Solving Logic-Based Benders Decomposition Master Problems with Constraint Programming and Domain-Independent Dynamic Programming

Jiachen Zhang and J. Christopher Beck

Department of Mechanical and Industrial Engineering, University of Toronto, Canada.

Contributing authors: jasonzjc@mie.utoronto.ca; jcb@mie.utoronto.ca;

Abstract

We investigate using Constraint Programming (CP) and Domain-Independent Dynamic Programming (DIDP) to solve the master problem in Logic-based Benders Decomposition (LBBD) models, focusing on the challenge of feasibility cut formulation. For CP, we exploit variable assignment bounds, variable assignment counting, and global constraints to construct three combinatorial cut encodings. For the state-based DIDP model, we propose two cut encoding approaches: using additional preconditions of state transitions or adding state constraints. Each of these approaches can be modeled using integer numeric variables or set variables, resulting in four novel encodings. We apply the three CP variants and four DIDP variants to the LBBD model of the simple assembly line balancing problems with sequence-dependent setup times type-1 (SUALBP-1). Experimental results show all approaches outperform a mixed-integer programming (MIP) based master problem and the state-of-the-art monolithic MIP model, with the three CP variants being superior to all of the DIDP approaches. Though the evaluation is specific to SUALBP-1, the proposed approaches can be used for other problems where LBBD with feasibility cuts is applicable.

Keywords: constraint programming, domain-independent dynamic programming, logic-based Benders decomposition, assembly line balancing, sequence-dependent setup

1 Introduction

Logic-Based Benders Decomposition (LBBD) is one of the most powerful and convenient patterns of problem decomposition for solving combinatorial optimization problems [1]. While the most common combination within the Constraint Programming (CP) literature uses Mixed Integer Programming (MIP) for master problems and CP for subproblems [2], LBBD is compatible with various modeling and solving techniques. For example, subproblems have been modeled and solved with Satisfiability Modulo Theories (SMT) [3], Binary Decision Diagrams [4], and problem-specific algorithms [5, 6]. However, work investigating modeling and solving methods other than MIP for master problems in LBBD is sporadic [7]. In this paper, we explore modeling and solving LBBD master problems with methods different from MIP.

As a constraint-based formalism, CP can readily accept cuts encoded as linear constraints. However, linear constraints tend to propagate weakly, resulting in poor master problem performance. The encoding methods proposed in this paper are more combinatorial and focus on key decision variables in the global constraints of the master problem CP model. As CP is competitive with MIP across a number of optimization problems [8], when the master problem is of the form that is better solved with CP, a CP-based master problem may outperform a corresponding MIP master problem if a good cut formulation can be achieved.

Domain-Independent Dynamic Programming (DIDP) is a recent exact framework to model and solve combinatorial optimization problems [9, 10]. Its success on well-known problems motivates us to investigate using DIDP for master problems in the LBBD framework. Since a DIDP model is defined as a state-transition system, encoding Benders cuts in DIDP differs fundamentally from the constraint-based encoding in MIP and CP.

As a case study, we use assembly line balancing problems with sequence-dependent setup times type-1 (SUALBP-1) [11]. The natural decomposition for this problem is to solve the Simple Assembly Line Balancing Problem type-1 (SALBP-1) as the master problem and to solve a traveling salesman problem with precedence constraints as a subproblem. Previous work shows that both CP and DIDP can outperform MIP for SALBP-1 [8], thus this choice allows us to test whether cuts can be formulated to maintain this advantage.

This paper is an extended version of a conference paper [12]. Our new contributions beyond that paper are as follows:

1. We apply cut strengthening to the Benders feasibility cuts to further improve the performance of the proposed approaches.
2. We re-run all the experiments with cut strengthening and the latest versions of Gurobi, didp-rs,[1] and CP Optimizer.
3. We analyze the difference in LBBD iterations among the seven LBBD models and hypothesize that solving the master problem with DIDP results in fewer iterations due to similarities in consecutive master solutions.

Our overall contributions are summarized as follows.

[1] https://github.com/domain-independent-dp/didp-rs: a Rust-based implementation of Dynamic Programming Description Language (DyPDL) and solvers for DIDP.

1. We develop an LBBD algorithm for SUALBP-1 with SALBP-1 master problem and TSP subproblems with precedence constraints.
2. We develop three novel representations of feasibility cuts for SUALBP-1 tailored to CP-based master problems.
3. We introduce four novel approaches for encoding Benders feasibility cuts in DIDP-modeled LBBD master problems, leveraging integer or set variables to represent transition preconditions or state constraints. Applying these approaches to SUALBP-1, we formulate four feasibility cut encodings specifically designed for DIDP-based master problems.
4. We obtain superior experimental results for SUALBP-1 when solving master problems with CP and DIDP compared to MIP, with CP emerging as the best-performing approach. We provide comprehensive statistical analysis and insights regarding the effectiveness of our seven novel cut formulations.

This paper is structured as follows. Section 2 provides the necessary background, while Section 3 presents the logic-based Benders decomposition of SUALBP-1, featuring MIP models for master problems and DIDP models for subproblems. In Section 4, we introduce three novel CP models for LBBD master problems of SUALBP-1. Section 5 formally defines the four feasibility cut encodings in DIDP, followed by Section 6 which details the application of these four encoding methods to SUALBP-1. Our experimental results and analysis are presented in Section 7, with a discussion of the proposed approaches and findings in Section 8. Finally, we offer our conclusions in Section 9.

2 Background

In this section, we first review the LBBD framework and the DIDP paradigm briefly, followed by an introduction and a literature review for the SUALBP-1.

2.1 Logic-Based Benders Decomposition

Logic-Based Benders Decomposition (LBBD) applies to problems that can be formulated as

$$\min_{\mathbf{x},\mathbf{y}}\{f(\mathbf{x},\mathbf{y})|C(\mathbf{x},\mathbf{y}),\mathbf{x}\in D_x,\mathbf{y}\in D_y\} \tag{1}$$

where $\mathbf{x}$ and $\mathbf{y}$ are decision variables in the domains D_x and D_y, while $f(\mathbf{x},\mathbf{y})$ and $C(\mathbf{x},\mathbf{y})$ represent the objective function and a set of constraints for these variables, respectively [13]. The variables are divided into two groups and, once some of the variables are fixed by solving a master problem and setting $\mathbf{x}=\overline{\mathbf{x}}$, where $\overline{\mathbf{x}}$ are the fixed values of $\mathbf{x}$, the remaining subproblem is defined, often in the form of multiple independent subproblems. The subproblem (SP) has the form

$$SP(\overline{\mathbf{x}}) = \min_{\mathbf{y}}\{f(\overline{\mathbf{x}},\mathbf{y})|C(\overline{\mathbf{x}},\mathbf{y}),\mathbf{y}\in D_y\}. \tag{2}$$

The SP solution is then analyzed to infer a function $B_{\overline{\mathbf{x}}}(\mathbf{x})$ that provides a lower bound on $f(\mathbf{x},\mathbf{y})$ for any given $\mathbf{x}\in D_x$. The bound is sharp for $\mathbf{x}=\overline{\mathbf{x}}$, i.e., $B_{\overline{\mathbf{x}}}(\mathbf{x}) = SP(\overline{\mathbf{x}})$ [1].

Each iteration of LBBD begins by solving a Master Problem (MP):

$$MP(\overline{\mathbf{X}}) = \min_{x,\beta}\{\beta | \beta \geq B_{\overline{\mathbf{x}}}(\mathbf{x}), \forall \overline{\mathbf{x}} \in \overline{\mathbf{X}}, \mathbf{x} \in D_x\} \tag{3}$$

where the inequalities $\beta \geq B_{\overline{\mathbf{x}}}(\mathbf{x})$ are Benders cuts obtained from the subproblem solutions given $\mathbf{x} = \overline{\mathbf{x}}$. $\overline{\mathbf{X}}$ is the set of master problem solutions and is usually empty initially.

Defining ϕ^* as the optimal value of the original problem (1), the optimal MP value $MP(\overline{\mathbf{X}})$ is a lower bound on ϕ^*. If $\overline{\mathbf{x}}$ is an optimal MP solution, the corresponding subproblem is then solved to obtain $SP(\overline{\mathbf{x}})$ as an upper bound on ϕ^*, and a Benders cut $\beta \geq B_{\overline{\mathbf{x}}}(\mathbf{x})$ for the master problem, with $\overline{\mathbf{x}}$ added to $\overline{\mathbf{X}}$. The process repeats until the lower and upper bounds converge, i.e., until $MP(\overline{\mathbf{X}}) = \min_{\overline{\mathbf{x}} \in \overline{\mathbf{X}}} SP(\overline{\mathbf{x}})$. The convergence is guaranteed after a finite number of iterations, if D_x is finite [13].

In general, there are two types of LBBD Benders cuts. When a subproblem is an optimization problem, we deduce a lower bound on ϕ^* in the form of a Benders optimality cut [14]. When a subproblem is a feasibility problem, a set of MP solutions are pruned by the corresponding Benders feasibility cut [15] according to the SP solution associated with $\overline{\mathbf{x}}$. In this work, we focus on encoding *Benders feasibility cuts*.

2.2 Domain-Independent Dynamic Programming

A DIDP model is described by Dynamic Programming Description Language (DyPDL), a solver-independent formalism to define a dynamic programming (DP) model [10]. In DyPDL, a problem is represented by states and transitions between states. A solution of the problem corresponds to a sequence of transitions satisfying particular conditions.

A DyPDL model is a tuple $\langle \mathcal{V}, S^0, \mathcal{T}, \mathcal{B}, \mathcal{C}, h \rangle$, where $\mathcal{V}$ is the set of state variables, S^0 is a state called the target state, $\mathcal{T}$ is the set of transitions, $\mathcal{B}$ is the set of base cases, $\mathcal{C}$ is the set of state constraints, and h is the set of dual bounds. A state variable is either an element, set, or numeric variable. A numeric state variable v may have a preference such as *less* (*more*), i.e., a state having smaller (larger) v dominates another state if the other state variables have the same value in the two states. Such a variable is called a resource variable.

Given a set of state variables $\mathcal{V} = \{v_1, ..., v_n\}$, a state is a tuple of values $S = (d_1, ..., d_n)$ where $d_i \in D_{v_i}$ for $i = 1, ..., n$, i.e., a state is a complete assignment to state variables. We denote the value d_i of variable v_i in state S by $S[v_i]$. Intuitively, the target state is the start of the state transition system and a base state is a goal, i.e., the end of the state transition system. State constraints are relations on state variables that must be satisfied by *all* states in a solution path.

A transition τ is a 4-tuple $\langle \text{eff}_\tau, \text{cost}_\tau, \text{pre}_\tau, \text{forced}_\tau \rangle$ where eff_τ is the set of effects, cost_τ is the cost, pre_τ is the set of preconditions, and $\text{forced}_\tau \in \{\top, \bot\}$, where $\top$ represents *true* and $\bot$ represents *false*. The preconditions of a transition define when we can use it while the effects of a transition define what the state variables become if the transition fires. For detailed DIDP models of various optimization problems, please see existing DIDP papers [8, 10].

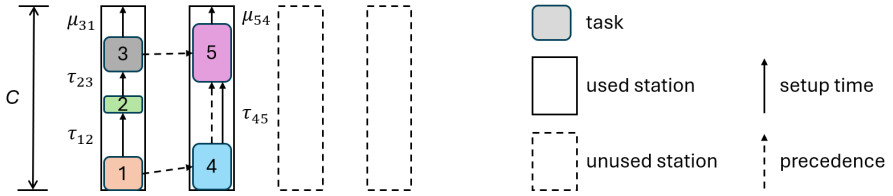

Fig. 1: Example of SUALBP-1.

2.3 SUALBP-1

The Simple Assembly Line Balancing Problem (SALBP) is a well-studied production planning problem [16]. As setup operations such as tool changes, curing, or cooling processes are often required between consecutive tasks in real production lines [17], SUALBP incorporates setup times into SALBP [18], as shown in Fig. 1.

2.3.1 Problem Definition

SUALBP-1 consists of n assembly tasks, partially ordered with precedence constraints, that require processing on m ordered assembly stations. The tasks on a machine must all sequentially execute within the cycle time c. In SUALBP-1, the cycle time c is fixed and the objective is to minimize the number of stations m. Though all stations can perform all assembly tasks, if a task is assigned to station j, all its successors as defined by the precedence constraints must be assigned to the same or subsequent stations (i.e., $j, j+1, ..., m$). Tasks assigned to the same station must also be sequenced to satisfy the precedence constraints, if any. The deterministic processing time of a task is provided a priori. However, the setup before a task (*forward setup*) is dependent upon the previous task in the processing sequence of the station it is assigned to. There is also a sequence-dependent setup (*backward setup*) from the last task on a machine to the first task on the same machine to model the setup required between the end of a cycle and the start of the next one.

The setups are not symmetric, i.e., the setup time from task i to j might be different from that from task j to i. Nevertheless, the setups satisfy the triangle inequality. The decisions to be made for SUALBP-1 are (i) the assignment of tasks to stations; and (ii) the sequence of the tasks assigned to each station. We use the notation proposed

Table 1: Notation and definition for SUALBP-1 [11].

Notation	Definition
$i, j \in V$	index and set of tasks
$k \in K$	index and set of stations
t_i	execution time for task $i \in V$
P_i (P_i^*)	set of direct (all) predecessors of task $i \in V$
S_i (S_i^*)	set of direct (all) successors of task $i \in V$
c	the cycle time
$\overline{m}$ ($\underline{m}$)	upper (lower) bound on the number of stations
τ_{ij} (μ_{ij})	forward (backward) setup times from task $i \in V$ to task j
$\underline{\tau}_i$ ($\underline{\mu}_i$)	the smallest forward (backward) setup time from any task to task $i \in V$
$\underline{t}_i$	a lower bound of the time contribution by task i, i.e., $\underline{t}_i = t_i + \min(\underline{\tau}_i, \underline{\mu}_i)$

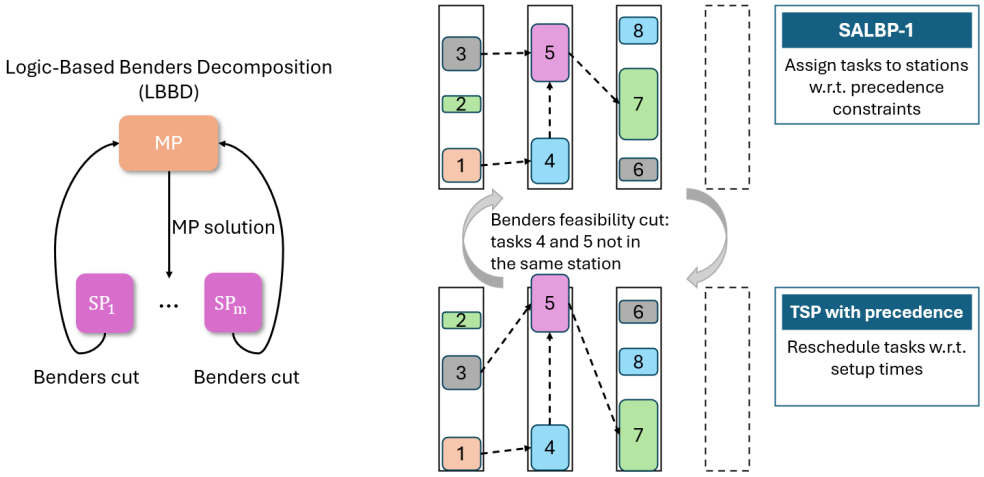

Fig. 2: LBBD for SUALBP-1.

by Esmaeilbeigi et al. [11], as shown in the Table 1 for SUALBP-1. To obtain all the parameters in the table, we adopt the preprocessing techniques in the literature [10, 11, 14].

SUALBP-1 has been solved with a number of approaches including MIP [11] and heuristics [19]. The state-of-the-art MIP model is the Second Station-Based Formulation (SSBF) [11] defined in the Appendix A. The model uses two-indexed binary variables to encode task assignment, three-indexed binary variables to represent the precedence relations of pairs of tasks on a station, and auxiliary variables to help express the objective and constraints.

