An Exact Decomposition-based Approach to the Conflict-Free-Transportation-Constrained Flexible Job-Shop Scheduling Problem

Roel W.M. van Os[a,*], Twan Basten[a]

[a]*Department of Electrical Engineering, Eindhoven University of Technology, Groene Loper 19, Eindhoven, 5612 AP, Netherlands*

Abstract

Automated material handling and transportation of semi-finished products has great potential to enhance the efficiency of Flexible Manufacturing Systems (FMSs). The resulting optimization problems for finding the shortest-makespan manufacturing schedules are complex and often hard to solve. Automated vehicles used for transportation must avoid colliding with one another while completing all required transports in the shortest possible time. The Conflict-Free-Transportation-constrained Flexible Job-Shop Scheduling Problem (CFTFJSSP) is the combination of flexible job-shop scheduling with conflict-free vehicle routing. To solve the CFTFJSSP, we propose an exact Logic-Based Benders Decomposition (LBBD) using both Constraint Programming (CP) and Integer Linear Programming (ILP) techniques. This LBBD-based approach is proven to outperform existing approaches to solving the CFTFJSSP in terms of solution quality on benchmark instances currently available in the literature. For most benchmark instances, our LBBD-based approach finds optimal makespan values. Because of time boxing, only in a few cases, the optimality of the found solution cannot be guaranteed. The solutions found by our LBBD-based approach show a makespan improvement of at least 10% for about half of the benchmarks instances, up to a 35% improvement in the best case, when compared to the heuristic solution approaches from the literature.

Keywords: Scheduling, Conflict-Free Routing, Logic-Based Benders Decomposition, Conflict-Driven Benders Cuts, Constraint Programming, Mathematical Programming

1. Introduction

With the modern advancements in Flexible Manufacturing Systems (FMSs), manufacturing processes require ever more flexibility. Traditional optimal job scheduling focuses on the sequencing of operations. Most of these problems assume fixed machine allocations leaving transportation of materials and semi-finished products out of scope. However, material handling and transportation of semi-finished products are an integral part of the manufacturing system and its automation is highly anticipated (Kusiak, 2018). Moreover, in modern FMSs, internal transportation is increasingly becoming a bottleneck that requires optimization (Hosseini et al., 2023). New affordable and flexible autonomous transportation technologies, such as Automated Guided Vehicles (AGVs), have great prospects to optimize manufacturing processes. With improved internal transportation, decoupling of machines and operations, where operations can be flexibly allocated to different machines, can further improve productivity and flexibility of FMSs. In these so-called *flexible job shops*, each operation can be assigned to a machine that best fits the production and vehicle routing scheme. Integral consideration of operation allocation and scheduling in flexible job shops and internal transportation routing and scheduling is a relevant recent research trend, as these kinds of FMSs characterize the most flexible manufacturing processes.

New challenges arise when combining internal transportation with optimal job scheduling. Moving autonomous vehicles must avoid problems such as collisions, deadlocks, and livelocks when moving

*Corresponding Author

Email addresses: r.w.m.v.os@tue.nl (Roel W.M. van Os), a.a.basten@tue.nl (Twan Basten)

semi-finished jobs or materials on the manufacturing floor. The optimization problem that arises is the Conflict-Free Routing Problem (CFRP). The CFRP has already been studied intensively. However, the combination of Conflict-Free Routing (CFR) with flexible job-shop scheduling is scarcely studied, despite its practical relevance. The combination of eliminating transportation conflicts and maximizing production efficiency is an interesting challenge in the integral scheduling of flexible jobs and conflict-free internal transportation.

The literature contains two prior studies that consider flexible job-shop scheduling with conflict-free internal transportation (Lyu et al., 2019; Liu et al., 2023). The two papers describe metaheuristic solution approaches, on two slightly different variants of the problem. Besides makespan optimization, Lyu et al. (2019) also considers determining the optimal number of vehicles for given combinations of system layouts and job sets. The implementations of the methods presented in (Lyu et al., 2019; Liu et al., 2023) are not available to the community at large, hampering further research on this problem.

In this paper, we integrate conflict-free internal transportation routing and flexible job-shop scheduling into the Conflict-Free-Transportation-constrained Flexible Job-Shop Scheduling Problem (CFTFJSSP). Our main objective is to find exact shortest makespan solutions for CFTFJSSP instances and to investigate scalability of such an exact approach. As the overall problem naturally decomposes into separate optimization problems, namely job scheduling and CFR, we propose an approach using the Logic-Based Benders Decomposition (LBBD) (Hooker and Ottosson, 2003; Hooker, 2024). Such a decomposition-based approach can be further refined using (meta)heuristics for subproblems and/or for cuts that steer the overall iterative process, if the need arises. The main contributions of this study are the following:

- The first exact approach to solving the CFTFJSSP, based on the Logic-Based Benders Decomposition. The decomposition uses an innovative *conflict-driven cut* that uses information from routing conflicts from CFR subproblems to speed up convergence of the iterative overall solution process.
- An evaluation of the proposed approach on existing benchmarks from literature, showing that our approach outperforms the existing heuristic approaches of (Lyu et al., 2019; Liu et al., 2023) in solution quality (makespan) on all available benchmarks. Our approach finds solutions that are guaranteed to be optimal in all but a few cases, when optimality cannot be guaranteed because of time boxing. Our experimental evaluation further shows that the approach of Lyu et al. (2019) to determine the optimal number of vehicles for given combinations of system layout and job sets provides suboptimal results for all but the smallest benchmark.
- Open-source software tools allowing further development on the presented CFTFJSSP implementation and performance comparison on benchmarks, https://github.com/TUE-EE-ES/TJSP-toolset/releases/tag/v1.2.0.

The remainder of this paper is organized as follows. Section 2 discusses the work related to the main topics of this study. Section 3 discusses the mathematical definition and constraints of the CFTFJSSP. Hereafter, the exact decomposition-based CFTFJSSP solution and its models are discussed in Section 4. Section 5 discusses the conflict-driven cuts used in the decomposition procedure. Then, Section 6 discusses the implementation for this new solution, and Section 7 evaluates its performance. Lastly, Section 8 summarizes the main conclusions of this work and discusses potential future work.

2. Related Work

This section gives an overview of studies related to the combination of job scheduling and transportation that preceded our research. We start by giving an overview of the foundation and classification of optimal job scheduling with transportation considerations. Then, we summarize the foregoing research for optimal job scheduling with fixed internal transportation times Hereafter, we introduce previous work related to AGV control, specifically conflict-free routing. Lastly, we look at the combined research of optimal job scheduling with conflict-free routing.

Optimal Job Scheduling with Transportation Classification. According to Lee and Chen (2001), the earliest optimal job scheduling publication with transportation considerations is the paper (Maggu and Das, 1980). It considers a flow shop makespan problem where transporters move semi-finished jobs from one machine to the next. At the same time, Potts (1980) considered a job-scheduling problem where the completion time of jobs is defined as the time when a job arrives at a customer. Publications (Maggu and Das, 1980) and (Potts, 1980) defined two branches of optimal job scheduling with transportation considerations. One focuses on the intermediate transport of jobs from one processing unit to another, while the other focuses on the distribution of finished jobs to customers. These two different types of transportation are classified by Lee and Chen (2001) as *type 1* and *type 2* transportation, respectively. Later, scheduling problems with type 2 transportation considerations would be commonly known as the Integrated Production and Outbound Distribution Scheduling (IPODS) problem, as introduced by Chen (2010). The research in this paper focuses on job scheduling problems with type 1 transportation considerations. Hosseini et al. (2023) names the problem setting for the integrated planning of manufacturing and internal logistics (type 1 transportation) as Scheduling Problems in Manufacturing with Transportation (SchedPT). However, we prefer a naming connecting to the standard naming in the domain: Transportation-constrained Job-Scheduling Problems (TJSPs), where *Transportation-constrained* refers to *internal* transportation constraints.

A rather small share of work on TJSPs considers complex manufacturing environments with multiple transporters and additional scheduling flexibility (Hosseini et al., 2023). In most TJSP research, transportation times are constant between machines, and detailed routing is often left out of scope.

Transportation-constrained Job-Scheduling Problems. One specific TJSP instance, the Transportation-constrained Job-Shop Scheduling Problem (TJSSP), has been frequently studied, as reviewed by Xie and Allen (2015) and Nouri et al. (2016). This problem considers a traditional job-shop environment, forming the basis of the flexible job-shop problem we explore in this work. Most of the current state-of-the-art TJSSP solutions are inspired by the foundations laid by Bilge and Ulusoy (1995). They consider a traditional job shop with a makespan minimization objective. The transportation is performed by AGVs that start at a load/unload station. AGVs do not return to the load/unload station after transportation and transfer directly to the next job, resulting in sequence-dependent travel times. They suggest a decomposition-based time-window heuristic in which a schedule with a transportation time window is provided and subsequently, a feasible transportation solution for that time window is searched. If no such solution is found, the process starts over with a revised schedule and a new time window. The algorithm was tested by introducing a benchmark set of problem instances. Later, Ulusoy et al. (1997) proposed a genetic algorithm for solving the same problem, outperforming the solution of (Bilge and Ulusoy, 1995) on the same benchmark. This genetic algorithm approach was eventually improved by Abdelmaguid et al. (2004), who introduced a hybrid scheme with a genetic algorithm to solve the scheduling of machines and a heuristic for scheduling the AGVs. The introduction of the heuristic increases the efficiency of the genetic algorithm search, resulting in even better benchmark results. Several improved heuristics for the benchmark of Bilge and Ulusoy (1995) followed, such as a tabu search algorithm in (Zheng et al., 2014) and a colored Petri net-based heuristic in (Baruwa and Piera, 2016). Also, several problem extensions were investigated, such as the multi-objective TJSSP in (Reddy and Rao, 2006). While these algorithms provide feasible solutions to the Bilge and Ulusoy (1995) benchmark, they fail to prove optimality for any of the found results.

El Khayat et al. (2006) took another research direction and provided the first exact mathematical and constraint programming models for the TJSSP. Both models were implemented and evaluated on their computation time to find an optimal solution. Later, Fontes and Homayouni (2019) pointed out that the model of El Khayat et al. (2006) was under-constrained, making the results unsuitable for the Bilge and Ulusoy (1995) benchmark. State-of-the-art mathematical programming models for the TJSSP are provided by Fontes and Homayouni (2019), Huang (2018), and more recently by Zhang et al. (2025). The presented models verify the optimal solutions to all but two instances of the Bilge and Ulusoy (1995) benchmark. Ham (2021) proposes a constraint programming model that outperforms the other exact approaches by proving optimal solutions to all benchmark instances. While these works fundamentally shaped the research landscape in TJSP, their scope is limited to predetermined operation-to-machine allocations.

To deal with the flexible allocation of operations to machines, Deroussi and Norre (2010) adapted the benchmark of Bilge and Ulusoy to deal with a Transportation-constrained *Flexible* Job-Shop Scheduling Problem (TFJSSP). To solve the problem, one of the authors suggested a hybrid particle swarm optimization algorithm with a local search in Deroussi (2014). Various other metaheuristic approaches have been proposed for effectively solving the benchmark's problem instances. Zhang et al. (2012) suggested a genetic algorithm for solving the machine allocation problem for operations and a tabu search procedure for sequencing the operations on each machine. Shortly after, the same authors improved this work in (Zhang et al., 2013) by incorporating a shifting bottleneck heuristic to address the sequencing of machine and vehicle tasks. Later, Homayouni and Fontes (2021) presented a late acceptance hill climbing method that demonstrated improved performance over the methods of Zhang et al. (2012, 2013) in solving the Deroussi benchmark. Only recently, Berterottière et al. (2024) proposed a tabu search heuristic solution with a novel and effective neighborhood search, outperforming all other state-of-the-art solutions at that point. While these studies show promising solutions to the TFJSSP, their implementations are heuristic-based and do not provide exact solutions to the benchmark instances.

The first exact mathematical programming model for the TFJSSP was proposed by Homayouni and Fontes (2019, 2021). The model was able to assess optimal solutions to all but one instance of the Deroussi and Norre (2010) benchmark. Later, Ham (2020) suggested the only constraint programming approach for the TFJSSP, providing the exact smallest makespan solution for all the benchmark instances of (Deroussi and Norre, 2010). Ham obtained these results with IBM's CP Optimizer solver, which uses machine-learning techniques to find the best-combined method of large neighborhood and completion strategies. These exact solutions form an important base for researching job scheduling with integrated transportation. However, all reviewed studies so far always assume fixed transportation times, ignoring conflict avoidance during transportation.

AGV Control & Conflict-Free Routing. Routing and scheduling of AGVs is widely studied as a separate topic. Qiu et al. (2002) discuss two main aspects for the control of AGV systems: scheduling and routing. Scheduling deals with the dispatching of AGVs to a pickup or drop-off task (transfer). In the case of a TJSP, this usually is part of the optimal job scheduling problem as seen before. Routing, on the other hand, deals with the determination of a suitable route to perform the scheduled task. Important contributions in routing are the introduction of the Conflict-Free Routing Problem (CFRP) by Broadbent et al. (1985) and its solution proposed by Kim and Tanchoco (1991). The latter paper proposed a Dijkstra-based algorithm to determine the shortest conflict-free route for an AGV in a bi-directional flow path network. Where Kim and Tanchoco (1991) focused only on conflict-free path finding for vehicles in a grid, Krishnamurthy et al. (1993) generalized the problem to a scheduling context. They proposed a column generation-based optimization approach to reduce the overall makespan. The method assumes that the allocation of tasks to vehicles is already available. Much research followed on the study of AGV control and the CFRP. However, most work does not consider simultaneous job and AGV scheduling and assumes optimal job schedules to start with.

Job Scheduling with Conflict-Free Routing. The first important combination of scheduling and routing was investigated by Corréa et al. (2007). They proposed a hybrid method for dispatching and conflict-free routing of AGVs in FMSs by decomposing routing and scheduling of vehicles using the Logic-Based Benders Decomposition (Hooker and Ottosson, 2003; Hooker, 2024). The research, however, does not consider the scheduling of machines, but only the scheduling of AGVs. More decomposition-based approaches followed. Nishi et al. (2011) proposed a decomposition-based approach considering simultaneous machine scheduling and the CFRP in flexible *flow* shops (referred to as hybrid flow shops in (Nishi et al., 2011)) to minimize the total weighted tardiness. Riazi et al. (2019) and Riazi and Lennartson (2021) expanded the work of Corréa et al. (2007). They improved the decomposition method to deal with bigger routing graphs and solve the subproblems using constraint programming and a simple heuristic that provides good solutions quickly. Despite these contributions, decomposition-based optimization for *flexible job-shop* scheduling and CFR of AGVs remains unexplored.

Besides decomposition-based approaches, several (meta)heurstic approaches exist for integrated job scheduling with conflict-free routing problems. (Saidi-Mehrabad et al., 2015) proposed an approach

for solving a TJSSP and CFRP without using a problem decomposition but applying a two-stage ant colony algorithm to a mixed-integer linear program. The introduced problem however takes vehicles that escort a job until completion, i.e., all transports for a single job are done by the same vehicle and vehicles only become available again after a job is complete. At the same time, Umar et al. (2015) suggested a hybrid genetic algorithm to integrate conflict-free AGV control activities in the TJSSP. The iterative approach first considers machine scheduling, AGV scheduling, and path planning, then detecting and avoiding route conflicts. Another TJSSP approach considering CFR was introduced by Zhou and Lei (2021), optimizing both makespan and energy consumption using a special grey wolf optimization algorithm. Recently, Sun et al. (2023) proposed a hybrid genetic algorithm for the TJSSP that incorporates the Dijkstra algorithm based on time windows to deal with the CFRP. Although the paper discusses flexible operation allocation, the genetic algorithm and experimental assessment only cover fixed operation allocations. Lyu et al. (2019) were the first to consider the Conflict-Free-Transportation-constrained Flexible Job-Shop Scheduling Problem (CFTFJSSP) that integrates both the TFJSSP and CFRP. They propose a genetic algorithm for solving both the job and AGV scheduling, combining it with a time-window-based Dijkstra algorithm for finding the shortest conflict-free paths. Recently, this work was expanded in (Liu et al., 2023), replacing the genetic algorithm with a self-learning genetic algorithm and showing that this successfully enhances the effectiveness of the algorithm. None of these implementations consider exact solutions for comparison of algorithms and they use slight variations in their problem descriptions, making it hard to compare approaches.

This paper introduces an exact decomposition-based approach to the CFTFJSSP using the Logic-Based Benders Decomposition from (Hooker and Ottosson, 2003). It employs a constraint programming model similar to (Ham, 2020) for job scheduling and an integer linear programming model for vehicle routing, inspired by (Saidi-Mehrabad et al., 2015) and (Lyu et al., 2019). It uses a novel conflict-driven cut that uses conflict information from infeasible CFRP subproblem instances to efficiently prune TFJSSP master problem solutions that lead to essentially the same CFRP subproblem conflicts. The approach is shown to outperform the state-of-the-art heuristic approaches in solution quality, finding optimal solutions for most of the benchmarks available in the literature. The tools developed in this work and the benchmarks are made available to the community at large to facilitate experimental comparison in future research. The tools and models are set up to facilitate experimentation with different problem variants and decomposition approaches.

A summary of the most important papers relevant to the work in this paper is given in Table 1.

