitable collision prohibits them from interfering with each other in the working area. In fact, few studies, Che et al. (2012) being one, cast light on the collision of robots in a multi-robot case or assign a subset of workstations to each robot (see Shabtay et al. (2014)).

Two methods have been suggested in the literature for exchanging parts between robots: shared stages and buffers. Shared stages are working stages to which more than one robot has access (Geismar et al., 2004). Geismar et al. (2008) studied a cyclic flow shop scheduling problem in a U-shaped robotic cell and demonstrated that there is a slight difference between the aforementioned approaches in terms of their impact on the cell's throughput. Considering studies using buffers, Kim et al. (2017) studied cyclic scheduling of sequentially connected cluster tools for wafer production. A buffer with a capacity of two was placed between adjacent cluster tools. One unit of the buffer's capacity was allocated to parts moving forward, while the other unit was assigned to parts moving backward. Yi et al. (2005) devised a decomposition method for turning a multi-cluster into several single-cluster tools. They also suggested the use of buffers with a capacity of two for connecting cluster tools. Chan et al. (2008) studied both buffers with a capacity of one and two for connecting cluster tools.

Search space reduction strategies, including constraint propagation, have been vastly explored for solving the job shop scheduling problem. Constraint propagation concentrates on removing solutions that cannot be optimal from the search space through applying various consistency tests iteratively (Dorndorf et al., 2000, 2002). Constraint programming also prioritizes finding a feasible solution through satisfying constraints. This method incorporates constraint propagation along with other techniques to quickly find a feasible solution and improve it over a number of iterations afterwards (Ham and Park, 2021). Furthermore, CP is an exact method that is capable of solving an optimization problem to optimality (Laborie et al., 2018).

It is worth mentioning that CP has several superiorities compared to other exact and metaheuristic methods—CP solvers do not require a problem to be either linear or convex (Russell, 2013), the number of variables and constraints used in CP is less than other exact methods (Meng et al., 2020), and metaheuristic methods require parameter tuning while CP does not (Ham and Cakici, 2016). Furthermore, CP is capable of solving large-scale problems (Gedik et al., 2018).

To the best of our knowledge, the literature on utilizing the CP method for solving robotic cell scheduling problems is scarce. In this regard, Booth et al. (2016a) studied a single-robot task allocation and scheduling problem. They devised MILP and CP models and demonstrated that both methods had merits for the devised problem. Booth et al. (2016b) designed both MILP and CP models to tackle the task allocation and scheduling problem within a nursing home, in which multiple robots were incorporated to provide the residents with healthcare services. The CP model outweighed the MILP model performance-wise. Behrens et al. (2019) applied CP to solve the task allocation and motion scheduling problem considering dual-arm robots in an assembly cell. Murín and Rudová (2019) also researched the job shop scheduling problem, considering multiple robots without any blocking condition. They focused on a situation in which robots could process parts on machines, as well as perform the material handling process. Ham (2021) studied a job shop scheduling problem without blocking, considering multiple robots that had access to all workstations. Ham (2020) considered a flexible job shop cell, in which multiple robots had access to all workstations. Furthermore, all workstations had input and output buffers that can be translated to a non-blocking condition.

It should be noted that the CP method has recently been implemented for solving distinct variants of the job shop scheduling problem (Ham and Cakici, 2016; Lunardi et al., 2020; Meng et al., 2020), parallel machine scheduling with sequence-dependent setups (Heinz et al., 2022), vehicle routing problem, timetabling, and assembly line balancing problem (Bukchin and Raviv, 2018). Moreover, recent studies have focused on hybrid and decomposition methods that combine CP with MILP (Enayaty-Ahangar et al., 2019; Polyakovskiy and M'Hallah, 2021; Vahedi-Nouri et al., 2022), or CP with a metaheuristic (Gecili and Parikh, 2022).

From the literature review above, it can be summarized that the prominence of exchanging parts among multiple robots, using intermediate buffers, while their access is restricted to specific workstations necessitates further research. This is especially the case when multiple robotic arms are in charge of the material handling process. The job shop variant of the above problem is missing from the literature; hence, the aim of the research at hand is to shed light on this area. Besides, dynamicity is the essence of non-cyclic studies in the field because many real-world applications, specifically those that concentrate on changes in customers' orders, aim to consider unpredictable incidents (i.e., the arrival of new jobs to the system). Therefore, the dynamic variant of the problem is studied in this research. Considering the solution methodology, the literature on the incorporation of the CP approach in the robotic cell scheduling problem has a great potential for extension. Specifically, dynamic robotic cell scheduling problems are missing. Two major presumptions of this paper are the restriction of robots' access to the workstations and their direct collaboration through buffers. Such complications give rise to new modeling challenges, and contribute to the CP literature.

3. Problem Description

As depicted in Figure 1, a U-shaped robotic cell consisting of several workstations is considered in this research. Robotic arms placed in the middle of the cell are responsible for the material handling process. The number of robots is an input parameter, determined based on the layout of the cell, number of workstations, available budget, and the robots' physical limitations in terms of movement. Each robotic arm has access to a predefined set of workstations within its region. Whenever a part needs to be moved from one workstation to another, the robotic arm serving the region where the corresponding workstation is located unloads the part from its current machine, transports the part to the next machine, and loads the part. For brevity of writing, a movement performed by a robot includes the steps mentioned above for all jobs. Furthermore, a no-wait situation is considered, which in turn necessitates a part to be unloaded from its current machine as soon as its processing is over.

Because robotic arms cannot reach every workstation, they require other means to exchange parts between each other. For this purpose, between each pair of consecutive robots, a buffer with a capacity of two is installed. For each buffer, one unit of its capacity is allocated to parts moving deeper into the cell, while the other unit of capacity is allotted to parts moving backward. For a part to leave the cell, it is required to move backward using intermediate buffers. This is because there is only one input/output station installed in the U-shaped manufacturing cell. The no-wait situation may or may not apply to intermediate buffers (both conditions are covered in this research). Apart from buffers placed between consecutive robots, no intermediate buffers are considered before or after workstations. Consequently, the blocking condition applies.

A job shop type cell is considered in this research. Hence, each part has a particular process plan that must include every buffer to be visited whenever a part moves from one region to another. Thus, the process plan for each part has to be updated based on the region of workstations and the layout of the cell. The following example better illustrates this matter, as well as how robots depicted in Figure 1 work in tandem to serve a part based on its process plan.

Illustrative example. Based on the manufacturing cell shown in Figure 1, a part may have the following process plan:

$$\{M_1, M_5, M_6, M_4, M_2\}$$

The first two processing steps, namely M_1 and M_5, are located in two distinct regions, 1 and 3, respectively. Therefore, when robot 1 (R_1) unloads the part from M_1, it should move the part to buffer B_{12} so that the next robot, R_2, can reach the part. At this point, R_2, can unload the part from B_{12} and load it on the next buffer B_{23}, so that R_3 which serves region 3 can have access to the part. Afterward, R_3 can pick up the part from B_{23} and load it on M_5. As it is witnessed, a part may not visit the workstations situated inside a region while passing it. In other words, a part may need to move forward several times, passing from one buffer to another consecutively. Finally, the process plan of the part is updated to the form:

$$\{I, M_1, B_{12}, B_{23}, M_5, M_6, B_{32}, M_4, B_{21}, M_2, O\}$$

Regarding the dynamic nature of the problem, it is presumed that several jobs may arrive at unforeseeable times while the cell operates based on its ongoing schedule. Hence, when new jobs arrive at the system, it should continue the ongoing operations for the sake of efficiency while a rescheduled plan is being prepared based on the system state at the very moment that the new schedule is expected to initiate. For this purpose, the upper bound of the time needed for acquiring the new schedule should be estimated in advance to be used for specifying the system state at the starting moment of the new schedule.

It is important to notice that rescheduling is likely to alter the previously planned completion time of parts that were scheduled beforehand. Drastic alterations in this regard may be problematic from different aspects; for example, a scheduled shipment for a part might require to be postponed based on the new schedule. Consequently, a penalty is needed to control the gap between the previous completion times of jobs and their new completion times. Finally, the newly developed schedule can initiate while taking all jobs, considering both the old and the newly arrived ones.

When new jobs arrive in the system, calculating the exact time the new schedule is supposed to begin, T_s, is of great significance since it is the basis for obtaining the system state at the rescheduling moment, which is an input parameter of the problem. Rendering a new schedule is a time-consuming process. Due to the problem's complexity, the maximum time the program is allowed to run should be decided in advance at a managerial level - the shorter the run time, the more agile the system becomes. However, shorter run time leads to a worse objective value (i.e., larger makespan) and more deviation in completion times of previous jobs. Hence, a trade-off is to be made regarding the run time between the solution quality and the agility of the system response in real-time.

The run time of the program is a known parameter called T_{cal}. Moreover, T_{Arr} is assumed to be the arrival time of new jobs. Yan et al. (2018) proposed a method for obtaining the time that a new schedule should begin for a dynamic robotic cell with one robotic arm. Their approach can be roughly extended for multiple robotic arms. Two scenarios are deemed as probable:

- If all robots are idle (either awaiting at workstations or moving empty) at $T_{cal} + T_{Arr}$, then the new schedule can take place at $T_s = T_{cal} + T_{Arr}$.
- If some robots are holding parts at $T_{cal} + T_{Arr}$, then they should all complete their movements. The time point where the last movement ends is the starting point of the new schedule.

At T_s, some parts might require to be picked up immediately. Therefore, all robots must be prepared at associated workstations based on the new schedule. To foresee this scenario, the most prolonged duration of a robot movement between two spots can be added to T_s. Since robotic arms are highly agile, the duration mentioned above would be negligible.

4. Mathematical Formulation

The job shop *robotic* cell scheduling problem is in the class of the job shop scheduling problem (i.e., scheduling operations on machines/buffers) with some additional constraints to ensure the feasibility of the robotic arm's movements in each region. The classical job shop scheduling has received considerable attention in the literature, e.g., (Ku and Beck, 2016) compared the performance of various models for the job shop scheduling problem using modern solvers. We refer interested readers to Xiong et al. (2022) for a comprehensive review on the job shop scheduling problem, and its applications in manufacturing, supply chain, and mine optimization. In this section, first, we provide the sets, indices, parameters, and the decision variables used for the mathematical formulation. Second, the MILP model which is formulated for the problem described in Section 3 is presented. Finally, the three designed SCs are described.