There is no existing LBBD approach specifically designed for SUALBP-1. The closest work is an LBBD algorithm for mixed-model assembly line balancing problem with sequence-dependent setups [15] that can be adapted (with significant simplification) to SUALBP-1. We discuss this model in Section 3.

In our parallel work [20], new state-of-the-art results are found with a monolithic DIDP model. Since our focus is on cut encoding in LBBD, we return to these results in the discussion.

3 LBBD for SUALBP-1

In this section, we present the logic-based Benders decomposition (LBBD) for SUALBP-1. LBBD has been applied to many variants of the assembly line balancing problems (ALBPs) [14, 15, 21]. The original problem is usually decomposed into an assignment master problem and multiple scheduling subproblems. We develop a similar decomposition approach for SUALBP-1, illustrated in Fig. 2.

In the decomposition of SUALBP-1, the master problem assigns tasks to stations, considers the precedence constraints between tasks, and minimizes the number stations used. The subproblems schedule the tasks within each station and take the setup times into account, but only towards constraint satisfaction instead of optimization.

Algorithm 1 LBBD for SUALBP-1

Input: a set of tasks V, a set of stations K

Output: complete task assignments to stations and schedules of all stations with precedence and cycle time constraints satisfied, if the problem is feasible; or two empty sets if the problem is infeasible

Initialize: o $\leftarrow False, C \leftarrow \emptyset$ ▷ Initialize the overall feasibility and Benders cuts

1: **while** $o = False$ **do**
2: $o = True$
3: $(f_{mp}, \{SP_k\}) \leftarrow \texttt{solveMp}(V, K, C)$ ▷ solve master problem, define subproblems
4: **if** $f_{mp} = False$ **then**
5: **Return** $\emptyset, \emptyset$ ▷ return empty sets due to infeasible problem
6: **end if**
7: **for** $k \in \mathcal{K}$ **do**
8: $(f_k, SC_k) \leftarrow \texttt{solveSp}(SP_k)$ ▷ solve subproblem, get station schedule
9: **if** $f_k = False$ **then**
10: $c_k = \texttt{getBenders}(SP_k)$ ▷ get Benders cuts if infeasible subproblem
11: $C \leftarrow C \cup \{c_k\}$
12: $o = False$
13: **end if**
14: **end for**
15: **end while**
16: **Return** $\{SP_k\}, \{SC_k\}$ ▷ return task assignments and station schedules

Whenever the best schedule (the shorter the station time the better) of the tasks in a station violates the fixed cycle time, a Benders feasibility cut is generated and added to the master problem.

The overall LBBD approach for SUALBP-1 is summarized in Algorithm 1. The algorithm initializes the overall feasibility to be false and the set of Benders cuts to be empty. Then the master problem and subproblems are iteratively solved until the overall feasibility is established, as shown in line 3 and 8. If the master problem is infeasible, the algorithm returns empty task assignments and schedules, as shown in lines 4 and 5. Otherwise, the set $\{SP_k\}$ is a set of complete task assignments to stations and defines the subproblems. If the subproblem of station k is feasible, SC_k is the corresponding feasible schedule of tasks assigned. Otherwise, Benders cuts are generated and added to the set of cuts, as shown in line 10 and 11. Finally if all subproblems are feasible, the algorithm returns the complete task assignments and station schedules, as shown in line 16.

Without any Benders cuts, the master problem is identical to the Simple Assembly Line Balancing Problem type-1 (SALBP-1) [22]. As the LBBD master problem is usually modeled and solved as a MIP, we first present the MIP model for the master problem.

3.1 MIP Model for the Master Problem

For the master problem, instead of a simplified MIP formulation proposed by Akpinar et. al [15] we use the state-of-the-art NF4 MIP formulation [23] for SALBP-1 and replace t_i by $\underline{t}_i$ to express the subproblem relaxation and accelerate the convergence of the decomposition. The decision variables of the MIP model are as follows:

- $x_{i,k}$ - binary variable that is 1 if task i is assigned to station k and 0 otherwise.
- y_k - binary variable that is 1 if station k is used and 0 otherwise.

Using the notation defined in Table 1, the MIP model for the master problem (MP) is as follows:

$$\min_{x,y} \quad \sum_{k=1,\dots,\overline{m}} y_k, \tag{4a}$$

$$\text{s.t.} \quad \sum_{k=E_i,\dots,L_i} x_{i,k} = 1, \qquad \forall i \in V, \tag{4b}$$

$$\sum_{k=E_i,\dots,L_i} x_{i,k} \cdot k \geq d_{j,i} + \sum_{k=E_j,\dots,L_j} x_{j,k} \cdot k, \qquad \forall i \in V, \forall j \in P_i^*, \tag{4c}$$

$$\sum_{i \in V, \text{ if } E_i \leq k \leq L_i} x_{i,k} \cdot \underline{t}_i \leq c, \qquad \forall k = 1, \dots, \underline{m}, \tag{4d}$$

$$\sum_{i \in V, \text{ if } E_i \leq k \leq L_i} x_{i,k} \cdot \underline{t}_i \leq c \cdot y_k, \qquad \forall k = \underline{m}+1, \dots, \overline{m} \tag{4e}$$

$$y_k = 1, \qquad \forall k = 1, \dots, \underline{m}, \tag{4f}$$

$$y_{k+1} \leq y_k, \qquad \forall k = \underline{m}, \dots, \overline{m}-1, \tag{4g}$$

$$x_{i,k} \in \{0,1\}, \qquad \forall i \in V, \forall k = E_i, \dots, L_i, \tag{4h}$$

$$y_k \in \{0,1\}, \qquad \forall k = 1, \dots, \overline{m}. \tag{4i}$$

The objective function (4a) minimizes the number of stations used. Constraints (4b) ensure that each task is assigned to exactly one station. Here E_i is defined as a lower bound on the number of stations required to schedule task i:

$$E_i = \left\lceil \frac{\underline{t}_i + \sum_{j \in P_i^*} \underline{t}_j}{c} \right\rceil. \tag{5}$$

L_i is an upper bound on the number of stations required to schedule task i:

$$L_i = \overline{m} + 1 - \left\lceil \frac{\underline{t}_i + \sum_{j \in F_i^*} \underline{t}_j}{c} \right\rceil. \tag{6}$$

Note that these terms are calculated based on $\underline{t}_i$ instead of t_i, because we want to incorporate the estimated setup times into task processing times to better approximate the subproblems.

Constraints (4c) guarantee that the precedence relations between tasks are satisfied. Constraints (4d) and (4e) ensure that the cycle time is not exceeded. These two

constraints also serve as the subproblem relaxation in the MP, because the station load time here is just a lower estimate as the sequence-dependent setup times can only be exactly calculated in the subproblems.

Constraints (4f) and (4g) assure that the first $\underline{m}$ stations are used and that the remaining stations are used with a preference of earlier stations to break symmetry. Finally, constraints (4h) and (4i) define the domains of the decision variables. Note that variables $x_{i,k}$ are not defined for $k < E_i$ and $k > L_i$, because the task i cannot be assigned to those stations.

3.2 DIDP Model for the Subproblem

In the LBBD framework for SUALBP-1, the MP solution assigns tasks to stations and hence determines the number of stations used, i.e., the objective value. Thus, each subproblem is a constraint satisfaction problem to find a schedule of the tasks, considering the precedence relation between tasks, the sequence-dependent setup times, and the cycle time. The task processing times are not included in the subproblem as they are constant after the task assignment is given; the sum of processing times is therefore subtracted from the cycle time when evaluating feasibility. The subproblem has the structure of the Traveling Salesman Problem (TSP) with precedence constraints. For this constrained TSP variant, our preliminary investigations showed that the DIDP model (presented below) and solver outperform the CP and MIP counterparts and we hence use DIDP as the sole subproblem solver. The state variables, base cases, and the recursive function are as follows.

State variables. For station j, the DIDP model has the following state variables:

- U: set variable for unscheduled tasks. In the target state, $U = \mathcal{I}^j$.
- s: element variable for the current task, with its value in $\mathcal{I}^j$. In the target state, $s = d_s$, where d_s is a dummy task with setup times from and to any other tasks set to zero.
- f: element variable for the first task, with its value in $\mathcal{I}^j$. In the target state, $f = d_s$.

Base cases. The base case of the DIDP model is: $U = \emptyset \ \wedge \ s = d_s$.

Recursive function. We use $\mathcal{V}(U, s, f)$ to represent the cost of a state. Let P_i^{j*} be the set of predecessors of task i on station j. Let $U' = \{i \in \mathcal{I}^j \mid \mathcal{I}^j \cap P_i^{j*} = \emptyset\}$ be the set of tasks whose predecessors are not assigned to station j.

$$\texttt{compute } \mathcal{V}(I^j, d_s, d_s) \tag{7a}$$

$$\mathcal{V}(U,s,f) = \begin{cases} 0 & \text{if } U = \emptyset \ \wedge \ s = d_s, \quad \text{(i)} \\ \mu_{sf} + \mathcal{V}(U, d_s, d_s) & \text{else if } U = \emptyset \ \wedge \ s \neq d_s, \quad \text{(ii)} \\ \mu_{si} + \min_{i \in U'} \mathcal{V}(U \backslash \{i\}, i, f) & \text{else if } U' \neq \emptyset \ \wedge \ s \neq d_s, \quad \text{(iii)} \\ \min_{i \in U'} \mathcal{V}(U \backslash \{i\}, i, i) & \text{else,} \quad \text{(iv)} \end{cases} \tag{7b}$$

$$\mathcal{V}(U,s,f) \geq \max \begin{cases} \underline{\mu}_f + \sum_{i \in U} \underline{\tau}_i, & \text{if } s = d_s, \quad \text{(i)} \\ 0, & \text{else.} \quad \text{(ii)} \end{cases} \tag{7c}$$

Case (7b-i) refers to the base case, while (7b-iv) corresponds to assigning the first task to the empty station. Case (7b-iii) represents assigning the next task to the station and adding the corresponding setup time. (7b-ii) represents closing the station and adding the setup time to the first task. (7c) is the dual bound [10].

Although this DIDP model is designed for optimization problems, there exist anytime DIDP solvers [8] implementing approaches such as complete anytime beam search [24] and anytime column progressive search [25]. With such solvers, we can set the primal bound and terminate search with the first feasible solution or when infeasibility is proved.

Note that the first master problem solution that is feasible for each subproblem is an optimal solution to the original problem. The process to reach that point usually takes several LBBD iterations and there is no intermediate feasible primal solution found before convergence. It is not uncommon in LBBD formulations (e.g., [6]) to devise a heuristic approach to transform a globally infeasible master solution into a feasible one in order to incrementally find sub-optimal global solutions. Such a heuristic could be employed regardless of the methodology to solve master problems. As our work focuses on the formulation of feasibility cuts for alternative master problem approaches, we leave the investigation of such heuristics for future work.

3.3 Benders Feasibility Cuts as Linear Constraints

Whenever the subproblem is infeasible, a Benders feasibility cut is generated and added to the master problem. For the cut representation, linear constraints [15] are directly applied. As $\mathcal{I}^j$ is the set of MP variable indices that appear in the j-th subproblem, the corresponding Benders feasibility cut in the MIP form is as follows:

$$\sum_{i \in \mathcal{I}^j} x_{i,k} \leq |\mathcal{I}^j| - 1, \ \forall k \in \{k \in K \mid E_i - 1 \leq k \leq L_i, \forall i \in \mathcal{I}^j\}. \tag{8}$$

Chu and Xia defined a valid Benders cut as a logical expression having two properties [26]:

P1: the cut must exclude the current MP solution if it is not globally feasible. (9a)

P2: the cut must not remove any globally feasible solutions. (9b)

P1 ensures finite convergence if the MP variables have finite domains. P2 assures optimality since the cut never removes globally feasible solutions. Here we provide a proof of the two properties for the Benders cut (8).

Proposition 1. *The linear constraint cut encoding* (8) *is valid.*

Proof. For any cut $j \in \mathcal{J}$, $\sum_{i \in \mathcal{I}^j} x_{i,k}$ counts the number of tasks that appear in the current station. With the count smaller than $|\mathcal{I}^j|$, as implemented in (8), the same set of tasks are never assigned to the same station and so P1 is satisfied. Since the solutions removed by this encoding are the solutions with the set of tasks $\mathcal{I}^j$ assigned to any station, they are all infeasible globally as the task processing times and setup times are independent of stations, and thus P2 is satisfied. □

The Benders cut (8) is stronger than the traditional combinatorial no-good cuts [27] because it cuts off multiple infeasible solutions at once instead of just one solution. In particular, any solution with any station containing all the tasks in $\mathcal{I}^j$ is pruned. It behaves similar to no-good cuts in the context of CSP [28, 29] and SAT [30].

The Benders cut (8) can be strengthened by multiple techniques [31–33]. A straightforward approach is to re-solve the subproblem with a subset of $\mathcal{I}^j$, say $\mathcal{I}^{j'} \subseteq \mathcal{I}^j$. If the subproblem is infeasible, we can obtain a stronger cut with respect to $\mathcal{I}^{j'}$ [32]. We use a similar and greedy approach to strengthen the Benders cuts for SUALBP-1.

Whenever a Benders cut is found, the tasks in the corresponding set $\mathcal{I}^j$ are sorted in ascending order of $\underline{t}_i$. Then by removing the first task i in the sorted set, a subset of tasks $\mathcal{I}^{j'} \subseteq \mathcal{I}^j$ is obtained and evaluated by re-solving the subproblem. If $\mathcal{I}^{j'}$ leads to an infeasibility, we get a stronger cut corresponding to $\mathcal{I}^{j'}$. We proceed to remove the next smallest task in terms of $\underline{t}_i$ from $\mathcal{I}^{j'}$ and evaluate the feasibility of the pruned subset. On the contrary, if $\mathcal{I}^{j'}$ results in a feasible subproblem, we return the task i to $\mathcal{I}^{j'}$ and remove the next smallest task in terms of $\underline{t}_i$, followed by another re-solving run of the subproblem. The procedure is terminated once all the tasks have been considered. As our preliminary experiments showed better performance when including this greedy tasks removal approach, we use it in all the LBBD models studied in this work.[2]

4 CP-LBBD for SUALBP-1

In this section, we present three LBBD formulations for SUALBP-1 with CP master problems and Benders feasibility cuts.

4.1 CP Master Problem

As stated in Section 3, the master problem is equivalent to the SALBP-1. For SALBP-1, Kuroiwa and Beck [10] improved the CP model proposed by Bukchin and Raviv [34] by using a `Pack` global constraint. Our models differ from theirs in two ways:

1. t_i is replaced by $\underline{t}_i$ for task i to model a subproblem relaxation in the master problem.
2. Three different combinatorial formulations of Benders feasibility cuts are used, one formulation in each model.

We define T_i as a lower bound on the number of stations between the station of task i and the last station, inclusive (namely the tail):

$$T_i = \left\lfloor \frac{\underline{t}_i - 1 + \sum_{j \in S_i^*} \underline{t}_j}{c} \right\rfloor. \tag{10}$$

Also, d_{ij} is a lower bound on the number of stations between the stations of tasks i and j, inclusive:

$$d_{ij} = \left\lceil \frac{\underline{t}_i + \underline{t}_j - 1 + \sum_{v \in S_i^* \cap P_j^*} \underline{t}_v}{c} \right\rceil. \tag{11}$$

[2] Numerical results without this cut strengthening can be found in Zhang & Beck [12].