Publication	**Job scheduling**	**Flexible job-shop scheduling**	**Conflict-Free Routing**	**Exact Solution**
(Corréa et al., 2007)			✓	✓
(Riazi and Lennartson, 2021)			✓	✓
(Ham, 2020)	✓	✓		✓
(Nishi et al., 2011)	✓		✓	✓
(Saidi-Mehrabad et al., 2015)	✓		✓	
(Umar et al., 2015)	✓		✓	✓
(Lyu et al., 2019)	✓	✓	✓	
(Liu et al., 2023)	✓	✓	✓	
This research	✓	✓	✓	✓

Table 1: Summary of important related literature.

3. Problem Formulation

In essence, we can describe the CFTFJSSP as the combination of the TFJSSP and CFRP. The TFJSSP can be described as a flexible job-shop scheduling problem that incorporates so-called transfer actions for vehicles to move materials or partially finished products between machines (locations). Transfer takes place between all operations of the same job, but note that for consecutive operations at the same location, the transfer time is often zero. For each vehicle, a transfer can be seen as the

sequence of an empty transfer and a loaded transfer. The empty transfer denotes the action of picking up the product at the correct location, where a loaded transfer denotes the actual movement of the product from the pickup location to the drop-off location. Each job starts with a loading operation at the loading station and ends with an unloading operation at the unloading station.

The CFRP is another problem that checks whether feasible conflict-free paths exist for a given set of transfers. Each transfer has a predefined vehicle assignment, release time, and deadline. A graph—called the *routing network*—is defined to represent the layout of the FMS. The nodes of the graph denote locations to which a vehicle can travel. Vehicles can only transition from one node to another if a directed edge between these nodes is defined. Finding a feasible path means finding a sequence of steps within the release-time and deadline time bounds. Each step represents a transition from one node to another and takes exactly a single time unit. Conflict-freeness is guaranteed by making sure that two vehicles cannot step towards the same node or swap nodes at one given moment in time. As feasible transfers occur within given time windows, solutions are deadlock- and livelock-free.

We propose a general-purpose problem description that generalizes the two CFTFJSSP variants from literature (Lyu et al., 2019; Liu et al., 2023). The formulations in Lyu et al. (2019) and Liu et al. (2023) deviate in terms of job loading and unloading and in the supported routing networks. Hence, in this paper, we opt for a formulation in terms of sets, functions, and predicate logic, which provides a neutral and abstract way to present the problem, without favoring any specific modeling and solution approaches (mathematical programming, constraint programming, SAT/SMT solving, heuristic approaches). This avoids overly constraining the problem definition. For instance, Lyu et al. (2019) and Liu et al. (2023) allow only grid-like routing networks, where our formulation allows any directed graph as a routing network definition. Our approach allows to freely choose solution strategies, which is particularly important given our intention to propose a hybrid solution that leverages both Constraint Programming (CP) and Mathematical Programming (MP) techniques.

A formal definition of the conflict-free-transportation-constrained flexible job shop is as follows. The sets of natural and real numbers are denoted by $\mathbb{N}$ and $\mathbb{R}$, respectively. $\mathbb{N}_0$ denotes the natural numbers including 0. Superscripts can specify a range within a set of numbers; $^+$ specifies all positive numbers and inequality symbols such as $<, >, \leq, \geq$ can be used to specify specific bounds. $\mathbb{R}_0^+$ denotes the non-negative real numbers. The CFTFJSSP definition distinguishes input parameters that define a problem instance and derived parameters that can be derived from given input parameters (and are useful in precisely defining the problem and its solutions).

Definition 3.1 (Conflict-Free-Transportation-constrained Flexible Job Shop (CFTFJS))**.** *A CFTFJS is a tuple* $(N, E, V, L, \tau^{\Delta}, J, r_j (j \in J), \mathcal{P})$*, with the following elements:*

Input Parameters	
N	*The set of nodes in the routing network.*
$E \subseteq N^2$	*The set of edges connecting the nodes in the routing network, specifying valid steps from one node to another. We assume that, for all* $n \in N$*,* $(n, n) \notin E$*; that is, it is not possible to make a step without changing location.*
V :	*The set of AGVs.*
$L \subseteq N$	*The set of special locations, including the loading station* ls*, unloading station* us*, and machine locations.*
$\tau^{\Delta} \in \mathbb{N}$:	*(Uniform) duration of a single step.*
$J, r_j \in \mathbb{N}$	*The set of jobs, with every job* $j \in J$ *consisting of a sequence of* r_j+2 *operations* $\langle o_{j,0}, \ldots, o_{j,r_j+1} \rangle$ *where operation* $o_{j,k}$ *denotes the* k*-th operation of the* j*-th job. Operation* $o_{j,0}$ *denotes the loading of materials at the loading station* ls*, and operation* o_{j,r_j+1} *denotes the unloading of the finished product at the unloading station* us*.*

Derived Parameters

$O = \{o_{j,k} \mid j \in J, k \in \mathbb{N}_0^{\leq r_j+1}\}$	*The set of operations including the loading and unloading operations for each job.*
$FN = \{ls, us\}$	*The set of free nodes where more than one vehicle $v \in V$ may be present simultaneously, which are the loading and the unloading station.*

Input Parameters (Continued)

$\mathcal{P} : O \times L \hookrightarrow \mathbb{R}_0^+$	*Partial function denoting the processing time of an operation $o \in O$ at a location $l \in L$. A respective operation o cannot be performed at the corresponding location l when $\mathcal{P}(o, l)$ is undefined. For the operations $o_{j,0}$ and o_{j,r_j+1} of any job $j \in J$, $\mathcal{P}$ is only defined for the loading station ls and unloading station us, respectively. Processing times for operations other than $o_{j,0}$ and o_{j,r_j+1} are assumed to be non-zero.*

Derived Parameters (Continued)

$L_o = \{l \in L \mid \mathcal{P}(o, l) \text{ defined}\}$	*The set of locations that can perform operation o.*
$LT = \{lt_{j,k} \in T \mid j \in J, k \in \mathbb{N}_0^{\leq r_j}\}$	*The set of loaded transfers where $lt_{j,k}$ is a shorthand notation for the loaded transfer between $o_{j,k}$ and $o_{j,k+1}$. The duration of this transfer can be 0 if two successive operations are performed on the same machine.*
$ET = \{et_{j,k} \in T, \mid j \in J, k \in \mathbb{N}_0^{\leq r_j}\}$	*The set of empty transfers for each loaded transfer in LT, with $et_{j,k}$ forming the shorthand notation for the corresponding empty transfer for $lt_{j,k}$. The duration of this transfer can be 0 if a vehicle is already at the correct location.*
$T = LT \cup ET$	*The complete set of all transfers.*

The following assumptions hold:

- *Job ordering is arbitrary.*
- *Product materials are initially available at the loading station.*
- *The considered vehicles are a fleet of identical automated guided vehicles.*
- *All vehicles are available at the loading station at the start.*
- *Vehicles have unlimited range and do not need charging/maintenance.*
- *A vehicle must be present to support loading and unloading when the same machine processes any two consecutive operations of a job.*
- *Loading/unloading times are included in the transfer and/or processing times.*
- *Machines have sufficient buffer capacity.*
- *The routing network is a unidirectional graph.*
- *Only in the loading and unloading stations multiple vehicles can be present.*
- *Conflicts are situations where two vehicles are present in the same node (except ls and us) or switch nodes at the same time instant.*

Following this definition, we define schedules, i.e., solutions, for a transportation-constrained flexible job-shop instance. Decision variables are those variables that need to be solved to find a solution. Derived variables and functions are variables and functions whose values can be derived from the values of decision variables. Those derived variables and functions are used in formulating constraints.

Definition 3.2 (Schedule)**.** *A* schedule $\mathcal{S}_{ot}, \mathcal{C}_t, \alpha, \beta, z_t, \mathcal{N}, \mathcal{S}_s$ *for a CFTFJS describes the starting times, completion times, machine allocation, vehicle allocation, and steps with their starting times for all operations and transfers.*

Decision Variables	
$\mathcal{S}_{ot} : O \cup T \to \mathbb{R}_0^+$	*Complete function giving the starting time of an operation or transfer.*
$\mathcal{C}_t : T \to \mathbb{R}_0^+$	*Complete function giving the completion time of a transfer.*
$\alpha : O \to L$	*Complete function giving the allocation of an operation to a machine.*
$\beta : T \to V$	*Complete function giving the allocation of a transfer to a vehicle.*
$z_t \in \mathbb{N}_0$	*The number of steps in transfer $t \in T$.*

Derived Variables	
$S = \{s_{t,i} \mid t \in T, i \in \mathbb{N}^{\leq z_t}\}$	*The set of all steps, where $s_{t,i}$ is a shorthand notation for the i-th step in transfer $t \in T$.*

Decision Variables (Continued)	
$\mathcal{N} : S \to N$	*Complete function giving the destination node of all steps.*
$\mathcal{S}_s : S \to \mathbb{R}_0^+$	*Complete function giving the starting time of a step.*

Derived Functions	
$\mathcal{C}_o : O \to \mathbb{R}_0^+$	*Complete function giving the completion time of an operation, where for all $o \in O$, $\mathcal{C}_o(o) = \mathcal{S}(o) + \mathcal{P}(o, \alpha(o))$.*
$\mathcal{C}_s : S \to \mathbb{R}_0^+$	*Complete function giving the completion time of a step, where for all $s \in S$, $\mathcal{C}(s) = \mathcal{S}(s) + \tau^{\Delta}$.*
$Nxt_v : S \hookrightarrow S$	*For each vehicle $v \in V$, partial function Nxt_v defines for each step of v, except the last one, the next step of that vehicle. Given a vehicle allocation β and a sequencing of all transfers per vehicle (as enforced by the constraints specified below), Nxt_v is well defined.*
$\beta_s : S \to V$	*Complete function giving the allocation of steps to vehicles, derivable from vehicle allocation β.*
$\epsilon : T \to N$	*Function giving a transfer's destination node. Given machine and vehicle allocations α, β, ϵ is well defined.*
$\gamma : T \to \mathbb{R}_0^+$:	*Specification of* transfer release times*, with for all $t \in T$, $\gamma(t) = \mathcal{S}_{ot}(t)$.*
$\delta : T \to \mathbb{R}_0^+$:	*Specification of the* transfer deadlines*, where $\delta = \mathcal{C}_t$. The release-time and deadline functions are introduced for conceptual clarity.*
$\mathcal{T} : L \times L \to \mathbb{R}_0^+$	*Shortest-path transfer times between any two locations in the routing network. These can be derived using a shortest-path algorithm.*

The following constraints apply:

C1. *Every machine can execute at most one operation at a time:*

$$\forall o_x, o_y \in O : \alpha(o_x) = \alpha(o_y) \implies (o_x = o_y \vee \mathcal{C}_o(o_x) \leq \mathcal{S}_{ot}(o_y) \vee \mathcal{C}_o(o_y) \leq \mathcal{S}_{ot}(o_x)).$$

C2. *Every vehicle can perform one transfer at a time:*

$$\forall t_x, t_y \in T : \beta(t_x) = \beta(t_y) \implies (t_x = t_y \vee \mathcal{C}_t(t_x) \leq \mathcal{S}_{ot}(t_y) \vee \mathcal{C}_t(t_y) \leq \mathcal{S}_{ot}(t_x)).$$

C3. *Operations can only be executed on specified machines.*

$$\forall o \in O : \alpha(o) \in L_o$$

C4. *An operation can only start when its preceding loaded transfer is completed.*

$$\forall j \in J : \forall k \in \mathbb{N}_0^{\leq r_j} : \mathcal{C}_t(lt_{j,k}) \leq \mathcal{S}_{ot}(o_{j,k+1})$$

C5. *A loaded transfer can only start when its preceding operation is completed.*

$$\forall j \in J : \forall k \in \mathbb{N}_0^{\leq r_j} : \mathcal{C}_o(o_{j,k}) \leq \mathcal{S}_{ot}(lt_{j,k})$$

C6. *Empty and loaded transfers for a particular operation are always performed by the same vehicle:*

$$\forall j \in J : \forall k \in \mathbb{N}_0^{\leq r_j} : \beta(et_{j,k}) = \beta(lt_{j,k})$$

C7. *A loaded transfer is always preceded by an empty transfer*

$$\forall j \in J : \forall k \in \mathbb{N}_0^{\leq r_j} : \mathcal{C}_t(et_{j,k}) = \mathcal{S}_{ot}(lt_{j,k})$$

C8. *The completion time of a loaded transfer is bounded by the start of the transfer plus the transfer time to the delivery location.*

$$\forall j \in J : \forall k \in \mathbb{N}_0^{\leq r_j} : \mathcal{C}_t(lt_{j,k}) \geq \mathcal{S}_{ot}(lt_{j,k}) + \mathcal{T}(\alpha(o_{j,k}),\, \alpha(o_{j,k+1}))$$

C9. *The completion time of an empty transfer is bounded by the transfer time to the pickup location from the vehicle's prior location.*

$$\begin{aligned}\forall j_1, j_2 \in J : \forall k_1 \in \mathbb{N}_0^{\leq r_{j_1}} : \forall k_2 \in \mathbb{N}_0^{\leq r_{j_2}} : \beta(lt_{j_1,k_1}) = \beta(et_{j_2,k_2}) \wedge \mathcal{S}(lt_{j_1,k_1}) < \mathcal{S}(et_{j_2,k_2}) \\ \wedge\ \nexists j_3 \in J, k_3 \in \mathbb{N}_0^{\leq r_{j_3}} : \beta(lt_{j_1,k_1}) = \beta(lt_{j_3,k_3}) \wedge \mathcal{S}(lt_{j_1,k_1}) < \mathcal{S}(lt_{j_3,k_3}) < \mathcal{S}(et_{j_2,k_2}) \\ \implies \mathcal{S}(et_{j_2,k_2}) = \mathcal{C}_t(lt_{j_1,k_1}) \wedge \mathcal{C}_t(et_{j_2,k_2}) \geq \mathcal{S}(et_{j_2,k_2}) + \mathcal{T}(\alpha(o_{j_1,k_1+1}),\, \alpha(o_{j_2,k_2}))\end{aligned}$$

C10. *Steps are performed sequentially during each transfer.*

$$\forall t \in T : \forall i \in \mathbb{N}^{<z_t} : \mathcal{C}_s(s_{t,i}) \leq \mathcal{S}_s(s_{t,i+1})$$

C11. *Vehicles can only move to adjacent nodes.*

$$\forall v \in V : \forall s \in S : (\mathcal{N}(s), \mathcal{N}(Nxt_v(s)) \in E$$

C12. *The last step for a transfer is to the transfer's destination node.*

$$\forall t \in T : z_t > 0 \implies \mathcal{N}(s_{t,z_t}) = \epsilon(t)$$

C13. *A transfer should start with its first step not earlier than its release time.*

$$\forall t \in T : z_t > 0 \implies \gamma(t) \leq \mathcal{S}_s(s_{t,1})$$

C14. *A transfer should end no later than its deadline.*

$$\forall t \in T : z_t > 0 \implies \mathcal{C}_s(s_{t,z_t}) \leq \delta(t)$$

C15. *Only one vehicle can be present in a non-free node at a given time.*

$$\forall s_1, s_2 \in S : s_1 \neq s_2 \wedge \mathcal{N}(s_1) = \mathcal{N}(s_2) \wedge \mathcal{N}(s_1) \notin FN$$
$$\implies \mathcal{S}_s(Nxt_{\beta_s(s_1)}(s_1)) \leq \mathcal{S}_s(s_2) \vee \mathcal{S}_s(Nxt_{\beta_s(s_2)}(s_2)) \leq \mathcal{S}_s(s_1)$$

C16. *Vehicles cannot swap nodes at the same time.*

$$\forall s_1, s_2 \in S : s_1 \neq s_2 \wedge \mathcal{N}(s_1) = \mathcal{N}(Nxt_{\beta_s(s_2)}(s_2)) \wedge \mathcal{N}(Nxt_{\beta_s(s_1)}(s_1)) = \mathcal{N}(s_2)$$
$$\implies \mathcal{C}_s(s_1) \leq \mathcal{S}_s(s_2) \vee \mathcal{C}_s(s_2) \leq \mathcal{S}_s(s_1)$$

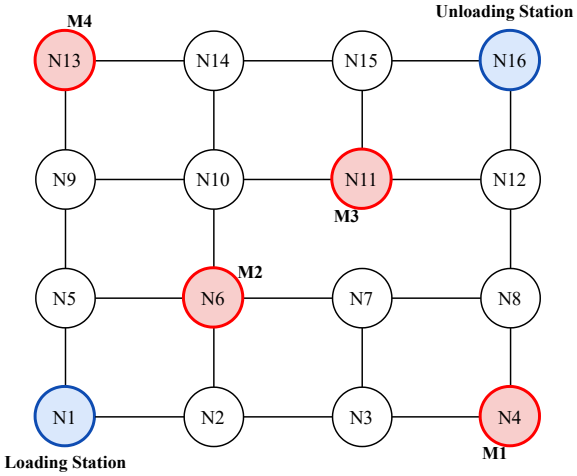
A schedule is feasible when all of these constraints are satisfied, and infeasible otherwise.

