Solving LBBD Master Problems with Constraint Programming and Domain-Independent Dynamic Programming

Jiachen Zhang
Department of Mechanical and Industrial Engineering, University of Toronto, Canada

J. Christopher Beck
Department of Mechanical and Industrial Engineering, University of Toronto, Canada

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, in particular addressing the challenge of feasibility cut formulation. For CP, we exploit key variable manipulation, constraint-based expressions, 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 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.

2012 ACM Subject Classification Mathematics of computing → Combinatorial optimization

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

Digital Object Identifier 10.4230/LIPIcs.CP.2024.34

Funding *J. Christopher Beck*: Natural Sciences and Engineering Research Council of Canada

1 Introduction

Logic-Based Benders Decomposition (LBBD) is one of the most powerful and convenient patterns of problem decomposition for solving combinatorial optimization problems [15]. While the most common combination within the Constraint Programming (CP) literature uses Mixed Integer Programming (MIP) for master problems and CP for subproblems [14], LBBD is compatible with various modeling and solving techniques. For example, subproblems have been modeled and solved with Satisfiability Modulo Theories (SMT) [22], Binary Decision Diagrams [11], and problem-specific algorithms [10, 29]. However, work investigating modeling and solution methods other than MIP for master problems in LBBD is sporadic [8]. In this paper, we explore the 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 [21], 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.

© Jaichen Zhang & J. Christopher Beck;
licensed under Creative Commons License CC-BY 4.0
30th International Conference on Principles and Practice of Constraint Programming (CP 2024).
Editor: Paul Shaw; Article No. 34; pp. 34:1–34:21
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Domain-Independent Dynamic Programming (DIDP) is a recent exact framework to model and solve combinatorial optimization problems [19, 20]. 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) [9]. 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 [21], thus this choice allows us to test whether cuts can be formulated to maintain this advantage.

Our contributions are summarized as follows.

1. We formulate three alternative representations of feasibility cuts for SUALBP-1 for a CP-based master problem.
2. We propose four approaches to encode Benders feasibility cuts in a DIDP model of LBBD master problems based on using integer or set variables to encode preconditions or state constraints. We apply these approaches to SUALBP-1 and develop four feasibility cut encodings for a DIDP-based master problem.
3. We obtain superior results for SUALBP-1 in solving master problems with CP and DIDP rather than MIP, with CP outperforming DIDP. We provide statistical analysis and insights on our seven novel cut formulations.

This paper is organized as follows. The background is covered in Section 2. The three novel CP feasibility cut formulations for SUALBP-1 are introduced in Section 3. The four encoding methods of Benders feasibility cuts in DIDP and their instantiations for SUALBP-1 are presented in Section 4. The experimental results are presented in Section 5. We discuss the proposed approaches and results in Section 6, followed by our conclusions.

2 Background

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 $\mathbf{x}=\overline{\mathbf{x}}$, the remaining subproblem is defined, often in the form of multiple independent subproblems. The subproblem (SP) has the form

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

LBBD analyzes the SP solution 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., $B_{\overline{\mathbf{x}}}(\mathbf{x})=SP(\overline{\mathbf{x}})$ [15].

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 LBBD variants, distinguished by subproblem types. When a subproblem is an optimization problem, we deduce a lower bound on ϕ^* in the form of a Benders optimality cut [31]. When a subproblem is a feasibility problem, a set of MP solutions are pruned by the corresponding Benders feasibility cut [1] 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 [20]. 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.

A transition τ is a 4-tuple $\langle eff_\tau, cost_\tau, pre_\tau, forced_\tau \rangle$ where eff_τ is the set of effects, $cost_\tau$ is the cost, pre_τ is the set of preconditions, and $forced_\tau \in \{\top, \bot\}$, 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 [20, 21].