Let z be an integer decision variable representing the number of stations, x_i be an integer decision variable for the station that task i is assigned to, and y_k be an integer decision variable for the sum of the lower bound time contribution of tasks scheduled in station k. Then the CP model for the master problem, CP-MP, is as follows:

$$\min\ z \tag{12a}$$

$$\text{s.t. } \texttt{Pack}(\{y_k|k \in K\}, \{x_i|i \in V\}, \{\underline{t}_i|i \in V\}), \tag{12b}$$

$$0 \le y_k \le c, \qquad \forall k \in K, \tag{12c}$$

$$E_i - 1 \le x_i \le z - 1 - T_i, \qquad \forall i \in V, \tag{12d}$$

$$x_i + d_{ij} \le x_j, \qquad \forall j \in V, \forall i \in P_j^*, \nexists v \in S_i^* \cap P_j^* : d_{ij} \le d_{iv} + d_{vj}. \tag{12e}$$

The constraint (12b) utilizes the `Pack` global constraint [35] that models the condition where a set of tasks must be packed into a limited number of stations without exceeding the total load of each station.[3] Specifically, the first argument $\{y_k|k \in K\}$ represents the total loads of all stations. The second argument $\{x_i|i \in V\}$ expresses the task assignment for all the tasks. The third argument $\{\underline{t}_i|i \in V\}$ stands for the load contribution (the lower bound of the time contribution) of these tasks. The `Pack` global constraint enforces the following equation:

$$y_k = \sum_{i \in V, x_i = k} \underline{t}_i, \quad \forall k \in K. \tag{13}$$

Constraints (12c) and (12d) state the domains of y_k and x_i, where E_i is the defined as in (5). Constraints (12b) and (12c) together ensure that the total task time on each station does not exceed the cycle time. Constraints (12e) are an enhanced version of the precedence constraint using d_{ij}.

4.2 CP Formulations for Benders Feasibility Cuts

For SUALBP-1, we develop three combinatorial CP formulations for Benders feasibility cuts by using variable assignment bounds, a `Count_Different` expression, and a `Pack` constraint.

4.2.1 Assignment Bound Encoding

For cut $j \in \mathcal{J}$, recall that $\mathcal{I}^j$ is the set of tasks assigned to the station that cannot be scheduled within the cycle time. Then the j-th Benders feasibility cut based on setting an upper bound for the number of decision variables that are assigned to a specific value is as follows:

$$\sum_{i \in \mathcal{I}^j} (x_i = k) \le |\mathcal{I}^j| - 1, \ \forall k \in \{k \in K \mid E_i - 1 \le k \le \overline{m} - 1 - T_i, \forall i \in \mathcal{I}^j\}. \tag{14}$$

The variable x_i represents the station that task i is assigned to. The set $\{k \in K \mid E_i - 1 \le k \le \overline{m} - 1 - T_i, \forall i \in \mathcal{I}^j\}$ excludes station k when there exists a task i that cannot

[3] https://ibmdecisionoptimization.github.io/docplex-doc/cp/docplex.cp.modeler.py.html

be assigned to k due to precedence relations, i.e., $k < E_i - 1$ or $k > \overline{m} - 1 - T_i$. The restriction is valid because the cut is trivially satisfied if only a subset of $\mathcal{I}^j$ can be assigned to station k.

Proposition 2. *Cut (14) is valid.*

Proof. Recall that Chu and Xia [26] defined a valid Benders cut as a logical expression having two properties that are introduced in (9a) and (9b). As $x_i = k$ specifies the station assignment and there are $|\mathcal{I}^j|$ tasks in $\mathcal{I}^j$, the cut prevents the tasks in $\mathcal{I}^j$ from being all assigned to the same station and satisfies P1. Since the solutions removed by this encoding are all infeasible globally with the set of tasks $\mathcal{I}^j$ assigned to any station, P2 is satisfied. □

4.2.2 Assignment Counting Encoding

Encoding (14) is similar to the linear constraints that are commonly used in MIP. But CP provides more compact encodings via constraint-based expressions and global constraints that can also take advantage of strong constraint propagation and domain filtering in CP.

The constraint-based expression `Count_Different` takes a list of (more than one) variables as input and returns the number of distinct values of these variables [36]. The j-th cut based on `Count_Different` is as follows:

$$\texttt{Count\_Different}(\{x_i | i \in \mathcal{I}^j\}) \geq 2. \tag{15}$$

Proposition 3. *Cut (15) is valid.*

Proof. This constraint guarantees that the number of distinct values in $\{x_i | i \in \mathcal{I}^j\}$ is at least 2 and implies (14). Thus, P1 and P2 are satisfied. □

4.2.3 Global Constraint Encoding

The j-th cut based on the global constraint `Pack` is as follows:

$$\texttt{Pack}(\{w_k | k \in K\}, \{x_i | i \in \mathcal{I}^j\}, \{\mathbf{1}_i | i \in \mathcal{I}^j\}), \tag{16}$$

where $0 \leq w_k \leq |\mathcal{I}^j| - 1$ and $\mathbf{1}_i = 1, \forall i \in \mathcal{I}^j$.

In the `Pack` constraint, the first argument $\{w_k | k \in K\}$ represents the total loads of all stations. The second argument $\{x_i | i \in \mathcal{I}^j\}$ expresses the task assignment for the relevant tasks in the cut j. The third argument $\{\mathbf{1}_i | i \in \mathcal{I}^j\}$ stands for the load contribution of these tasks. Thus, the domain of w_k ensures that each station can accommodate at most $|\mathcal{I}^j| - 1$ tasks in $\mathcal{I}^j$ given that each task contributes only 1 to the station load.

Proposition 4. *Cut (16) is valid.*

Proof. Since $\mathbf{1}_i$ has unit length and $w_k \leq |\mathcal{I}^j| - 1$, this cut assures that no more than $|\mathcal{I}^j| - 1$ tasks in $\mathcal{I}^j$ are assigned to any station and satisfies P1. Since the solutions removed by this encoding are all infeasible globally with the set of tasks $\mathcal{I}^j$ assigned to any station, P2 is satisfied. □

The CP-LBBD models with cut (14), (15), and (16) are referred as CP-LBBD$_a$, CP-LBBD$_c$, and CP-LBBD$_p$, corresponding to 'assignment', 'count', and 'pack', respectively. We have applied the same cut strengthening technique described in Section 3.3 to the three CP-LBBD models.

Note that the CP-LBBD$_c$ and CP-LBBD$_p$ models, unlike CP-LBBD$_a$, cannot prune unnecessary cuts for a station k if it does not satisfy the condition $E_i - 1 \leq k \leq \overline{m} - 1 - T_i, \forall i \in \mathcal{I}^j$, because the `Count_Different` expression and the `Pack` constraint do not have the flexibility to exclude a specific station.

5 Feasibility Cut Encoding in DIDP-LBBD

In this section, we present four *general* encoding methods for the feasibility cuts in the LBBD framework with DIDP master problems.

LBBD is often defined with MIP master problems and we first use MIP context. Let $\mathbf{x}$ be the decision variables in the master problem and let $\overline{\mathbf{x}}$ be the optimal solution of the latest MP iteration. Let $\mathcal{I}$ be the set of all MP variable indices and let $\mathcal{I}^j$ be the set of MP variable indices that appear in the j-th subproblem, then the Benders feasibility cut obtained from this subproblem is of the following form:

$$\sum_{i \in \mathcal{I}^j} (x_i = \overline{x}_i) \leq |\mathcal{I}^j| - 1. \tag{17}$$

This form is often formulated as a linear constraint in the MIP master problem and we call it the j-th cut. We define the variable-value pair $(x_i, \overline{x}_i)$ to be *active* if $x_i = \overline{x}_i$ and *inactive* if $x_i \neq \overline{x}_i$.

In DIDP, however, a cut of the form (17) cannot be directly represented with state variables. Thus, instead of adding only a constraint to the DIDP model, we add a new state variable for each cut, with relevant transitions updating the variable value. New preconditions or state constraints are also added to enforce the logic of Benders cuts.

5.1 Counting-based Encoding

Our first two encoding methods are based on integer variables in DIDP. The essence of the encodings is to track the active variable-value pairs, i.e., $(x_i, \overline{x}_i)$ in the LHS of (17). However, the variable x_i is in the MIP form and does not exist in DIDP models. Often, we can still extract the concrete interpretation of the value assignment of x_i from the state variables and transitions in DIDP models.

Here our encoding methods identify the transitions that increase or decrease the LHS of (17), by using a counter that is correspondingly incremented or decremented by the identified transitions. Then we add a precondition or state constraint on this counter to enable the DIDP pruning when the cut (17) is violated.

To formally define our encoding methods, we rely on a mapping from DIDP models to the variable-value pairs. Let $\xi(S)$ be the sequence of transitions from the target state S^0 to state S. Define a function $\mathcal{M}$ that maps $\xi(S)$ to the value assignments of

decision variables $\{x_i|\forall i \in \mathcal{I}\}$ in the MIP form such that:

$$\mathcal{M}(\xi(S)) = \{x_i|\forall i \in \mathcal{I}\}. \tag{18}$$

Here the value of x_i, is interpreted/translated from $\xi(S)$ and can be undetermined, i.e., not assigned any value yet in DIDP since the corresponding decision has not been made. Then we define g^j to be the number of the active variable-value pairs in the left-hand side (LHS) of the cut (17) that can be extracted from $\mathcal{M}(\xi(S))$, i.e.,

$$g^j = \sum_{i \in \mathcal{I}^j} (x_i = \overline{x}_i), \tag{19}$$

where $\mathcal{I}^j$ is the set of decision variable indices in the MIP form mapped from the associated subproblem. Note that $\mathcal{I}$ is added to the DIDP model as a constant set when initializing the model, while $\mathcal{I}^j$ is a constant subset of $\mathcal{I}$.

In the target state, since the $\xi(S^0)$ contains only the target state and no actual decisions have been made, we have $\sum_{i \in \mathcal{I}^j}(x_i = \overline{x}_i) = 0$ and g^j is 0. Let $\mathcal{F}^j$ be the function that updates the value of g^j according to transitions. If the effects eff_τ of transition τ imply that $x_i = \overline{x}_i$ for some $i \in \mathcal{I}^j$ and $x_k \neq \overline{x}_k$ for some $k \in \mathcal{I}^j$, we have

$$\mathcal{F}^j(\tau) = |\mathcal{U}^j_\tau| - |\mathcal{D}^j_\tau|, \tag{20}$$

where $\mathcal{U}^j_\tau$ ($\mathcal{D}^j_\tau$) is the set of the variable indices of the variable-value pairs that are changed from inactive (active) to active (inactive) by transition τ with respect to the j-th cut, with $i \in \mathcal{U}^j_\tau$ and $k \in \mathcal{D}^j_\tau$.

In other words, let S be the state where the preconditions of transition τ are satisfied, and let $S' = S[[\tau]]$ be the state reachable from S by τ, we have

$$(x_i \neq \overline{x}_i \wedge x_i \in \mathcal{M}(\xi(S^{'})), \forall i \in \mathcal{U}^j_\tau) \wedge (x_i = \overline{x}_i \wedge x_i \in \mathcal{M}(\xi(S)), \forall i \in \mathcal{U}^j_\tau) \tag{21a}$$

$$(x_i = \overline{x}_i \wedge x_i \in \mathcal{M}(\xi(S^{'})), \forall i \in \mathcal{D}^j_\tau) \wedge (x_i \neq \overline{x}_i \wedge x_i \in \mathcal{M}(\xi(S)), \forall i \in \mathcal{D}^j_\tau) \tag{21b}$$

Therefore, we have

$$S'[g^j] = S[g^j] + \mathcal{F}^j(\tau). \tag{22}$$

In practice, the implementation of $\mathcal{F}$ depends on the problem and we define the encoding for SUALBP-1 in Section 6.2. With the LHS of cut (17) modeled, we use preconditions or state constraints to model the right-hand side (RHS).

5.1.1 Counting-based Precondition Encoding

Our first method for modeling the RHS of (17) is based on preconditions. Specifically, for the cut with the form (17), we add a precondition for each transition in the DIDP model that can model the Benders cut as follows:

$$S[g^j] + \mathcal{F}^j(\tau) \leq |\mathcal{I}^j| - 1, \tag{23}$$

where τ is the transition. If the precondition is violated, the transition τ is not permitted.

5.1.2 Counting-based State Constraint Encoding

Our second method for modeling the RHS of (17) is based on state constraints that need to be satisfied by all states. The state constraint for the j-th cut is as follows:

$$S[g^j] \leq |\mathcal{I}^j| - 1, \tag{24}$$

where S is any state. A state constraint is evaluated after a state is created but a violated precondition prevents a state from being created.

5.2 Set-based Encoding

Our other two encoding methods are based on set variables in DIDP. Let Ω^j be a set variable that keeps track of the active variable-value pairs in the LHS of the cut (17), according to the mapping $\mathcal{M}$ and the constant set $\mathcal{I}^j$. More specifically, the set variable Ω^j contains an element e_i iff $\hat{x}_i = \overline{x}_i$ is satisfied in a state. In the target state, $\Omega^j = \emptyset$. If the effects eff_τ of transition τ imply that $\hat{x}_i = \overline{x}_i$ for some $i \in \mathcal{I}^j$ and $\hat{x}_k \neq \overline{x}_k$ for some $k \in \mathcal{I}^j$, let $\mathcal{U}^j_\tau$ be the set containing all such i and $\mathcal{D}^j_\tau$ be the set containing all such k, then the Ω^j is changed to

$$(S[\Omega^j] \cup \mathcal{U}^j_\tau) \backslash \mathcal{D}^j_\tau. \tag{25}$$

Let S be a state and $S' = S[[\tau]]$ be the state reachable from S by τ, we have $S'[\Omega^j] = (S[\Omega^j] \cup \mathcal{U}^j_\tau) \backslash \mathcal{D}^j_\tau$. Similar to the counting-based encoding, we use preconditions or state constraints to model the RHS.

5.2.1 Set-based Precondition Encoding

For the cut (17), we add a precondition for each transition that can modify Ω^j in the DIDP model as follows:

$$\mathcal{I}^j \not\subseteq (S[\Omega^j] \cup \mathcal{U}^j_\tau) \backslash \mathcal{D}^j_\tau, \tag{26}$$

where τ is the transition. $(S[\Omega^j] \cup \mathcal{U}^j_\tau) \backslash \mathcal{D}^j_\tau$ gives the value of Ω^j after the transition and may contain items that are not in $\mathcal{I}^j$. The precondition prevents Ω^j from including all the items in $\mathcal{I}^j$.

5.2.2 Set-based State Constraint Encoding

The state constraint for the j-th cut is as follows:

$$\mathcal{I}^j \not\subseteq S[\Omega^j], \tag{27}$$

where S is any state.

5.3 Weakness of the DIDP Encoding

There is a fundamental weakness in the aforementioned DIDP encodings compared to constraint-based models: adding a cut expands the search space. All four DIDP encoding methods rely on adding a new state variable to the MP to keep track of the changes to the LHS of (17) caused by transitions. After adding a new state variable corresponding to the j-th cut, the original state space size is multiplied by the cardinality of the $\mathcal{I}^j$. We return to this point in Section 8.

6 DIDP-LBBD for SUALBP-1

In this section, we present the DIDP model for the master problem for SUALBP-1 and the instantiation of the four encoding methods in Section 5 to the Benders feasibility cuts for SUALBP-1.

6.1 DIDP Master Problem

Since the LBBD master problem for SUALBP-1 is equivalent to the SALBP-1 as stated in Section 3, our DIDP formulations for the master problem are inspired by an existing DIDP model for SALBP-1 [10], which is defined as follows.

State variables.

- U: set variable for unscheduled tasks. In the target state (i.e., the initial state), $U = V$.
- r: integer resource variable for the remaining time (cycle time minus used time) of the current station. In the target state, $r = 0$. A larger r is better.

Base case. A base case is a set of conditions to terminate the recursion. The base case of the DIDP model is: $U = \emptyset$.

Transitions. Recall that a transition τ is a 4-tuple $\langle \text{eff}_\tau, \text{cost}_\tau, \text{pre}_\tau, \text{forced}_\tau \rangle$ where eff_τ is the effect, cost_τ is the cost expression, pre_τ is the set of preconditions, and $\text{forced}_\tau \in \{\bot, \top\}$.

Table 2: Summary of the DIDP model for the master problem.

State	Type	Objects	Preference	
U	set	tasks V		
r	integer resource		larger	
Target state	$U = V, r = 0$			
Base case	$U = \emptyset$			
Transition	Preconditions	Effects	Cost	Forced
$\texttt{assign}_i$	$i \in U \ \wedge \ \underline{t}_i \leq r \wedge U \cap P_i^* = \emptyset$	$U \to U \backslash \{i\} \wedge r \to r - \underline{t}_i$	0	$\bot$
open	$(i \notin U \ \vee \ r < \underline{t}_i \vee U \cap P_i^* \neq \emptyset) \mid \forall i \in V$	$r \to c$	1	$\bot$

- $Assign_i = \langle U \to U \backslash \{i\} \ \wedge \ r \to r - \underline{t}_i, \ 0, \ i \in U \ \wedge \ \underline{t}_i \leq r \ \wedge \ U \cap P_i^* = \emptyset, \ \bot \rangle$: assign task i to the current station.
- $Open = \langle r \to c, \ 1, \ (i \notin U \ \vee \ r < \underline{t}_i \ \vee \ U \cap P_i^* \neq \emptyset) \mid \forall i \in V, \ \bot \rangle$: open a new station.