Sets and indices:

$\mathcal{J}$	Set of all jobs to be processed, $j, j' \in \mathcal{J}$
$\mathcal{J}_{in}$	Set of jobs being processed in the cell at the moment the new schedule initiates, $\mathcal{J}_{in} \subset \mathcal{J}$
$\mathcal{J}_{old}$	Set of jobs from the previous schedule (set of all jobs excluding the newly-arrived) ones, $\mathcal{J}_{old} \subset \mathcal{J}$
$\mathcal{M}$	Set of all stations in the cell including workstations, intermediate buffers, and input/output station, $m \in \mathcal{M}$
Buf_{set}	Set of all buffers in the cell, $Buf_{set} \subset \mathcal{M}$
Buf'_{set}	Set of all stations, except for the buffers, $Buf'_{set} \subset \mathcal{M}$
Reg	Set of all service regions, $Reg = Reg_1, Reg_2, ...$
$\mathcal{K}$	Ordered set of all regions and intermediate buffers connecting them, $\mathcal{K} = Reg_1, Buf_{1,2}, Reg_2, ..., k \in \mathcal{K}, Reg \subset \mathcal{K}$
$\mathcal{O}$	Set of all operations, $i, i', i'', i''' \in \mathcal{O}$
$\mathcal{O}_j$	Set of job j's operations, $\mathcal{O} = \bigcup_{j=1}^{\lvert\mathcal{J}\rvert} \mathcal{O}_j$
i, i', i'', i'''	Operations' indices to be performed on each working part based on its corresponding process plan

Parameters:

OC_j	Number of job j's operations, $OC_j = \lvert\mathcal{O}_j\rvert$
E_j	Time elapsed for the ongoing operation of job j when the new schedule begins
PCT_j	Time old jobs were supposed to be completed based on the initial schedule
$p_{i,j}$	Processing time of operation i of job j on $n_{i,j}$
$n_{i,j}$	Machine responsible for performing operation i of job j
$d_{i,j}$	Duration of time a robot requires to move job j from $n_{i,j}$ to $n_{i+1,j}$
$e_{i,j,i',j'}$	Duration of time a robot requires to move from $n_{i,j}$ to $n_{i',j'}$ without holding any job
$reg_{i,j}$	Region wherein $n_{i,j}$ is located
γ	Coefficient for controlling the schedule deviation penalty incurred on the objective function
T_s	Estimated time the new schedule starts
ε	Minimum time required for a part to stay on a buffer before a robot can unload it
M	A large positive value

Decision variables:

$S_{i,j}$	Time when a robot starts unloading job j from $n_{i,j}$ to transfer it to $n_{i+1,j}$
$R_{i,j}$	Time when job j is placed on $n_{i,j}$, for $n_{i,j} \in Buf_{set}$
CTD^+_j	Positive deviation of completion time of $j \in \mathcal{J}_{old}$ in the new schedule

CTD^-_j	Negative deviation of completion time of $j \in \mathcal{J}_{old}$ in the new schedule
$Y_{i,j,i',j'}$	Equals 1 if job j is unloaded from $n_{i,j}$ before j' is unloaded from $n_{i',j'}$; 0, otherwise
C_{max}	Maximum completion time of all jobs in the cell (makespan)

4.1. MILP Model

The devised MILP model is as follows:

$$minimize \quad Z = C_{max} + \gamma \sum_{j \in \mathcal{J}_{old}} (CTD^+_j + CTD^-_j) \tag{1}$$

subject to

$$C_{max} \geqslant S_{OC_j,j} \qquad \forall j \in \mathcal{J} \tag{2}$$

$$S_{OC_j,j} - (PCT_j - T_s) = CTD^+_j - CTD^-_j \qquad \forall j \in \mathcal{J}_{old} \tag{3}$$

The model's objective function, displayed by Equation 1, is minimizing the maximum completion time of all jobs while controlling the gap between the new and previous completion times of jobs scheduled beforehand. Constraints 2 set the C_{max} variable to be greater than the completion time of the last operation of all jobs. Constraints 3 calculate the deviation of completion time of jobs previously scheduled.

$$S_{i,j} - S_{i-1,j} - d_{i-1,j} = p_{i,j} \qquad \forall j \in \mathcal{J}, 1 \leqslant i \leqslant OC_j, n_{i,j} \in Buf'_{set} \tag{4}$$

$$R_{i,j} - S_{i-1,j} - d_{i-1,j} = 0 \qquad \forall j \in \mathcal{J}, 1 \leqslant i \leqslant OC_j, n_{i,j} \in Buf_{set} \tag{5}$$

$$S_{i,j} \geqslant R_{i,j} \qquad \forall j \in \mathcal{J}, 1 \leqslant i \leqslant OC_j, n_{i,j} \in Buf_{set} \tag{6}$$

Constraints 4 state that job j cannot be unloaded from $n_{i,j}$, unless it has already been moved to $n_{i,j}$ and processing duration $p_{i,j}$ has elapsed. They hence represent the processing time of jobs on workstations. Constraints 5 demonstrate that the time when a job arrives at a buffer is equal to the departure time from the previous machine plus the moving time. Constraints 6 state that as soon as a job is placed on $n_{i,j} \in Buf_{set}$, it can be unloaded by the next robot. It should be noted that Constraints 6 can be altered in two ways. Firstly, if the no-wait condition is needed to hold for buffers as well as workstations, then the inequality form should turn into equality. Hence, the constraint can be rewritten as $S_{i,j} = R_{i,j}$. Secondly, there may exist a condition that parts need to stay a minimum amount of time for the sake of physical stability before the next robot can grab them. Under such circumstances, the inequality should be updated to $S_{i,j} - R_{i,j} \geqslant \varepsilon$.

$$S_{i,j} + E_j = p_{i,j} \qquad \forall j \in \mathcal{J}_{in}, i = 1, n_{i,j} \in Buf'_{set} \tag{7}$$

$$R_{i,j} = 0 \qquad \forall j \in \mathcal{J}_{in}, i = 1, n_{i,j} \in Buf_{set} \tag{8}$$

Considering the current state of the system when the new schedule is expected to begin is of paramount significance. For jobs already undergoing distinct processes in the cell, at the moment when the new schedule begins, E_j represents the time that has elapsed from the time their current process on $n_{i,j}$ has begun. Constraints 7 control the processing time of ongoing operations when the new schedule initiates. Constraints 8 ensure that parts are placed on buffers at the moment when the new scheduled initiates are ready to be picked up.

$$S_{i'-1,j'} \geqslant S_{i,j} + d_{i,j} + e_{i+1,j,i'-1,j'} - (1 - Y_{i,j,i',j'})M \qquad \begin{array}{l} \forall j, j' \in \mathcal{J}, 1 < i < OC_j, 1 < i' < OC_{j'} \\ j < j', n_{i,j}, n_{i',j'} \in Buf'_{set}, n_{i,j} = n_{i',j'} \end{array} \tag{9}$$

$$S_{i-1,j} \geqslant S_{i',j'} + d_{i',j'} + e_{i'+1,j',i-1,j} - Y_{i,j,i',j'}M \qquad \begin{array}{l} \forall j, j' \in \mathcal{J}, 1 < i < OC_j, 1 < i' < OC_{j'} \\ j < j', n_{i,j}, n_{i',j'} \in Buf'_{set}, n_{i,j} = n_{i',j'} \end{array} \tag{10}$$

Constraints 9 (and Constraints 10 likewise) show that part j' cannot be unloaded to be moved to $n_{i',j'} \in Buf'_{set}$, unless part j, which is currently loaded on $n_{i,j} = n_{i',j'}$ is moved to its next destination, and the robot serving the region has reached for j'. Since only one robot can serve each region, the constraint holds for workstations within the region. Hence, Constraints 9 and 10 demonstrate that solely one job can be processed by each workstation in any region at any time.

$$S_{i'-1,j'} + d_{i'-1,j'} \geqslant S_{i,j} - (1 - Y_{i,j,i',j'})M \quad \forall j, j' \in \mathcal{J}, 1 \leqslant i \leqslant OC_j, 1 \leqslant i' \leqslant OC_{j'} \quad j < j', n_{i,j}, n_{i',j'} \in Buf_{set}, n_{i,j} = n_{i',j'} \tag{11}$$

$$S_{i-1,j} + d_{i-1,j} \geqslant S_{i',j'} - Y_{i,j,i',j'}M \quad \forall j, j' \in \mathcal{J}, 1 \leqslant i \leqslant OC_j, 1 \leqslant i' \leqslant OC_{j'} \quad j < j', n_{i,j}, n_{i',j'} \in Buf_{set}, n_{i,j} = n_{i',j'} \tag{12}$$

Constraints 11 and 12 guarantee that for each movement direction, only one job can be placed on any buffer at any time. A part cannot arrive at a buffer unless the part previously occupying the buffer is already unloaded by its corresponding robot. It is worth mentioning that since two robots serve consecutive regions, the capacity constraints for workstations (Constraints 9 and 10) differ from that of buffers (Constraints 11 and 12).

$$S_{i',j'} \geqslant S_{i,j} + d_{i,j} + e_{i+1,j,i',j'} - (1 - Y_{i,j,i',j'})M \tag{13}$$

$$\begin{aligned}
&[\forall j, j' \in \mathcal{J}, j < j', i \leqslant OC_j, i' \leqslant OC_{j'}]\wedge \\
&[(reg_{i+1,j} = reg_{i',j'}, reg_{i,j} \notin Reg, reg_{i',j'} \in Reg)\vee \\
&(reg_{i,j} = reg_{i'+1,j'}, reg_{i',j'} \notin Reg, reg_{i,j} \in Reg)\vee \\
&(reg_{i+1,j} = reg_{i'+1,j'}, [(reg_{i',j'} \wedge reg_{i,j}) \notin Reg])\vee \\
&(reg_{i,j} = reg_{i',j'}, reg_{i,j} \in Reg)]
\end{aligned} \tag{13a}$$

$$\begin{aligned}
&[\forall j, j' \in \mathcal{J}, j < j', i \leqslant OC_j, i' \leqslant OC_{j'}, \forall k \in \mathcal{K}, k \neq Reg_1]\wedge \\
&[(reg_{i,j} = k-1, reg_{i+1,j} = k+1, (reg_{i',j'} \vee reg_{i'+1,j'}) = k, reg_{i,j}, reg_{i+1,j} \notin Reg)\vee \\
&(reg_{i,j} = k+1, reg_{i+1,j} = k-1, (reg_{i',j'} \vee reg_{i'+1,j'}) = k, reg_{i,j}, reg_{i+1,j} \notin Reg)\vee \\
&(reg_{i',j'} = k-1, reg_{i'+1,j'} = k+1, (reg_{i,j} \vee reg_{i+1,j}) = k, reg_{i',j'}, reg_{i'+1,j'} \notin Reg)\vee \\
&(reg_{i',j'} = k+1, reg_{i'+1,j'} = k-1, (reg_{i,j} \vee reg_{i+1,j}) = k, reg_{i',j'}, reg_{i'+1,j'} \notin Reg)\vee \\
&(reg_{i,j} = k-1, reg_{i+1,j} = k+1, reg_{i',j'} = k-1, reg_{i'+1,j'} = k+1, reg_{i,j}, reg_{i',j'} \notin Reg)\vee \\
&(reg_{i,j} = k-1, reg_{i+1,j} = k+1, reg_{i',j'} = k+1, reg_{i'+1,j'} = k-1, reg_{i,j}, reg_{i',j'} \notin Reg)\vee \\
&(reg_{i,j} = k+1, reg_{i+1,j} = k-1, reg_{i',j'} = k-1, reg_{i'+1,j'} = k+1, reg_{i,j}, reg_{i',j'} \notin Reg)\vee \\
&(reg_{i,j} = k+1, reg_{i+1,j} = k-1, reg_{i',j'} = k+1, reg_{i'+1,j'} = k-1, reg_{i,j}, reg_{i',j'} \notin Reg)]
\end{aligned} \tag{13b}$$

$$S_{i,j} \geqslant S_{i',j'} + d_{i',j'} + e_{i'+1,j',i,j} - Y_{i,j,i',j'}M \tag{14}$$

$$\begin{aligned}
&[\forall j, j' \in \mathcal{J}, j < j', i \leqslant OC_j, i' \leqslant OC_{j'}]\wedge \\
&[(reg_{i+1,j} = reg_{i',j'}, reg_{i,j} \notin Reg, reg_{i',j'} \in Reg)\vee \\
&(reg_{i,j} = reg_{i'+1,j'}, reg_{i',j'} \notin Reg, reg_{i,j} \in Reg)\vee \\
&(reg_{i+1,j} = reg_{i'+1,j'}, [(reg_{i',j'} \wedge reg_{i,j}) \notin Reg])\vee \\
&(reg_{i,j} = reg_{i',j'}, reg_{i,j} \in Reg)]
\end{aligned} \tag{14a}$$