Lastly, the Conflict-Free-Transportation-constrained Flexible Job-Shop Scheduling Problem (CFTFJSSP) is a CFTFJS scheduling problem characterized by an optimization objective. In this research, we focus on a makespan objective, which is defined as:

Definition 3.3 (Makespan)**.** *The makespan C_{max} of a schedule $\mathcal{S}_{ot}$, $\mathcal{C}_t$, α, β, z_t, $\mathcal{N}$, $\mathcal{S}_s$ is the latest completion time of any transfer (including those for the unloading operation). Note that C_{max} only depends on decision variables $\mathcal{S}_{ot}$, through the derived function $\mathcal{C}_o$: $C_{max}(\mathcal{S}_{ot}) = \max_{j \in J}(\mathcal{C}_o(o_{j,r_j+1}))$.*

Definition 3.4 (Conflict-Free-Transportation-constrained Flexible Job-Shop Scheduling Problem (CFTFJSSP))**.** *The optimal solution for a CFTFJSSP is a feasible schedule $\mathcal{S}_{ot}$, $\mathcal{C}_t$, α, β, z_t, $\mathcal{N}$, $\mathcal{S}_s$ with the smallest possible makespan $C_{max}(\mathcal{S}_{ot})$.*

Example 3.1. Consider an FMS characterized by the routing network in Figure 1 (where the undirected edges are represent two directed edges) and the given job set. The jobs J, operations O, and processing times $\mathcal{P}$ for given machines m_j, loading station ls, and unloading station us defining a specific problem instance are given in the table to the right.



Job	Operation	ls	m_1	m_2	m_3	m_4	us
j_0	$o_{0,0}$	0	-	-	-	-	-
	$o_{0,1}$	-	2	2	-	-	-
	$o_{0,2}$	-	-	-	3	3	-
	$o_{0,3}$	-	-	-	-	-	0
j_1	$o_{1,0}$	0	-	-	-	-	-
	$o_{1,1}$	-	4	4	-	-	-
	$o_{1,2}$	-	-	-	2	2	-
	$o_{1,3}$	-	-	-	-	-	0

Figure 1: A CFTFJSSP instance using FMS layout 2 from (Lyu et al., 2019) and a simple illustrative job set.

The schedule shown in Figure 2 for the given problem instance is a shortest makespan solution. The first panel shows the operation scheduling, where `J0-00` denotes operation $o_{0,0}$, and so on. The second and third panels show the loaded and empty transfers, respectively, where `Tjk` in the first of these two panels denotes loaded transfer $lt_{j,k}$, and the black bars in the second of these two panels indicate empty transfers. The last panel shows the presence of a vehicle in a node, allowing us to verify the vehicle's chosen path. Together, the panels give the full schedule, consisting of the operation and transfer starting times $\mathcal{S}_{ot}$, the transfer completion times $\mathcal{C}_t$, the machine allocation α, vehicle allocation β, number of steps z_t, destination nodes of steps $\mathcal{N}$, and the starting times of steps $\mathcal{S}_s$. The makespan of the schedule is $C_{max} = 14$.

Consider for instance job j_0 in the schedule of Figure 2, whose handling can be tracked with the red markings. Job j_0 starts with a loading operation $o_{0,0}$ with duration 0 seconds in the loading station location ls (marking 1). The solution includes machine allocation $\alpha(o_{0,1}) = m_1$, meaning the loaded materials need to be moved to location m_1. This is achieved by the loaded transfer $lt_{0,0}$ (`T00` in the figure), which is allocated to vehicle v_0 in line with $\beta(lt_{0,0}) = v_0$ (marking 2). The vehicle takes steps from node 1 to node 4 (in de layout of Figure 1) in subsequent time units moving from the loading station ls to machine m_1 (marking 3 in Figure 2). Upon completing the transfer, operation $o_{0,1}$ (`J0_O1` in the figure) can start taking 2 time units in line with specification $\mathcal{P}(o_{0,1}, m_1) = 2$ from the table above (marking 4). Next are the second loaded transfer $lt_{0,1}$ (`T01` in the figure with vehicle allocation $\beta(lt_{0,1}) = v_0$) and operation $o_{0,2}$ (`J0_O2` in the figure with machine allocation $\alpha(o_{0,2}) = m_3$). After operation $o_{0,2}$, the product needs to be delivered to the unloading station location us. The respective loaded transfer $lt_{0,2}$ (`T02`) cannot start immediately after completion of $o_{0,2}$ (marking 5). Up to this point, all transfers for job j_0 were performed by v_0, and v_0 exclusively served the transfers within job j_0. However, $lt_{0,2}$ is allocated to v_1, which was previously performing transfer $lt_{1,1}$ (`T11`) for job j_1. Hence, $lt_{0,2}$ needs to wait for vehicle v_1 to arrive at location m_3 by completing the empty transfer $et_{0,2}$ (the bottom-right black bar in Figure 2, marking 6). The steps of this empty transfer start in location m_4 at node 13 (see Figure 1) and go to location m_3 at node 11. Hereafter, job 0 is completed by performing loaded transfer $lt_{0,2}$ (`T02`) and the unloading operation $o_{0,3}$ (`J0_O3`). (Note that switching the vehicles between the jobs is not necessary for obtaining an optimal schedule. In fact, not switching the vehicles would allow to complete job j_0 one time unit earlier. This does not affect the makespan of the overall schedule though.) ■

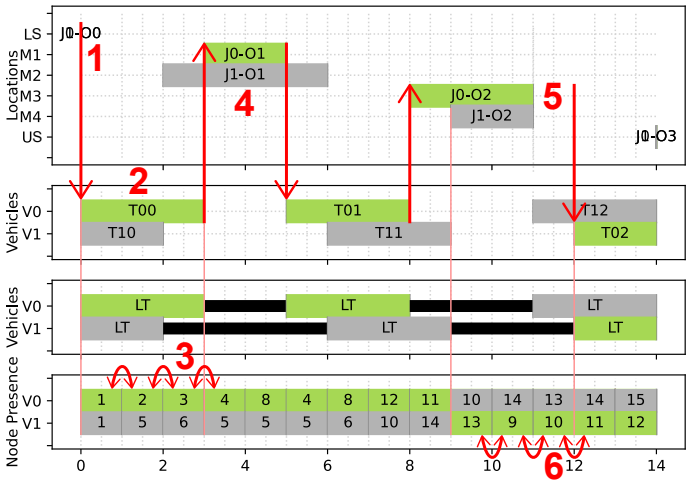


Figure 2: Optimal schedule for the illustrative CFTFJSSP instance.

4. Decomposition-based Solution – Models

Given the problem statement, we present our solution for the CFTFJSSP. The approach is based on the Logic-Based Benders Decomposition (LBBD). This approach decomposes the complex CFTFJSSP into smaller, less complex subproblems. An added advantage of this approach is that the decomposed problems can be solved with different techniques that best suit the subproblem at hand.

4.1. Logic-Based Benders Decomposition

The LBBD (Hooker and Ottosson, 2003) is a powerful iterative method for solving complex optimization problems. The general idea of the LBBD is to fix a part of the decision variable value assignments *(trial values)* in such a way that the remaining problem (subproblem) becomes easier to solve. Smart evaluation of this subproblem's solution enables the selection of new, better trial values for a next iteration of the procedure. When done properly, the procedure converges towards finding the optimal solution to the overall problem.

The optimization problem to be solved using the LBBD is referred to as the *overall problem*. The optimization problem for which trial values are sought is the *master problem*. The optimization problem that remains after assigning the trial values is called the *subproblem*. The power of the LBBD is primarily related to the so-called *Benders cuts*. Benders cuts are constraints added in each iteration of the LBBD to the master problem. The goal of these Benders cuts is to guide the selection of the next set of trial values towards the optimal solution of the overall problem by using information on the found subproblem solution.

4.2. Decomposition Setup

The CFTFJSSP of Definition 3.4 is our overall problem. The CFTFJSSP naturally decomposes in the earlier mentioned TFJSSP as the master problem and the CFRP as the subproblem. We can decompose the decision variables of a CFTFJSSP instance as defined in Definition 3.2 into two vectors. Vector $\boldsymbol{x} = [\mathcal{S}_{ot}, \mathcal{C}_t, \alpha, \beta]$ contains the starting times of operations and transfers, completion times of transfers, the machine allocation, and vehicle allocation decision variables needed to construct the TFJSSP master problem. Additionally, vector $\boldsymbol{y} = [z_t, \mathcal{N}, \mathcal{S}_s]$ contains the number of steps, the respective step destinations, and step timing decision variables that construct the CFRP subproblem. This gives us the following definitions.

Definition 4.1 (Decomposed CFTFJSSP Master Problem)**.** *The TFJSSP master problem for the decomposed CFTFJSSP is the problem with decision variables* $\boldsymbol{x} = [\mathcal{S}_{ot}, \mathcal{C}_t, \alpha, \beta]$ *and a constraint set* $C(\boldsymbol{x})$ *that includes the constraints* ***C1.-C9.*** *(as defined in Definition 3.2) in which only the decision variables of* $\boldsymbol{x}$ *occur.*

Definition 4.2 (Decomposed CFTFJSSP Subproblem)**.** *The CFRP subproblem for the decomposed CFTFJSSP is the problem with decision variables* $\boldsymbol{y} = [z_t, \mathcal{N}, \mathcal{S}_s]$ *and the constraint sets* $C(\boldsymbol{y})$ *with constraints* ***C10.-C12.*** *(as defined in Definition 3.2) in which only the decision variables of* $\boldsymbol{y}$ *occur, and* $C(\boldsymbol{x}, \boldsymbol{y})$ *with constraints* ***C13.-C16.*** *(as defined in Definition 3.2) in which both the decision variables of* $\boldsymbol{x}$ *and* $\boldsymbol{y}$ *occur.*

The goal is to optimize the CFTFJSSP for makespan as specified by Definition 3.3. The makespan objective only depends on decision variables $\mathcal{S}_{ot}$, showing that the subproblem is a feasibility problem. The decomposition process operates as follows. The master problem, the TFJSSP instance at hand, is optimized for makespan. Machines and vehicles are allocated to operations and transfers and the starting and completion times for all operations and transfers are determined by solving $C(\boldsymbol{x})$. The resulting solution defines the trial values $\boldsymbol{x}^k$ in iteration k of the process. Hereafter, the process continues to solve the CFRP subproblem by solving $C(\boldsymbol{y})$ and $C(\boldsymbol{x}, \boldsymbol{y})$ where the trial values $\boldsymbol{x}^k$ are substituted for $\boldsymbol{x}$ in $C(\boldsymbol{x}, \boldsymbol{y})$. The aim of the subproblem is then to find a feasible vehicle routing assignment based on these trial values. If a feasible vehicle routing assignment is found, the valuation of the decision variables $\boldsymbol{x}$ from the master problem and $\boldsymbol{y}$ of the subproblem together form the optimal solution for the overall problem.

4.3. Constraint Programming Model for the Master Problem

To find an optimal solution to the TFJSSP master problem, a Constraint Programming (CP) model was created. CP has been shown to be a powerful mechanism in obtaining optimal solutions to scheduling problems, outperforming mathematical programming approaches such as seen in (Laborie et al., 2018; Ham, 2020; Lunardi et al., 2020; Ham, 2021; Eigbe et al., 2023). The CP model used for our TFJSSP master problem is mostly similar to the specification by Ham (2020). The CP model is implemented using the IBM ILOG CP Optimizer (CP Optimizer) solver.

Input Parameters

$L, J, r_j (j \in J), V, \mathcal{P}, \mathcal{T}$	As in Definition 3.1 and Definition 3.2. Also derived notations are adopted.

Derived Parameters

$\mathbf{T}$	A matrix specifying transfer times between each possible pair of loaded transfers in line with transfer time function $\mathcal{T}$. For any pair of transfers t_1 from location l_1 to location l_1' and t_2 from location l_2 to location l_2', $\mathbf{T}[(l_1, l_1'), (l_2, l_2')] = \mathcal{T}(l_1', l_2)$.

Decision Variables

$operations_{o_{j,k}}$	Interval variable for the k-th operation of job j.
$machine\_operation\_options_{o_{j,k},l}$	Optional interval variable with processing time $\mathcal{P}(o_{j,k}, l)$ for each location l able to perform the k-th operation of job j.
$transfers_{o_{j,k}}$	Interval variable for each loaded transfer $lt_{j,k}$.
$vehicle\_transfer\_options_{o_{j,k},v,sl,dl}$	Optional interval variable for each vehicle v performing the loaded transfer following the k-th operation of job j, by moving from source location $sl \in L_{o_{j,k}}$ to destination location $dl \in L_{o_{j,k+1}}$.
$transfer\_sequence_v$	Sequence variable of loaded transfers for each vehicle v.

The program is then constructed accordingly:

$\textbf{\textit{alternative}}(operations_o, [machine\_operation\_options_{o,l}])$	For all $o \in O$ with $l \in L_o$	(1)
$\textbf{\textit{alternative}}(transfers_o, [vehicle\_transfer\_options_{o,v,sl,dl}])$	For all $o \in O, v \in V$, source location $sl \in L_{o_{j,k}}$ and destination location $dl \in L_{o_{j,k+1}}$	(2)
$\textbf{\textit{no\_overlap}}(machine\_operation\_options_{o,l})$	For all $l \in L$	(3)
$\textbf{\textit{no\_overlap}}(transfer\_sequence_v, \mathbf{T})$	For all $v \in V$	(4)
$\textbf{\textit{end\_before\_start}}(transfers_{o_{j,k}}, operations_{o_{j,k+1}})$	For all $j \in J$ and $k \in \mathbb{N}_0^{\leq r_j}$	(5)
$\textbf{\textit{end\_before\_start}}(operation_{o_{j,k}}, transfers_{o_{j,k}})$	For all $j \in J$ and $k \in \mathbb{N}_0^{\leq r_j}$	(6)
$\textbf{\textit{presence\_of}}(vehicle\_transfer\_options_{o_{j,k},v,sl,dl}) \implies$ $\textbf{\textit{presence\_of}}(machine\_operation\_options_{o_{j,k},sl}) \wedge$ $\textbf{\textit{presence\_of}}(machine\_operation\_options_{o_{j,k+1},dl})$	For all $j \in J, k \in \mathbb{N}_0^{\leq r_j}, v \in V$, $sl \in L_{o_{j,k}}$ and $dl \in L_{o_{j,k+1}}$	(7)
$\textbf{\textit{size\_of}}(vehicle\_transfer\_options_{o_{j,k},v,sl,dl}) \geq \mathcal{T}(sl, dl)$	For all $j \in J, k \in \mathbb{N}_0^{\leq r_j}, v \in V$, $sl \in L_{o_{j,k}}$ and $dl \in L_{o_{j,k+1}}$	(8)

Equation 1 constrains that each operation is executed on one of the available machines. Equation 2 specifies that the loaded transfer for an operation has multiple alternative starting and destination locations and can be performed by any available vehicle. Equation 3 and Equation 4 constrain that machines and vehicles can only perform one operation or loaded transfer at a time. Furthermore, Equation 4 ensures that empty transfers are performed correctly, through the transfer time matrix $\mathbf{T}$. Equation 5 and Equation 6 specify the precedence constraints between operations and loaded transfers. Equation 7 ensures that the start and destination locations of a loaded transfer match with the chosen machines for the relevant operations. Lastly, Equation 8 ensures that a loaded transfer takes up a minimally specified time. Table 2 shows how the CP constraints match with the master problem constraints mentioned in Definition 4.1.

As the TFJSSP is a common studied problem in the literature, we evaluated the correctness of our CP formulation in finding optimal solutions by running the Deroussi and Norre (2010) benchmark. We compare our obtained makespan values with the optimal makespan values found by the state-of-the-art mathematical programming model of Homayouni and Fontes (2021) and the CP2 model as described in Ham (2020). Homayouni and Fontes (2021) obtained optimal solutions for nine instances, Ham (2020) was able to prove optimality for all ten instances. Our CP model obtained the same makespan as these references for the ten instances of the Deroussi and Norre (2010) benchmark.

Definition constraints	CP constraints
C1.	Equation 3
C2.	Equation 4
C3.	Equation 1 and the definition of $machine\_operation\_options_{o_{j,k},l}$
C4.	Equation 5
C5.	Equation 6
C6., **C7.**	Empty transfers are captured as setup times between loaded transfers and are therefore automatically coupled to and preceding the appropriate loaded transfer
C8.	Equation 8
C9.	Equation 2, Equation 7, and the transition matrix in Equation 4

Table 2: Connections between the master problem constraints and the CP model

4.4. Mathematical Programming Model for the Subproblem

For the CFRP subproblem, a Mathematical Programming (MP) model is used. Generally, the literature shows two mathematical programming approaches to solving the CFRP. The first approach considers a binary variable for a vehicle traversing from one node to another at a certain time as seen in Krishnamurthy et al. (1993); Nishi et al. (2005); Corréa et al. (2007); Liu et al. (2023). The second approach considers whether a vehicle is present in a node at a certain time, such as in Saidi-Mehrabad et al. (2015); Lyu et al. (2019); Sun et al. (2023). An MP, specifically an Integer Linear Programming (ILP), model following the second approach results in the best solution (van Os, 2024). The model is implemented using the IBM ILOG CPLEX Optimizer (CPLEX) solver and is defined as follows:

Input Parameters

$N, E, V, \tau^{\Delta}, FN, \delta, \epsilon$	As in Definition 3.1 and Definition 3.2. Also derived notations are adopted.