Note that we use $\underline{t}_i$ instead of t_i in the master problem to estimate the setup times that are exactly calculated in the subproblems.

Theoretically, the transition *Open* can be used at any state. However, a state with a closed station that can accommodate an unscheduled task is dominated by an otherwise identical one that schedules such a task. Thus, a transition dominance rule (see Beck et. al [37]), stating that a station can only be opened if no task can be assigned to the current station, is encoded in the preconditions for transition *Open*. This dominance rule plays an important role in the efficiency of the DIDP model [10] but presents a complication for our cut formulations (see Section 6.2.2). The state variables, target state, base case, and transitions are summarized in Table 2.

Recursive function. We use $f(U, r)$ to represent the cost of a state. Let $U_1 = \{i \in U \mid r \geq \underline{t}_i \ \wedge \ U \cap P_i^* = \emptyset\}$ be the set of tasks with all their predecessors scheduled that can fit on the current station. The recursive function of the DIDP model is as follows:

$$\texttt{compute } f(V, 0) \tag{28a}$$

$$f(U,r) = \begin{cases} 0 & \text{if } U = \emptyset, \quad \text{(i)} \\ 1 + f(U, c) & \text{else if } U_1 = \emptyset, \quad \text{(ii)} \\ \min_{i \in U_1} f(U \backslash \{i\}, r - \underline{t}_i) & \text{else}, \quad \text{(iii)} \end{cases} \tag{28b}$$

$$f(U, r) \leq f(U, r^{'}), \quad \text{if } r \geq r^{'}, \tag{28c}$$

$$f(U,r) \geq \max \begin{cases} \left\lceil \frac{\sum_{i \in U} \underline{t}_i - r}{c} \right\rceil, & \text{(i)} \\ \sum_{i \in U} w_i^2 + \sum_{i \in U} w_i^{'2} - \mathbb{1}(r \geq C/2), & \text{(ii)} \\ \sum_{i \in U} w_i^3 - \mathbb{1}(r \geq C/3). & \text{(iii)} \end{cases} \tag{28d}$$

The term (28a) is to compute the cost of the target state. Equation (28b) is the main recursion of the DIDP model. Specifically, (28b-i) refers to the base case, while (28b-ii) corresponds to opening a new station and (28b-iii) refers to assigning task i to the current station. Inequality (28c) formulates state domination due to the resource variable: if other variables are equal, a state with a larger remaining time dominates. (28d-i), (28d-ii), and (28d-iii) are valid dual bounds proposed by Scholl and Klein [38] with numeric constants $w^2, w^{'2}, w^3$ indexed by a task i and depending on $\underline{t}_i$, as shown in Table 3. In addition, the expression $\mathbb{1}(r \geq C/2)$ returns 1 if $r \geq C/2$ and 0 otherwise. Similarly, the expression $\mathbb{1}(r \geq C/3)$ returns 1 if $r \geq C/3$ and 0 otherwise.

6.2 Encoding DIDP-LBBD Cuts for SUALBP-1

The formulations in Section 5.1 and Section 5.2 can be used for any cut of the form (17). Here we formally present four cut formulations for SUALBP-1.

Recall that Chu and Xia [26] defined a valid Benders cut as a logical expression having two properties that are described in (9a) and (9b). We prove that these two properties are satisfied for each of the four cut encodings.

6.2.1 Counting-based Precondition Encoding

Let $\mathcal{J}$ be the set of subproblems leading to Benders cuts. Consider subproblem $j \in \mathcal{J}$ corresponding to station k, define $\mathcal{I}^j$ as the set of tasks assigned to the station that cannot all be scheduled within the cycle time. Also, define function $\mathcal{F}^j$ such that $\mathcal{F}^j(i) = 1$ if $i \in \mathcal{I}^j$ and 0 otherwise. In order to encode this cut, we add a new state variable g^j with its value being 0 at the target state. We then modify the recursive formulation (28b) as follows.

$$f(U, r, \{g^j \mid \forall j \in J\}) = \begin{cases} 0 & \text{if } U = \emptyset, \quad \text{(i)} \\ 1 + f(U, c, \{0 \mid \forall j \in J\}) & \text{else if } U_2 = \emptyset, \quad \text{(ii)} \\ \min_{i \in U_2} f(U \backslash \{i\}, r - \underline{t}_i, \{g^j + \mathcal{F}^j(i) \mid \forall j \in J\}) & \text{else.} \quad \text{(iii)} \end{cases} \tag{29}$$

where $U_2 = \{i \in U \mid r \geq \underline{t}_i \ \wedge \ U \cap P_i^* = \emptyset \ \wedge \ (\forall j \in \mathcal{J}, g^j + \mathcal{F}^j(i) \leq |\mathcal{I}^j| - 1)\}$.

Case (29-i) is the base case when all tasks are assigned to stations. Case (29-ii) represents the situation where a new station needs to be opened and all the state variables $\{g^j | j \in \mathcal{J}\}$ are reset to 0 to indicate that in the new station, no task is scheduled yet. The case (29-iii) indicates that when task i is assigned to the current station, the state variables $\{g^j | j \in \mathcal{J}\}$ are increased (or kept the same) accordingly. Since case (29-iii) implies that U_2 is not empty, any task that can be assigned to the current station must not make any cut violated according to the newly added preconditions: $\forall j \in \mathcal{J}, g^j + \mathcal{F}^j(i) \leq |\mathcal{I}^j| - 1$, because $g^j + \mathcal{F}^j$ can be mapped back to the LHS of the Benders cut (17) while $|\mathcal{I}^j| - 1$ can be mapped back to the RHS.

Proposition 5. *The counting-based precondition encoding is valid.*

Proof. For any cut $j \in \mathcal{J}$, g^j counts the number of variable-value pairs that appear in the current station. With transition *Open*, the current station changes to the next station and $g^j = 0$, as shown in (29-ii). As shown in (29-iii), with transition $Assign_i$ for any i, since $\mathcal{F}^j$ is non-negative and $g^j + \mathcal{F}^j(i) \leq |\mathcal{I}^j| - 1$ is the precondition stated in U_2, we have $S[g^j] \leq |\mathcal{I}^j| - 1$ at any state S of the DIDP model. This guarantees that the same set of tasks are never assigned to the same station and satisfies P1. Since the solutions removed by this encoding are the solutions with the set of tasks $\mathcal{I}^j$ assigned to any station, they are all infeasible globally as the task processing times and setup times are independent of stations, and thus P2 is satisfied. □

Table 3: Numeric constants for calculating a knapsack-based dual bound.

$\underline{t}_i$	(0, c/2)	c/2	(c/2, c]	$\underline{t}_i$	(0, c/3)	c/3	(c/3, c/2)	2c/3	(2c/3, c]
w_i^2	0	0	1	w_i^3	0	1/3	1/2	2/3	1
$w_i'^2$	0	1/2	0						

6.2.2 Counting-based State Constraint Encoding

We can also enforce the cuts by keeping the modified effects and using state constraints instead of preconditions to enforce the logic of feasibility cuts. The recursive formulation (28b) becomes:

$$f(U, r, \{g^j \mid \forall j \in J\}) = \begin{cases} 0 & \text{if } U = \emptyset, \quad \text{(i)} \\ 1 + f(U, c, \{0 \mid \forall j \in J\}) & \text{else if } U_2 = \emptyset, \quad \text{(ii)} \\ \min_{i \in U_1} f(U \backslash \{i\}, r - \underline{t}_i, \{g^j + \mathcal{F}^j(i) \mid \forall j \in J\}) & \text{else if } U_1 \neq \emptyset. \quad \text{(iii)} \end{cases} \tag{30}$$

Cases (30-i) and (30-ii) refer to the base case and opening new station, respectively, which are the same as (29-i) and (29-ii). Case (30-iii) considers assigning task i to the current station and updating the state variables $\{g^j | j \in \mathcal{J}\}$ accordingly.

In (30-iii), there is no precondition preventing a task assignment that violates Benders cut. Instead, state constraints are added to prune the resulting states as follows:

$$g^j \leq |\mathcal{I}^j| - 1, \forall j \in \mathcal{J}. \tag{31}$$

However, as noted, there is an interaction between the cut and the dominance rule associated with the preconditions of transition *Open*: if we maintain the original precondition on *Open* (i.e., $U_1 = \emptyset$), then a state where only tasks that violate the cut can be scheduled will result in a dead-end. The transitions satisfying (30-iii) will fire and the resulting states will all violate the state constraints. Thus, no state is reachable from the current state. However, a new station should be opened in the state when no tasks can be scheduled. To ensure the correctness of the model, either we remove the dominance and allow *Open* at any time, or we maintain it by allowing *Open* when no tasks, including those violating cuts, can be scheduled (the new preconditions become $U_2 = \emptyset$). We select the latter option to maintain the efficiency of the proposed DIDP model.

Proposition 6. *The counting-based state constraint encoding is valid.*

Proof. Similar to the proof for Proposition 5, for any cut $j \in \mathcal{J}$, g^j counts the number of variable-value pairs that appear in the current station. Explicitly enforced by state constraint (31), we have $S[g^j] \leq |\mathcal{I}^j| - 1$ at any state S of the DIDP model. P1 and P2 are hence satisfied. □

6.2.3 Set-based Precondition Encoding

To encode this cut, we add a new state variable Ω^j with its value being $\emptyset$ at the target state. We then modify the recursive formulation (28b) in the DIDP model to express

all the Benders feasibility cuts:

$$f(U, r, \{\Omega^j \mid \forall j \in J\}) = \begin{cases} 0 & \text{if } U = \emptyset, \quad \text{(i)} \\ 1 + f(U, c, \{\emptyset \mid \forall j \in J\}) & \text{else if } U_3 = \emptyset, \quad \text{(ii)} \\ \min_{i \in U_3} f(U \backslash \{i\}, r - \underline{t}_i, \{\Omega^j \cup \{i\} \mid \forall j \in J\}) & \text{else.} \quad \text{(iii)} \end{cases} \tag{32}$$

where $U_3 = \{i \in U \mid r \geq \underline{t}_i \ \wedge \ U \cap P_i^* = \emptyset \ \wedge \ (\forall j \in \mathcal{J}, \mathcal{I}^j \not\subseteq \Omega^j \cup \{i\})\}$.

Case (32-i) is the base case. Case (32-ii) represents the transition where a new station is opened and all the state variables $\{\Omega^j | j \in \mathcal{J}\}$ are reset to empty sets. Case (32-iii) indicates that when task i is assigned to the current station, the state variables $\{\Omega^j | j \in \mathcal{J}\}$ are updated accordingly. Since case (32-iii) requires that U_3 is not empty, any task that can be assigned to the current station must not make any cut violated according to the newly added preconditions: $\forall j \in \mathcal{J}, \mathcal{I}^j \not\subseteq \Omega^j \cup \{i\}$, because this set relation can be mapped back to the Benders cut j.

Proposition 7. *The set-based precondition encoding is valid.*

Proof. For any cut $j \in \mathcal{J}$, Ω^j keeps track of the variable-value pairs that appear in the current station. Case (32-ii) represents that, with transition *Open*, the current station changes to the next station and $\Omega^j = \emptyset$. As shown in (32-iii), with transition $Assign_i$ for any i, since the effects never remove any element from Ω^j and $\mathcal{I}^j \not\subseteq \Omega^j \cup \{i\}$ is the precondition stated in U_3, we have $\mathcal{I}^j \not\subseteq S[\Omega^j]$ at any state S of the DIDP model. This guarantees that the same set of tasks would never appear in the same station and satisfies P1. Similar to the proof for Proposition 5, P2 is satisfied. □

6.2.4 Set-based State Constraint Encoding

We can also adapt the state constraint encoding to the set-based formulation. The recursive formulation (28b) becomes:

$$f(U, r, \{\Omega^j \mid \forall j \in J\}) = \begin{cases} 0 & \text{if } U = \emptyset, \quad \text{(i)} \\ 1 + f(U, c, \{\emptyset \mid \forall j \in J\}) & \text{else if } U_3 = \emptyset, \quad \text{(ii)} \\ \min_{i \in U_1} f(U \backslash \{i\}, r - \underline{t}_i, \{\Omega^j \cup \{i\} \mid \forall j \in J\}) & \text{else if } U_1 \neq \emptyset. \quad \text{(iii)} \end{cases} \tag{33}$$

Cases (33-i) and (33-ii) refer to the base case and opening new station, respectively, which are the same as (32-i) and (32-ii). Case (33-iii) represents scheduling task i in the current station and update the state variables $\{\Omega^j | j \in \mathcal{J}\}$ accordingly.

The added state constraint is:

$$\mathcal{I}^j \not\subseteq \Omega^j, \forall j \in \mathcal{J}. \tag{34}$$

Similar to (30), we maintain the dominance specified by the preconditions of the transition *Open* by inserting the case violating state constraints (34) into the preconditions (the new preconditions become $U_3 = \emptyset$).

Proposition 8. *The set-based state constraint encoding is valid.*

Proof. Similar to the proof for Proposition 7, for any cut $j \in \mathcal{J}$, Ω^j keeps track of the variable-value pairs that appear in the current station. Enforced by the explicit state constraint (34), we have $\mathcal{I}^j \nsubseteq S[\Omega^j]$ at any state S of the DIDP model. P1 and P2 are hence satisfied. □

The DIDP-LBBD models with recursive formulation (29), (30), (32), and (33) replacing (28b) are referred as DIDP-LBBD$_{cPre}$, DIDP-LBBD$_{cCon}$, DIDP-LBBD$_{sPre}$, and DIDP-LBBD$_{sCon}$, respectively, where 'c' and 's' correspond to 'count' and 'set' and 'Pre' and 'Con' map to 'precondition' and 'constraint'.

7 Experimental Evaluation

In this section, we compare the performance of our CP-LBBD, DIDP-LBBD, and MIP-LBBD models against the state-of-the-art MIP model [11] as shown in the Appendix A. We use the 788 instances of the SBF2 data set [19].[4]

7.1 Experiment Setting

Originally proposed by Scholl et al. [19], the SBF2 data set has been widely used in the literature [39–41] and was recently modified by Zohali et al. [14]. We use the modified version and follow their clustering of the instances into four classes:

- Data set A: small (132 instances) with up to 25 tasks.
- Data set B: medium (140 instances) with 28 to 35 tasks.
- Data set C: large (188 instances) with 45 to 70 tasks.
- Data set D: extra-large (328 instances) with 75 to 111 tasks.

Each class has four different settings according to a parameter α that specifies the ratio of the average setup time to the average task processing time: 0.25, 0.50, 0.75, and 1.00.

For the DIDP models, we use the state-of-the-art solver based on CABS [8] in didp-rs v0.9.0.[5] For the CP models, we use CP Optimizer 22.1.1 [36]. For the MIP models, we use Gurobi 12.0.1 [42]. All the experiments are implemented in Python 3.10.11. Each instance is run for 1800 seconds on a single thread on a Ubuntu 22.04.2 LTS machine with Intel Core i7 CPU and 16 GB memory.

7.2 Experiment Results

The results on SUALBP-1 are shown in Fig. 3. Better performance is indicated by curves closer to the top left corner of the graph. The performance of our approaches on datasets A, B, C, and D separately are presented in Fig. 4 - 7. In addition to the overall performance on average, we provide more granular and detailed results in the Table 4.

[4] https://assembly-line-balancing.de/sualbsp/data-set-of-scholl-et-al-2013/
[5] https://github.com/domain-independent-dp/didp-rs/releases/tag/v0.9.0

In the LBBD models compared in this work, the first feasible master problem solution that is also feasible for all subproblems is an optimal solution to the original problem and the LBBD algorithm terminates. Thus, for a problem instance, we define an approach to be the "winner" if it is able to prove optimality for the instance faster than all the other approaches. When multiple approaches are equivalently fast, they are all winners. When no approach can prove optimality for the instance, there is no winner. The detailed win rates (the proportion of instances for which an approach is the winner) and the optimality rates (the proportion of instances for which an approach can solve to optimality) for the four datasets separately and together are presented in Table 4.