$$\begin{aligned}
&[\forall j, j' \in \mathcal{J}, j < j', i \leqslant OC_j, i' \leqslant OC_{j'}, \forall k \in \mathcal{K}, k \neq Reg_1]\wedge \\
&[(reg_{i,j} = k-1, reg_{i+1,j} = k+1, (reg_{i',j'} \vee reg_{i'+1,j'}) = k, reg_{i,j}, reg_{i+1,j} \notin Reg)\vee \\
&(reg_{i,j} = k+1, reg_{i+1,j} = k-1, (reg_{i',j'} \vee reg_{i'+1,j'}) = k, reg_{i,j}, reg_{i+1,j} \notin Reg)\vee \\
&(reg_{i',j'} = k-1, reg_{i'+1,j'} = k+1, (reg_{i,j} \vee reg_{i+1,j}) = k, reg_{i',j'}, reg_{i'+1,j'} \notin Reg)\vee \\
&(reg_{i',j'} = k+1, reg_{i'+1,j'} = k-1, (reg_{i,j} \vee reg_{i+1,j}) = k, reg_{i',j'}, reg_{i'+1,j'} \notin Reg)\vee \\
&(reg_{i,j} = k-1, reg_{i+1,j} = k+1, reg_{i',j'} = k-1, reg_{i'+1,j'} = k+1, reg_{i,j}, reg_{i',j'} \notin Reg)\vee \\
&(reg_{i,j} = k-1, reg_{i+1,j} = k+1, reg_{i',j'} = k+1, reg_{i'+1,j'} = k-1, reg_{i,j}, reg_{i',j'} \notin Reg)\vee \\
&(reg_{i,j} = k+1, reg_{i+1,j} = k-1, reg_{i',j'} = k-1, reg_{i'+1,j'} = k+1, reg_{i,j}, reg_{i',j'} \notin Reg)\vee \\
&(reg_{i,j} = k+1, reg_{i+1,j} = k-1, reg_{i',j'} = k+1, reg_{i'+1,j'} = k-1, reg_{i,j}, reg_{i',j'} \notin Reg)]
\end{aligned} \tag{14b}$$

Constraints 13 and 14 represent the robot capacity, meaning that each robot can hold only one part at a time. Only if a robot unloads its current part and spends adequate time to get to the workstation where the other part is awaiting, can it pick up the new part. Several convoluted conditions may occur in the cell to which the abovementioned constraints apply. Conditions denoted as 13a and 14a refer to three situations:

1. Part j (or j') is located on a buffer, before or after a region, and is intending to visit a machine within the region while the other part j' (or j) is to be unloaded from a machine within the region.
2. Both parts, j and j' are located on buffers outside the region and intending to visit a machine within the region.
3. Both parts j and j' are to be unloaded from machines located in the robot's service region.

Also, conditions denoted as 13b and 14b refer to two distinct situations:

1. Part j (or j') is to be moved by a robot from one buffer to the next buffer (either in a forward or backward direction) while the other part j' (or j) is either to be unloaded from a machine located in the service region or from a buffer to get inside the region.
2. Both parts, j and j' are to be moved consecutively from one buffer to another buffer using the same robot (either in a forward or backward direction).

$$Y_{i,j,i',j'} = 1 \qquad \forall j \in \mathcal{J}_{in}, \forall j' \in \mathcal{J}, i = 1, 1 < i' \leqslant OC_{j'}, n_{i,j} = n_{i',j'}, j \neq j' \tag{15}$$

Constraints 15 ensure that when the new schedule begins. If a machine or a buffer is processing a part, it finishes its current operation before accepting any new parts. In other words, if any part is to be moved to $n_{i,j}$, the part $j \in \mathcal{J}_{in}$, which is currently loaded on the machine should be unloaded beforehand. Constraints 16-21 represent the decision variables of the model.

$$Y_{i,j,i',j'} \in \{0,1\} \qquad \forall j, j' \in \mathcal{J}, 1 \leqslant i \leqslant OC_j, 1 \leqslant i' \leqslant OC_{j'}, j < j' \tag{16}$$

$$S_{i,j} \geqslant 0 \qquad \forall j \in \mathcal{J}, 0 \leqslant i \leqslant OC_j \tag{17}$$

$$R_{i,j} \geqslant 0 \qquad \forall j \in \mathcal{J}, 0 \leqslant i \leqslant OC_j \tag{18}$$

$$CTD_j^+ \geqslant 0 \qquad \forall j \in \mathcal{J}_{in} \tag{19}$$

$$CTD_j^- \geqslant 0 \qquad \forall j \in \mathcal{J}_{in} \tag{20}$$

$$C_{max} \geqslant 0 \tag{21}$$

4.2. Speed-up Constraints

The only binary variable of the MILP model is $Y_{i,j,i',j'}$. The MILP model is also formulated in a way where there is no relationship among $Y_{i,j,i',j'}$ variables for different sets of indices. On the other hand, when different $Y_{i,j,i',j'}$ variables are assigned a value of 1, they impose strict restrictions on the value of some other $Y_{i,j,i',j'}$. This idea is the cornerstone of the three devised SCs. New constraints are therefore added to the MILP model to prune combinations of $Y_{i,j,i',j'}$, the attainment of which is logically and operationally infeasible. While this course of action may compromise the efficiency of the search space, as new constraints are added to the MILP model, it may also contribute to improving the system's tolerance for the complexity of solving the problem. Therefore, a thorough numerical experiment should be carried out on all combinations of these constraints to scrutinize their performance when added to the model (see Section 6). It should be noted that, devising numerical experiments to assess the quality of such valid inequalities is a well-accepted approach (Lanza et al., 2022; Delorme and Santini, 2022).

4.2.1. SC1

SC1 is as follows:

$$(1 - Y_{i,j,i',j'}) + Y_{i'',j,i''',j'} \geqslant 1 \qquad [\forall j, j' \in \mathcal{J}, j < j', n_{i,j} = n_{i',j'}, i \leqslant OC_j, i''' \leqslant OC_{j'}, i'' \leqslant i, i''' \geqslant i'] \wedge (i \neq i'' \vee i' \neq i''') \tag{22}$$

As Constraints 22 show, if parts j and j' share the same workstation for operations i and i' (respectively), then in the instance where part j is unloaded before part j', all operations of part j before i should also take place before all operations of j' after i'. Figure 2 provides a visual depiction of how SC1 works.

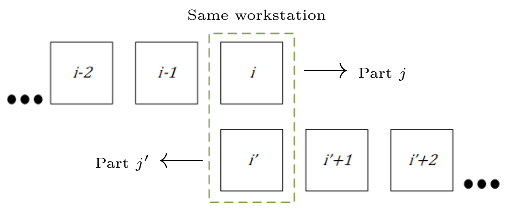


Figure 2: Visual justification of SC1

4.2.2. SC2

SC2 is as follows:

$$(1 - Y_{i-1,j,i'-1,j'}) + Y_{i,j,i',j'} \geqslant 1 \qquad \forall j, j' \in \mathcal{J}, j < j', n_{i,j} = n_{i',j'}, i < OC_j, i' < OC_{j'} \tag{23}$$

As Constraints 23 show, if parts j and j' share the same workstation for operations i and i' respectively ($n_{i,j} = n_{i',j'}$), then in case part j is to be unloaded before part j', it should also be unloaded before j', when operation $i-1$ finishes. Otherwise, part j' is loaded on $n_{i',j'}$ before part j which contradicts our initial assumption. Figure 3 provides a visual depiction of how SC2 works.

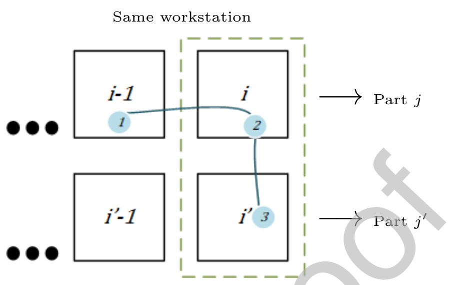


Figure 3: Visual justification of SC2

4.2.3. SC3

SC3 is as follows:

$$(1 - Y_{i-1,j,i'-1,j'}) + Y_{i,j,i'-1,j'} \geqslant 1 \qquad \forall j, j' \in \mathcal{J}, j < j', n_{i,j} = n_{i',j'}, i < OC_j, i' < OC_{j'} \tag{24}$$

As Constraints 24 show, if parts j and j' share the same workstation for operations i and i' respectively ($n_{i,j} = n_{i',j'}$), then in case part j is to be unloaded before part j', it should also be unloaded before j' regarding operations $i - 1$ and $i' - 1$. Otherwise, part j' is loaded on $n_{i',j'}$ before part j, which contradicts our initial assumption. Figure 4 provides a visual depiction of how SC3 works.

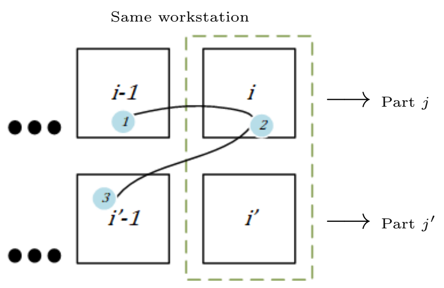


Figure 4: Visual justification of SC3

It should be noted that Brucker et al. (2012) used the notion behind SC2 and SC3 to specify feasible solutions within a tree search method. Figure 5 aggregates the logic behind SC2 and SC3 in a more comprehensive visual representation.

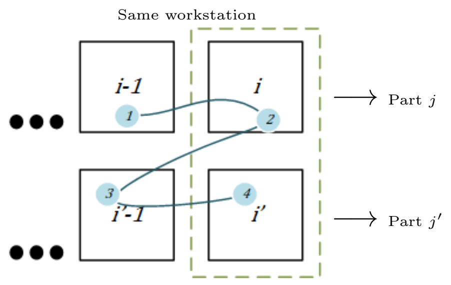


Figure 5: Visual justification of SC2 and SC3 aggregated

5. Proposed CP Model

5.1. Background

Before proceeding with the proposed CP Model, let us initially discuss the complexity of the problem studied in Section 3. Assume a special case of the problem where the intermediate buffers have unlimited capacity and the robots can perform both, parts transportation and empty movements in zero time ($d_{i,j} = 0$, $e_{i',j',i,j} = 0$). Under above circumstances, the robot and the buffer capacity constraints will become obsolete. The workstation capacity constraint still holds; however, in this case, it is more fathomable to interpret $S_{i,j}$ variables as the time when the processing of job j ends on $n_{i,j}$. Using such simplified presumptions, any solution approach that is proposed for this problem can solve the classical job shop scheduling problem as well. Therefore, the classical job shop scheduling problem can be reduced to the problem at hand. Since the job shop scheduling problem is $\mathcal{NP}$-hard (Applegate and Cook, 1991), the problem considered in this research is also $\mathcal{NP}$-hard.

The inclusion of parts transportation using a multi-robot handling system, along with the dynamic nature of the problem, clearly result in a complex problem setting. Therefore, providing a solution approach capable of yielding feasible and quality solutions in T_{cal} is instrumental. Choice of solution approach is critical in such a scenario as it should prioritize finding a quality feasible solution quickly over solving the scheduling problem to optimality in a longer run time. As elaborated in Section 2, CP is an exact method that prioritizes finding quality feasible solutions. Therefore, it is opted as the solution approach to tackle the research problem.

In order to implement the CP method for modeling the problem proposed in Section 3, the *DOcplex.cp* module from *IBM ILOG Optimization Studio* is added as a library to *Python* programming language. A CP model consists of variables, functions, and constraints. Two main variable types exist in the CP method, namely interval and sequence variables.

- An interval variable represents any operation with some duration, starting, and ending time, i.e., each operation of a job based on its process plan.
- Sequence variables represent different permutations for a set of interval variables.

For a thorough description of different types of CP variables, functions, constraints, and CP's modeling logic, visit IBM (2020).