Derived Parameters

$A_n = \{a \in N \mid (n, a) \in E\} \cup \{n\}$	The adjacent nodes for each node $n \in N$.
$H = \max_{t \in T}(\delta(t))$	Time horizon, being the maximum deadline of all transfers $t \in T$.
$y_v \in \mathbb{N}_0, T_v = \langle t_{v,1}, \ldots, t_{v,y_v} \rangle$	The sequence of transfers $t_{v,k}$ for vehicle $v \in V$ with y_v denoting the number of transfers for vehicle $v \in V$. These can be derived from the trial values $\boldsymbol{x}^k$.
$\mathbb{T} \subseteq \mathbb{N}_0$	The discretized time domain.

Decision Variables

$$x^{\tau}_{v,n} = \begin{cases} 1 & \text{if AGV } v \text{ is present in node } n \text{ at time } \tau \in \mathbb{T}^{\leq H} \\ 0 & \text{otherwise} \end{cases}$$

The program is constructed accordingly:

$$\sum_{n \in N} x^{\tau}_{v,n} = 1, \qquad \forall \tau \in \mathbb{T}^{\leq H}, \forall v \in V \tag{9}$$

$$x^{\tau}_{v,n_x} = \sum_{n_y \in N \mid n_x \in A_{n_y}} x^{\tau+\tau^{\Delta}}_{v,n_y}, \qquad \forall \tau \in \mathbb{T}^{\leq H-\tau^{\Delta}}, \forall v \in V, \forall n_x \in N \tag{10}$$

$$x^{0}_{v,ls} = 1, \qquad \forall v \in V \tag{11}$$

$$x^{\delta(t_{v,k})}_{v,\epsilon(t_{v,k})} = 1, \qquad \forall v \in V, \forall k \in \mathbb{N}^{\leq y_v} \tag{12}$$

$$\sum_{v \in V} x^{\tau}_{v,n} \leq 1, \qquad \forall \tau \in \mathbb{T}^{\leq H}, \forall v \in V, \forall n \in N \setminus FN \tag{13}$$

$$x^{\tau+\tau^{\Delta}}_{v_x,n_x} + x^{\tau+\tau^{\Delta}}_{v_y,n_y} \leq 3 - (x^{\tau}_{v_x,n_y} + x^{\tau}_{v_y,n_x}), \qquad \forall \tau \in \mathbb{T}^{\leq H-\tau^{\Delta}}, \forall v_x, v_y \in V, \forall n_x, n_y \in N \text{ with } n_x \in A_{n_y}, n_y \in A_{n_x} \tag{14}$$

Equation 9 specifies that a vehicle is present at a node at all times. Equation 10 requires a vehicle to travel to an adjacent node or stay in its current position at each time step. Equation 11 ensures the first step is always from the loading station. Equation 12 ensures all destination nodes must be visited at the respective transfer's deadline. Note that the related constraint **C14.** allows for early arrivals. Such early arrivals are also possible in the MP, as a vehicle can simply stay in the destination for one or more time steps after its arrival. Equation 13 restricts that only one vehicle can simultaneously be present in a node, excluding the free nodes. Equation 14 restricts that vehicles cannot swap nodes. Table 3 shows how the MP model relates to the subproblem constraints of Definition 4.2.

Definition constraints	**MP constraints**
Input assumption: first transfer starts in ls	Equation 11
C10.	As the model uses time steps instead of physical steps, sequencing is automatically guaranteed by Equation 9.
C11.	Equation 10
C12.	Equation 12
C13.	As the model uses time steps, and assumes for all $v \in V$ and $k \in \mathbb{N}^{<y_v}$, $\delta(t_{v,k}) = \gamma(t_{v,k+1})$, this is automatically guaranteed.
C14.	Equation 12
C15.	Equation 13
C16.	Equation 14

Table 3: Connections between the subproblem constraints and the MP model.

5. Conflict-Driven Benders Cuts

The efficiency of any LBBD-based solution is characterized by the strength of the respective Benders cuts. Each iteration of the procedure, these cuts refine the master problem to facilitate convergence towards an optimal solution. We present two types of cuts that help us speed up to convergence of the LBBD process. Both cuts are so-called *feasibility cuts* (Hooker, 2024), meaning that they aim to refine the TFJSSP master problem specification such that it does not produce infeasible CFRP subproblems.

5.1. Simple Feasibility Cuts

The simplest feasibility cut is one that simply disregards trial values that produced an infeasible result in past iterations. Such a cut—often called a *nogood* cut—is very weak in terms of how it affects the convergence rate towards the optimal solution.

The nogood cut can be strengthened by evaluating the subproblem at hand. In the case of the CFTFJSSP, we know an infeasible solution occurs when no feasible routing assignment within the defined transfers' release times and deadlines can be found. With this knowledge, we can derive that an infeasible set of trial values $\boldsymbol{x}^k$ might be changed in a feasible set if a certain transfer time is increased. This means that either the release time is earlier or the deadline is later. This is what Corréa et al. (2007) noticed for their combined scheduling and conflict-free routing problem, which can be seen as a simplified CFTJSSP. Hence, they introduced a feasibility cut that either completely disregards the trial values $\boldsymbol{x}^k$ or forces to change some variables in $\boldsymbol{x}^k$. Later, Riazi and Lennartson (2021) applied these same cuts for a similar problem variant.

We can apply the same concepts as used by Corréa et al. (2007); Riazi and Lennartson (2021). Hence, a feasibility cut in iteration k for the CFTFJSSP is given as the *disjunction* of two options:

- Find a completely new arbitrary solution (nogood cut)

$$F_1^k(\boldsymbol{x}) := \boldsymbol{x} \neq \boldsymbol{x}^k \tag{15}$$

- Fix allocations and operation and transfer ordering but increase at least one transfer in length

$$F_2^k(\boldsymbol{x}) := \alpha = \alpha^k \wedge (\forall o_1, o_2 \in O : \mathcal{S}^k(o_1) \leq \mathcal{S}^k(o_2) \implies \mathcal{S}(o_1) \leq \mathcal{S}(o_2)) \wedge \\ (\forall t_1, t_2 \in T : \mathcal{S}^k(t_1) \leq \mathcal{S}^k(t_2) \implies \mathcal{S}(t_1) \leq \mathcal{S}(t_2))) \wedge \\ \Big(\exists t \in T : \mathcal{C}(t) - \mathcal{S}(t) > \mathcal{C}^k(t) - \mathcal{S}^k(t)\Big) \tag{16}$$

The cut is then defined as:

$$F_3^k(\boldsymbol{x}) := F_1^k(\boldsymbol{x}) \vee F_2^k(\boldsymbol{x}) \tag{17}$$

The resulting cut provides a suggestion of how an infeasible $\boldsymbol{x}^k$ can be transformed to result in a feasible $\boldsymbol{x}^{k+1}$. While this makes the cut stronger than the nogood cut $F_1^k(\boldsymbol{x})$, it still is limited in its impact on reducing the algorithm's convergence rate.

5.2. Conflict-Driven Feasibility Cuts

Simple feasibility cuts are merely based on the observation that a subproblem has an infeasible schedule. The cuts described in the previous section only suggest that some variables should change to obtain a feasible schedule for the overall problem. However, there is no guarantee that the changed variables are related to the infeasibility of the subproblem schedule. Hence, the conflict may reoccur in the next iteration. Our novel *conflict-driven feasibility cuts*, on the other hand, use extra information from an infeasible subproblem schedule to identify which variables caused the infeasibility in the first place. Hence, rather than changing any variable, such a cut can directly enforce changes to prevent the conflict from reoccurring.

5.2.1. Weak Conflict-Driven Feasibility Cuts for the CFTFJSSP

In the CFTFJSSP, conflicts occur when two or more vehicles visit the same node (presence conflict) or swap nodes simultaneously (swap conflict). The conflict can be resolved by avoiding the specific steps that led to it. Our CFTFJSSP implementation has the capability to provide information on the steps that lead to a conflict. We can eliminate solutions with the same conflict structure by applying a cut directed at these steps.

To illustrate these conflict-driven feasibility cuts, we first introduce a useful transfer notation:

$t_{o,v,sl,dl,s,c}$ Variable denoting a transfer for operation $o \in O$, performed by vehicle v, which starts from source location $sl \in L$, moves to the destination location $dl \in L$, with starting time $s \in \mathbb{R}_0^+$ and completion time $c \in \mathbb{R}_0^+$.

Example 5.1. Consider the example CFTFJSSP instance with job set 3, three vehicles, and routing network layout 2 as presented by Lyu et al. (2019). The FMS layout of this CFTFJSSP instance was already shown in Figure 1. Solving the TFJSSP master problem in the first iteration results in trial values $\boldsymbol{x}^1$. A fragment of a possible schedule corresponding to $\boldsymbol{x}^1$ can be seen in Figure 3.

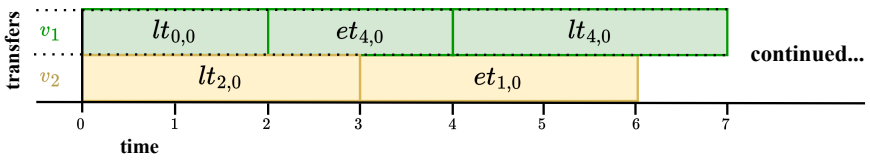


Figure 3: Gantt chart showing the transfer schedule results from iteration 1.

Using the notations introduced earlier, $lt_{0,0} = t_{o_{0,0},1,ls,m_2,0,2}$, $et_{4,0} = t_{o_{4,0},1,m_2,ls,2,4}$, $lt_{4,0} = t_{o_{4,0},1,ls,m_4,4,7}$, $lt_{2,0} = t_{o_{2,0},2,ls,m_4,0,3}$ and $et_{1,0} = t_{o_{1,0},2,m_4,ls,3,6}$, where $ls, m_2, m_4 \in L$ denote the loading station, machine 2 and machine 4, respectively. Note that $et_{0,0} = t_{o_{0,0},ls,ls,0,0}$ and $et_{2,0} = t_{o_{2,0},ls,ls,0,0}$ that are also part of schedule $\boldsymbol{x}^1$ are not shown in the figure as their respective transfer times are 0.

The CFR subproblem now needs to find a routing schedule $\boldsymbol{y}$ for the vehicles for the given trial values $\boldsymbol{x}^1$. The release-time and deadline time bounds limit the number of possible routing assignments—the sequences of steps and their destination nodes $\mathcal{N}$ through the routing network—as can be seen in Table 4. Transfers $lt_{4,0}, lt_{2,0}, et_{1,0}$ have only one routing assignment option.

Transfer	Release time	Deadline	Start Node	End Node	Routing Assignments
$lt_{0,0} = t_{o_{0,0},1,1,6,0,2}$	0	2	1	6	$1 \to 2 \to 6$ $1 \to 5 \to 6$
$et_{4,0} = t_{o_{4,0},1,6,1,2,4}$	2	4	6	1	$6 \to 2 \to 1$ $6 \to 5 \to 1$
$lt_{4,0} = t_{o_{4,0},1,1,13,4,7}$	4	7	1	13	$1 \to 5 \to 9 \to 13$
$lt_{2,0} = t_{o_{2,0},2,1,13,0,3}$	0	3	1	13	$1 \to 5 \to 9 \to 13$
$et_{1,0} = t_{o_{1,0},2,13,1,3,6}$	3	6	13	1	$13 \to 5 \to 9 \to 1$

Table 4: Transfer parameters for the transfers from Figure 3.

Consider a *possible, partial* assignment of steps for the considered transfers shown in Figure 4, where steps for vehicle 1 are shown in green and steps for vehicle 2 in yellow. The steps for $lt_{0,0}$ are omitted and the routing assignment for $et_{4,0}$ is chosen as $6 \to 2 \to 1$. The CFR subproblem is infeasible as an unavoidable presence conflict occurs in node 5. The conflict can be avoided if we prevent these *conflicting steps* and their respective timing from reoccurring. This can only be done by avoiding the related transfers in the master problem. ■

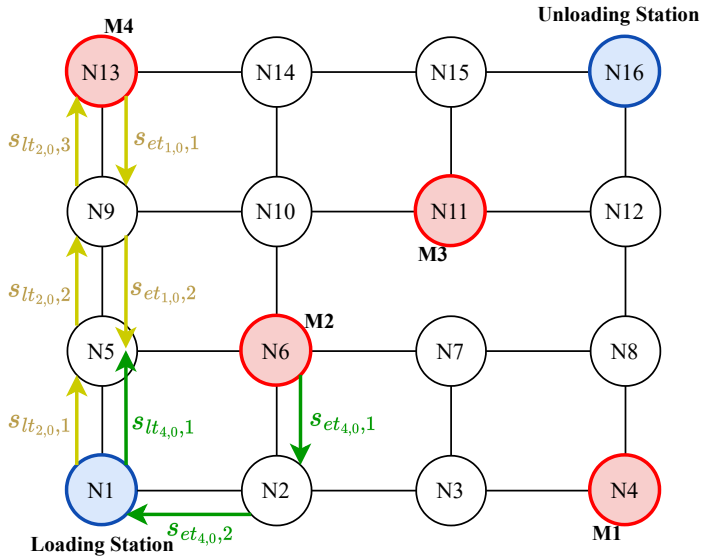


Figure 4: Routing network graph showing the steps in the subproblem tried at iteration 1. These steps unavoidably lead to infeasibility.

Let Q^k be a set of *conflict steps* obtained in iteration k of the subproblem. Set Q^k shows us the steps to avoid when solving the subproblem with the given input constraints. Since our cut needs to be phrased in terms of master problem decision variables $\boldsymbol{x}$, we specify the set of conflicting *transfers* that caused the subproblem conflict in iteration k.

$$QT^k = \{t \in T \,|\, \exists i \in \mathbb{N}^{\leq z_t} : s_{t,i} \in Q^k\} \tag{18}$$

Example. Reconsidering Example 5.1, we can define the set of conflicting transfers as:

$$QT^1 = \{t_{o_{4,0},1,m_2,ls,2,4}, t_{o_{4,0},1,ls,m_4,4,7}, t_{o_{2,0},2,ls,m_4,0,3}, t_{o_{1,0},2,m_4,ls,3,6}\} \tag{19}$$

To ensure the conflict does not resurface, we must guarantee a way to exclude this set QT^1 from reoccurring when solving a master problem instance. ■

We can prohibit the occurrence of a specific set of conflicting transfers QT^k for this vehicle assignment in a conflict-driven feasibility cut:

$$F_4^k(\boldsymbol{x}) := QT^k \not\subseteq T \tag{20}$$

5.2.2. Strengthened Conflict-Driven Feasibility Cut

While the cut from Equation 20 makes sure that at least one of the transfers in set QT^k cannot reoccur, we can still find a similar set QT^{k+1} in the next iteration that causes an identical conflict, namely by having transfers between the same set of locations with the same release times and deadlines, but for different vehicles and/or operations. By strengthening the cut, we can exclude those similar conflict situations that can be derived from the set QT^k.

Example. The vehicles in a CFTFJS as defined in Definition 3.4 are all identical vehicles and all start at the exact same location $ls \in L$. Hence, when allocating conflicting transfers in QT^k to different vehicles, the conflict in the subproblem persists. For instance, when assigning $lt_{2,0}$ and $et_{1,0}$ as shown in Figure 3 to vehicle v_3 instead of v_2, the same conflict occurs. With the same vehicle assignment, locations, and starting times and completion times but different operations, the conflict also re-occurs. For instance, the set of transfers

$$\{t_{\boldsymbol{o_{3,0}},1,m_2,ls,2,4}, t_{\boldsymbol{o_{3,0}},1,ls,m_4,4,7}, t_{o_{2,0},\boldsymbol{0},ls,m_4,0,3}, t_{o_{1,0},\boldsymbol{0},m_4,ls,3,6}\} \tag{21}$$

where operation 0 of job 3 takes the role of operation 0 of job 4 and vehicle 0 takes the role of vehicle 2, leads to essentially the same subproblem conflict. ■

Let $f_o : O \leftrightarrow O$ and $f_v : V \leftrightarrow V$ denote bijections on the set of operations O and the set of vehicles V, respectively. Functions f_o and f_v may be seen as renamings of operations and vehicles. For a given f_o, f_v and QT^k, a similar (isomporhic) set of conflicting transfers SQT^k can be derived:

$$SQT^k(f_o, f_v) = \{t_{f_o(o),f_v(v),sl,dl,s,c} \mid t_{o,v,sl,dl,s,c} \in QT^k\} \tag{22}$$

A strengthened feasibility cut excludes all sets of conflicting transfers that are isomorphic to the original set QT^k.

$$F_5^k(\boldsymbol{x}) := (\forall f_o \in O \leftrightarrow O : \forall f_v \in V \leftrightarrow V : SQT^k(f_o, f_v) \not\subseteq T) \tag{23}$$

Note that this cut includes set QT^k, since the identity functions on O and V are also bijections.

The presented decomposition setup in Section 4.2 and the valid Benders cuts presented in this section form an exact solution technique for solving the CFTFJSSP. For the remainder of this paper, we refer to this exact solution as the *LBBD-based CFTFJSSP*.

5.2.3. Time-Shifted Conflict-Driven Feasibility Cut

As a final remark, one may also observe that besides different vehicles and operations, a similar conflict can be found anywhere in time as long as their relative starting times and completion times are shifted by an exact amount.