First if we look at the overall performance shown in Fig. 3, all of our proposed techniques outperform the MIP model, the current state of the art. CP-LBBD$_a$ achieves the best performance at the time limit with 72.3% of instances proved optimal. CP-LBBD$_c$ performs best before 500 seconds. In particular, CP-LBBD$_c$ achieves 66.9% in 300 seconds while CP-LBBD$_a$ is 100 seconds slower to achieve that level. This performance difference indicates the speedup brought by the constraint-based expression `Count_Different`. CP-LBBD$_p$, though trailing the other two CP-LBBD models, performs better than DIDP-LBBD, MIP-LBBD, and MIP approaches.

Though the three CP-LBBD variants differ substantially in Fig. 3, there is no significant performance difference among the four DIDP-LBBD variants. The DIDP-LBBD models find and prove optimal solutions for more instances in a shorter computation time than MIP-LBBD and MIP. In 100 seconds, all four DIDP-LBBD models find and prove optimality on 55% of the instances. MIP cannot achieve the same performance in 450 seconds. At 1800 seconds, DIDP-LBBD has found and proved optimality for around 61% of the problem instances compared to 60.4% and 54.9% for MIP-LBBD and MIP, respectively.

For the overall win rates in Table 4, all LBBD models except for DIDP-LBBD$_{cPre}$ outperform the MIP model, with the CP-LBBD$_c$ model achieving the highest win rate. While CP-LBBD$_a$ finds more optimal solutions than CP-LBBD$_c$, the latter typically finds solutions more quickly. More interestingly, the DIDP-LBBD$_{sCon}$ model achieves the second best win rate and outperforms other DIDP-LBBD variants. The DIDP-LBBD$_{cPre}$ model performs the worst, with the monolithic MIP model being

Table 4: Win rates and optimality rates per approach.

	Win rate (%) / Optimality rate (%)				
Model \ Dataset	Dataset A	Dataset B	Dataset C	Dataset D	Overall
CP-LBBD$_a$	0.0 / **100.0**	1.4 / **100.0**	**20.2** / **63.8**	9.1 / 54.3	8.9 / **72.3**
CP-LBBD$_c$	4.5 / **100.0**	**34.3** / **100.0**	16.5 / 55.9	**32.0** / **54.6**	**24.1** / 70.6
CP-LBBD$_p$	3.0 / **100.0**	20.7 / **100.0**	4.8 / 44.7	6.7 / 52.1	8.1 / 66.9
DIDP-LBBD$_{cPre}$	1.5 / **100.0**	2.1 / **100.0**	0.0 / 52.1	0.0 / 33.8	0.6 / 61.0
DIDP-LBBD$_{cCon}$	0.8 / **100.0**	9.3 / **100.0**	9.6 / 54.8	0.6 / 34.1	4.3 / 61.8
DIDP-LBBD$_{sPre}$	40.9 / **100.0**	3.6 / **100.0**	0.0 / 51.1	0.0 / 33.5	7.5 / 60.7
DIDP-LBBD$_{sCon}$	**43.2** / **100.0**	20.7 / **100.0**	18.6 / 53.7	7.9 / 33.8	18.7 / 61.4
MIP-LBBD	6.1 / **100.0**	5.7 / **100.0**	6.4 / 36.2	0.9 / 41.5	3.9 / 60.4
MIP	0.0 / **87.9**	2.1 / **100.0**	5.3 / 47.9	0.9 / 26.5	2.0 / 54.9

slightly better. Note that strong win-rate results mean that an approach is the fastest for more instances compared to other approaches but may trail on or fail to solve other instances.

Fig. 3 and Table 4 show that directly bounding the assignment of core decision variables x_i in the CP model or using light-weighted constraint-based expressions are advantageous compared to global constraints, especially when using a global constraint requires extra variables such as w_k in the `Pack` constraint. DIDP-LBBD models, though possessing similar capability in solving SUALBP-1 instances to optimality within the time limit, exhibit different speeds. Specifically, set-based DIDP-LBBD models are often faster than counting-based models, while state constraint-based models are often faster than precondition-based models.

For the smallest dataset A, all the LBBD models prove optimality for all the instances, while MIP achieves optimality for 87.9% instances: MIP struggles for even small instances. Note that all lines except for the line for MIP overlap on the top of Fig. 4 and we do not plot the entire 1800 seconds because the results do not change after 2 seconds. However, the two set-based DIDP-LBBD models are often faster, as reflected in their win rates in Table 4.

For the mid-size instances, CP-LBBD models outperform MIP-LBBD and DIDP-LBBD models, with MIP trailing all the other approaches. The CP-LBBD$_c$ model demonstrates the best performance according to the win rates, with CP-LBBD$_p$ and DIDP-LBBD$_{cCon}$ tied for second place.

For the large instances, as shown in Fig. 6, the compared approaches behave quite differently. The CP-LBBD$_a$ and CP-LBBD$_c$ models lead the performance ranking but the performance of CP-LBBD$_c$ drops after 1200 seconds. CP-LBBD$_a$ performs the best and achieves proved optimality for 64% instances. However, CP-LBBD$_p$ again performs poorly, only better than the MIP-LBBD model and achieves proved optimality for 44% instances. This result likely indicates that the inference of the `Pack` constraint is too expensive. The four DIDP-LBBD models perform similarly and only worse than CP-LBBD$_a$ and CP-LBBD$_c$, with DIDP-LBBD$_{sPre}$ trailing the other three models. Surprisingly, the monolithic MIP model outperforms MIP-LBBD and achieves a similar percentage of instances proved to optimality as DIDP-LBBD$_{sPre}$. The MIP-LBBD model only proves 35% instances to optimality. According to the win rates in Table 4, CP-LBBD$_a$ is the best for the large instances. The DIDP-LBBD$_{cCon}$ model is consistently fast but the two precondition-based DIDP-LBBD models do not win at all.

For the extra large instances, as shown in Fig. 7, the performance rankings differ from the rankings for large instances. Although the three CP-based LBBD models outperform the other approaches, they cannot prove optimality for more than 55% instances. The global-constraint-based CP-LBBD$_p$ is consistently worse than the other two CP-LBBD models. While the MIP-based LBBD model underperforms the DIDP-based models before 300 seconds, it starts to consistently outperform the DIDP-based models after 360 seconds and proves 9% more instances at the time limit. The monolithic MIP model is the worst and proves less than 30% of the instances to optimality. According to the win rates in Table 4, the CP-LBBD$_c$ model is frequently faster

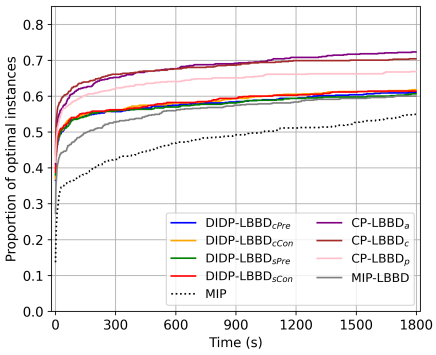

Fig. 3: Ratio of instances solved to optimality over time for SUALBP-1.

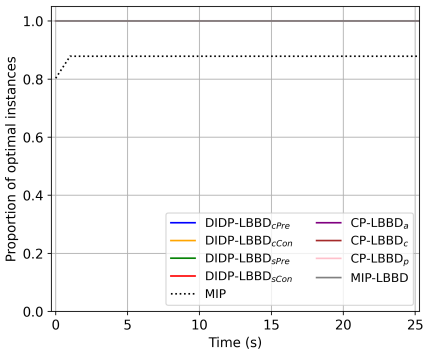

Fig. 4: Ratio of instances solved to optimality over time for dataset A.

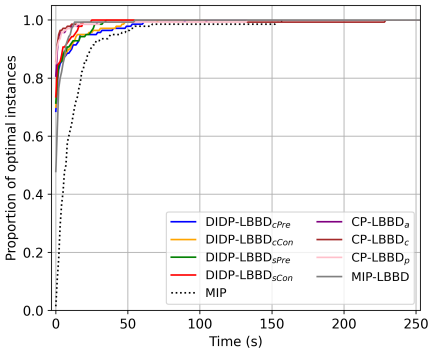

Fig. 5: Ratio of instances solved to optimality over time for dataset B.

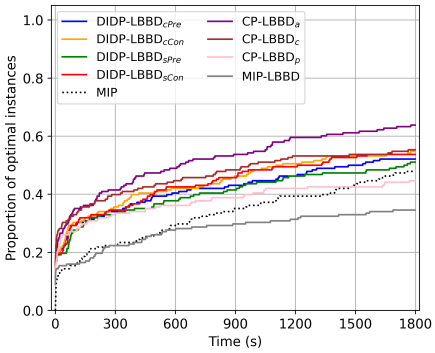

Fig. 6: Ratio of instances solved to optimality over time for dataset C.

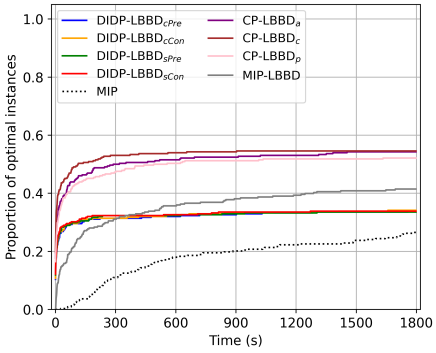

Fig. 7: Ratio of instances solved to optimality over time for dataset D.

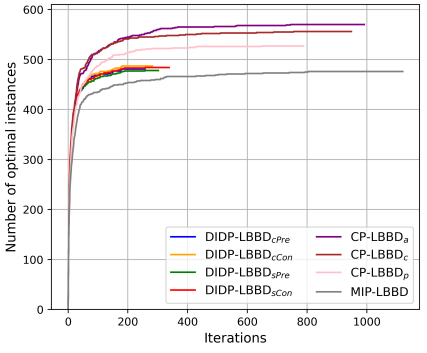

Fig. 8: Instances solved to optimality over LBBD iterations required.

than other models, with the CP-LBBD$_a$, CP-LBBD$_p$, and DIDP-LBBD$_{cCon}$ models winning occasionally.

Overall, the differences between optimality rates and win rates implies that the CP-LBBD models are not always faster than DIDP-LBBD models, especially when problem instances are small. However, CP-LBBD approaches scale better.

Focusing on the eight LBBD models as shown in Fig. 3, the relative rankings are: CP-LBBD, DIDP-LBBD, and MIP-LBBD. According to the win rates in Table 4, if we compare the best CP-LBBD model (i.e., CP-LBBD$_c$), the best DIDP-LBBD model (i.e., DIDP-LBBD$_{sCon}$), and the MIP-LBBD model, we reach the same performance rankings, demonstrating the promise of CP-LBBD and DIDP-LBBD.

It is worth mentioning that the time spent on subproblems is very short, e.g., 0.001s, suggesting that a branch-and-check[6] approach may result in further performance improvements [43].

7.3 Algorithm Analysis

To better understand the performance of the eight LBBD models and the differences among them, we conduct a series of analyses.

7.3.1 Instances Solved to Optimality vs. Iterations Required

We first report the number of instances that are proved optimal versus the number of LBBD iterations for all eight LBBD models and all four datasets, as shown in Fig. 8. If for a model, after iteration i, there is no instance running due to the time limit, the plot stops at iteration i.

From Fig. 8 we can see that the CP-LBBD models prove the most instances optimal with CP-LBBD$_a$ performing the best. The CP-LBBD$_c$ model proves fewer instances optimal than CP-LBBD$_a$ but more instances than the CP-LBBD$_p$ model. DIDP-LBBD models, with similar performance among four variants, trail the CP-LBBD models but outperform the MIP-LBBD model.

Note that the plots of DIDP-LBBD models stop the earliest (i.e., before 350 iterations) and cannot prove more instances optimal after around 200 iterations. The state-constraint-based DIDP-LBBD models perform slightly better than the precondition-based models. The interpretations here are two-fold: (i) DIDP-LBBD models cannot conduct more iterations within the time limit; (ii) DIDP-LBBD models require relatively fewer iterations to prove some instances optimal. We present further analysis below to verify these interpretations.

It appears that DIDP-LBBD does many fewer iterations but performs better than MIP-LBBD. Given that the only difference is in the solutions to the MP (i.e., the cuts are mathematically equivalent), then it must be the differences arise from the different techniques finding different optimal MP solutions. We conduct further evaluations to investigate: (i) whether this phenomenon is real and (ii) what causes this phenomenon.

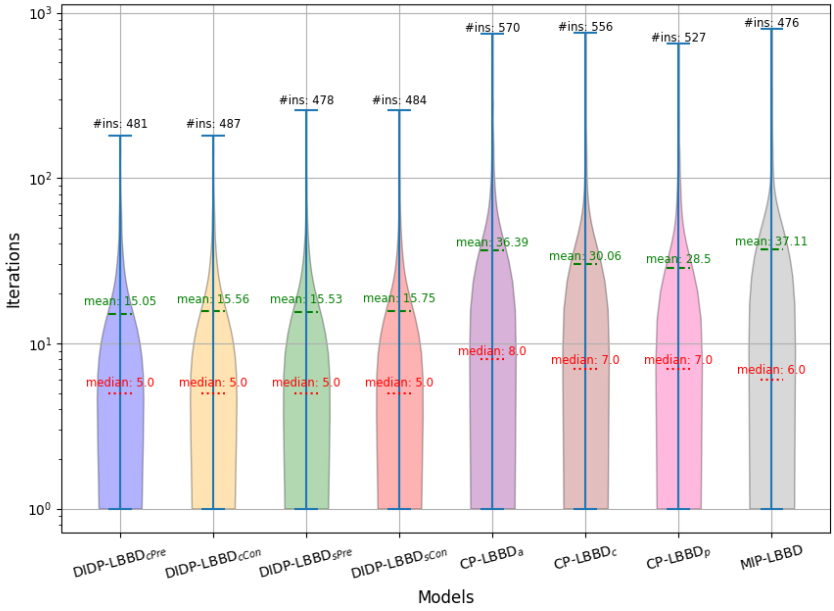

Fig. 9: Distribution of the LBBD iterations. The number of instances considered for each model is reported by the black text "#ins". The mean is indicated by the short solid horizontal green line and reported by the green text "mean". The median by the dotted horizontal red line and reported by the red text "median". Note the logarithmic scale of the y-axis

7.3.2 Distribution of the Number of Iterations

We analyze the distribution of the number of iterations required for the instances proved optimal by each of the eight LBBD models. The results are shown in Fig. 9. Since each LBBD model proves a different number of instances to optimality, the number of instances considered in each violin plot corresponding to each model is reported by the black text "#ins". We also report the mean and median number of iterations required as the text "mean" and "median", respectively.

We can see that the four DIDP-LBBD models are indeed more efficient than CP-LBBD and MIP-LBBD models in terms of the number of iterations required to prove optimality. For each model, the median number of iterations (indicated by the dotted horizontal red line) is smaller than the mean (indicated by the solid horizontal green line): most of the instances need only few LBBD iterations to be proved optimal and occasionally some instance needs more iterations. Compared to CP-LBBD and MIP-LBBD models, DIDP-LBBD models demonstrate the smallest mean and median number of iterations, but all median values are relatively close.

[6]Branch-and-check solves the master problem only once with each of the feasible solutions found in the solving process triggering subproblem evaluation.

While the mean iterations for MIP-LBBD are the largest among all models, the median for MIP-LBBD is smaller than CP-LBBD models: MIP-LBBD has more instances that require many iterations to prove optimal and the distribution of the number of iterations is similar to bimodal distribution.

Interestingly, the CP-LBBD$_p$ model has the smallest mean and median iterations compared to CP-LBBD$_c$ and CP-LBBD$_a$. The interpretations here are also two-fold: (i) there are few instances with extremely many iterations when using the CP-LBBD$_p$ model; (ii) some instances need many more iterations than the current time limit allowed for CP-LBBD$_p$ to prove optimal, and if these instances are included under a longer time limit, the mean and median iterations will exceed that for CP-LBBD$_c$ and CP-LBBD$_a$.