Two types of robot movements exist in the problem studied by this paper. A robot may be either holding a part or empty while moving. For the first case, it is considered that each movement of a part between the corresponding pair of workstations, is an operation that is performed by a robot. In other words, robots are thought of as workstations in this scenario. Such operations should be added to the process plan of a part. In this regard, set $\mathcal{M}$ should be updated to include all robots, since they are treated as workstations in the developed CP model.

$$\mathcal{M} = \{Buf_{set} \cup Buf'_{set} \cup Robots\}$$

Illustrative example (continued). Based on the cell depicted in Figure 1, if a part has the following process plan,

$$\{I, M_1, B_{12}, B_{23}, M_5, M_6, B_{32}, M_4, B_{21}, M_2, O\},$$

then the following updated process plan should be provided as an input to the CP model:

$$\{I, R_1, M_1, R_1, B_{12}, R_2, B_{23}, R_3, M_5, R_3, M_6, R_3, B_{32}, R_2, M_4, R_2, B_{21}, R_1, M_2, R_1, O\}$$

The duration of robot operations, which are represented by the updated process plan, is equal to $d_{i,j}$. For the empty movement situation, two modeling approaches are possible. While both approaches are valid, their performances vary conspicuously considering the problem at hand.

One such approach is to use a built-in CP function called *transition_matrix*(). The output of *transition_matrix*() is an input argument of the *no_overlap*() method for all robots, which in turn, imposes a minimum waiting time for the situation when a robot decides to make an empty movement from one station to another. The waiting time is equal to the distance between two specific locations. The alternative approach which is strongly advised for the problem at hand is using the *if_then*() function to devise a CP constraint for this purpose. The constraint restricts the staring time of a part's movement if an empty movement by the corresponding robot is required to take place beforehand. This approach is superior in terms of the memory usage and quality of the outcome for the problem studied in this paper. In the rest of this section, CP variables and built-in functions used for modeling are presented, and then the CP formulation is provided. Defined variables and built-in CP functions in our model are as follows:

CP variables:

$job_{o[j][i]}$	Interval variable representing each operation of job j based on its process plan
$machine_{o[m]}$	Sequence variable demonstrating different permutations of all operations to be performed by each robot, workstation, and buffer

CP built-in functions:

set_end()	Sets the ending time of an interval variable
set_start()	Sets the starting time of an interval variable
get_end()	Gets the ending time of an interval variable
get_start()	Gets the starting time of an interval variable
end_at_start()	Sets the ending time of an interval variable equal to the starting time of another interval variable

$transition\_matrix()$	Constructs a matrix to store the duration of movements for a shared resource when moving from one station to another
$no\_overlap()$	Prohibits a set of interval variables from overlapping
$if\_then()$	Constructs if/then rule
$minimize()$	Minimizes input
$max()$	Maximizes input
$abs()$	Absolute function
$sum()$	Sum function

The CP model is designed to be equivalent to the MILP model. Hence, the dynamicity of the problem proposed in Section 3 should also be considered. To do so, the $job_{o[j][i]}$ interval variables for $\forall j \in \mathcal{J}_{in}$ are defined, such that value 0 is assigned to their starting times. Therefore, whether $job_{o[j][i]}$ stands for an ongoing operation on a buffer, or a workstation, it should start at time point zero. Owing to the modeling logic behind the CP method, the above condition can implicitly model dynamicity, and no further course of action is necessary. The CP model is as follows:

$$end\_at\_start(job_{o[j][i]}, job_{o[j][i+1]}) \qquad \forall j \in \mathcal{J}, 1 \leqslant i \leqslant OC_j \tag{25}$$

$$job_{o[j][i]}.set\_end(job_{o[j][i+1]}.get\_start()) \qquad \forall j \in \mathcal{J}, i \leqslant OC_j, n_{i,j} \in Buf_{set} \tag{26}$$

$$job_{o[j][i]}.set\_start(job_{o[j][i-1]}.get\_end()) \qquad \forall j \in \mathcal{J}, i \leqslant OC_j, n_{i,j} \in Buf_{set} \tag{27}$$

$$machine_{o[m]} = sequence\_var(job_{o[j][i]}) \qquad \forall j \in \mathcal{J}, i \leqslant OC_j, n_{i,j} = m \tag{28}$$

$$no\_overlap(machine_{o[m]}) \qquad \forall m \in \mathcal{M} \tag{29}$$

$$\begin{aligned} &if\_then(\\ &job_{o[j][i-1]}.get\_end() \leqslant job_{o[j'][i'-1]}.get\_end(), && \forall j,j' \in \mathcal{J}, i \leqslant OC_j, i' \leqslant OC_{j'}, n_{i,j} = n_{i',j'}, \\ &job_{o[j][i]}.get\_end() \leqslant job_{o[j'][i']}.get\_start() - e_{i+1,j,i'-1,j'}) && n_{i,j} \notin \{Buf_{set} \cup Buf'_{set}\} \end{aligned} \tag{30}$$

$$\begin{aligned} &if\_then(\\ &job_{o[j][i]}.get\_end() \leqslant job_{o[j'][i']}.get\_start(), \\ &job_{o[j][i+1]}.get\_end() \leqslant job_{o[j'][i'-1]}.get\_start()) && \forall j,j' \in \mathcal{J}, i \leqslant OC_j, i' \leqslant OC_{j'}, n_{i+1,j} = n_{i'-1,j'} \end{aligned} \tag{31}$$

$$\begin{aligned} &minimize[\max(job_{o[j][OC_j]} \ for\ all\ j \in \mathcal{J}) \\ &+ \gamma\ sum[abs(job_{o[j][i]}.get\_end() + T_s - PCT_j)\ for\ all\ j \in \mathcal{J}_{in}]] \end{aligned} \tag{32}$$

Equations 25 demonstrate that each operation of part j should take place as soon as their previous operation ends, based on the updated process plan. Therefore, all operations, including robot movements of part j, are sequentially sorted by considering the no-wait condition. It should be noted that $job_{o[j][i]}$ interval variables can either accept a duration as a parameter or let it be determined by the model. Except for $job_{o[j][i]}$ associated with buffer placements for any part j, operations are given durations based on associated $p_{i,j}$ and $d_{i,j}$ for operations related to workstations and robots' movements, respectively.

Equations 26 and 27 determine the length of $job_{o[j][i]}$ for operations regarding buffer stages. Equations 26 state that buffer placement ends as soon as the corresponding robot unloads part j from the buffer. Similarly, Equations 27 ensure that the buffer-placement interval variable starts as soon as the corresponding robot places part j on the buffer. Equations 26 and 27 represent the free pick-up criterion for buffers. However, if the no-wait condition is to be applied to buffers, the above expressions should be removed from the model. Furthermore, the size of $job_{o[j][i]}$ interval variables related to buffers should be equal to zero. Equations 28 construct the sequence variable for all workstations, robots, and buffers. Equations 29 forbid overlapping in performed operations, while Equations 30 considers a situation in which a robot should move empty from one workstation to another one to unload a part. Equations 31 create an if-then rule to control the sequence of robot movements. Finally, Equation 32 displays the objective function of the problem.

6. Numerical Experiment

In this section, the performance of the MILP model is tested in the presence of all combinations of the SCs against that of the CP model. Next, in order to further understand the SCs' impact on the MILP model's search space, an analogy is drawn based on the average number of nodes explored by each method. The performance of the CP model is analyzed during the run time to further emphasize the effect of elapsed time on the quality of the solution. To run the MILP and CP models in each experiment, CPLEX and DOcplex.cp libraries are imported in the Python 3 programming language, respectively. All scripts are executed on a system with AMD Ryzen 7 PRO 4750U processor and 32GBs of RAM.

Based on the cell depicted in Figure 1, 30 large-scale instances are randomly generated. The processing times of operations are random integers uniformly distributed in the range of 1 to 10. The process plan for each job is randomly chosen such that for each pair of consecutive processing steps, required visits to buffers are added to the process plan based on the cell layout. The number of jobs and the number of assigned operations to each job are decided in advance. The duration of time for the robot to move between workstations and buffers is calculated based on the cell layout. Mentioned values are in the range of 1 to 2. Two rescheduling weights (γ=0.1 and 0.3) are examined for each instance. To diversify problem instances, the number of jobs ($|\mathcal{J}|$), the total number of operations ($|\mathcal{O}|$), the number of new jobs arriving into the system due to the dynamic nature of the problem (Arr), and the value of γ differ for distinct instances. Each instance is represented as a 4-tuple ($|\mathcal{J}|, |\mathcal{O}|, Arr, \gamma$). Based on the rescheduling strategy adopted in this research, instances are all solved in a fixed run time, T_{cal} (which is considered to be 120s). It should be noted that, T_s is set to 0 for all instances, to make the results comparable for different run times.

The power set of SCs described in Section 4.2 is as follows:

$$\{\emptyset, \{SC1\}, \{SC2\}, \{SC3\}, \{SC1, SC2\}, \{SC1, SC3\}, \{SC2, SC3\}, \{SC1, SC2, SC3\}\}$$

As displayed in Table 1, 60 numerical instances of large size are solved in T_{cal}, using the CP model, the MILP model, and the MILP model in the presence of all SC combinations. Since the instances in Table 1 cannot reach proven optimality, a lower bound should provide a baseline for comparison. For this purpose, a Relaxation of the Problem (RP) is considered which can be solved to proven optimality (see Appendix A). Note that the optimal solution of the RP is a valid lower bound for the original problem. We label this lower bound as the Lower Bound obtained from the Relaxed Problem (LB-RP). In order to calculate LB-RPs for all instances presented in this paper, the relaxed versions of the CP model, as proposed in Appendix A, are solved to optimality. Interested readers are referred to Appendix B to see the gap of optimal solution with LB-RP for medium-size instances (which are solved to optimality). The gap of the optimal solution obtained by each method with the LB-RP is calculated using Equation 33, where Z^*_{method} and $Z^*_{\text{LB-RP}}$ correspond to the objective value

Table 1: Comparison of the solution gap with the LB-RP in T_{cal}, among the results of the CP model and the MILP model in presence of all combinations of the SCs