Example. Reconsider the set of transfers seen in Equation 21. Let us assume all transfers are shifted by time Δ. This leads to a similar set of transfers

$$\{t_{o_{3,0},1,m_2,ls,\boldsymbol{2+\Delta},\boldsymbol{4+\Delta}}, t_{o_{3,0},1,ls,m_4,\boldsymbol{4+\Delta},\boldsymbol{7+\Delta}}, t_{o_{2,0},3,ls,m_4,\boldsymbol{0+\Delta},\boldsymbol{3+\Delta}}, t_{o_{1,0},3,m_4,ls,\boldsymbol{3+\Delta},\boldsymbol{6+\Delta}}\} \tag{24}$$

that once more leads to essentially the same subproblem conflict. ■

We have not found a means to implement this in our CP solution for the TFJSSP master problem. Hence, we leave it for future work to implement this strengthened cut into the CFTFJSSP implementation.

6. Implementation

We proceed with the implementation of the LBBD-based CFTFJSSP. We outline two ways to strengthen the master problem formulation to effectively exclude possible infeasible subproblem instances. This is followed by an overview of the complete LBBD-based CFTFJSSP procedure. All model implementations and tools for running the LBBB-based CFTFJSSP are publicly available at: https://github.com/TUE-EE-ES/TJSP-toolset/releases/tag/v1.2.0.

6.1. Strengthening the Master Problem

While the master problem model in Section 4.3 covers all the TFJSSP constraints, it does not consider whether the resulting trial values lead to a feasible subproblem result. Determining which combinations of master problem transfer starting and completion times lead to a routing conflict is complicated. This motivates the use of an LBBD. However, there are some distinguishable situations that lead to conflicts in any CFTFJSSP. It is beneficial to exclude such situations in the specification of the master problem in the LBBD. For the LBBD-based CFTFJSSP, we determined two constraint additions to strengthen the master problem such that it can eliminate these conflicting situations in generating subproblem trial values.

The first master problem strengthening avoid collisions between transfers with the same start and end locations. It is not straightforward to formulate constraints to avoid conflicts between vehicles in any arbitrary location. But it is possible to exclude specific cases, in particular, the cases where multiple transfers end at the same location at the same time and where transfers for different operations and vehicles complete and start at the same location at the same time:

$$\begin{aligned} &\textbf{\textit{end\_of}}(vehicle\_transfer\_options_{o_1,v_1,sl_1,dl}) \neq \\ &\quad \textbf{\textit{end\_of}}(vehicle\_transfer\_options_{o_2,v_2,sl_2,dl}) \end{aligned} \quad \begin{aligned} &\text{For all } o_1, o_2 \in O \text{ with } o_1 \neq o_2, \\ &\quad v_1, v_2 \in V, \text{ with } v_1 \neq v_2, \\ &\quad sl_1 \in L_{o_1}, sl_2 \in L_{o_2} \text{ and } dl \in L_{o_1} \setminus FN \end{aligned} \tag{25}$$

$$\begin{aligned} &\textbf{\textit{start\_of}}(vehicle\_transfer\_options_{o_1,v_1,sl_1,dl_1}) \neq \\ &\quad \textbf{\textit{end\_of}}(vehicle\_transfer\_options_{o_2,v_2,sl_2,dl_2}) \end{aligned} \quad \begin{aligned} &\text{For all } o_1, o_2 \in O \text{ with } o_1 \neq o_2, \\ &\quad v_1, v_2 \in V, \text{ with } v_1 \neq v_2, \text{ and} \\ &\quad sl_1, dl_1 \in L_{o_1}, sl_2, dl_2 \in L_{o_2} \text{ with} \\ &\quad sl_1 = dl_2, dl_2 \notin FN \end{aligned} \tag{26}$$

The second master problem strengthening limits conflicts when vehicles leave the loading station. Let us consider once more the routing network by Lyu et al. (2019) in Figure 5 with three vehicles. By

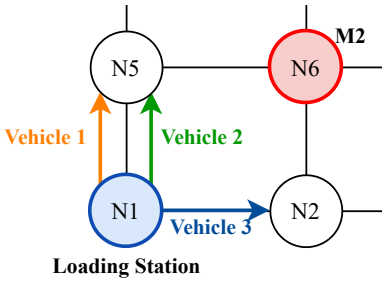


Figure 5: Conflict where more than three transfers leave the loading station

definition, all vehicles start in node 1, which marks the loading station. From that point, all vehicles can be assigned a transfer to one of the machines. All vehicles can either travel to node 2 or node 5. However, as the figure shows, if all these transfers start simultaneously, we end up in a conflicting situation. In this particular example, two vehicles are to move to node 5 at the same time; this results in a conflict in the CFRP subproblem. In general, we can safely constrain the number of transfers starting at the same time from the loading station to the out-degree (the number of outgoing edges) of the loading station.

To capture this as a CP constraint for our master problem, we introduce ST as the set of first transfers $transfer_{o_{j,0}}$ for all $j \in J$. Let od be the out-degree of the starting location. We can then

capture this in a constraint as follows:

$$|\{st_2 \in ST \mid \textbf{\textit{start\_of}}(st_1) = \textbf{\textit{start\_of}}(st_2), st_1 \neq st_2\}| < od \quad \text{for all } st_1 \in ST \tag{27}$$

We refer to the master problem including these extra strengthening constraints as the strengthened CP master problem. Later, in Section 7. we see that the addition of these extra constraints improves the solving speed, reduces the iteration count, and improves solution quality of the LBBD-based CFTFJSSP. By adding these constraints, intermediate master problem solutions lead less often to infeasible trial values.

6.2. Overall LBBD implementation

The complete LBBD-based CFTFJSSP implementation is captured in the software tooling that is provided along with this paper and is illustrated in Figure 6. Upon solving the strengthened CP master problem, the software tool extracts the trial values $\boldsymbol{x}^k$ from the strengthened CP master problem's solution. The tool then derives the sequence of transfers T_v for each vehicle v and their respective deadlines δ and destination nodes ϵ, which forms a new MP subproblem instance. By **C7.** and **C9.** from Definition 3.2, a transfer's release time is the deadline of the preceding transfer with the same vehicle.

Upon an infeasible subproblem result, either the simple feasibility cut F_3 (Equation 17) or the conflict-driven feasibility cut F_5 (Equation 23) is applied. Applying conflict-driven cuts requires us to extract the conflicting steps from the subproblem instance. The CPLEX solver used for MP subproblem implementation includes a conflict refiner tool that helps identifying the variables responsible for an infeasible result. This conflict refiner determines which $x^{\tau}_{v,n}$ valuations of the MP subproblem formulation in Section 4.4 caused the infeasibility, forming our set of conflicting steps Q^k. From here, it is straightforward to proceed with the remaining steps outlined in Section 5 in order to specify the conflict-driven cuts for our master CP problem.

The conflict-driven cuts form the heart of the LBBD-based CFTFJSSP enabling fast convergence towards an optimal solution. However, for some trial values $\boldsymbol{x}^k$ the conflict refiner might have difficulties determining the set of conflicting steps Q^k, resulting in long solving times. In these situations, it is a better idea to apply a simple feasibility cut in order to obtain a new set of trial values $\boldsymbol{x}^k$. With this new set of trial values, the conflict refiner might find the set Q^k relatively fast, therefore leading to faster iterations of the LBBD-based CFTFJSSP. The implementation of this mechanism is realised by introducing a timeout for the conflict refiner. For each infeasible subproblem result, after a certain time, the conflict-refinement process is aborted, and the LBBD resorts to the simple feasibility cut from Equation 17.

Using the conflict refiner brings some additional risks. There is little information available on how CPLEX's Conflict Refiner operates. To ensure the correctness of the conflict refiner result, a checking mechanism is applied. The checking mechanism verifies whether the found set of conflicting transfers, indeed, always results in an infeasible subproblem solution. To obtain this verification, an additional CFRP MP model is introduced. This CFRP MP model takes the set of conflicting transfers given by Equation 18 as input and computes the CFRP without the other transfers that are part of $\boldsymbol{x}^k$. If the infeasibility remains, we can conclude that the set of conflicting transfers is indeed a valid set of conflicting transfers. When this is not the case, the LBBD procedure settles on once again using the simple feasibility cut from Equation 17. (In all the benchmark results reported in the next section, the reported conflicting transfers were always proper conflicts.)

As shown in the next section, the convergence speed of our LBBD-based CFTFJSSP is strongly affected by the use of the conflict refiner in constructing the conflict-driven feasibility cuts. While it is still uncommon to use these kinds of infeasibility explanation mechanisms of solvers to devise cuts, Hooker (2024) mentions that such mechanisms are very promising as they provide new, powerful Benders cuts.

7. Experimental Evaluation

To evaluate the LBBD-based CFTFJSSP solution, we compare the approach to existing solutions in the literature. All results, solutions, and code related to the experimental work can be found at:

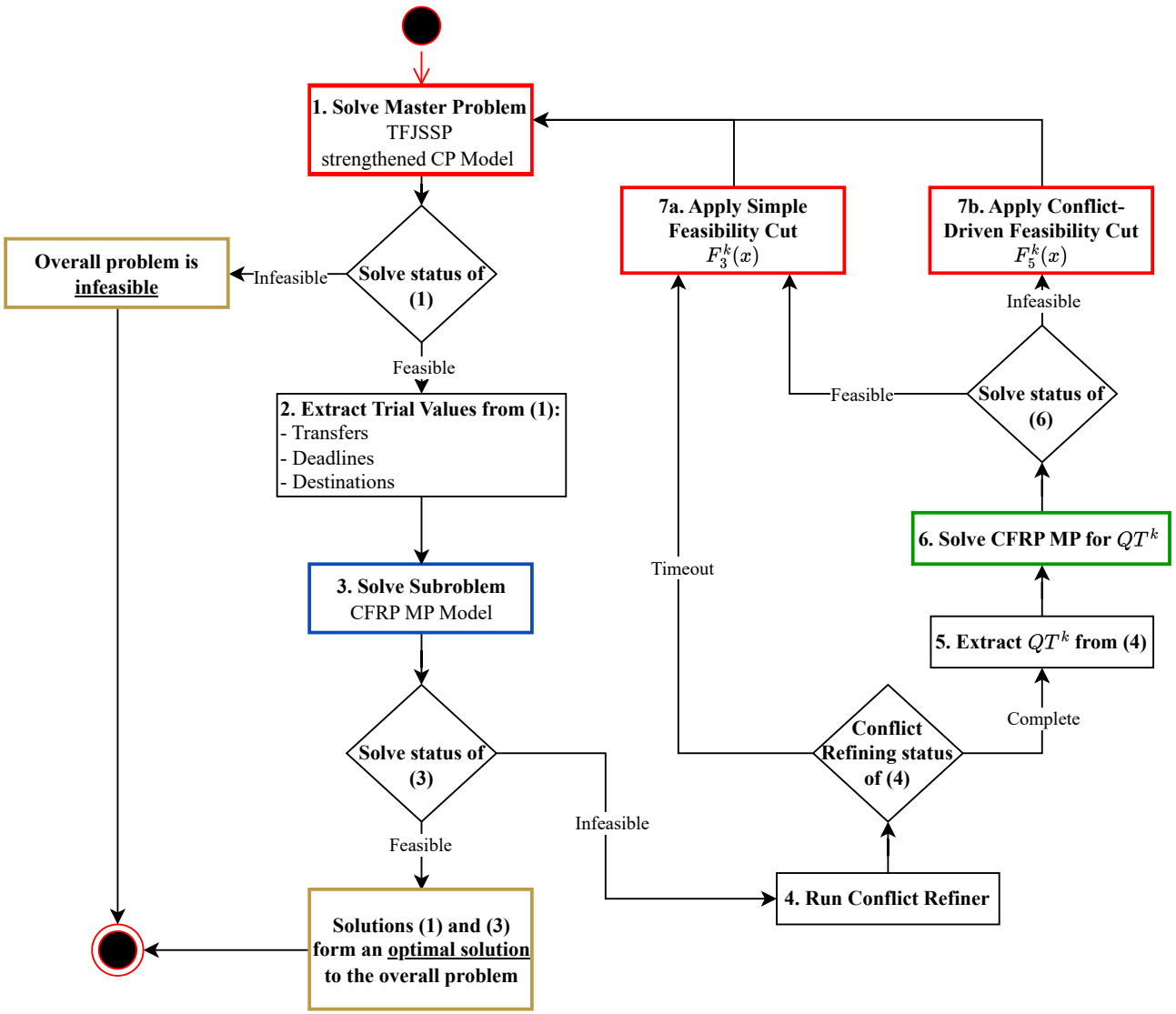


Figure 6: Flowchart showing the procedure of the LBBD-based CFTFJSSP implementation.

https://github.com/TUE-EE-ES/TJSP-toolset/releases/tag/v1.2.0.

7.1. Benchmarks

There is no standardized benchmark for the CFTFJSSP. Therefore, for our assessment, both the benchmarks published by Lyu et al. (2019) and Liu et al. (2023) are considered. These are the only two prior studies on CFTFJSSP variants in the literature and both use their own benchmarks.

Lyu et al. (2019) consider the most extensive set of problem instances. They provide 14 job sets and 6 differently sized FMS layout graphs. The combination of a job set, one of these layouts, and a number of vehicles forms a CFTFJSSP instance. Their goal is to evaluate how the makespan is influenced by the number of vehicles. Hence, every combination of a job set and layout is tested several times, each time with a different number of vehicles.

Liu et al. (2023) provide multiple small experiments to test various aspects of their CFTFJSSP solution. Several experiments evaluate how different numbers of vehicles and jobs influence the makespan and resource utilization. Other experiments are used to verify the effectiveness of the proposed algorithm. Seven of these experiments form a small benchmark that can be used for comparison with the LBBD-based CFTFJSSP.

7.2. Goals of the Experimental Assessment

The main objective of the experiments is to evaluate the performance of the LBBD-based CFTFJSSP While throughout TJSP research, various metaheuristic approaches were proposed for solv-

ing the considered optimization problems, we need methods to assess the solution quality of these metaheuristic approaches. By comparing the objective values of metaheuristic solutions with the ones found in an exact solution, we can assess the differences in solution quality. As the LBBD-based CFTFJSSP is such an exact solution, our first goal is to assess the solution quality, i.e., the reported makespan.

EG1. Find exact solutions for the benchmark instances. The goal is to validate that, indeed, the exact solutions outperform the metaheuristic approaches from the literature in terms of solution quality. These results provide insight in the optimality gap of the heuristic approaches.

Initial experimentation suggests that the time required to solve the master problem significantly affects the overall solving time of the CFTFJSSP. To limit the solving time of the master problem, we can simply introduce a timeout period to the solver (time boxing). If the solver cannot find an optimal solution within this time but has found at least a feasible solution, the process can continue with this feasible solution. In these cases, the optimality of the solution cannot be guaranteed, but it allows us to find decent solutions within a reasonable time. A second goal of the assessment is to investigate the impact of time boxing the master problem solver in the LBBD-based CFTFJSSP.

EG2. Evaluate the impact on the solution quality and solving time of different solver timeout periods for the master problem in the LBBD-based CFTFJSSP.

The LBBD-based CFTFJSSP applies various strenghtening elements such as the use of the strengthened conflict-driven feasibility cuts (Section 5.2), and the additional constraints to the master problem (Section 6.1). A third goal of the assessment is to investigate the impact of each of these elements and show their influence on the performance of the LBBD-based CFTFJSSP.

EG3. Evaluate the impact of various strengthening elements in the LBBD-based CFTFJSSP on solving time, iteration count, and the ability to find feasible solutions.

Finally, similar to what is done by Lyu et al. (2019), the LBBD-based CFTFJSSP can be used to evaluate the optimal number of vehicles for a given FMS layout and job set.

EG4. Evaluate the optimal number of vehicles for the benchmark of Lyu et al. (2019).

7.3. Experimental Setup

As explained earlier, both benchmarks available in literature Lyu et al. (2019); Liu et al. (2023) are used for comparison. Within the LBBD-based CFTFJSSP implementation, the CP master problem is solved using the IBM ILOG CP Optimizer solver, and the MP subproblem using the IBM ILOG CPLEX Optimizer solver. All tests apply the default solver settings and were conducted on an Intel i7-9750H 2.60 GHz with 16 GB RAM memory. The conflict refiner was given a default time limit of 180 seconds. Besides a master problem timeout, there is also a total solving time timeout of 5 hours per experiment.

7.4. Exact Benchmark Solutions and the Impact of Time Boxing the Master Problem.

We first present experiments for the first two assessment goals **EG1.** and **EG2.**.

7.4.1. Experimental Results on the Benchmark by Lyu et al. (2019).

We start by comparing our LBBD-based CFTFJSSP to the benchmark solutions from Lyu et al. (2019).

The purpose of the benchmark was to evaluate the exact number of vehicles needed for each of the 14 proposed job sets, where each job set was tested with one of 6 FMS layout graphs. This resulted in 14 sets of experiments. For each of the 14 sets of experiments, two to five problem instances were solved using a different number of vehicles. The experiment to solve each problem instance was executed from the least number to the highest number of vehicles as it is assumed that the makespan decreases when more vehicles are available. If the makespan decrease between subsequent experiments was at most 5%, the set of experiments was completed.