7.3.3 Scatter Plots of Iterations Required

The second evaluation we conduct is to look into individual instances and compare the number of iterations required for a pair of selected LBBD models. To better understand the difference between the DIDP-LBBD, CP-LBBD, and MIP-LBBD models, we select the DIDP-LBBD$_{cCon}$, CP-LBBD$_a$, and MIP-LBBD models for further analysis.

The Fig. 10 is a scatter plot for the subset of instances that can be proved optimal by both the DIDP-LBBD$_{cCon}$ and CP-LBBD$_a$ models. Each point in the plot corresponds to an instance with the x-value and y-value being the number of iterations required when solved with the CP-LBBD$_a$ model and the DIDP-LBBD$_{cCon}$ model, respectively.

From the scatter plot, we can see that the DIDP-LBBD$_{cCon}$ model requires many fewer iterations than the CP-LBBD$_a$ model. In fact, in the plot, only a few instances need more iterations when solved with the DIDP-LBBD$_{cCon}$ model compared to the CP-LBBD$_a$ model.

A similar scatter plot for the DIDP-LBBD$_{cCon}$ and MIP-LBBD models is shown in Fig. 11. We can see that the DIDP-LBBD$_{cCon}$ model also needs many fewer iterations than the MIP-LBBD model. The scatter plot for the CP-LBBD$_a$ and MIP-LBBD models is shown in Fig. 12. The CP-LBBD$_a$ needs slightly fewer iterations than the MIP-LBBD model, which is consistent with our observations in Fig. 8 and Fig. 9.

Note that the three scatter plots consider three different subsets of instances. Each plot includes only the instances that can be solved optimally by both methods involved.

7.3.4 MP Solution Similarity vs. Iterations Required

This section presents efforts to understand the difference in the number of iterations between different LBBD models.

Since the only difference between the proposed LBBD models is in solving the MP and all cut formulations are mathematically equivalent, the reason for the performance difference appears to arise from different optimal solutions being found by the different MP solvers. Once two approaches find two different optimal MP solutions at an iteration, their search processes diverge because the cuts generated from different MP solutions will be different. However, the results above suggest that there is a conceivable systematic effect: DIDP models appear to be consistently finding MP solutions that lead to fewer overall iterations.

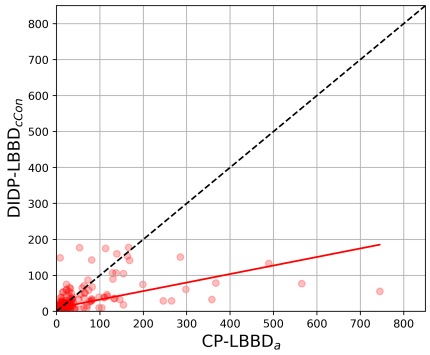

Fig. 10: Number of iterations required for DIDP-LBBD$_{cCon}$ and CP-LBBD$_a$ models.

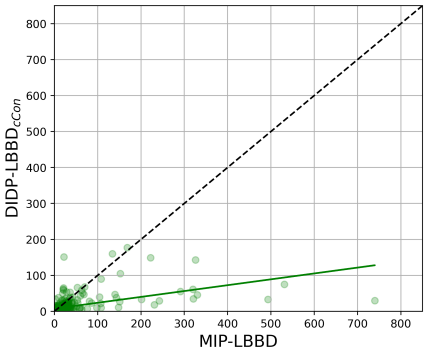

Fig. 11: Number of iterations required for DIDP-LBBD$_{cCon}$ and MIP-LBBD models.

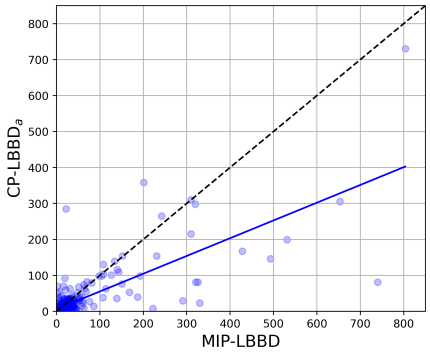

Fig. 12: Number of iterations required for CP-LBBD$_a$ and MIP-LBBD models.

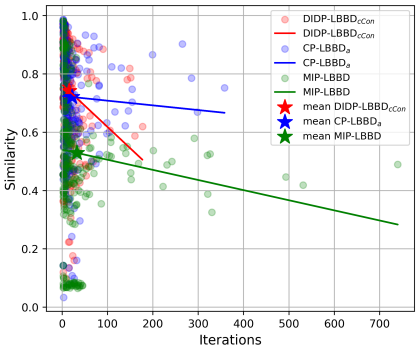

Fig. 13: MP solution similarity vs. iterations required for DIDP-LBBD$_{cCon}$, CP-LBBD$_a$ and MIP-LBBD models

As an attempt to explain the systematic effect observed, we propose a hypothesis: the similarity between the optimal MP solutions obtained in consecutive LBBD iterations is negatively correlated with the number of LBBD iterations required to prove optimal. This hypothesis is inspired by a column stabilization technique to alleviate the *tailing off* effect at the end of the column generation (CG) process and to accelerate its convergence [44]. Specifically in CG, using simplex re-optimization after adding a column instead of optimization from scratch can bring stabilization toward the previous primal solution of the restricted master problem (RMP) and reduce the number of iterations required to find the optimal CG solution [45]. This technique implies that similar primal solutions of RMPs in CG lead to fewer iterations required for convergence. We hypothesize that a similar phenomenon could happen in LBBD models for SUALBP-1. We now introduce the tools needed to investigate this hypothesis.

We define MP solution similarity as follows. Given an instance i, after the instance is proved optimal by an approach a, a sequence of MP solutions $(s^1_{i,a}, ..., s^T_{i,a})$ found in each iteration is also obtained. For a pair of adjacent MP solutions $(s^j_{i,a}, s^{j+1}_{i,a})$ in the sequence, we compare the tasks assigned to the station k in both solutions. Let $A^{j,k}_{i,a}$ and $A^{j+1,k}_{i,a}$ be the sets of tasks assigned to the station k in the solutions $s^j_{i,a}$ and $s^{j+1}_{i,a}$, respectively. Then we define $c^{j,k}_{i,a}$ as the number of the tasks shared by the two solutions in station k, i.e., $c^{j,k}_{i,a} = |A^{j,k}_{i,a} \cap A^{j+1,k}_{i,a}|$. Then the similarity between $(s^j_{i,a}$ and $s^{j+1}_{i,a})$ is

$$\theta^j_{i,a} = \frac{\sum_{k=1,...,K} c^{j,k}_{i,a}}{N}, \tag{35}$$

where N is the total number of tasks of instance i. The similarity score of the sequence of MP solutions (corresponding to instance i and approach a) is

$$\Theta_{i,a} = \frac{\sum_{j=1,...,J-1} \theta^j_{i,a}}{J-1}, \tag{36}$$

where J is the number of MP solutions in the sequence. Note that a higher similarity score means less change between consecutive MP solutions in the solution sequence.

In Fig. 13, we plot the similarity score versus the number of iterations required to prove optimality for all the instances that can be solved optimally by DIDP-LBBD$_{cCon}$, CP-LBBD$_a$ and MIP-LBBD models. We exclude instances that can be solved by any approach with only one LBBD iteration as we cannot calculate its MP solution similarity. We also plot the regression line to approximate the distribution.

We can see that the mean similarity of the DIDP-LBBD$_{cCon}$ model is the highest while the mean iterations required are the fewest. The DIDP-LBBD$_{cCon}$ model demonstrates a clear negative correlation between the MP solution similarity and the number of iterations required to prove optimal. For CP-LBBD$_a$ and MIP-LBBD models, the negative correlations are less apparent.

Thus, for the DIDP-LBBD$_{cCon}$ model, the MP solution similarity is high, the changes between consecutive MP solutions are small, and the number of iterations required to prove optimality is low. We speculate that the search behavior of the DIDP model leads to this phenomenon. The DIDP-LBBD$_{cCon}$ model for the master problem constructs the solution of SUALBP-1 by assigning each of the tasks to stations. The assignment starts from a station and fills tasks into it until no more tasks can be placed. Then a new station is opened and the task assignment continues. When there is a Benders cut encoded by transition preconditions or state constraints included in the DIDP model of the MP, it does not immediately affect the solution construction process (or search behavior). The cut is only triggered when the tasks in the current station satisfy the conditions associated with the cut. Thus, the adjacent MP solutions obtained with the DIDP-LBBD$_{cCon}$ model could be very similar since the corresponding DIDP models for the two MPs differ by only a few cuts.

For CP-LBBD and MIP-LBBD models, the Benders cuts are added as constraints and have an immediate impact on the search space. Since the solutions in MIP models

of the MP are not explicitly constructed from the first to the last station, we could see very different adjacent MP solutions. As a result, for SUALBP-1, the adjacent MP solutions obtained with DIDP models usually have higher similarity scores than the adjacent MP solutions obtained with MIP models.

We speculate that the negative correlation between the MP solution similarity and the number of iterations required to prove optimal is caused by the problem structure of SUALBP-1. Since the objective is to minimize the number of stations used, the cost function is very coarse. Thus, in most instances, there are multiple optimal solutions. If the algorithm searches in a small region and improves the solution quality based on a given MP solution, as our DIDP model does with Benders cuts, it could likely lead to an optimal solution soon. Instead, exploring different regions may increase the unnecessary computational efforts and LBBD iterations, as our MIP model does with Benders cuts.

In the Benders decomposition literature, stabilization techniques such as regularization [46] and trust-regions [47] are often used to accelerate the convergence [48]. Specifically, the regularization technique modifies the MP objective function to penalize deviations from previous solutions and the trust-regions technique limits the MP solution to a localized area around a reference point, preventing erratic oscillations between solutions. These techniques can lead to similar MP solutions and thus fewer iterations required to prove optimality for many problems [49]. We speculate that as the DIDP-LBBD models find similar MP solutions, they result in performance similar to a regularized model but without explicit regularization. We make the observation to inspire theoretical research in future work.

Interestingly, note that the average similarity score of the CP-LBBD$_a$ model is high. This means that the CP models of the master problems could be exploiting a small search region. The relatively low average iterations required to prove optimal for the CP-LBBD$_a$ model are consistent with our speculated negative correlation between the similarity and iterations. Therefore, the CP-LBBD models can perform a relatively small number of LBBD iterations when searching in a small region. The default search strategy of CP Optimizer consists of several search techniques (including large neighborhood search, genetic algorithms, etc.) that are dynamically changed during the search by adapting to the problem of interest [50]. Thus, it is unclear what algorithms are used in the CP master problem. With the coarse objective function of SUALBP-1, focusing on similar solutions allows CP (and DIDP) to cut off solutions in a concentrated area, leading to a primal solution that then cuts off solutions in other areas. Thus, the number of iterations required to prove optimality is significantly reduced. However, we must re-emphasize that this is speculation and further investigation and fuller analysis is needed.

7.3.5 Runtime vs. Iterations

As shown in the last section, both DIDP-LBBD and CP-LBBD models demonstrate a negative correlation between the MP solution similarity and the number of iterations required to prove optimality, but why do the CP-LBBD models outperform the DIDP-LBBD models? To answer this question, we conduct a further analysis.

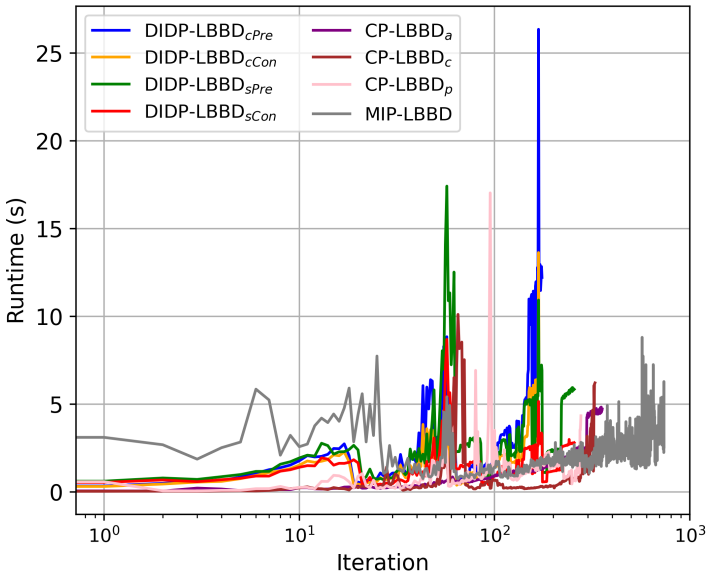

Fig. 14: Mean MP runtime over iterations. Note the logarithmic scale of the x-axis.

For the SBF2 data set, 409 of the 788 instances are proved optimal by all of the eight LBBD models. The mean MP runtimes of the 409 instances over iterations for all eight LBBD models are shown in Fig. 14.

CP-LBBD and MIP-LBBD have relatively consistent MP runtime across different iterations. Since CP-LBBD models need fewer iterations and spend less or similar runtime in each iteration than the MIP-LBBD model, CP-LBBD models are generally faster than the MIP-LBBD model. For DIDP-LBBD models, although starting from a small magnitude, the MP runtimes increase drastically as the iterations increase. As discussed in Section 5.3, with more state variables added to the DIDP model of the master problem, the state space of the model is enlarged and the search needs more search effort to find and prove optimality: hence the MPs become more time-consuming to solve. We have already shown in Fig. 13 that the number of iterations required for the DIDP-LBBD models is slightly fewer than that for the CP-LBBD models. But since the runtime of the DIDP-LBBD models increases sharply with the number of iterations, the overall performance of the CP-LBBD models is better than the DIDP-LBBD models. Therefore, this performance degradation can partially explain the worse results of DIDP-LBBD compared to CP-LBBD.

The mean MP runtimes over iterations for dataset A, B, C, and D are separately shown in Fig. 15 - 18, respectively. Also note the logarithmic scale of the x-axis in Fig. 15 - 18.

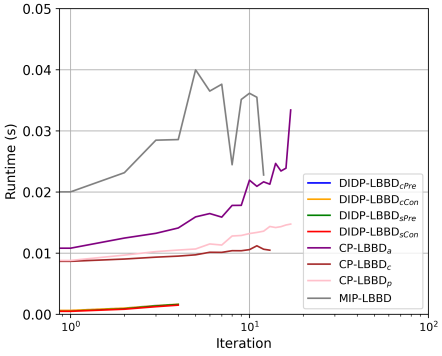

Fig. 15: Mean MP runtime over iterations for dataset A.

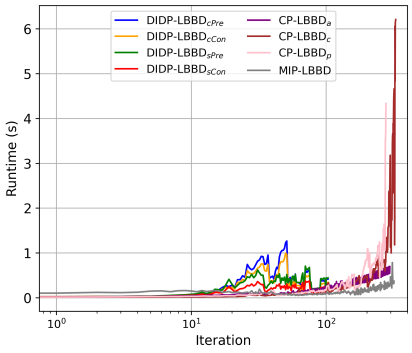

Fig. 16: Mean MP runtime over iterations for dataset B.

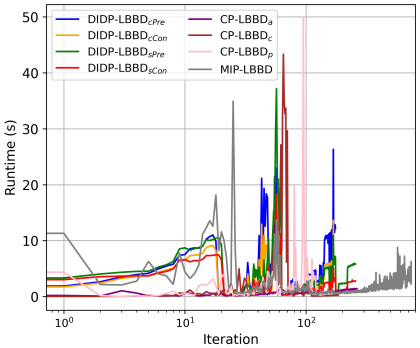

Fig. 17: Mean MP runtime over iterations for dataset C.

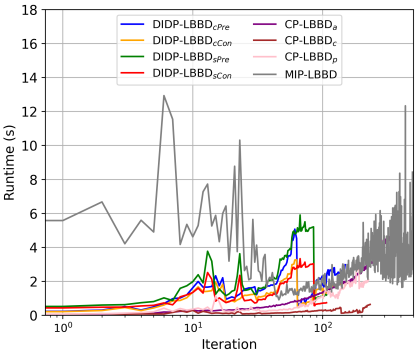

Fig. 18: Mean MP runtime over iterations for dataset D.