No.	$(\lvert\mathcal{J}\rvert, \lvert\mathcal{O}\rvert, Arr, \gamma)$	Gap with LB-RP (%)									LB-RP	
		CP	MILP	SC1	SC2	SC3	SC(1,2)	SC(1,3)	SC(2,3)	SC(1,2,3)	Solution	Time(s)
1	(15, 127, 0, 0.1)	**47.10**	52.33	53.14	49.07	52.33	57.29	53.14	49.69	49.38	82.00	3.40
2	(16, 137, 1, 0.1)	**37.54**	51.48	51.89	51.30	51.87	50.57	52.01	54.00	50.57	131.10	885.43
3	(17, 145, 2, 0.1)	**43.12**	52.41	52.96	50.95	52.82	47.81	54.38	51.73	49.94	131.10	607.77
4	(18, 149, 3, 0.1)	**44.92**	52.91	54.22	53.26	54.29	54.00	53.11	56.80	54.70	131.10	356.25
5	(19, 157, 4, 0.1)	**51.01**	56.34	56.50	58.50	57.95	53.84	58.46	56.33	55.74	131.50	300.55
6	(20, 163, 5, 0.1)	**48.50**	57.29	57.01	56.63	55.93	56.36	57.86	56.04	54.96	131.80	335.60
7	(21, 171, 6, 0.1)	**53.95**	54.78	61.03	64.64	64.50	60.42	65.29	61.33	61.81	132.80	194.74
8	(22, 178, 7, 0.1)	**58.55**	59.31	66.66	76.62	60.72	60.29	66.30	60.04	60.76	133.30	126.11
9	(23, 187, 8, 0.1)	**54.73**	63.55	69.34	62.89	64.20	66.43	78.37	67.60	70.50	135.50	218.46
10	(24, 193, 9, 0.1)	60.96	**58.83**	68.98	62.67	65.58	67.90	65.52	77.34	63.82	137.50	919.93
11	(15, 135, 0, 0.1)	**54.17**	56.74	59.69	56.25	57.22	56.50	59.26	54.71	58.15	77.00	789.25
12	(16, 144, 1, 0.1)	39.89	**37.10**	51.93	44.02	47.34	43.84	58.02	47.20	47.78	138.50	4930.24
13	(17, 154, 2, 0.1)	**40.99**	51.74	56.17	55.92	51.99	55.39	47.85	51.57	47.22	138.50	3103.77
14	(18, 160, 3, 0.1)	**42.96**	68.27	68.52	61.52	68.60	63.14	63.19	67.84	59.60	138.50	2866.01
15	(19, 168, 4, 0.1)	**52.50**	63.24	68.08	67.04	65.13	62.92	65.67	70.23	61.31	138.50	2251.08
16	(20, 175, 5, 0.1)	**50.14**	72.18	68.95	68.44	70.05	64.04	63.70	69.14	65.95	138.50	1358.27
17	(21, 182, 6, 0.1)	**57.04**	68.30	66.23	68.18	69.80	65.31	64.96	69.08	67.72	138.50	848.01
18	(22, 189, 7, 0.1)	**52.38**	71.56	73.72	71.83	74.17	68.31	69.84	66.75	69.77	139.20	937.52
19	(23, 193, 8, 0.1)	**56.65**	69.69	69.41	74.51	72.60	67.68	72.29	72.06	68.03	140.20	260.09
20	(24, 203, 9, 0.1)	**60.24**	74.21	72.90	73.50	76.41	66.67	71.82	73.99	69.31	141.40	1740.64
21	(15, 148, 0, 0.1)	**53.09**	59.01	58.26	54.73	61.28	56.67	60.94	60.26	59.01	91.00	2186.15
22	(16, 159, 1, 0.1)	**25.58**	54.65	49.84	51.66	54.84	50.19	57.28	53.90	54.01	172.20	4627.76
23	(17, 170, 2, 0.1)	**39.94**	60.13	55.23	57.11	57.20	53.14	55.72	55.81	55.70	172.20	4239.10
24	(18, 179, 3, 0.1)	**48.66**	56.14	58.46	56.27	59.44	57.88	60.24	61.26	57.57	172.20	3457.70
25	(19, 187, 4, 0.1)	**50.01**	58.15	59.23	54.64	57.79	58.79	58.61	55.61	57.75	172.20	1905.40
26	(20, 197, 5, 0.1)	**49.90**	66.00	61.42	61.42	60.50	59.97	62.44	68.43	61.79	172.20	1988.42
27	(21, 207, 6, 0.1)	**54.76**	65.77	65.12	68.41	68.14	59.83	67.45	64.94	61.42	172.50	1470.29
28	(22, 213, 7, 0.1)	**54.47**	69.18	68.78	67.97	63.80	65.03	68.58	69.36	66.61	172.70	898.95
29	(23, 223, 8, 0.1)	**55.36**	71.34	71.07	73.77	67.38	66.06	70.31	73.12	69.95	172.90	905.64
30	(24, 234, 9, 0.1)	**62.42**	83.68	78.55	73.91	64.37	69.23	78.67	71.12	67.51	173.30	1153.00
31	(15, 127, 0, 0.3)	**47.10**	52.33	53.14	49.07	52.33	57.29	53.14	49.69	49.38	82.00	2.22
32	(16, 137, 1, 0.3)	50.85	62.49	**30.45**	33.73	30.55	56.90	63.45	64.42	61.87	150.30	31.95
33	(17, 145, 2, 0.3)	50.85	65.39	63.95	66.35	62.61	61.64	63.84	60.54	**47.85**	150.30	30.60
34	(18, 149, 3, 0.3)	**54.39**	68.12	66.64	68.92	66.16	56.23	65.93	65.22	62.46	150.30	39.28
35	(19, 157, 4, 0.3)	**59.99**	68.29	65.62	65.63	68.36	65.62	65.23	69.66	64.50	150.30	26.85
36	(20, 163, 5, 0.3)	**60.52**	69.84	66.14	68.60	71.63	66.44	62.76	65.55	66.78	150.30	30.93
37	(21, 171, 6, 0.3)	63.88	63.94	72.06	69.90	**63.42**	68.48	67.64	77.75	70.08	150.30	38.26
38	(22, 178, 7, 0.3)	67.61	75.63	67.18	**54.09**	68.47	72.76	69.99	70.64	70.33	150.30	35.23
39	(23, 187, 8, 0.3)	**70.58**	74.65	71.06	78.01	81.36	72.75	73.47	74.13	74.29	150.30	42.56
40	(24, 193, 9, 0.3)	70.80	**67.65**	74.61	73.87	80.62	72.50	73.56	79.50	72.14	150.30	50.28
41	(15, 135, 0, 0.3)	**54.17**	56.74	59.69	56.25	57.22	56.50	59.26	54.71	58.15	77.00	789.25
42	(16, 144, 1, 0.3)	39.21	49.81	48.14	20.34	**11.17**	56.49	49.58	45.71	31.40	160.60	24.81
43	(17, 154, 2, 0.3)	53.29	39.44	50.80	48.89	59.43	56.52	71.74	**36.52**	54.95	160.60	30.10
44	(18, 160, 3, 0.3)	54.67	76.57	74.19	48.01	**43.57**	73.11	76.90	46.41	71.15	160.60	26.55
45	(19, 168, 4, 0.3)	62.24	76.51	77.40	62.63	63.55	76.13	76.79	**44.91**	73.15	160.60	32.06
46	(20, 175, 5, 0.3)	**62.31**	77.33	75.02	75.91	77.76	74.92	79.16	78.31	74.06	160.60	23.21
47	(21, 182, 6, 0.3)	**58.24**	78.64	76.32	80.55	78.68	75.52	76.36	76.15	80.19	160.60	37.03
48	(22, 189, 7, 0.3)	**65.61**	81.56	76.20	80.02	77.05	78.94	75.64	81.53	73.13	160.60	32.33
49	(23, 193, 8, 0.3)	**67.24**	81.99	81.07	81.07	79.49	78.36	77.61	78.07	77.86	160.60	31.86
50	(24, 203, 9, 0.3)	**72.07**	82.62	79.11	81.29	82.27	77.50	80.50	80.62	78.15	160.60	45.74
51	(15, 148, 0, 0.3)	**53.09**	59.01	58.26	54.73	61.28	56.67	60.94	60.26	59.01	91.00	2186.15
52	(16, 159, 1, 0.3)	42.52	20.58	66.51	**12.10**	13.84	62.73	32.53	46.26	20.80	196.80	43.97
53	(17, 170, 2, 0.3)	38.96	68.10	65.68	**31.24**	67.60	68.38	66.57	68.13	63.74	196.80	34.39
54	(18, 179, 3, 0.3)	57.02	70.21	68.19	68.94	55.31	69.36	69.45	**45.77**	69.04	196.80	43.11
55	(19, 187, 4, 0.3)	**59.52**	69.21	71.16	69.17	68.79	69.01	70.96	71.63	67.24	196.80	56.56
56	(20, 197, 5, 0.3)	**63.36**	67.31	68.02	66.17	73.33	70.09	70.57	64.99	72.13	196.80	61.05
57	(21, 207, 6, 0.3)	**67.27**	74.19	71.42	68.69	71.20	69.14	71.95	71.35	67.72	196.80	67.92
58	(22, 213, 7, 0.3)	**67.83**	72.64	72.80	76.23	74.71	71.01	77.00	78.90	72.94	196.80	85.93
59	(23, 223, 8, 0.3)	75.79	78.16	77.85	76.94	76.37	**72.57**	74.46	78.76	74.99	196.80	73.40
60	(24, 234, 9, 0.3)	77.61	85.40	81.92	79.24	85.38	**74.51**	81.18	82.22	76.96	196.80	65.26
Avg. Gap with LB-RP		**54.84**	64.51	65.06	61.90	62.86	63.73	65.66	63.93	62.58		

of the method and the RP respectively.

$$\text{Gap with LB-RP} = \frac{Z^*_{\text{method}} - Z^*_{\text{LB-RP}}}{Z^*_{\text{method}}} \times 100 \tag{33}$$

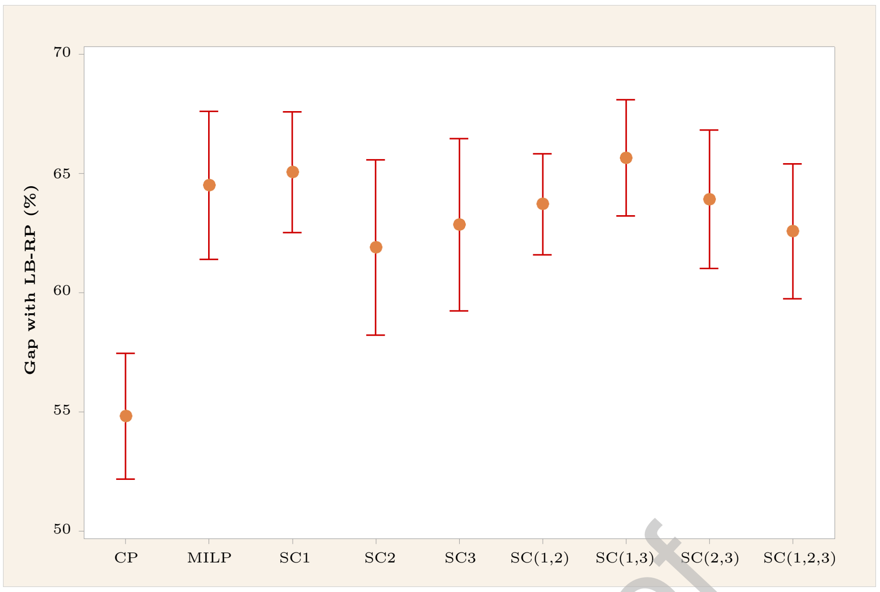


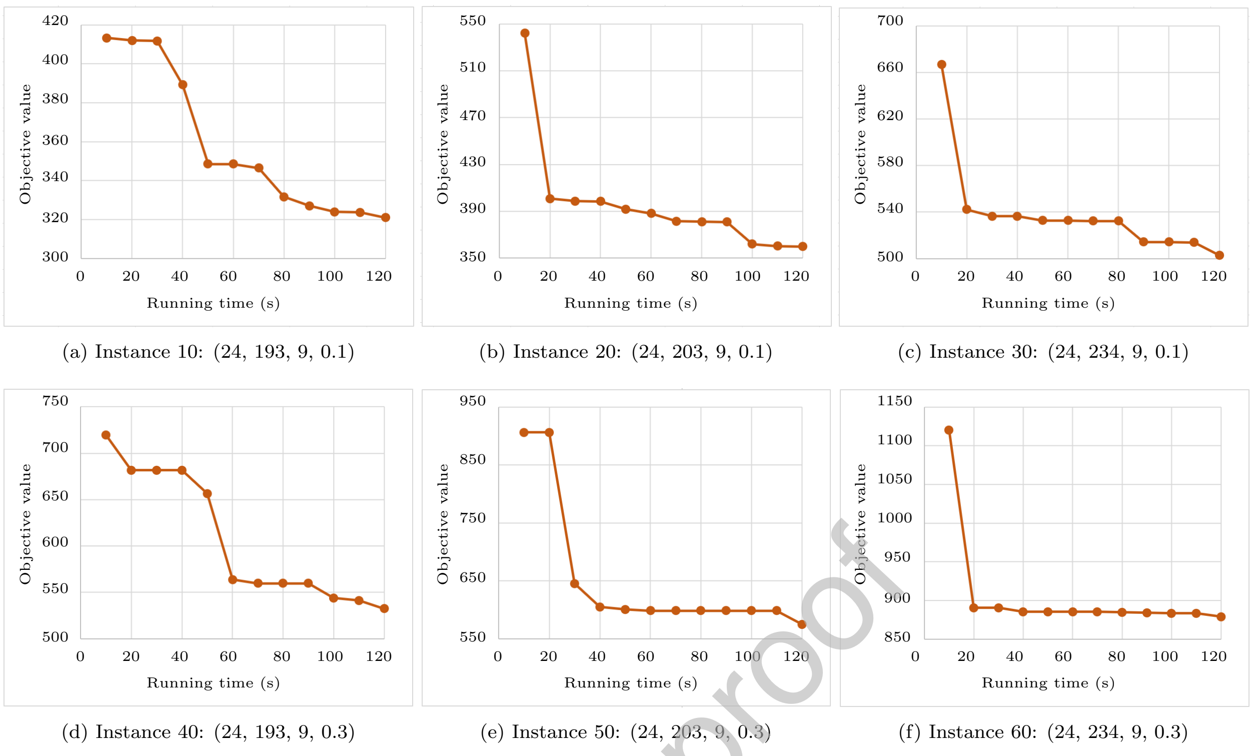
Figure 6: Interval plot with a 95% confidence interval for comparing the proposed approaches based on data provided in Table 1