The purpose of the comparison is now to observe whether the LBBD-based CFTFJSSP indeed finds better makespan solutions compared to the genetic algorithm from (Lyu et al., 2019). We denote each experiment as EXij-k, which indicates the experiment of running job set i with FMS layout graph j using k vehicles. To investigate the impact of time boxing the master problem on the solution quality and solving time, initially, 30-second and 900-second master-problem timeout periods are tested, with a total timeout period of 5 hours. By using a timeout period for the master problem, we may lose the ability to find optimal solutions for all instances, but we can still find solutions that are feasible (and likely near-optimal). For benchmark instances without optimal solutions, an additional run with a 4500-second master timeout and no total timeout was performed.

There are some differences between the experimental setups in (Lyu et al., 2019) and in our experiments. First, the reported solving times in (Lyu et al., 2019) refer to the time it takes to find the optimal number of vehicles (i.e., to find a makespan decrease of at most 5%). We are interested in the solving time for individual problem instances. Hence, the reported solving times in (Lyu et al., 2019) are not useful as reference. Second, as Lyu et al. (2019) propose a genetic algorithm, they run each experiment 5-10 times to obtain a mean and a minimum makespan value. We are interested in the minimum makespan values. Hence, when referring to the makespan from (Lyu et al., 2019), we consider the minimum makespan that was found.

The results and comparison of the benchmark instances can be seen in Table 5. The table shows the experiment numbers in the first column, followed by the numbers of jobs and machines for that experiment, the makespan found by Lyu et al. (2019), and the solving time, solving status, and makespan found by the LBBD-based CFTFJSSP. Lastly, for each of the found solutions, the improvement percentage in makespan is given. The experiments with a 30-second and 900-second master-problem timeout period and a 5-hour total time limit are referred to as $\mathbf{30^5}$ and $\mathbf{900^5}$ respectively. The experiments for the 4500-second master-problem timeout period and no total time limit are denoted as $\mathbf{4500^\infty}$. The latter experiments were only done for problem instances for which the first two experiments did not provide an optimal solution. The solving status is reported as 'Optimal' if an optimal solution was found, as 'Feasible' if a solution is found that is feasible but not necessarily optimal because the master problem solution that is found is not necessarily optimal, and as 'Unknown' when the process runs out of time without finding a feasible solution. If only a single result is reported, this means the result is the same for all timeout settings. The best makespan values are denoted in bold. In addition to the results in Table 5, Appendix A shows the solution found for problem instance EX74-3; the solutions for the other benchmark instances can be found in the Git repository.

It is important to note that Table 5 is not in line with Table 6 as presented in (Lyu et al., 2019), as this table is not in line with the presented benchmark instances in Appendix B of (Lyu et al., 2019). Table 5 strictly follows the experiments as presented in Appendix B of Lyu et al. (2019).

Conclusions EG1. In all cases, the LBBD-based CFTFJSSP outperforms Lyu's genetic algorithm in terms of solution quality. We find optimal solutions in all but two problem instances (because of the time boxing). Lyu et al. (2019) found optimal solutions for only four problem instances, for two of the layout/job-set combinations. The overestimation by Lyu et al. (2019) of the achievable makespan is up to 37.9% (for EX115-3), which corresponds to the highest improvement percentage of 27.5 % reported in Table 5.

In the case of EX74-1 and EX115-3, only feasible solutions are found, without optimality guarantee. CP Optimizer provides a lower bound for the value being optimized in such cases. For the solution of EX74-1, CP Optimizer obtained a lower bound of 79. Given the makespan of 140 found by the LBBD-based CFTFJSSP, this gives an optimality gap of 43.57%. This might indicate that there is room for further improvement, although it is often the case that the reported lower bounds are not tight. For EX115-3, CP Optimizer obtained a lower bound of 55 which gives an optimality gap of 5.17%; this indicates that our solution is close to the optimum.

The solving time for the LBBD-based CFTFJSSP varies from less than 1 second to several hours (for most of the $\mathbf{4500^\infty}$ experiments plus instances EX115-3, EX115-4, and EX115-5 from the $\mathbf{900^5}$ experiments; all other instances are solved within one hour). To what extent such solving times are acceptable in practical application is context-dependent. Solving times of several hours may be acceptable when schedules are generated, for instance, overnight.

Experiment	$\|J\|/\|M\|$	C_{max} (Lyu et al., 2019)	**Solving Time (s)**			**Solving Status**	C_{max}	**Difference (%)**
			$\mathbf{30^5}$	$\mathbf{900^5}$	$\mathbf{4500^\infty}$	$\mathbf{30^5/900^5/4500^\infty}$	$\mathbf{30^5/900^5/4500^\infty}$	$\mathbf{30^5/900^5/4500^\infty}$
EX11-1	3/3	44	0.123	0.120		Optimal	**42**	4.55
EX11-2		41	0.235	0.222		Optimal	**40**	2.44
EX11-3		41	0.592	0.588		Optimal	**40**	2.44
EX22-1	4/4	75	1.147	0,905		Optimal	**63**	16.00
EX22-2		51	152.860	163.877		Optimal	**46**	9.80
EX22-3		47	0.830	0.829		Optimal	**46**	2.13
EX22-4		47	24.747	25.829		Optimal	**46**	2.13
EX32-1	5/4	89	15.112	14.100		Optimal	**72**	19.10
EX32-2		52	0.457	0.480		Optimal	**44**	15.38
EX32-3		47	1.962	2.054		Optimal	**44**	6.38
EX32-4		47	103.270	108.295		Optimal	**44**	6.38
EX43-1	5/5	99	11.675	14.367		Optimal	**81**	18.18
EX43-2		61	17.799	18.880		Optimal	**51**	16.36
EX43-3		54	0.964	1.106		Optimal	**51**	5.56
EX43-4		54	2.738	2.970		Optimal	**51**	5.56
EX53-1	6/5	119	30.365	900.394	4304.425	Feasible/Feasible/Optimal	100/**98/98**	15.97/17.65/17.65
EX53-2		68	12.469	14.221		Optimal	**53**	22.06
EX53-3		57	27.391	28.324		Optimal	**46**	19.30
EX53-4		52	53.782	58.389		Optimal	**46**	11.54
EX53-5		51	313.160	319.601		Optimal	**46**	9.80
EX64-1	6/6	131	30.800	900.838	2013.797	Feasible/Feasible/Optimal	**114**	12.98
EX64-2		82	3.032	3.097		Optimal	**75**	8.54
EX64-3		**75**	20.057	21.094		Optimal	**75**	0.00
EX64-4		**75**	5.673	5.991		Optimal	**75**	0.00
EX74-1	7/6	150	30.987	900.975	4500.020	Feasible	142/**140/140**	5.33/6.67/6.67
EX74-2		90.5	8.188	8.689		Optimal	**73**	19.34
EX74-3		77	2.432	2.348		Optimal	**72**	6.49
EX74-4		**72**	5.505	5.394		Optimal	**72**	0.00
EX74-5		**72**	8.871	8.950		Optimal	**72**	0.00
EX84-2	8/6	116.5	6.999	8.013		Optimal	**93**	20.17
EX84-3		100	5.170	5.057		Optimal	**93**	7.00
EX84-4		95	421.779	421.970		Optimal	**93**	2.11
EX84-5		94	69.711	70.552		Optimal	**93**	1.06
EX95-2	7/7	107	89.238	137.610		Optimal	**83**	22.43
EX95-3		92	102.143	106.310		Optimal	**81**	11.96
EX95-4		84	2187.682	745.417		Optimal	**81**	3.57
EX95-5		83	2029.989	548.691		Optimal	**81**	2.41
EX105-3	8/7	75	40.104	45.425		Optimal	**64**	14.67
EX105-4		70	722.223	742.626		Optimal	**64**	8.57
EX105-5		67	47.137	47.136		Optimal	**64**	4.48
EX105-6		67	9.646	9.595		Optimal	**64**	4.48
EX115-3	9/7	80	1072.837	18014.253	359153.801	Feasible/Unknown/Feasible	59/**-/58**	26.25/-/27.50
EX115-4		71.5	13214.607	18056.652	25273.345	Feasible/Unknown/Optimal	99/**-/55**	-38.46/-/23.07
EX115-5		67.5	2444.219	13503.533		Unknown/Optimal	**-/55**	-/18.52
EX115-6		64	2149.294	1424.057		Optimal	**55**	14.06
EX115-7		63	14.987	12.186		Optimal	**55**	12.7
EX126-2	8/8	107	5.136	4.258		Optimal	**84**	21.50
EX126-3		95	56.468	63.733		Optimal	**84**	11.58
EX136-3	9/8	105	1109.559	1158.383		Optimal	**95**	9.52
EX136-4		100.5	66.635	70.122		Optimal	**95**	5.47
EX136-5		97	44.540	46.750		Optimal	**95**	2.06
EX136-6		96	1374.842	1413.516		Optimal	**95**	1.04
EX136-7		96	112.847	115.139		Optimal	**95**	1.04
EX146-4	10/8	118.5	82.805	83.096		Optimal	**92**	22.36
EX146-5		111.5	12.118	12.125		Optimal	**92**	17.49
EX146-6		102	16.599	16.767		Optimal	**92**	9.80
EX146-7		102	22.924	23.085		Optimal	**92**	9.80

Table 5: Table comparing the smallest makespan results of Lyu et al. (2019) and the LBBD-based CFTFJSSP with a 30, 900, and 4500-second timeout period for the master problem.

Conclusions EG2. Except for EX115-5, in both the $\mathbf{30^5}$ and $\mathbf{900^5}$ experiments, the same problem instances were solved to optimality. Furthermore, for these instances the solving times with both experiments are roughly similar. This indicates that in these problem instances the solving time for each master iteration is almost always below 30 seconds.

Experiments EX53-1, EX64-1, EX74-1, EX115-3, EX115-4, and EX115-5 resulted in non-optimal solutions for either or both the $\mathbf{30^5}$ and $\mathbf{900^5}$ experiments. In all these experiments, the LBBD procedure takes various iterations to obtain valid solutions. Within a 5-hour total timout period, this means a smaller master problem timeout period increases the maximal number of iterations that the total LBBD procedure can take. Shortening the master problem timeout period therefore increases the chance of finding a feasible solution for these instances. This behavior is well reflected in experiments EX115-3 and EX115-4 where a feasible solution was found in the $\mathbf{30^5}$ experiments, but the total timeout period was reached in the $\mathbf{900^5}$ experiments.

However, there are disadvantages in having a tight time budget for the master problem. Reducing the timeout period might come at the cost of the solution quality as finding optimal solutions may require a longer master problem timeout period. This is the case for EX115-5 in the $\mathbf{30^5}$ experiments where at one point no feasible solution for the master problem can be found within the 30-seconds timeout period, terminating the procedure. By increasing the master timeout period to 900 seconds, we do find an optimal solution.

These observations motivate the $\mathbf{4500^\infty}$ experiments, as the absence of a total time period removes iteration limitations and the increased master problem period improves the solution quality potential. Running these extra experiments made it possible to find optimal solutions for EX53-1, EX64-1, and EX115-4. Further increasing the time limits for EX74-1 and EX115-3 will eventually lead to finding optimal solutions to these instances. However, it may be expected that this will result in excessive solving times.

7.4.2. Experimental Results on the Benchmark by Liu et al. (2023)

Besides the benchmark from Lyu et al. (2019), the LBBD-based CFTFJSSP was also tested on the benchmark of Liu et al. (2023).

In contrast to the benchmark in (Lyu et al., 2019), (Liu et al., 2023) allows diagonal steps between nodes in the routing network. The MP subproblem in the LBBD-based CFTFJSSP allows diagonal steps under the assumption that these steps take unit time, as uniform step times are required for synchronizing vehicle movements. While the mathematical problem formulation in (Liu et al., 2023) assumes unit-time steps as well, the implemented Dijkstra's algorithm with time windows allows step times proportional to the euclidean distance, that is, $\sqrt{2}$ time units for a diagonal step. Experiments in (Liu et al., 2023) were done with travel times proportional to distance. Travel times proportional to distance could be approximated in an MP by choosing a finer discretization. However, this substantially changes the model since also conflict modeling would need to be adapted.

To enable a fair comparison between our approach and the work of (Liu et al., 2023) without reworking the model, we therefore defined two sets of experiments, one using the original routing networks defined by (Liu et al., 2023) but assuming unit-time steps for all directions (horizontal, vertical, diagonal) and one set of experiments where all benchmark instances are redefined with a routing network that allows only horizontal and vertical movements. The optimal makespan of the latter instances is never better than the optimal makespan of the same instances allowing diagonal steps of $\sqrt{2}$ time units. That is, the optimal makespan values found by the LBBD-based CFTFJSSP on the instances without diagonal steps serve as upper bounds on the optimal makespans of the instances in (Liu et al., 2023).

The results of our experiments can be found in Table 6. The table shows the experiment (exp.) numbers in the first column, followed by the numbers of jobs, machines, and vehicles for that experiment. Experiments with the -HV extension indicate experiments where, for the LBBD-based CFTFJSSP, only horizontal and vertical movements are allowed in the routing network. The third up to the sixth columns show the different solution methods tested in (Liu et al., 2023). The GA-D method refers to a conventional genetic algorithm integrated with Dijkstra's algorithm with time windows. The AGA-D method refers to an adaptive genetic algorithm integrated with Dijkstra's algorithm with time windows. The SLGA-D method refers to the method combining a self-learning

genetic algorithm with Dijkstra's algorithm with time windows. This is the main algorithm presented in Liu et al. (2023). The HPSO-D method refers to a hybrid particle swarm optimization integrated with Dijkstra's algorithm with time windows. The remaining columns present the solution of the LBBD-based CFTFJSSP and the relative improvement percentage (diff.) in makespan compared to the best-found makespan by one of the algorithms reported by (Liu et al., 2023). Liu et al. (2023) only report the iteration time of their algorithm; they do not report overall solving times that can be used for comparison. For EX1-HV up to EX7-HV, the relative improvement percentage (diff.) is relative to the results in (Liu et al., 2023) for EX1 up to EX7, respectively. In addition to the results in Table 6, Appendix A shows the obtained solution for problem instance EX5; solutions for the other benchmark instances can be found in the Git repository.

Exp.	$\|J\|/\|M\|$ $/\|V\|$	**GA-D** C_{max} (Liu et al., 2023)	**AGA-D** C_{max} (Liu et al., 2023)	**SLGA-D** C_{max} (Liu et al., 2023)	**HPSO-D** C_{max} (Liu et al., 2023)	**Solving Time** (s)		**Solving Status**	C_{max}	**Diff.** (%)
						$\mathbf{30^5}$	$\mathbf{900^5}$	$\mathbf{30^5/900^5}$	$\mathbf{30^5/900^5}$	$\mathbf{30^5/900^5}$
EX1	2/3/2	17.20	17.20	17.20	17.20	1.075	1.221	Optimal	**13**	24.42
EX2	3/3/2	18.00	18.00	18.00	18.00	0.197	0.234	Optimal	**15***	16.67
EX3	4/4/3	37.20	37.20	37.20	37.20	3.663	3.052	Optimal	**33**	11.29
EX4	5/5/4	46.60	41.00	39.60	45.40	7.509	7.174	Optimal	**31**	21.72
EX5	6/4/4	56.00	50.60	49.00	53.40	16.903	10.159	Optimal	**34**	30.61
EX6	7/5/4	86.40	74.60	74.80	84.80	53.369	834.931	Feasible/ Optimal	70/**45**	6.17/39.68
EX7	8/3/3	92.80	90.20	87.40	87.60	33.001	582.954	Feasible/ Optimal	80/**78**	8.47/10.76
EX1-HV	2/3/2	-	-	-	-	0.508	0.243	Optimal	**16**	6.98
EX2-HV	3/3/2	-	-	-	-	0.232	0.369	Optimal	**16***	11.11
EX3-HV	4/4/3	-	-	-	-	3.786	3.869	Optimal	**36**	3.23
EX4-HV	5/5/4	-	-	-	-	19.889	17.789	Optimal	**35**	11.62
EX5-HV	6/4/4	-	-	-	-	15.081	14.005	Optimal	**38**	22.45
EX6-HV	7/5/4	-	-	-	-	61.487	409.439	Feasible/ Optimal	86/**49†**	-15,28 /34.32
EX7-HV	8/3/3	-	-	-	-	238.113	139.580	Feasible/ Optimal	82 /**79**	6.18 /9.61

Table 6: Table comparing the smallest makespan results of Liu et al. (2023) and the LBBD-based CFTFJSSP.

The * marked experiment EX2 is not in line with experiment EX2 as presented in Table 6 of (Liu et al., 2023). We believe that, due to a spacing error, the operation processing times in the experimental description in (Liu et al., 2023) were shifted, leading to an inconsistent CFTFJSSP instance. Experiment EX2 shown in Table 6 was run on a version that was corrected to the best of our understanding. Our corrected version can be found in the software repository.

For the † marked experiment EX6-HV, applying the default solver settings resulted in an out-of-memory error in CP Optimizer. By default, CP Optimizer parallizes the solving process into workers, that match with the number of available CPU cores. However, the memory usage for the solving process grows linearly with the number of workers that are being used. Hence, to reduce the memory usage for EX6-HV the number of workers was set to 1.

Conclusions EG1. In all experiments, the LBBD-based CFTFJSSP outperforms Liu's self-learning genetic algorithm in terms of solution quality. Furthermore, for all the setups, we were able to find optimal solutions with the $\mathbf{900^5}$ experiments. We also see that, by including diagonal steps, the optimal makespan of EX1 up to EX7 improves compared to EX1-HV up to EX7-HV.