For dataset A, the DIDP-LBBD models achieve the best performance, while the CP-LBBD and MIP-LBBD models exhibit longer MP runtimes. Notably, the CP-LBBD models experience a relatively prolonged runtime in the initial phase, which decreases after the first iteration. Since the number of iterations required is low, the runtime for dataset A is generally short and the performance degradation of DIDP-LBBD models is not obvious.

For dataset B, the DIDP-LBBD models and the CP-LBBD$_p$ model show steep increases in runtime during the first 60 iterations, whereas the CP-LBBD$_a$, CP-LBBD$_c$, and MIP-LBBD models display a more gradual rise. Specifically, the MIP-LBBD model maintains a relatively consistent runtime over nearly 350 iterations, highlighting the insensitivity of MIP runtime to the number of cuts added.

For dataset C, a noticeable surge in runtime begins around iteration 40, with the DIDP-LBBD model performing worse than the CP-LBBD models in this regard.

Despite some fluctuations, the MIP-LBBD model maintains a relatively consistent runtime across iterations.

For dataset D, the MIP-LBBD model exhibits the worst runtime performance. The DIDP-LBBD models experience two sharp runtime increases before iteration 100. Initially, the CP-LBBD$_p$ and CP-LBBD$_c$ models perform well but are eventually outpaced by the two set-based DIDP-LBBD models. Among all the models, CP-LBBD$_a$ performs the best, delivering consistent runtime throughout the iterations.

In summary, the MIP-LBBD model demonstrates the most consistent mean runtime across iterations. As the number of iterations increases, the DIDP-LBBD models exhibit significant performance degradation, characterized by sharp increases in runtime. While the CP-LBBD models also experience runtime surges, these are generally less severe and are often followed by a stabilization period, during which the CP-LBBD models maintain consistent performance. Combined with our findings in Section 7.3.4, this explains the better performance of the CP-LBBD models compared to the DIDP-LBBD and MIP-LBBD models. In the next section, we analyze the performance of the three CP-LBBD models in more detail.

7.3.6 Analysis of CP-LBBD

In order to investigate the differences among the three CP-LBBD models, for all 788 instances in the SBF2 dataset, we added the cuts generated by CP-LBBD$_a$ model at each MP iteration to all models, in their corresponding cut forms. Thus for each MP iteration, the three models solve identical problems except for the differences in the form of the cuts. We set the time limit to 3600 seconds for this experiment.

The ratio of instances solved and proved optimal over time for datasets A, B, C, and D are presented in Fig. 19, 20, 21, and 22, respectively. All four graphs show a substantial cluster in the lower-left corner demonstrating broadly similar performance. However, both CP-LBBD$_c$ and, to a greater extent, CP-LBBD$_p$ exhibit a number of instances with a large number of nodes and large runtimes when CP-LBBD$_a$ has relatively small values of these measures.

These graphs are consistent with the overall results of the CP models in Figure 3. In terms of the number of nodes generated, the graphs suggest that the difference comes less from a systematic performance difference among the models and more from a small number of outliers with large node counts for CP-LBBD$_c$ and CP-LBBD$_p$. In contrast, the runtime graphs for CP-LBBD$_p$ and, though to a lesser extent, CP-LBBD$_c$ show vertical clusters of instances with relatively low CP-LBBD$_a$ runtimes implying that the higher computational effort of the global constraint-based models does not pay off in terms of performance.

The runtime over the number of nodes of the MPs in CP-LBBD models for the SBF2 dataset is shown in Fig. 23. The results in Fig. 3 also coincide with these findings (CP-LBBD$_a$ and CP-LBBD$_c$ outperform CP-LBBD$_p$) from a regression perspective.

8 Discussion

Our experiments and analysis show a surprising difference in the behavior of different solvers in solving the MP in LBBD. Despite the fact that the models and cuts are

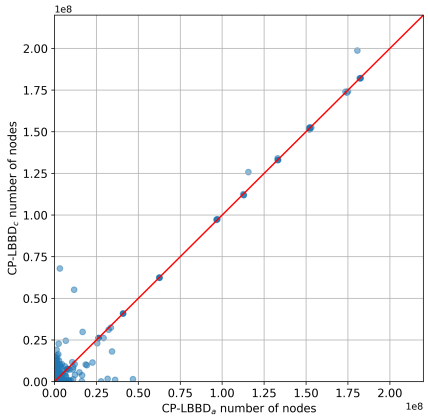

Fig. 19: Number of nodes of the MPs in CP-LBBD$_c$ and CP-LBBD$_a$.

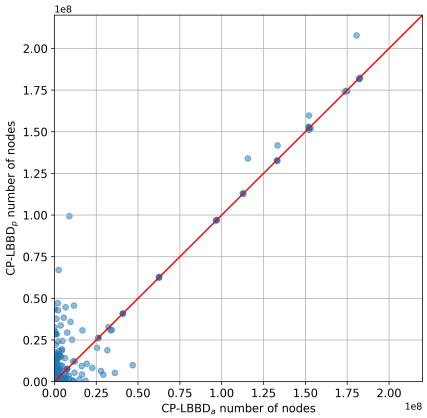

Fig. 20: Number of nodes of the MPs in CP-LBBD$_p$ and CP-LBBD$_a$.

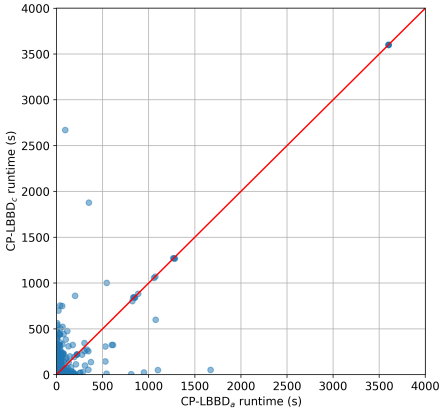

Fig. 21: Runtime of the MPs in CP-LBBD$_c$ and CP-LBBD$_a$.

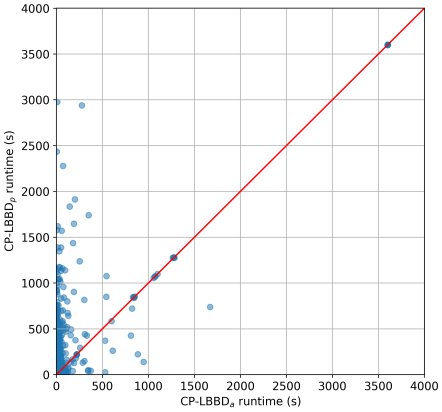

Fig. 22: Runtime of the MPs in CP-LBBD$_p$ and CP-LBBD$_a$.

mathematically equivalent, it appears that different approaches find MP solutions that push the global search behavior toward more or fewer iterations. We provide some evidence that the similarity of consecutive MP solutions may play a role here but further research is required to fully understand our observations.

Global constraints in CP can increase domain propagation and the overall solving performance but have a limit, after which the improved propagation, if any, is not worth the effort required [51]. This dynamic may be observed by the worse results of CP-LBBD$_p$ compared to CP-LBBD$_a$ and CP-LBBD$_c$. By contrast, CP-LBBD$_a$ and

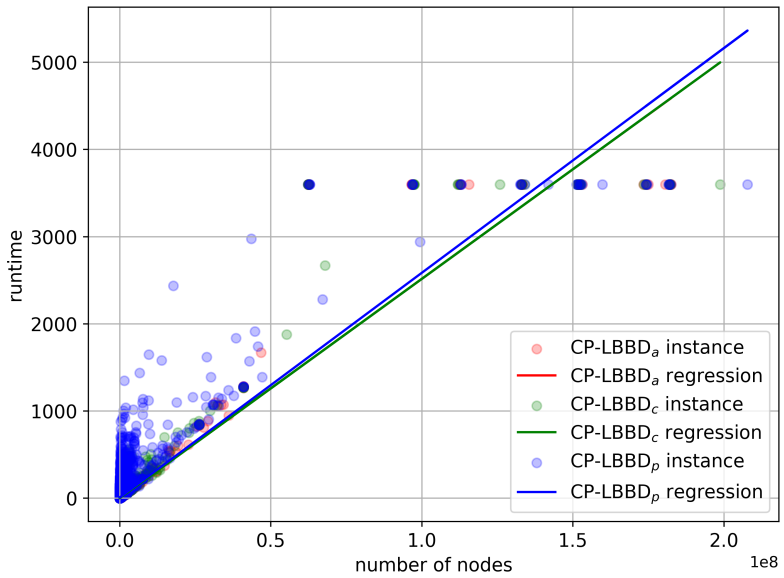

Fig. 23: Runtime over number of nodes of the MPs in CP-LBBD models for the SBF2 dataset. Note that the regression lines for CP-LBBD$_a$ and CP-LBBD$_c$ overlap.

CP-LBBD$_c$ manipulate the main decision variables more directly while not inducing much larger constraint models. This trade-off between stronger propagation and increasing model size is prevalent in applications of CP techniques.

The validity of the proposed four DIDP cut encoding methods depends on the effective extraction of the useful information, i.e., the change of the variable-value pairs in the Benders cuts. Such information is often hidden in the transitions of DIDP models. Thus, it is difficult to create a cut using the existing state variables. An important question is to understand if this state-space expansion is an inherent weakness for DIDP and, indeed, state-based models in general. There exists similar work examining the addition of trajectory constraints to AI planning problems which similarly expand the state space [52, 53]. Also, it is worth investigating how to improve the DIDP cut encodings. Methods such as cut aggregation [47, 54], cut selection [55], adaptive cuts [56] and cut removal [57] are promising to improve the cut quality or reduce the number of cuts to overcome the performance degradation caused by the state-space expansion.

In a parallel work, a monolithic DIDP model for SUALBP-1 was shown to perform better than all the LBBD models presented here [20]. This is a surprising result as the state of the art for similar problems with sequence-dependent setup times is typically based on decomposition [14, 58]. Further research is required to understand why DIDP models for SUALBP-1 do not follow this pattern. We speculate that the expanded state space plays a role.

It is noteworthy to mention that Benders feasibility cuts are relatively straightforward to formulate as the essence is to prune a subset of solutions directly. For Benders optimality cuts, however, the pruning becomes indirect and objective-related, which is more challenging to develop, especially for DIDP. It would be interesting to develop DIDP-based and CP-based encoding methods for the Benders optimality cuts.

9 Conclusions

In this paper, we investigated a rarely studied topic: using non-MIP methods to solve the master problem in the logic-based Benders decomposition (LBBD) framework. This idea is simple yet novel as Benders cuts are expressed as linear mathematical constraints in the form of MIP constraints. However, we show that alternative methods are possible, effective, and efficient. Additionally, the different MP solving methods provide surprisingly varied performance.

Specifically, we proposed novel LBBD models with master problems modeled and solved with domain-independent dynamic programming (DIDP) and constraint programming (CP), using simple assembly line balancing problem with sequence-dependent setup times type-1 (SUALBP-1) as a testbed. In the state transition system of DIDP, we proposed four encoding methods for Benders feasibility cuts by using integer or set variables and preconditions or state constraints. As it is challenging to transform linear constraints to an equivalent form in DIDP, we use a new state variable to keep track of the status of the cut. The body of the cut is updated with the state transitions and the cut logic is evaluated by either transition preconditions or state constraints. Although the four LBBD models are for SUALBP-1, the four encoding methods are general and can be applied to the DIDP-LBBD models for other combinatorial optimization problems.

We also developed three CP-based master problem formulations with Benders feasibility cuts formulated as variable assignment bounds, variable assignment counting, and global constraints, with increasing strength and complexity of propagation. Though designed for SUALBP-1, these three modeling ideas in CP can be borrowed by other optimization problems to formulate Benders cuts as well.

Experimental results on SUALBP-1 show superior performance for the CP-LBBD models and good performance of the four DIDP-LBBD models, compared to MIP-LBBD and monolithic MIP models, demonstrating the promise of DIDP hybrid and CP hybrid approaches. However, the DIDP encodings exhibit state-space expansion and performance degradation as the number of cuts increases, while the CP encoding with global constraints does not pay off in terms of the computational effort required to enforce the stronger propagation. Thus, further improvements are needed for these cut encoding methods. Considering the long history of the research and development for the LBBD models with MIP master problems, DIDP-based and CP-based LBBD models have great potential to be explored.

Acknowledgements.

Declarations

Funding. This work was financially supported by the Natural Sciences and Engineering Research Council of Canada.

Conflict of interest. Author J. Christopher Beck is the past President of the Executive Committee of the Association for Constraint Programming. The authors declare they have no financial or non-financial interests.

Ethics approval and consent to participate. Not applicable.

Consent to publication. Not applicable.

Data availability. The problem instances used in this work are available online (https://github.com/JasonZhangjc/cp_didp_lbbd).

Materials availability. Not applicable.

Code availability. The code will be available online.

Author contribution. Jiachen Zhang proposed the idea and developed the decomposition algorithms. J. Christopher Beck is the advisor of this work. Jiachen Zhang and J. Christopher Beck wrote the manuscript.

Appendix A Monolithic MIP Model of SUALBP-1

To present the monolithic MIP model of SUALBP-1, additional parameters are required, as shown in Table A1. We base our formulation on the Second Station-Based Formulation (SSBF) [11], the state-of-the-art MIP formulation for SUALBPs. Since the SSBF model can be adapted to both SUALBP-1 and SUALBP-2, we name it SSBF-1 [11]. The decision variables are:

- x_{ik}: binary variable with value 1, iff task $i \in V$ is assigned to station $k \in FS_i$.
- z_i: integer variable for encoding the index of the station that task i is assigned to.
- u_k: binary variable with value 1, iff any task is assigned to station k.
- g_{ijk}: binary variable $= 1$, iff task i is performed right before task j on station k.
- h_{ijk}: binary variable $= 1$, iff task i is the last and task j is the first task on station k.
- r_i: integer variable representing the rank of task i in a sequence of all tasks. The sequence is composed of the task sequences on all the active stations.

The SSBF-1 MIP model proposed by Esmaeilbeigi et al. [11] is as follows.

$$\min \sum_{k \in KP} u_k + \underline{m} \tag{A1a}$$

$$\text{s.t.} \sum_{k \in FS_i} x_{ik} = 1, \qquad \forall i \in V, \tag{A1b}$$

$$\sum_{k \in FS_i} k \cdot x_{ik} = z_i, \qquad \forall i \in V, \tag{A1c}$$

Table A1: Additional parameters for SUALBP-1 [11].