Results provided in Table 1 show the superiority of the CP model to the MILP model and all SC combinations, as it yields the best average solution gap with the LB-RP (54.84%). Furthermore, it obtains the lowest solution gap with LB-RP in 73.34% of the instances. The MILP model works best when combined with SC2 and SC(1,2,3), where its performance improves by 4.21% and 2.99%, respectively. We also observe that the combination of the MILP model with SC3, SC(1,2), SC(2,3), and SC(1,2,3) improves over its solo performance. Figure 6 demonstrates the interval plot drawn based on data in Table 1 with a 95% confidence interval. Based on this comparison, statistically, there is a significant difference between the CP model and all other approaches. Furthermore, CP has a tighter interval in comparison with the MILP model and SC2, demonstrating that CP's performance is more consistent over all instances.

It is worth mentioning that the above comparison is limited to the instances in Table 1. This is to say, the comparison cannot tell which solution approach is absolutely superior, especially in instances for which all solution approaches yield large gaps. Broadly speaking, the well-known no-free-lunch theorems (Wolpert and Macready, 1997) suggest that we should not expect a single algorithm to dominate all other algorithms across all possible problem instances. If a researcher succeeds in demonstrating the superiority of one algorithm over a portfolio of other algorithms, then one may claim that there are, for example, untested instances where the algorithm may be outperformed (Smith-Miles and Lopes, 2012; Smith-Miles et al., 2014; Smith-Miles and Bowly, 2015).

Table 2: Comparison of the average objective value and average number of nodes for all solution approaches

Approach	Avg. Objective Value	Avg. Number of Nodes
MILP	491.34	17391.85
SC1	477.82	10699.37
SC2	456.84	17108.27
SC3	469.19	15601.85
SC(1,2)	448.57	9266.98
SC(1,3)	483.92	10536.75
SC(2,3)	474.02	12546.65
SC(1,2,3)	443.83	9410.97



(a) Instance 10: (24, 193, 9, 0.1) (b) Instance 20: (24, 203, 9, 0.1) (c) Instance 30: (24, 234, 9, 0.1)

(d) Instance 40: (24, 193, 9, 0.3) (e) Instance 50: (24, 203, 9, 0.3) (f) Instance 60: (24, 234, 9, 0.3)

Figure 7: Improvement in the objective value of the CP model during the run time

Table 2 shows the average number of nodes explored by the MILP model in the presence of all SCs. The MILP model has the least number of constraints and manages to explore the highest number of nodes in T_{cal}. Moreover, SC(1,2,3) which has the highest number of constraints manages to visit the second least number of nodes in comparison with other approaches. However, the average objective value of SC(1,2,3) improves over that of the MILP model by 9.67%. Hence, it can be concluded that while SCs require visiting fewer nodes, they improve the average solution quality of the MILP model.

As the dynamic nature of the problem necessitates, rescheduling of the current schedule is likely to happen on a regular basis. Therefore, the decision maker might do a trade-off between the quality of the obtained solution and the duration of the run time. As a result, a time-based analysis of the CP model's performance is required to understand how its objective value improves during the run time. Figure 7 illustrates the objective values for test instances 10, 20, 30, 40, 50 and 60 solved in T_{cal}, where all instances are selected from Table 1. During the run time, their objective values are reported every 20s. As the figure suggests, for similar hardware configurations and problem settings, the CP model tends to have the largest improvement in the solution quality in the first 90s. Consequently, the decision maker is strongly advised to perform a similar analysis, based on their hardware capabilities and problem setting, to understand the trade-off between the extent to which the objective value improves and the additional run time allocated to the solver.

7. Discussion

Our focus in this section is three-fold. First, two pick-up criteria, namely no-wait and free pick-up, are compared for buffers. Second, the impact of changes in robot speed on the objective value is assessed. Knowledge extracted from the above analysis provides valuable insight into managing resources in a

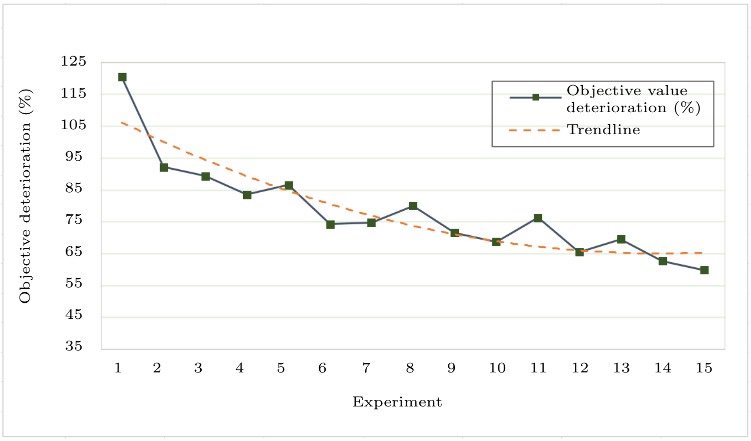


Figure 8: Objective deterioration in the wake of shifting from free pick-up to a no-wait condition for the intermediate buffers

robotic cell. Finally, the sensitivity of the model's outputs to the rescheduling coefficient is analyzed. All instances in this section are solved using the proposed SC(1,2,3) model.

7.1. Buffers' Pick-up Criterion

Two distinct approaches, namely no-wait and free pick-up, can be adopted regarding buffers' pick-up criterion. The no-wait condition can conspicuously deteriorate the objective function since it decreases the schedule flexibility. In contrast, the free pick-up criterion allows a robot to pick up a job from a buffer station when prepared, remarkably improving the problem's objective. To further analyze this matter, 15 problem instances with 5 to 19 jobs are tested under both conditions. The relative amount by which the objective deteriorates is depicted in Figure 8. Based on the experiment presented in this figure, shifting from a free pick-up to a no-wait condition worsens the problem's objective by 78.36% (on average). Furthermore, as the problem instance grows in size, the amount by which the objective deteriorates experiences a downward trend with a decreasing slope showing that the problem's size effect is diminishing.

As deciding about buffers' pick-up criterion appears to be a decision dependent on product needs, it is suggested that operations performed by workstations located in different regions not be related to each other in accordance with the no-wait relationship as much as possible. In other words, if several operations and their corresponding workstations have a no-wait relationship, they should be placed in the same region. Such a design approach helps intermediate buffers abide by the free pick-up rule which in turn improves the cell efficiency. This issue can be addressed at managerial levels while specifying the manufacturing cell layout, so that it is optimized for the particular system in question and its specific needs. Note that the essence of this matter drastically escalates when fewer jobs are scheduled at each turn.

7.2. Robot Speed

Changes in robot speed can also alter the objective value. Faster movement of robots might result in better values of the objective. However, such a decision escalates power usage and imposes more costs on the system. Hence, there exists a trade-off between robot speed and production costs to be considered at a managerial level. To better understand the proportion to which the objective

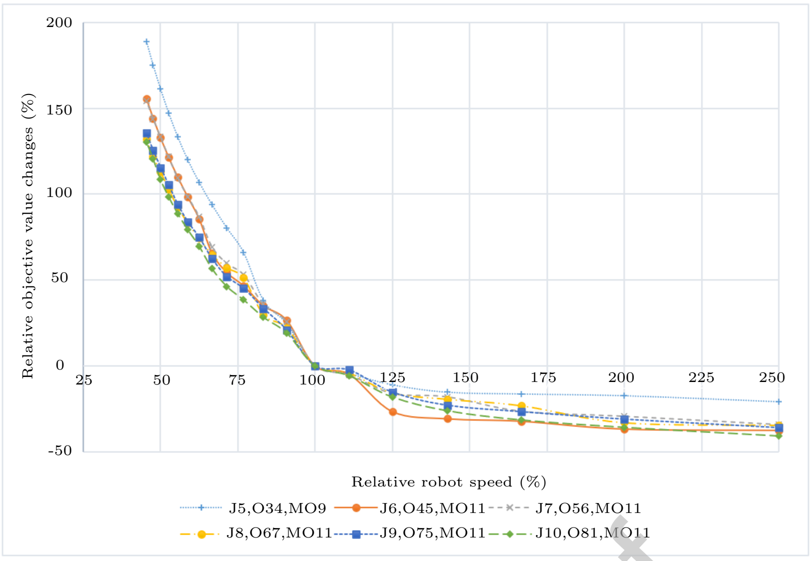


Figure 9: Impact of changes in robot speed on the output of the model

value changes based on the robot speed, six problem instances with 5 to 10 jobs are considered. The total number of operations and the maximum number of operations (MO) per job vary among the instances. Nineteen values for robot speed, ranging from 45.45% to 250% of its initial value, are examined.

The results obtained from this experiment are displayed in Figure 9. As shown in this figure, increasing robot speed ameliorates the objective function in a decreasing manner—the amount by which the objective value improves declines as the speed grows. Furthermore, the objective value is less sensitive to speed, more dependent on the processing duration at a higher speed. As depicted in Figure 9, when the relative robot speed goes beyond 200% of its initial value, the objective value remains almost unchanged. At this point, increasing robot speed any further will not yield sensibly superior results, while it is still likely to raise imposed costs on the system. Therefore, when management is inclined to increase cell productivity (through reducing C_{max}), increasing robots' speed should take place with respect to a similar analyse. Besides, this measure should be taken only as long as the objective value is highly sensitive to robots' speed and imposed costs do not undermine the achieved level of productivity.

7.3. Rescheduling Coefficient

The rescheduling approach, adopted for tackling the dynamic nature of the problem, can alter the ongoing schedule to a great extent. This may not be preferred as previously scheduled products will experience changes in their former completion times. Therefore, such changes should incur a penalty on the objective of the problem, which is controlled by γ. This penalty coefficient enables the manager to decide on the extent to which it is preferred to maintain the current schedule while adding new jobs. Greater values for γ increase the objective value of the problem. However, the value of C_{max} does not necessarily deteriorate every time γ increases. In fact, C_{max} deteriorates at some steps and remains unchanged for the rest. Figure 10 demonstrates alterations of the objective function and C_{max} for a problem instance with nine jobs and two arriving jobs which will be added to the current schedule, for 18 values of γ. As it is observed in Figure 10, raising the value of γ beyond a specific threshold results in a situation in which the model gives the highest priority to the rescheduling cost (due to the high penalty imposed by this component). Therefore, the difference between the objective value and C_{max} becomes zero. After this threshold, the model becomes insensitive to the γ parameter. Besides, at this point, the previous completion times for jobs already existing in the system remain unchanged.