As discussed earlier, EX1-HV up to EX7-HV provide upper bounds on the optimal makespan values for the corresponding benchmark instance of (Liu et al., 2023). From this, we see that the SLGA-D algorithm by (Liu et al., 2023) was not able to find any optimal solution, as their heuristic overestimates the makespan for all experiments. The overestimation of their algorithms is up to 52.24% (in EX6-HV).

The solving times for our LBBD-CFTFJSSP range from less than 1 second to about 14 minutes.

Conclusions EG2. Similar to the results seen for the benchmark by (Liu et al., 2023), we see that for EX1 up to EX5 and for EX1-HV up to EX5-HV we find optimal solutions with similar solving times for both timeout periods for the master problem. For EX6, EX7, EX6-C, and EX7-C, we see that $\mathbf{30^5}$ experiments are only able to find feasible solutions, while the $\mathbf{900^5}$ experiments lead to optimal solutions. The 900-second timeout is never triggered though, so we may conclude that the master problem solving time is apparently between 30 and 900 seconds. As before, we see that larger timeouts may lead to an improved solution quality as more optimal solutions are found.

7.4.3. Overall Conclusions EG1 and EG2

Conclusions EG1. The LBBD-based CFTFJSSP finds exact solutions for all but two benchmark instances available in the literature. It outperforms all existing benchmark experiments in terms of solution quality. Figure 7 shows that for about half of the problem instances the maximal improvement is 10% or more; for about a quarter of all instances the improvement is 17% or more. Note that the estimates for the benchmark instances of (Liu et al., 2023) are conservative, because we use the LBBD-CFTFJSSP '-HV' experiments that do not allow diagonal movement as a basis for the comparison.

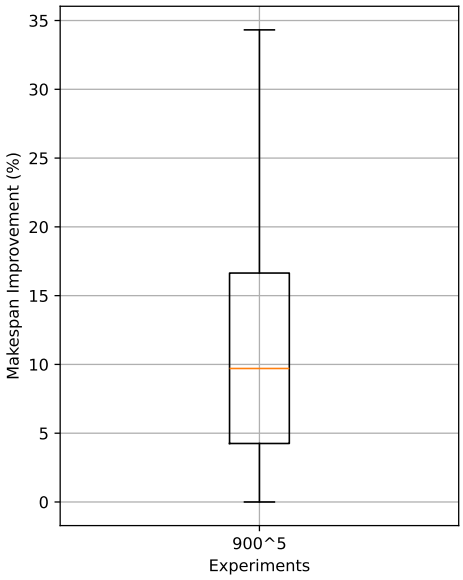


Figure 7: Makespan improvement for all the evaluated benchmark instances.

Conclusions EG2. The tested timeout settings for the master problem do not affect the solution for most of the benchmark instances presented, because no timeouts occur. Shorter timeout periods can lead to a reduced solution quality when no optimal solution can be found for a master problem instance within the given time budget. The best timeout value in a practical setting may depend on the context and application.

7.5. Evaluating the Influence of Each Strengthening Element

Our LBBD solution embeds various strengthening elements. The first strengthening elements were enhancements of the simple feasibility cut found in (17), resulting in three feasibility cuts:

- (SIC) Simple feasibility cut from (17).
- (WC) Weak conflict-driven feasibility cut from (20).
- (STC) Strengthened conflict-driven feasibility cut from (23).

Additionally, we have shown how additional master problem constraints can avoid various conflicting situations in the generated subproblem instance, resulting in three potential constraint sets to strengthen the original master problem:

- (TM) Constraints (25) and (26) to forbid transfers to start and end in the same location.
- (LM) Constraint (27) to limit the outgoing degree of vehicles leaving the loading station.
- (SM) The combination of (TM) and (LM), which we refer to as the strengthened CP master problem.

The proposed LBBD-based CFTFJSSP applies the strengthened CP master problem (SM) and the strengthened conflict-driven feasibility cut (STC). However, it is possible to combine these strengthening elements in other ways to form alternative LBBD-based CFTFJSSP approaches. Any combination of elements influences the total solving time, iteration count of the decomposition process, and the ability to find feasible and optimal solutions. To evaluate the influence of the various strengthening elements on these performance metrics, we ran the benchmark by Lyu et al. (2019) with nine alternative LBBD-based CFTFJSSP approaches. Each alternative combines a specific master problem strengthening with a different feasibility cut. We do not consider LBBD-based CFTFJSSP alternatives that apply the default master problem without any strengthening constraints. Experiments show that this does not provide an effective approach to solve the CFTFJSSP.

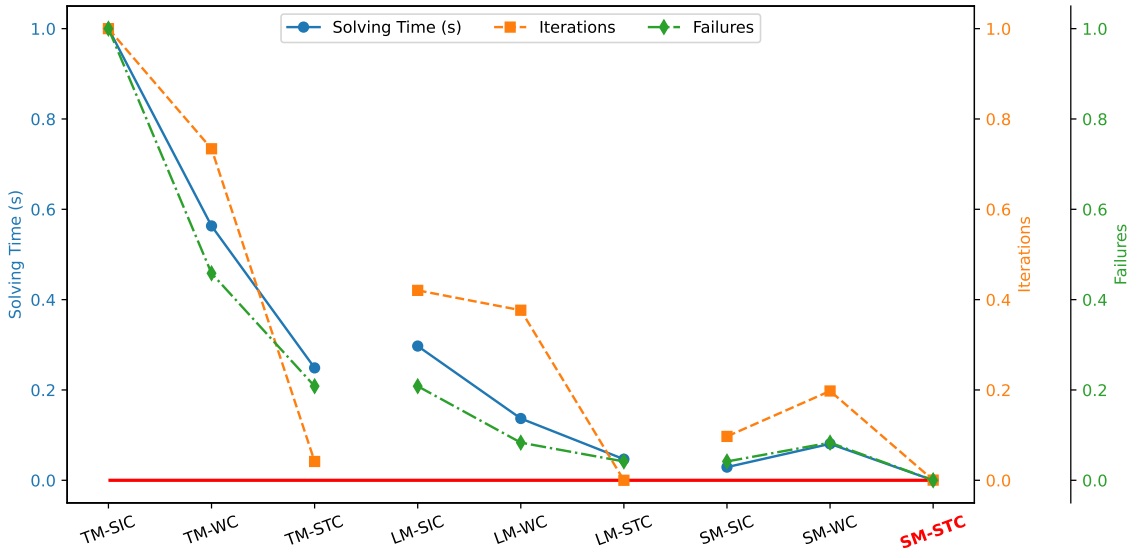


Figure 8: Performance comparison of LBBD-based CFTFJSSP alternatives on the benchmark by Lyu et al. (2019).

Figure 8 shows the comparison of all LBBD-based CFTFJSSP alternatives on each of the performance metrics. The x-axis shows the combination of strengthening elements that form each LBBD-based CFTFJSSP alternative. The LBBD-based CFTFJSSP proposed in this paper combining the SM-STC strengthening elements forms the baseline. It is shown in red and marked as a horizontal red line. Each performance metric is represented with a different color. The blue line represents the normalized total solving time each of the alternative approaches needed to complete the entire benchmark. The orange line represents the normalized total number of iterations needed to complete the benchmark. Finally, the green line shows the normalized number of failures, which indicates how many times the approach was not able to find a feasible solution. For each of the performance metrics holds that a smaller number equals better performance.

Conclusions EG3. From Figure 8, it can be seen that a strengthening of the feasibility cuts indeed corresponds to an improvement in all performance metrics of the respective LBBD-based CFTFJSSP alternative. A single outlier for this behavior is seen in the performance of the SM-WC combination, that in a few instances performs worse than its SIC counterpart. In all alternatives, the strengthened cut STC performs best of all feasibility cuts. Most notably, in all cases, the STC significantly reduces the number of iterations, confirming that strengthening the cuts results in faster convergence towards a solution.

Among the master problem strengthening elements, the LM constraints give a better performance on all metrics than the TM constraints. This can be expected, as without the LM constraints, loading station conflicts would occur in more than half of the benchmark instances, leading to more iterations and thus solving time to find a feasible solution. The combination of all constraints SM gives an improvement in all performance metrics compared to its TM and LM alternatives.

The baseline performance by the proposed LBBD-based CFTFJSSP combining the SM-STC strengthening elements outperforms all LBBD-based CFTFJSSP alternatives on all performance met-

rics. The total time to solve the entire benchmark with this approach was roughly 17 hours, which is a significant improvement over the 144 hours needed for the TM-SIC combination.

7.6. Evaluating the Optimal Number of Vehicles.

The original experimental setup by Lyu et al. (2019) was to decide upon the optimal number of vehicles for each of the 14 sets of experiments. Lyu et al. (2019) determined that an optimal number of vehicles for their genetic algorithm was found when the decrease in makespan with one additional vehicle is at most 5%. As the LBBD-based CFTFJSSP provides optimal solutions, we can conclude the optimal number of vehicles when the same makespan is found with an additional vehicle.

Table 7 summarizes results from the earlier experiments and Lyu et al. (2019). The first column shows the considered sets of experiments, the second column specifies the optimal number of vehicles reported by Lyu et al. (2019), and the last column shows the optimal number of vehicles determined by the LBBD-based CFTFJSSP using the results from Table 5. All found results, except possibly the result for EX115, are optimal. Experiment EX115-3 was not solved to optimality. Moreover, the reported lower bound for EX115-3 is 55, which matches the optimal makespan for EX115-4. If the optimal solution to EX115-3 is in fact 55, the optimal vehicle count for EX115 would be 3.

Experiment	**Lyu *et al.***	**LBBD-based CFTFJSSP**
EX11	2	2
EX22	3	2
EX32	3	2
EX43	3	2
EX53	4	3
EX64	3	2
EX74	4	3
EX84	4	2
EX95	4	3
EX105	5	3
EX115	6	4 (possibly 3)
EX126	4	2
EX136	5	3
EX146	6	4

Table 7: The optimal number of vehicles for each set of experiments.

Conclusions EG4. The LBBD-based CFTFJSSP outperforms (Lyu et al., 2019) in finding the optimal number of vehicles. The approach of Lyu et al. (2019) overestimates the number of vehicles needed for all layout/job-set combinations, except for the smallest setup, experiment set EX11.

8. Conclusions & Future Work

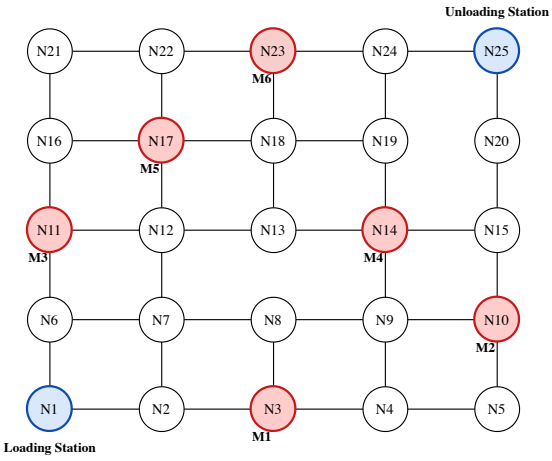
The main objective of this work was to define an exact approach to finding shortest makespan solutions for CFTFJSSP instances using a Logic-Based Benders Decomposition (LBBD). This was achieved by developing an LBBD-based CFTFJSSP solution using a CP model for the TFJSSP as a master problem and an ILP model for the CFRP as subproblem. This LBBD-based CFTFJSSP is the first exact approach for solving the CFTFJSSP. The procedure applies an innovative conflict-driven Benders cut based on the conflict-refiner capability of the IBM ILOG CPLEX Optimizer solver. The cut analyses the conflict structure and excludes the occurrence of many similar solutions to the master problem instance at hand that cause similar conflicts in the derived subproblem instance. The LBBD-based CFTFJSSP finds better makespan solutions for all benchmark instances avaiable in the literature (Lyu et al., 2019; Liu et al., 2023). The software implementation and tools used in this research are made publicly available for further experimentation and research.

There are several interesting directions for further research. Choosing different solver settings may lead to different solving performance. As a first direction for future work, it would be interesting to further investigate the tuning of the hyperparameters of our method (solver settings, time box for the master problem), possibly dynamically during the search, to further improve the performance. Additionally, the current LBBD-based CFTFJSSP optimizes the single objective of minimum makespan.

A second direction for future work may be to investigate multi-objective CFTFJSSP, including for instance energy cost for transportation. Another direction might be to speed up the LBBD-based CFTFJSSP or improve its scalability to larger problem instances by replacing solving techniques used for the master problem and/or subproblem by other exact techniques or heuristics. Given the results obtained in this work, it is moreover interesting to expand the use of the LBBD to other complex optimization problems, particularly in combination with smart Benders cuts using information about infeasibility causes from infeasible subproblem instances.

Appendix A. Solutions

This appendix shows visual representations for two benchmark solutions obtained by our LBBD-based CFTFJSSP implementation. The visual representation is the same as the one explained in Example 3.1. From the benchmark by Lyu et al. (2019), Figure A.9 shows a visual representation of problem instance EX74-3. The LBBD-based CFTFJSSP obtained solution for this instance can be seen in Figure A.10. From the benchmark by Liu et al. (2023), Figure A.11 shows a visual representation of problem instance EX5, with its LBBD-based CFTFJSSP obtained solution in Figure A.12.



Job	Operation	ls	m_1	m_2	m_3	m_4	m_5	m_6	us
j_0	$o_{0,0}$	0	-	-	-	-	-	-	-
	$o_{0,1}$	-	-	-	10	-	-	-	-
	$o_{0,2}$	-	6	-	-	-	-	-	-
	$o_{0,3}$	-	-	-	-	5	-	7	-
	$o_{0,4}$	-	-	-	-	-	-	-	0
j_1	$o_{1,0}$	0	-	-	-	-	-	-	-
	$o_{1,1}$	-	-	-	-	-	-	8	-
	$o_{1,2}$	-	-	-	-	5	-	-	-
	$o_{1,3}$	-	9	-	12	-	-	-	-
	$o_{1,4}$	-	-	-	-	-	-	-	0
j_2	$o_{2,0}$	0	-	-	-	-	-	-	-
	$o_{2,1}$	-	5	-	6	8	-	-	-
	$o_{2,2}$	-	-	-	-	-	-	-	0
j_3	$o_{3,0}$	0	-	-	-	-	-	-	-
	$o_{3,1}$	-	-	10	12	-	-	-	-
	$o_{3,2}$	-	-	-	-	-	14	-	-
	$o_{3,3}$	-	-	-	-	-	-	-	0
j_4	$o_{4,0}$	0	-	-	-	-	-	-	-
	$o_{4,1}$	-	-	8	-	-	-	-	-
	$o_{4,2}$	-	-	-	-	11	-	10	-
	$o_{4,3}$	-	9	-	-	-	-	-	-
	$o_{4,4}$	-	-	-	-	-	-	-	0
j_5	$o_{5,0}$	0	-	-	-	-	-	-	-
	$o_{5,1}$	-	-	-	-	8	-	-	-
	$o_{5,2}$	-	-	10	-	-	-	-	-
	$o_{5,3}$	-	11	-	-	-	-	13	-
	$o_{5,4}$	-	-	-	-	-	7	-	-
	$o_{5,5}$	-	-	-	14	-	-	-	-
	$o_{5,6}$	-	-	-	-	-	-	-	0
j_6	$o_{4,0}$	0	-	-	-	-	-	-	-
	$o_{4,1}$	-	-	-	5	-	-	-	-
	$o_{4,2}$	-	-	-	-	9	-	-	-
	$o_{4,3}$	-	-	-	-	-	-	-	0

Figure A.9: CFTFJSSP instance EX74-3 as defined in (Lyu et al., 2019)

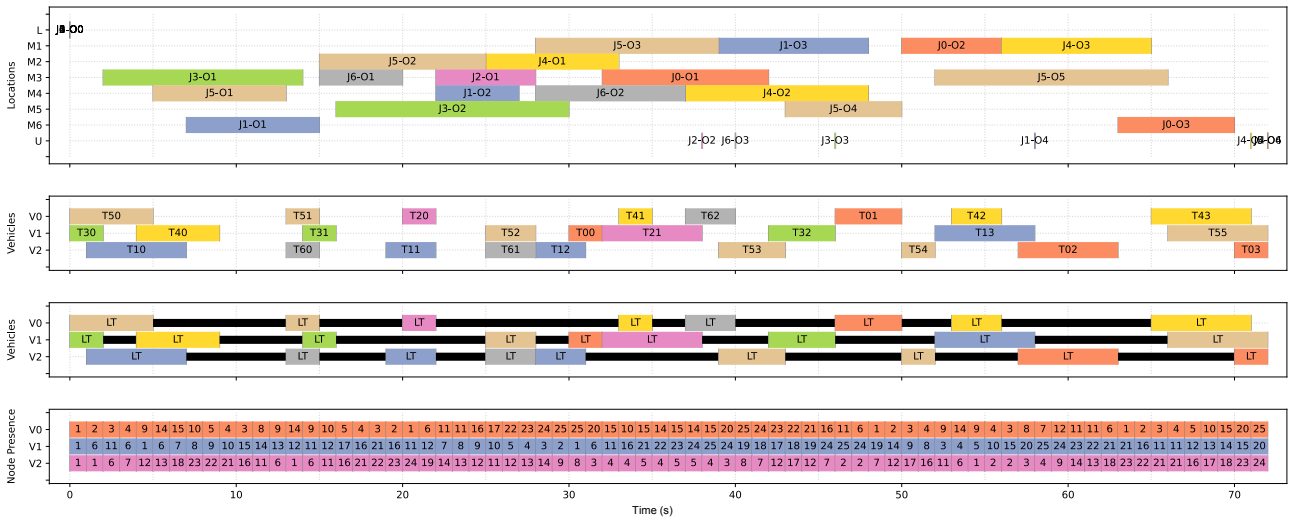
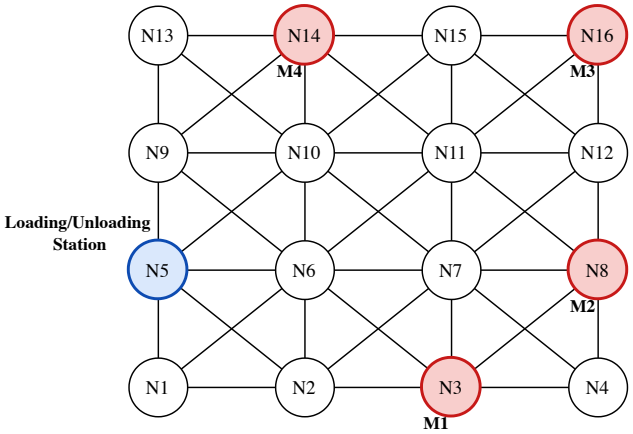