Notation	Definition
$\mathcal{E}$	set of all precedence relations
E_i	earliest station for task $i \in V$, e.g., $E_i = \left\lceil \frac{t_i + \sum_{j \in P_i^*} t_j}{\overline{c}} \right\rceil$
L_i	latest station for task $i \in V$, e.g., $L_i = \overline{m} + 1 - \left\lceil \frac{t_i + \sum_{j \in F_i^*} t_j}{\overline{c}} \right\rceil$
d_{ij}	lower bound on the number of stations between the stations of tasks i and j, inclusive, e.g., $d_{ij} = \left\lceil \frac{t_i + t_j - 1 + \sum_{v \in S_i^* \cap P_j^*} t_v}{c} \right\rceil$
$KD(KP)$	set of definite (possible) stations, i.e., $KD = \{1, ..., \underline{m}\}$, $KP = \{\underline{m}+1, ..., \overline{m}\}$, and $K = KD \cup KP$
FS_i	set of stations to which task $i \in V$ can be assigned, i.e., $FS_i = \{E_i, E_i + 1, ..., L_i\}$
FT_k	set of tasks which can be assigned to station $k \in K$, i.e., $FT_k = \{i \in V \vert k \in FS_i\}$
A_i	set of tasks that cannot be assigned to the station to which task i is assigned, e.g., $A_i = \{j \in V \vert FS_j \cap FS_i = \emptyset\}$
$F_i^F(P_i^F)$	set of tasks which may directly follow (precede) task i in forward direction, i.e., $F_i^F = \{j \in V - (F_i^* - F_i) - P_i^* - A_i - \{i\}\}$ and $P_i^F = \{j \in V \vert i \in F_j^F\}$
$F_i^B(P_i^B)$	set of tasks which may directly follow (precede) task i in backward direction, i.e., $F_i^B = \{j \in V - F_i^* - A_i\}$ and $P_i^B = \{j \in V \vert i \in F_j^B\}$

$$\sum_{i \in FT_k \cap F_i^F} g_{ijk} + \sum_{i \in FT_k \cap F_i^B} h_{ijk} = x_{ik}, \qquad \forall i \in V,\ \forall k \in FS_i, \qquad \text{(A1d)}$$

$$\sum_{i \in FT_k \cap P_j^F} g_{ijk} + \sum_{i \in FT_k \cap P_j^B} h_{ijk} = x_{jk}, \qquad \forall j \in V,\ \forall k \in FS_j, \qquad \text{(A1e)}$$

$$\sum_{i \in FT_k} \sum_{j \in (FT_k \cap F_i^B)} h_{ijk} = 1, \qquad \forall k \in KD, \qquad \text{(A1f)}$$

$$\sum_{i \in FT_k} \sum_{j \in (FT_k \cap F_i^B)} h_{ijk} = u_k, \qquad \forall k \in KP, \qquad \text{(A1g)}$$

$$(n - |F_i^*| - |P_j^*|) \cdot \Big(\sum_{k \in (FS_i \cap FS_j)} g_{ijk} - 1 \Big) + r_i + 1 \leq r_j, \qquad \forall i \in V,\ \forall j \in F_i^F, \qquad \text{(A1h)}$$

$$r_i + 1 \leq r_j, \qquad \forall (i,j) \in \mathcal{E}, \qquad \text{(A1i)}$$

$$z_i \leq z_j, \qquad \forall (i,j) \in \mathcal{E}, \qquad \text{(A1j)}$$

$$\sum_{i \in FT_k} t_i x_{ik} + \sum_{i \in FT_k} \sum_{j \in (FT_k \cap F_i^F)} \tau_{ij} g_{ijk} + \sum_{i \in FT_k \cap P_i^B} \mu_{ij} h_{ijk} \leq c, \qquad \forall k \in KD, \qquad \text{(A1k)}$$

$$\sum_{i \in FT_k} t_i x_{ik} + \sum_{i \in FT_k} \sum_{j \in (FT_k \cap F_i^F)} \tau_{ij} g_{ijk} + \sum_{i \in FT_k \cap P_i^B} \mu_{ij} h_{ijk} \leq c \cdot u_k, \qquad \forall k \in KP, \qquad \text{(A1l)}$$

$$\sum_{i \in FT_k \setminus \{j\}} x_{ik} \leq (n - \underline{m} + 1) \cdot (1 - h_{jjk}), \qquad \forall k \in K,\ \forall j \in FT_k, \quad \text{(A1m)}$$

$$u_{k+1} \leq u_k, \qquad \forall k \in KP \setminus \{\overline{m}\}, \quad \text{(A1n)}$$

$$g_{ijk} \in \{0,1\}, \qquad \forall k \in K, \quad \forall i \in FT_k, \quad \forall j \in (FT_k \cap F_i^F), \quad \text{(A1o)}$$

$$h_{ijk} \in \{0,1\}, \qquad \forall k \in K, \quad \forall i \in FT_k, \quad \forall j \in (FT_k \cap F_i^B), \quad \text{(A1p)}$$

$$|P_i^*| + 1 \leq r_i \leq n - |F_i^*|, \qquad \forall i \in V, \quad \text{(A1q)}$$

$$x_{ik} \in \{0,1\}, \qquad \forall i \in V,\ \forall k \in FS_i, \quad \text{(A1r)}$$

$$r_i, z_i \in \mathbb{Z}^+, \qquad \forall i \in V, \quad \text{(A1s)}$$

The objective (A1a) minimizes the number of stations. Constraints (A1b) ensure that a task is assigned to a station. Constraints (A1c) link x_{ik} and z_i. Constraints (A1d) and (A1e) assure that a task on station k is followed and preceded by exactly one other task in the cyclic sequence of this station. According to constraints (A1f) and (A1g), in each cycle exactly one of the relations is a backward setup. Constraints (A1h) and (A1i) establish the precedence relations among the tasks within each station. Note that the constraints (A1h) are inactive if tasks i and j are assigned to different stations. We add the constraints (A1j) to make sure that the precedence relations among the tasks of different stations are satisfied. Knapsack constraints (A1k) and (A1l) ensure that no station time exceeds the cycle time. Constraints (A1m) guarantee that only task j is allocated to station k when $h_{jjk} = 1$. Constraints (A1n) guarantee that stations are used in the correct order and no empty station is in the middle of used stations. Constraints (A1o) to (A1s) specify the domain of the variables.

Note that the decision variables r_i and z_i are set to continuous in the original SSBF model [11]. However, doing so results in infeasible solutions being labeled as feasible for some problem instances. In addition to the MIP model, Esmaeilbeigi et al. [11] developed pre-processing techniques to reduce the number of variables and constraints. We implement all these techniques, as well.

References

[1] Hooker, J.N., Ottosson, G.: Logic-based Benders decomposition. Mathematical Programming **96**(1), 33–60 (2003)

[2] Hooker, J.N.: Planning and scheduling by logic-based Benders decomposition. Operations Research **55**(3), 588–602 (2007)

[3] Leutwiler, F., Corman, F.: A logic-based Benders decomposition for microscopic railway timetable planning. European Journal of Operational Research **303**(2), 525–540 (2022)

[4] Guo, C., Bodur, M., Aleman, D.M., Urbach, D.R.: Logic-based Benders decomposition and binary decision diagram based approaches for stochastic distributed operating room scheduling. INFORMS Journal on Computing **33**(4), 1551–1569 (2021)

[5] Forbes, M.A., Harris, M.G., Jansen, H., Van Der Schoot, F.A., Taimre, T.: Combining optimisation and simulation using logic-based Benders decomposition. European Journal of Operational Research **312**(3), 840–854 (2024)

[6] Tran, T.T., Beck, J.C.: Logic-based Benders decomposition for alternative resource scheduling with sequence dependent setups. In: Proceedings of the 20th European Conference on Artificial Intelligence, pp. 774–779 (2012)

[7] Daryalal, M., Pouya, H., DeSantis, M.A.: Network migration problem: A hybrid logic-based Benders decomposition approach. INFORMS Journal on Computing **35**(3), 593–613 (2023)

[8] Kuroiwa, R., Beck, J.C.: Solving domain-independent dynamic programming problems with anytime heuristic search. In: the 33rd International Conference on Automated Planning and Scheduling (ICAPS 2023), 245–253. (2023)

[9] Kuroiwa, R., Beck, J.C.: Domain-independent dynamic programming. arXiv preprint arXiv:2401.13883 (2024)

[10] Kuroiwa, R., Beck, J.C.: Domain-independent dynamic programming: Generic state space search for combinatorial optimization. In: the 33rd International Conference on Automated Planning and Scheduling (ICAPS 2023), 236–244. (2023)

[11] Esmaeilbeigi, R., Naderi, B., Charkhgard, P.: New formulations for the setup assembly line balancing and scheduling problem. OR Spectrum **38**, 493–518 (2016)

[12] Zhang, J., Beck, J.C.: Solving LBBD master problems with constraint programming and domain-independent dynamic programming. In: 30th International Conference on Principles and Practice of Constraint Programming (CP 2024), pp. 611–631 (2024). Schloss Dagstuhl–Leibniz-Zentrum für Informatik

[13] Hooker, J.: Logic-based Methods for Optimization: Combining Optimization and Constraint Satisfaction. John Wiley & Sons, New York (2011)

[14] Zohali, H., Naderi, B., Roshanaei, V.: Solving the type-2 assembly line balancing with setups using logic-based Benders decomposition. INFORMS Journal on Computing **34**(1), 315–332 (2022)

[15] Akpinar, S., Elmi, A., Bektaş, T.: Combinatorial Benders cuts for assembly line balancing problems with setups. European Journal of Operational Research **259**(2), 527–537 (2017)

[16] Becker, C., Scholl, A.: A survey on problems and methods in generalized assembly line balancing. European Journal of Operational Research **168**(3), 694–715 (2006)

[17] Kumar, N., Mahto, D.: Assembly line balancing: a review of developments and trends in approach to industrial application. Global Journal of Research in Engineering - G: Industrial Engineering **13**(2), 29–50 (2013)

[18] Andres, C., Miralles, C., Pastor, R.: Balancing and scheduling tasks in assembly lines with sequence-dependent setup times. European Journal of Operational Research **187**(3), 1212–1223 (2008)

[19] Scholl, A., Boysen, N., Fliedner, M.: The assembly line balancing and scheduling problem with sequence-dependent setup times: Problem extension, model formulation and efficient heuristics. OR Spectrum **35**, 291–320 (2013)

[20] Zhang, J., Beck, J.C.: Domain-independent dynamic programming and constraint programming approaches for assembly line balancing problems with setups. INFORMS Journal on Computing **37**(4), 977–997 (2025)

[21] Michels, A.S., Lopes, T.C., Sikora, C.G.S., Magatão, L.: A benders' decomposition algorithm with combinatorial cuts for the multi-manned assembly line balancing problem. European Journal of Operational Research **278**(3), 796–808 (2019)

[22] Baybars, I.: A survey of exact algorithms for the simple assembly line balancing problem. Management Science **32**(8), 909–932 (1986)

[23] Ritt, M., Costa, A.M.: Improved integer programming models for simple assembly line balancing and related problems. International Transactions in Operational Research **25**(4), 1345–1359 (2018)

[24] Zhang, W.: Complete anytime beam search. In: Proceedings of the 1998 15th National Conference on Artificial Intelligence, AAAI (1998)

[25] Vadlamudi, S.G., Gaurav, P., Aine, S., Chakrabarti, P.P.: Anytime column search. AI 2012: Advances in Artificial Intelligence, 254

[26] Chu, Y., Xia, Q.: Generating Benders cuts for a general class of integer programming problems. In: Régin, J.-C., Rueher, M. (eds.) Proceedings of the 1st International Conference on the Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems (CPAIOR 2004), vol. 3011, pp. 127–136. Springer, Berlin Heidelberg (2004)

[27] Codato, G., Fischetti, M.: Combinatorial Benders' cuts for mixed-integer linear programming. Operations Research **54**(4), 756–766 (2006)

[28] Dechter, R.: Enhancement schemes for constraint processing: Backjumping, learning, and cutset decomposition. Artificial Intelligence **41**(3), 273–312 (1990)

[29] Katsirelos, G., Bacchus, F.: Generalized nogoods in CSPs. In: Proceedings of the 20th National Conference on Artificial intelligence-Volume 1, pp. 390–396 (2005)

[30] Bayardo Jr, R.J., Schrag, R.: Using CSP look-back techniques to solve real-world sat instances. In: Proceedings of the 14th National Conference on Artificial Intelligence, pp. 203–208 (1997)

[31] Sadykov, R.: A branch-and-check algorithm for minimizing the weighted number of late jobs on a single machine with release dates. European Journal of Operational Research **189**(3), 1284–1304 (2008)

[32] Coban, E., Hooker, J.N.: Single-facility scheduling by logic-based Benders decomposition. Annals of Operations Research **210**, 245–272 (2013)

[33] Lam, E., Gange, G., Stuckey, P.J., Van Hentenryck, P., Dekker, J.J.: Nutmeg: a MIP and CP hybrid solver using branch-and-check. SN Operations Research Forum **1**, 1–27 (2020)

[34] Bukchin, Y., Raviv, T.: Constraint programming for solving various assembly line balancing problems. Omega **78**, 57–68 (2018)

[35] Shaw, P.: A constraint for bin packing. In: Proceedings of the 10th International Conference on Principles and Practice of Constraint Programming (CP 2004), pp. 648–662 (2004). Springer

[36] IBM: IBM ILOG CPLEX Optimizer. Accessed on 2024-04-20. https://www.ibm.com/products/ilog-cplex-optimization-studio/cplex-cp-optimizer

[37] Beck, J.C., Kuroiwa, R., Lee, J.H., Stuckey, P.J., Zhong, A.Z.: Transition dominance in domain-independent dynamic programming. In: 31st International Conference on Principles and Practice of Constraint Programming (CP 2025), pp. 5–1 (2025). Schloss Dagstuhl–Leibniz-Zentrum für Informatik

[38] Scholl, A., Klein, R.: Salome: A bidirectional branch-and-bound procedure for assembly line balancing. INFORMS Journal on Computing **9**(4), 319–334 (1997)

[39] Akpinar, Ş., Baykasoğlu, A.: Modeling and solving mixed-model assembly line balancing problem with setups. part I: A mixed integer linear programming model. Journal of Manufacturing Systems **33**(1), 177–187 (2014)

[40] Yilmaz, H.: Modeling and solving assembly line worker assignment and balancing problem with sequence-dependent setup times. Soft Computing **25**(20), 12899–12914 (2021)

[41] Furugi, A.: Sequence-dependent time-and cost-oriented assembly line balancing problems: A combinatorial Benders' decomposition approach. Engineering Optimization **54**(1), 170–184 (2022)

[42] Gurobi Optimization, L.: Gurobi Optimizer Reference Manual. Accessed on 2024-04-10 (2021). http://www.gurobi.com

[43] Beck, J.C.: Checking-up on branch-and-check. In: Cohen, D. (ed.) Proceedings of the 16th International Conference on Principles and Practice of Constraint Programming (CP 2010), pp. 84–98. Springer, Berlin Heidelberg (2010). https://doi.org/10.1007/978-3-642-15396-9_10

[44] Amor, H.B., Desrosiers, J., Frangioni, A.: Stabilization in column generation. Les Cahiers du GERAD ISSN **711**, 2440 (2004)

[45] Desaulniers, G., Desrosiers, J., Solomon, M.M.: Column Generation vol. 5. Springer, New York (2006)

[46] Ruszczyński, A.: A regularized decomposition method for minimizing a sum of polyhedral functions. Mathematical Programming **35**, 309–333 (1986)

[47] Linderoth, J., Wright, S.: Decomposition algorithms for stochastic programming on a computational grid. Computational Optimization and Applications **24**, 207–250 (2003)

[48] Rahmaniani, R., Crainic, T.G., Gendreau, M., Rei, W.: The Benders decomposition algorithm: A literature review. European Journal of Operational Research **259**(3), 801–817 (2017)

[49] Zverovich, V., Fábián, C.I., Ellison, E.F., Mitra, G.: A computational study of a solver system for processing two-stage stochastic LPs with enhanced Benders decomposition. Mathematical Programming Computation **4**, 211–238 (2012)

[50] IBM: IBM ILOG CP Optimizer. Accessed on 2024-04-20. https://ibmdecisionoptimization.github.io/docplex-doc/cp.html

[51] Rossi, F., Van Beek, P., Walsh, T.: Constraint programming. Foundations of Artificial Intelligence **3**, 181–211 (2008)

[52] Hsu, C.-W., Wah, B.W., Huang, R., Chen, Y.: Constraint partitioning for solving planning problems with trajectory constraints and goal preferences. In: Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI 2007), pp. 1924–1929 (2007)

[53] Baier, J.A., Bacchus, F., McIlraith, S.A.: A heuristic search approach to planning with temporally extended preferences. Artificial Intelligence **173**(5-6), 593–618 (2009)

[54] Rahmaniani, R., Crainic, T.G., Gendreau, M., Rei, W.: The Asynchronous Benders Decomposition Method. CIRRELT (2018)

[55] Hosseini, M., Turner, J.: Deepest cuts for Benders decomposition. Operations Research **73**(5), 2591–2609 (2024)

[56] Ramírez-Pico, C., Ljubić, I., Moreno, E.: Benders adaptive-cuts method for two-stage stochastic programs. Transportation Science **57**(5), 1252–1275 (2023)

[57] Lee, M., Ma, N., Yu, G., Dai, H.: Accelerating generalized Benders decomposition for wireless resource allocation. IEEE Transactions on Wireless Communications **20**(2), 1233–1247 (2020)

[58] Tran, T.T., Araujo, A., Beck, J.C.: Decomposition methods for the parallel machine scheduling problem with setups. INFORMS Journal on Computing **28**(1), 83–95 (2016)