8. Conclusion

Conclusively, this research studied a dynamic scheduling problem within a U-shaped robotic cell. Robotic arms were placed in the middle of the cell to perform the material handling process. Each robot had access to a predefined subset of workstations based on its physical limitations, with intermediate buffers used by robots to exchange parts. As the job shop situation was considered, each part had a particular process plan. Workstations abided by the no-wait rule, while two pick-up criteria, namely no-wait and free pick-up, were considered for buffers. As the schedule began, new parts could arrive at the cell at unpredictable times. For this purpose, a complete rescheduling approach by considering the current system state was adopted.

To solve the proposed problem, first, a MILP model was developed. Second, three SCs were devised based on feasible and logical robot behavior. Third, the CP method was utilized for modeling and solving the problem. Through extensive numerical experiments, an analogy was drawn between the CP and MILP models in presence of all combinations of SCs for solving large-size problems in the same run time. The CP model outperformed all approaches and proved to be capable of finding quality feasible solutions in T_{cal} which perfectly fits the dynamic nature of the problem. Furthermore, the superior combinations of the SCs were identified for the MILP model. Next, a sensitivity analysis was performed to assess pick-up criteria for buffer stations. It was demonstrated that the free pick-up criterion conspicuously outperformed the no-wait situation in terms of cell performance. Moreover, the impact of changes in the problem's size was analyzed regarding this matter. Next, the effect of alterations in robots' speed on the objective function was studied. It was demonstrated that increasing the robot's speed beyond a threshold fails to further improve the cell performance, as the workstations' processing rates become more influential when the robot's speed rises. Additionally, the proportion to which cell performance improves declines as the robot speed grows. The mentioned analysis provides managerial insight regarding resource management in the studied robotic cell. Finally, the alterations in the model's outputs were assessed subject to changes in the rescheduling parameter. It was concluded that putting more emphasis on maintaining the current schedule through

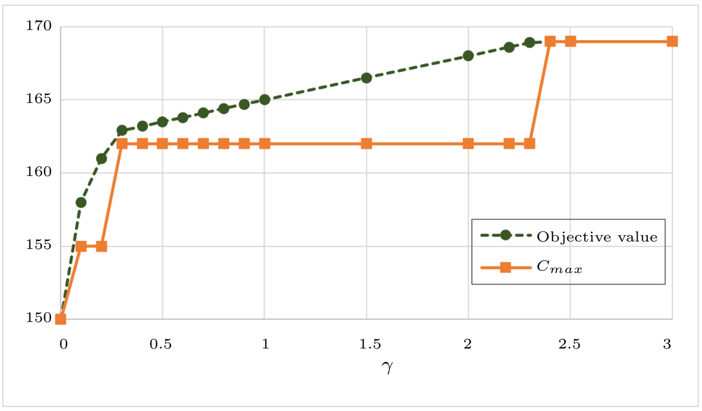


Figure 10: Impact of changes in γ on the objective value and the value of C_{max}

increasing the penalty incurred on the objective function does not necessarily deteriorate the C_{max} of the schedule. Further, giving absolute importance to the rescheduling penalty results in an objective value equal to the C_{max}.

For further research, a flexible job shop environment can be studied. Considering a subset of machines that can operate alternatively increases the problem's complexity. Dynamicity is also not limited to the arrival of new jobs at a cell. As a result, machine breakdowns and other interruptions to the system can be taken into account. Another prominent aspect to shed light on is utilizing dual-arm robots. This course of action can have a drastic impact on buffers, specifically when buffers abide by the no-wait condition, which was the focus of this study.

Appendix A. A Procedure to Obtain a Lower Bound from a Relaxed Problem

As it is demonstrated in Section 6, the large-scale instances in Table 1 cannot be solved to proven optimality. On the other hand, a problem which is sufficiently relaxed can be solved to optimality. In order to obtain such RP, complicated assumptions should be dismissed. In this regard, the capacity constraints for workstations and buffers, along with the empty movement constraints for the robots are relaxed. A minor modification to the CP model proposed in Section 5 yields the following relaxed problem:

$$\text{RP} = minimize\{(32) \mid (25) - (28), (29) \text{ for all } m \in \mathcal{M} \setminus \{Buf_{set} \cup Buf'_{set}\}\}$$

Note that Equation 29 prevents overlaps of the interval variables on a common resource. Therefore, excluding Buf_{set} and $Buf^{'}_{set}$ from $\mathcal{M}$ is equivalent to relaxing the capacity constraints over buffers and workstations respectively. To verify the validity of the obtained lower bound, we emphasize that the lower bound obtained from any relaxation of the original problem is valid. The reason behind this is that such a relaxed problem optimizes over some set containing all feasible solutions of the original problem as a proper subset. This is to say, we can conclude that the optimal solution of the relaxed problem gives a valid lower bound for the original problem.

Appendix B. An Analysis over Instances Solved to Optimality

The large-scale instances provided in Table 1 do not reach proven optimality, and thus it is critical to understand the characteristics of instances which can be solved to optimality. For this purpose, 14 medium-size sets of instances are generated. The $(|\mathcal{J}|, |\mathcal{O}|, Arr, \gamma)$ is similar for each set, while the processing times are different. Within each set, the instance labeled as "M" is generated using the procedure described in Section 6. For all 14 sets, two other scenarios are considered: in the "L" scenario, from each job which is arriving to the system, two operations are randomly selected and their processing times are decreased by 30%. In the "H" scenario, for all such operations, the processing times are increased by 30%.

As displayed in Table B.3, all instances are solved to optimality using the SC(1,2,3) model. Those instances which found proof of optimality using the SC(1,2,3) model in $T_{cal} \leq 1200s$ are labeled as "Yes". Besides, all instances are solved with the CP model in $T_{cal} = 60s$. For the CP model, the time when each instance reaches the reported solution is also given to provide further insight into CP's performance (similar to the analysis which is provided in Figure 7). It should be noted that, since $T_s = 0$ for all instances, changing T_{cal} does not change the input parameters of the problem, making

Table B.3: Performance of CP and LB-RP on medium-size instances which are solved to optimality by SC(1,2,3)

Set	$(\lvert\mathcal{J}\rvert, \lvert\mathcal{O}\rvert, Arr, \gamma)$	Processing time	SC(1,2,3)		CP ($T_{cal}=60$s)		LB-RP	
			Optimal ($T_{cal} \leqslant 1200$s)	Time (s)	Gap with Optimality (%)	Time (s)	Gap with Optimality (%)	Time (s)
1	(9, 55, 4, 0.1)	L	Yes	573.92	0	7.91	47.60	2.10
		M	Yes	632.95	0	19.69	43.73	3.26
		H	Yes	467.63	0	10.47	45.23	1.99
2	(9, 55, 4, 0.3)	L	Yes	533.13	0	46.7	48.20	3.28
		M	Yes	576.36	0.13	8.52	43.69	2.55
		H	Yes	612.92	0	36.63	45.48	1.98
3	(9, 61, 4, 0.1)	L	Yes	593.72	0.26	5.2	45.05	4.00
		M	Yes	573.72	0.76	9.72	45.31	5.38
		H	Yes	531.81	1	9.25	44.66	5.34
4	(9, 61, 4, 0.3)	L	Yes	525.45	2.41	15.99	44.75	3.64
		M	Yes	502.41	0	31.6	45.72	3.27
		H	Yes	615.67	0	53.98	46.39	3.97
5	(10, 54, 5, 0.1)	L	Yes	562.55	0	9.44	45.47	2.02
		M	Yes	598.82	0.14	2.32	49.59	3.90
		H	Yes	701.81	0	3.18	45.47	2.83
6	(10, 54, 5, 0.3)	L	Yes	698.81	0	6.94	47.08	1.18
		M	Yes	573.03	0	43.31	49.33	1.65
		H	Yes	778.50	0	7.19	46.24	1.78
7	(10, 59, 5, 0.1)	L	Yes	778.50	6.62	14.86	44.68	2.14
		M	Yes	584.88	0	15	49.94	6.60
		H	Yes	544.53	1	18.87	47.19	3.20
8	(10, 59, 5, 0.3)	L	Yes	595.72	0	19.77	49.02	1.55
		M	Yes	633.03	0	11.96	50.00	3.12
		H	Yes	648.31	0	49.73	47.57	1.87
9	(11, 63, 6, 0.1)	L	No	1538.75	0.86	7.62	47.76	1.44
		M	Yes	786.98	0	5.5	52.43	4.84
		H	Yes	843.53	1	7.65	50.35	4.23
10	(11, 63, 6, 0.3)	L	No	1305.94	0	5.2	48.28	0.98
		M	Yes	948.38	0	5.73	52.76	1.95
		H	Yes	1100.38	2	7.49	51.78	2.99
11	(11, 68, 6, 0.1)	L	No	2752.33	0	30.1	48.87	2.35
		M	Yes	1158.42	0.44	8.72	52.74	8.27
		H	Yes	826.30	1	46.63	49.19	8.96
12	(11, 68, 6, 0.3)	L	No	2076.06	1.50	17.37	49.53	1.45
		M	No	1412.88	0.65	6.85	53.04	6.01
		H	Yes	962.13	2	6.25	50.45	7.57
13	(12, 68, 7, 0.1)	L	No	16172.36	0	52.26	47.19	1.31
		M	No	7664.52	0.11	4.66	50.27	1.14
		H	No	1225.20	2	44.38	47.90	1.70
14	(12, 68, 7, 0.3)	L	No	12579.45	0.45	41.62	47.67	0.98
		M	No	2704.34	0.32	6.66	50.80	1.10
		H	Yes	1020.89	1.73	24.37	49.39	1.62
Average				1702.79	0.62	18.75	48.04	3.13

it possible to do a valid comparison between the SC(1,2,3) model and the CP model under above circumstances. Furthermore, using Equation 33, the gap of the optimal solution of each instance with its LB-RP is obtained. In order to calculate the LB-RP, a constraint programming model of the RP is solved to optimality, as it is described in Appendix A.

Based on the results presented in Table B.3, there are several observations to consider: to begin with, the complexity of the problem depends on the processing times of operations such that shorter processing times result in a problem with higher complexity. As it is demonstrated in the table, while the process plans within each group are analogous among "L", "M", and "H" instances, in groups 9, 10, and 11 only the "L" instances fail to be solved to optimality. The same observation holds for groups 12 and 14 where the "H" instance is the only variant with the "Yes" label. This is mainly due to the no-wait pick-up criterion–as the processing times get shorter, the robot should unload parts from their current workstations faster. Since robots should also perform empty movements when shifting from one job to another, faster unloading of jobs from their workstations makes more empty movement constraints binding, and this can add to the complexity of the problem. Another observation is that the gap of optimal solution for each instance with its LB-RP is fairly robust towards the changes in the processing times. As the RP dismisses the constraints concerning the empty movements of robots along with the capacity constraints of workstations and buffers, it is expected that changes in the processing times of operations adversely affect the quality of the LB-RP. However, results presented in Table B.3 show no such pattern. Furthermore, due to extensive relaxation of the original problem, the relaxed version of the problem takes 3.13s on average to reach proof of optimality. Finally, as results provided in Table 1 already demonstrated, the CP model has the best performance in finding quality feasible solutions in fairly short run times. In fact, the results provided in Table B.3 reveal that the CP model can manage to find near optimal solutions (with an average gap of 0.62%) in 60s for all medium-size instances.

Acknowledgement

The authors would like to thank the anonymous referees for their constructive suggestions and comments. The third author acknowledges support from the University of Melbourne and the Albert Shimmins Research Continuity Fund.