Figure A.10: Optimal solution to problem instance EX74-3 (Lyu et al., 2019) as found by the LBBD-based CFTFJSSP



Job	Operation	ls/us	m_1	m_2	m_3	m_4
j_0	$o_{0,0}$	0	-	-	-	-
	$o_{0,1}$	-	13	12	9	-
	$o_{0,2}$	-	5	-	7	9
	$o_{0,3}$	0	-	-	-	-
j_1	$o_{1,0}$	0	-	-	-	-
	$o_{1,1}$	-	6	9	11	4
	$o_{1,2}$	-	-	8	3	10
	$o_{1,3}$	-	3	11	13	-
	$o_{1,4}$	0	-	-	-	-
j_2	$o_{2,0}$	0	-	-	-	-
	$o_{2,1}$	-	15	17	-	12
	$o_{2,2}$	0	-	-	-	-
j_3	$o_{3,0}$	0	-	-	-	-
	$o_{3,1}$	-	1	6	4	-
	$o_{3,2}$	-	4	9	7	11
	$o_{3,3}$	-	13	10	17	-
	$o_{3,4}$	-	-	-	8	12
	$o_{3,5}$	0	-	-	-	-
j_4	$o_{4,0}$	0	-	-	-	-
	$o_{4,1}$	-	-	14	-	6
	$o_{4,2}$	-	17	9	11	-
	$o_{4,3}$	0	-	-	-	-
j_5	$o_{5,0}$	0	-	-	-	-
	$o_{5,1}$	-	5	13	-	7
	$o_{5,2}$	-	17	-	10	14
	$o_{5,3}$	-	9	6	5	-
	$o_{5,4}$	0	-	-	-	-

Figure A.11: CFTFJSSP instance EX5 as defined in (Liu et al., 2023)

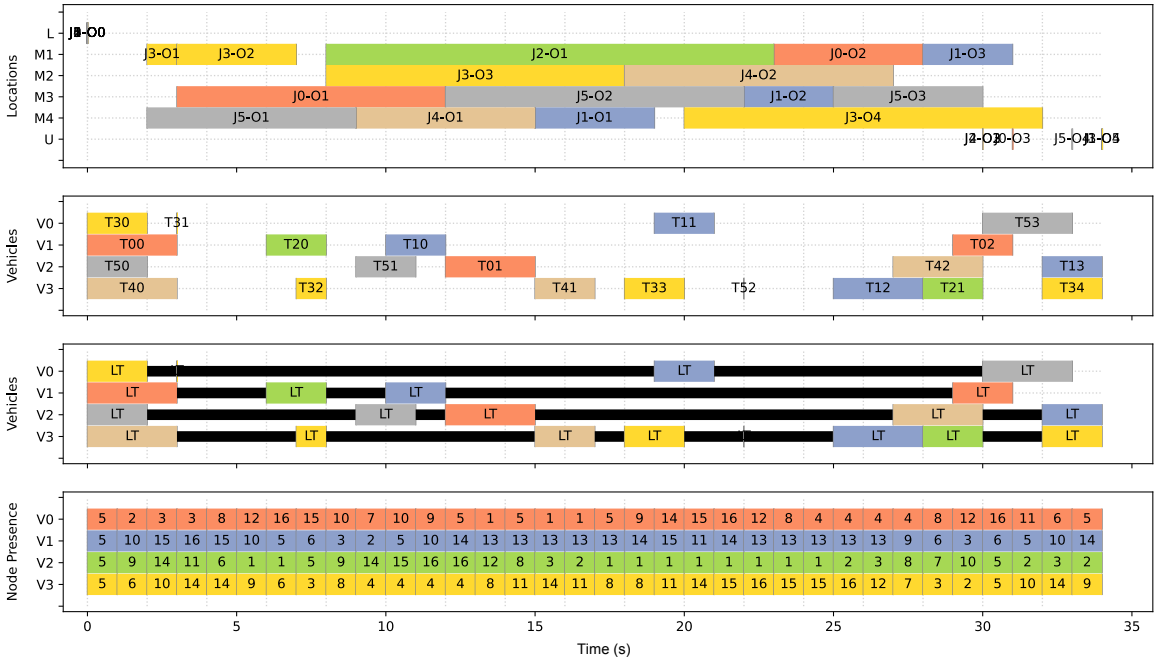


Figure A.12: Optimal solution to problem instance EX5 (Liu et al., 2023) as found by the LBBD-based CFTFJSSP

References

Abdelmaguid, T.F., Nassef, A.O., Kamal, B.A., Hassan, M.F., 2004. A hybrid GA/heuristic approach to the simultaneous scheduling of machines and automated guided vehicles. International Journal of Production Research 42, 267–281. doi:10.1080/00207540320001235 79.

Baruwa, O.T., Piera, M.A., 2016. A coloured Petri net-based hybrid heuristic search approach to simultaneous scheduling of machines and automated guided vehicles. International Journal of Production Research 54, 4773–4792. doi:10.1080/00207543.2015.1087656.

Berterottière, L., Dauzère-Pérès, S., Yugma, C., 2024. Flexible job-shop scheduling with transportation resources. European Journal of Operational Research 312, 890–909. doi:10.1016/j.ejor.2023.07.036.

Bilge, Ü., Ulusoy, G., 1995. A Time Window Approach to Simultaneous Scheduling of Machines and Material Handling System in an FMS. Operations Research 43, 1058–1070. doi:10.1287/opre.43.6.1058.

Broadbent, A.J., Besant, C.B., Premi, S.K., Walker, S.P., 1985. Free ranging AGV systems : promises, problems and pathways, in: Proceedings of the 2nd International Conference on Automated Materials Handling, pp. 221–237.

Chen, Z.L., 2010. Integrated Production and Outbound Distribution Scheduling: Review and Extensions. Operations Research 58, 130–148. doi:10.1287/opre.1080.0688.

Corréa, A.I., Langevin, A., Rousseau, L.M., 2007. Scheduling and routing of automated guided vehicles: A hybrid approach. Computers & Operations Research 34, 1688–1707. doi:10.1016/j.cor.2005.07.004.

Deroussi, L., 2014. A Hybrid PSO Applied to the Flexible Job Shop with Transport, in: Swarm Intelligence Based Optimization, Cham. pp. 115–122. doi:10.1007/978-3-319-12970-9_13.

Deroussi, L., Norre, S., 2010. Simultaneous scheduling of machines and vehicles for the flexible job shop problem, in: International conference on metaheuristics and nature inspired computing, pp. 1–2.

Eigbe, E.A., De Schutter, B., Nasri, M., Yorke-Smith, N., 2023. Sequence- and Time-Dependent Maintenance Scheduling in Twice Re-Entrant Flow Shops. IEEE Access 11, 103461–103475. doi:10.1109/ACCESS.2023.3317533.

El Khayat, G., Langevin, A., Riopel, D., 2006. Integrated production and material handling scheduling using mathematical programming and constraint programming. European Journal of Operational Research 175, 1818–1832. doi:10.1016/j.ejor.2005.02.077.

Fontes, D.B.M.M., Homayouni, S.M., 2019. Joint production and transportation scheduling in flexible manufacturing systems. Journal of Global Optimization 74, 879–908. doi:10.1007/s10898-018-0681-7.

Ham, A., 2020. Transfer-robot task scheduling in flexible job shop. Journal of Intelligent Manufacturing 31, 1783–1793. doi:10.1007/s10845-020-01537-6.

Ham, A., 2021. Transfer-robot task scheduling in job shop. International Journal of Production Research 59, 813–823. doi:10.1080/00207543.2019.1709671.

Homayouni, S.M., Fontes, D.B.M.M., 2019. Joint scheduling of production and transport with alternative job routing in flexible manufacturing systems. AIP Conference Proceedings 2070, 020045. doi:10.1063/1.5090012.

Homayouni, S.M., Fontes, D.B.M.M., 2021. Production and transport scheduling in flexible job shop manufacturing systems. Journal of Global Optimization 79, 463–502. doi:10.1007/s10898-021-00992-6.

Hooker, J., 2024. Logic-Based Benders Decomposition: Theory and Applications. Synthesis Lectures on Operations Research and Applications, Springer International Publishing, Cham. doi:10.1007/978-3-031-45039-6.

Hooker, J.N., Ottosson, G., 2003. Logic-based Benders decomposition. Mathematical Programming 96, 33–60. doi:10.1007/s10107-003-0375-9.

Hosseini, A., Otto, A., Pesch, E., 2023. Scheduling in manufacturing with transportation: Classification and solution techniques. European Journal of Operational Research 315, 821–843. doi:10.1016/j.ejor.2023.10.013.

Huang, S., 2018. Optimization of job shop scheduling with material handling by automated guided vehicle. Ph.D. thesis. Iowa State University.

Kim, C.W., Tanchoco, J.M.A., 1991. Conflict-free shortest-time bidirectional AGV routeing. International Journal of Production Research , 2377–2391URL: http://www.tandfonline.com/doi/abs/10.1080/00207549108948090, doi:10.1080/00207549108948090.

Krishnamurthy, N.N., Batta, R., Karwan, M.H., 1993. Developing Conflict-Free Routes for Automated Guided Vehicles. Operations Research 41, 1077–1090. doi:10.1287/opre.41.6.1077.

Kusiak, A., 2018. Smart manufacturing. International Journal of Production Research 56, 508–517. doi:10.1080/00207543.2017.1351644.

Laborie, P., Rogerie, J., Shaw, P., Vilím, P., 2018. IBM ILOG CP optimizer for scheduling: 20+ years of scheduling with constraints at IBM/ILOG. Constraints 23, 210–250. doi:10.1007/s10601-018-9281-x.

Lee, C.Y., Chen, Z.L., 2001. Machine scheduling with transportation considerations. Journal of Scheduling 4, 3–24. doi:10.1002/1099-1425(200101/02)4:1<3::AID-JOS57>3.0.CO;2-D.

Liu, J., Sun, B., Li, G., Chen, Y., 2023. An integrated scheduling approach considering dispatching strategy and conflict-free route of AMRs in flexible job shop. The International Journal of Advanced Manufacturing Technology 127, 1979–2002. doi:10.1007/s00170-022-10619-z.

Lunardi, W.T., Birgin, E.G., Laborie, P., Ronconi, D.P., Voos, H., 2020. Mixed Integer linear programming and constraint programming models for the online printing shop scheduling problem. Computers & Operations Research 123, 105020. doi:10.1016/j.cor.2020.105020.

Lyu, X., Song, Y., He, C., Lei, Q., Guo, W., 2019. Approach to Integrated Scheduling Problems Considering Optimal Number of Automated Guided Vehicles and Conflict-Free Routing in Flexible Manufacturing Systems. IEEE Access 7, 74909–74924. doi:10.1109/ACCESS.2019.2919109.

Maggu, P.L., Das, G., 1980. On 2× n sequencing problem with transportation times of jobs. Pure and Applied Mathematika Sciences 12, 1–6.

Nishi, T., Ando, M., Konishi, M., 2005. Distributed route planning for multiple mobile robots using an augmented Lagrangian decomposition and coordination technique. IEEE Transactions on Robotics 21, 1191–1200. doi:10.1109/TRO.2005.853489.

Nishi, T., Hiranaka, Y., Grossmann, I.E., 2011. A bilevel decomposition algorithm for simultaneous production scheduling and conflict-free routing for automated guided vehicles. Computers & Operations Research 38, 876–888. doi:10.1016/j.cor.2010.08.012.

Nouri, H.E., Driss, O.B., Ghédira, K., 2016. A Classification Schema for the Job Shop Scheduling Problem with Transportation Resources: State-of-the-Art Review, in: Artificial Intelligence Perspectives in Intelligent Systems, pp. 1–11. doi:10.1007/978-3-319-33625-1_1.

van Os, R., 2024. Optimizing Conflict-Free-Transportation-Constrained Flexible Manufacturing Systems. Master's thesis. Eindhoven University Of Technology.

Potts, C.N., 1980. Technical Note—Analysis of a Heuristic for One Machine Sequencing with Release Dates and Delivery Times. Operations Research 28, 1436–1441. doi:10.1287/opre.28.6.1436.

Qiu, L., Hsu, W.J., Huang, S.Y., Wang, H., 2002. Scheduling and routing algorithms for AGVs: A survey. International Journal of Production Research 40, 745–760. doi:10.1080/00207540110091712.

Reddy, B.S.P., Rao, C.S.P., 2006. A hybrid multi-objective GA for simultaneous scheduling of machines and AGVs in FMS. The International Journal of Advanced Manufacturing Technology 31, 602–613. doi:10.1007/s00170-005-0223-6.

Riazi, S., Diding, T., Falkman, P., Bengtsson, K., Lennartson, B., 2019. Scheduling and Routing of AGVs for Large-scale Flexible Manufacturing Systems, in: 2019 IEEE 15th International Conference on Automation Science and Engineering (CASE), pp. 891–896. doi:10.1109/COASE.2019.8842849.

Riazi, S., Lennartson, B., 2021. Using CP/SMT Solvers for Scheduling and Routing of AGVs. IEEE Transactions on Automation Science and Engineering 18, 218–229. doi:10.1109/TASE.2020.3012879.

Saidi-Mehrabad, M., Dehnavi-Arani, S., Evazabadian, F., Mahmoodian, V., 2015. An Ant Colony Algorithm (ACA) for solving the new integrated model of job shop scheduling and conflict-free routing of AGVs. Computers & Industrial Engineering 86, 2–13. doi:10.1016/j.cie.2015.01.003.

Sun, J., Xu, Z., Yan, Z., Liu, L., Zhang, Y., 2023. An Approach to Integrated Scheduling of Flexible Job-Shop Considering Conflict-Free Routing Problems. Sensors 23, 4526. doi:10.3390/s23094526.

Ulusoy, G., Sivrikaya-Şerifoğlu, F., Bilge, Ü., 1997. A genetic algorithm approach to the simultaneous scheduling of machines and automated guided vehicles. Computers & Operations Research 24, 335–351. doi:10.1016/S0305-0548(96)00061-5.

Umar, U.A., Ariffin, M.K.A., Ismail, N., Tang, S.H., 2015. Hybrid multiobjective genetic algorithms for integrated dynamic scheduling and routing of jobs and automated-guided vehicle (AGV) in flexible manufacturing systems (FMS) environment. The International Journal of Advanced Manufacturing Technology 81, 2123–2141. doi:10.1007/s00170-015-7329-2.

Xie, C., Allen, T.T., 2015. Simulation and experimental design methods for job shop scheduling with material handling: a survey. The International Journal of Advanced Manufacturing Technology 80, 233–243. doi:10.1007/s00170-015-6981-x.

Zhang, H., , Pengju, S., , Fu, Y., 2025. The synchronised asymmetric multiple travelling salesman problem in a job shop with transportation environment. Journal of Control and Decision 0, 1–14. doi:10.1080/23307706.2024.2441871.

Zhang, Q., Manier, H., Manier, M.A., 2012. A genetic algorithm with tabu search procedure for flexible job shop scheduling with transportation constraints and bounded processing times. Computers & Operations Research 39, 1713–1723. doi:10.1016/j.cor.2011.10.007.

Zhang, Q., Manier, H., Manier, M.A., 2013. Metaheuristics for Job Shop Scheduling with Transportation. John Wiley & Sons, Ltd. chapter 17. pp. 465–493. doi:https://doi.org/10.1002/9781118731598.ch17.

Zheng, Y., Xiao, Y., Seo, Y., 2014. A tabu search algorithm for simultaneous machine/AGV scheduling problem. International Journal of Production Research 52, 5748–5763. doi:10.1080/00207543.2014.910628.

Zhou, B., Lei, Y., 2021. Bi-objective grey wolf optimization algorithm combined Levy flight mechanism for the FMC green scheduling problem. Applied Soft Computing 111, 107717. doi:10.1016/j.asoc.2021.107717.