References

Applegate, D., Cook, W., 1991. A computational study of the job-shop scheduling problem. ORSA Journal on Computing 3, 149 – 156.

Behrens, J., Lange, R., Mansouri, M., 2019. A constraint programming approach to simultaneous task allocation and motion scheduling for industrial dual-arm manipulation tasks. International Conference on Robotics and Automation , 8705 – 8711.

Bock, F., Bruhn, H., 2021. Case study on scheduling cyclic conveyor belts. Omega 102, 102339.

Booth, K., Tran, T., Zejat, G., Beck, J., 2016a. Mixed-integer and constraint programming techniques for mobile robot task planning. IEEE Robotics and Automation Letters 1, 500 – 507.

Booth, K., Zejat, G., Beck, J., 2016b. A constraint programming approach to multi-robot task allocation and scheduling in retirement homes. International conference on principles and practice of constraint programming , 539 – 555.

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

Bukchin, Y., Raviv, T., 2018. Constraint programming for solving various assembly line balancing problems. Omega 78, 57 – 68.

Carlier, J., Haouari, M., Kharbeche, M., Moukrim, A., 2010. An optimization-based heuristic for the robotic cell problem. European Journal of Operational Research 202, 636 – 645.

Chan, W., Yi, J., Ding, S., Song, D., 2008. Optimal scheduling of k-unit production of cluster tools with single-blade robots. IEEE International Conference on Automation Science and Engineering , 335 – 340.

Che, A., Chabrol, M., Gourgand, M., Wang, Y., 2012. Scheduling multiple robots in a no-wait re-entrant robotic flowshop. International Journal of Production Economics 135, 199 – 208.

Che, A., Chu, C., 2009. Multi-degree cyclic scheduling of a no-wait robotic cell with multiple robots. European Journal of Operational Research 199, 77 – 88.

Che, A., Chu, C., Levner, E., 2003. A polynomial algorithm for 2-degree cyclic robot scheduling. European Journal of Operational Research 145, 31 – 44.

Che, A., Hu, H., Chabrol, M., Gourgand, M., 2011. A polynomial algorithm for multi-robot 2-cyclic scheduling in a no-wait robotic cell. Computers & Operations Research 38, 1275 – 1285.

Che, A., Kats, V., Levner, E., 2017. An efficient bicriteria algorithm for stable robotic flow shop scheduling. European Journal of Operational Research 260, 964 – 971.

Dang, Q., Nguyen, C., Rudová, H., 2019. Scheduling of mobile robots for transportation and manufacturing tasks. Journal of Heuristics 25, 175 – 213.

Dawande, M., Geismar, H., Sethi, S., Sriskandarajah, C., 2007. Throughput optimization in robotic cells. volume 101. Springer Science & Business Media.

Delorme, M., Santini, A., 2022. Energy-efficient automated vertical farms. Omega 109, 102611.

Dorndorf, U., Pesch, E., Phan-Huy, T., 2000. Constraint propagation techniques for the disjunctive scheduling problem. Artificial Intelligence 122, 189 – 240.

Dorndorf, U., Pesch, E., Phan-Huy, T., 2002. Constraint propagation and problem decomposition: A preprocessing procedure for the job shop problem. Annals of Operations Research 115, 125 – 145.

Elmi, A., Topaloglu, S., 2016. Multi-degree cyclic flow shop robotic cell scheduling problem: Ant colony optimization. Computers & Operations Research 73, 67 – 83.

Elmi, A., Topaloglu, S., 2017. Cyclic job shop robotic cell scheduling problem: Ant colony optimization. Computers & Industrial Engineering 111, 417 – 432.

Enayaty-Ahangar, F., Rainwater, C.E., Sharkey, T.C., 2019. A logic-based decomposition approach for multi-period network interdiction models. Omega 87, 71–85.

Feng, J., Che, A., Chu, C., 2015. Dynamic hoist scheduling problem with multi-capacity reentrant machines: A mixed integer programming approach. Computers & Industrial Engineering 87, 611 – 620.

Gecili, H., Parikh, P., 2022. Joint shelf design and shelf space allocation problem for retailers. Omega 111, 102634.

Gedik, R., Kalathia, D., Egilmez, G., Kirac, E., 2018. A constraint programming approach for solving unrelated parallel machine scheduling problem. Computers & Industrial Engineering 121, 139 – 149.

Geismar, H., Pinedo, M., Sriskandarajah, C., 2008. Robotic cells with parallel machines and multiple dual gripper robots: a comparative overview. IIE Transactions 40, 1211 – 1227.

Geismar, H., Sriskandarajah, C., Ramanan, N., 2004. Increasing throughput for robotic cells with parallel machines and multiple robots. IEEE Transactions on Automation Science and Engineering 1, 84 – 89.

Gultekin, H., Coban, B., Akhlaghi, V., 2018. Cyclic scheduling of parts and robot moves in m-machine robotic cells. Computers & Operations Research 90, 161 – 172.

Hall, N., Sriskandarajah, C., 1996. A survey of machine scheduling problems with blocking and no-wait in process. Operations Research 44, 510 – 525.

Ham, A., 2020. Transfer-robot task scheduling in flexible job shop. Journal of Intelligent Manufacturing 31, 1783 – 1793.

Ham, A., 2021. Transfer-robot task scheduling in job shop. International Journal of Production Research 59, 813 – 823.

Ham, A., Cakici, E., 2016. Flexible job shop scheduling problem with parallel batch processing machines: Mip and cp approaches. Computers & Industrial Engineering 102, 160 – 165.

Ham, A., Park, M.J., 2021. Human robot task allocation and scheduling: Boeing 777 case study. IEEE Robotics and Automation Letters 6, 1256 – 1263.

Hashemi-Petroodi, S.E., Thevenin, S., Kovalev, S., Dolgui, A., 2022. Model-dependent task assignment in multi-manned mixed-model assembly lines with walking workers. Omega 113, 102688.

Heinz, V., Novák, A., Vlk, M., Hanzálek, Z., 2022. Constraint programming and constructive heuristics for parallel machine scheduling with sequence-dependent setups and common servers. Computers & Industrial Engineering 172, 108586.

IBM, 2020. Constraint programming modeling for python (docplex.cp) , Retrieved from https://ibmdecisionoptimization.github.io/docplex--doc/cp/index.html.

IFR, 2021. International federation of robotics , Retrieved from https://ifr.org/downloads/press2018/Forecast_installations_WR2021.jpg.

Jolai, F., Foumani, M., Tavakoli-Moghadam, R., Fattahi, P., 2012. Cyclic scheduling of a robotic flexible cell with load lock and swap. Journal of Intelligent Manufacturing 23, 1885 – 1891.

Kharbeche, M., Carlier, J., Haouari, M., Moukrim, A., 2011. Exact methods for the robotic cell problem. Flexible services and manufacturing journal 23, 242 – 261.

Kim, D., Kim, H., Lee, T., 2017. Optimal scheduling for sequentially connected cluster tools with dual-armed robots and a single input and output module. International Journal of Production Research 55, 3092 – 3109.

Ku, W.Y., Beck, C., 2016. Mixed integer programming models for job shop scheduling: a computational analysis. Computers & Operations Research 73, 165 – 173.

Laborie, P., Rogerie, J., Shaw, P., Vilím, P., 2018. Ibm ilog cp optimizer for scheduling. Constraints 23, 210 – 250.

Lanza, G., Passacantando, M., Scutellà, M.G., 2022. Assigning and sequencing storage locations under a two level storage policy: Optimization model and matheuristic approaches. Omega 108, 102565.

Li, X., Chan, F., Chung, S., 2015. Optimal multi-degree cyclic scheduling of multiple robots without overlapping in robotic flowshops with parallel machines. Journal of Manufacturing Systems 36, 62 – 75.

Liu, S., Kozan, E., 2017. A hybrid metaheuristic algorithm to optimise a real-world robotic cell. Computers & Operations Research 84, 188 – 194.

Lunardi, W., Birgin, E., Laborie, P., Ronconi, D., Voos, H., 2020. Mixed integer linear programming and constraint programming models for the online printing shop scheduling problem. Computers & Operations Research 123, 105020.

Meng, L., Zhang, C., Ren, Y., Zhang, B., Lv, C., 2020. Mixed-integer linear programming and constraint programming formulations for solving distributed flexible job shop scheduling problem. Computers & Industrial Engineering 142, 106347.

Murín, S., Rudová, H., 2019. Scheduling of mobile robots using constraint programming. International Conference on Principles and Practice of Constraint Programming Springer, Cham, 456 – 471.

Nouri, H., Driss, O., Ghédira, K., 2016. Simultaneous scheduling of machines and transport robots in flexible job shop environment using hybrid metaheuristics based on clustered holonic multiagent model. Computers & Industrial Engineering 102, 488 – 501.

Polyakovskiy, S., M'Hallah, R., 2021. Just-in-time two-dimensional bin packing. Omega 102, 102311.

Russell, R., 2013. A constraint programming approach to designing a newspaper distribution system. International Journal of Production Economics 145, 132 – 138.

Shabtay, D., Arviv, K., Stern, H., Edan, Y., 2014. A combined robot selection and scheduling problem for flow-shops with no-wait restrictions. Omega 43, 96 – 107.

da Silva, N., Scarpin, C., Jr, J.P., Ruiz, A., 2019. Online single machine scheduling with setup times depending on the jobs sequence. Computers & Industrial Engineering 129, 251 – 258.

Smith-Miles, K., Baatar, D., Wreford, B., Lewis, R., 2014. Towards objective measures of algorithm performance across instance space. Computers & Operations Research 45, 12–24.

Smith-Miles, K., Bowly, S., 2015. Generating new test instances by evolving in instance space. Computers & Operations Research 63, 102–113.

Smith-Miles, K., Lopes, L., 2012. Measuring instance difficulty for combinatorial optimization problems. Computers & Operations Research 39, 875–889.

Vahedi-Nouri, B., Tavakkoli-Moghaddam, R., Hanzálek, Z., Dolgui, A., 2022. Workforce planning and production scheduling in a reconfigurable manufacturing system facing the covid-19 pandemic. Journal of Manufacturing Systems 63, 563–574.

Watanabe, K., Inada, S., 2020. Search algorithm of the assembly sequence of products by using past learning results. International Journal of Production Economics 226, 107615.

Wolpert, D., Macready, W., 1997. No free lunch theorems for optimization. IEEE Transactions on Evolutionary Computation 1, 67–82.

Xiong, H., Shi, S., Ren, D., Hu, J., 2022. A survey of job shop scheduling problem: the types and models. Computers & Operations Research , 105731.

Yan, P., Che, A., Cai, X., Tang, X., 2014. Two-phase branch and bound algorithm for robotic cells rescheduling considering limited disturbance. Computers & Operations Research 50, 128 – 140.

Yan, P., Liu, S., Sun, T., Ma, K., 2018. A dynamic scheduling approach for optimizing the material handling operations in a robotic cell. Computers & Operations Research 99, 166 – 177.

Yang, Y., Chen, Y., Long, C., 2016. Flexible robotic manufacturing cell scheduling problem with multiple robots. International Journal of Production Research 54, 6768 – 6781.

Yi, J., Ding, S., Song, D., 2005. Steady-state throughput and scheduling analysis of multi-cluster tools for semiconductor manufacturing: A decomposition approach. IEEE International Conference on Robotics and Automation , 292 – 298.

Zhao, C., Fu, J., Xu, Q., 2013. Real-time dynamic hoist scheduling for multistage material handling process under uncertainties. AIChE Journal 59, 465 – 482.

Zhou, B., Li, M., 2018. Scheduling method of robotic cells with machine–robot process and time window constraints. Proceedings of the Institution of Mechanical Engineers, Part E: Journal of Process Mechanical Engineering 232, 650 – 661.