2.3 SUALBP-1

The Simple Assembly Line Balancing Problem (SALBP) is a well-studied production planning problem [5]. As setup operations such as tool changes, curing, or cooling processes are often required between consecutive tasks in real production lines [18], SUALBP incorporates setup times into SALBP [2], 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

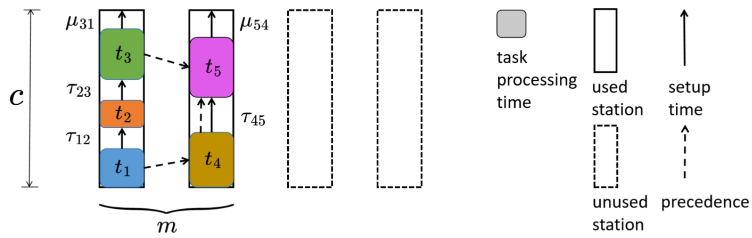


Figure 1 Example of SUALBP-1.

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., $j, j+1, j+2, ..., m$). 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 by Esmaeilbeigi et al. [9], as shown in the Table 1 for SUALBP-1. To obtain all the parameters in the table, we adapt the preprocessing techniques in the literature [20, 9, 31].

SUALBP-1 has been solved with a number of approaches including MIP [9] and heuristics [25]. The state-of-the-art MIP model is the Second Station-Based Formulation (SSBF) [9] defined in 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 [1] that can be adapted (with significant simplification) to SUALBP-1. We

Table 1 Notation and definition for SUALBP-1 [9].

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., $\underline{t}_i = t_i + \min(\underline{\tau}_i, \underline{\mu}_i)$

discuss this model in Section 5.

In our parallel work currently under review [30], 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 CP-LBBD for SUALBP-1

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

3.1 CP Master Problem

SUALBP-1 fixes the cycle time (maximum station time) and seeks to minimize the number of stations used. In the LBBD framework, we decompose the problem to an assignment master problem and a scheduling subproblem for each station.

In all our approaches, the master problem assigns tasks to stations, minimizing the number of stations used, and ensuring that the precedence constraints between tasks and the cycle time limit are not violated. Without any Benders cuts, this master problem is identical to the Simple Assembly Line Balancing Problem type-1 (SALBP-1) [4].

For SALBP-1, Kuroiwa and Beck [20] improved the CP model proposed by Bukchin and Raviv [6] by using `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 and (2) three different combinatorial formulations of Benders feasibility cuts are used, one formulation in each model.

We define E_i as a lower bound on the number of stations required to schedule task i, L_i as a lower bound on the number of stations between the station of task i and the last station, inclusive, and d_{ij} as a lower bound on the number of stations between the stations of tasks i and j, inclusive:

$$E_i = \left\lceil \frac{\underline{t}_i + \sum_{j \in P_i^*} \underline{t}_j}{c} \right\rceil, \quad L_i = \left\lfloor \frac{\underline{t}_i - 1 + \sum_{j \in S_i^*} \underline{t}_j}{c} \right\rfloor, \quad d_{ij} = \left\lceil \frac{\underline{t}_i + \underline{t}_j - 1 + \sum_{v \in S_i^* \cap P_j^*} \underline{t}_v}{c} \right\rceil.$$

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:

$$\min z \tag{4a}$$
$$\text{s.t. } \texttt{Pack}(\{y_k | k \in K\}, \{x_i | i \in V\}, \{\underline{t}_i | i \in V\}), \tag{4b}$$
$$0 \leq y_k \leq c, \qquad \forall k \in K, \tag{4c}$$
$$E_i - 1 \leq x_i \leq z - 1 - L_i, \qquad \forall i \in V, \tag{4d}$$
$$x_i + d_{ij} \leq x_j, \qquad \forall j \in V, \forall i \in P_j^*, \nexists v \in S_i^* \cap P_j^* : d_{ij} \leq d_{iv} + d_{vj}. \tag{4e}$$

The `Pack` global constraint [27] ensures that for tasks 'packed' onto stations, $y_k = \sum_{i \in V, x_i = k} \underline{t}_i$. Constraints (4c) and (4d) state the domains of y_k and x_i. Constraint (4b) and (4c) together ensure that the total task time on each station does not exceed the cycle time. Constraint (4e) is an enhanced version of the precedence constraint using d_{ij}.

3.2 CP Formulations for Benders Feasibility Cuts

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

Let $\mathcal{J}$ be the set of subproblems leading to Benders cuts. Consider subproblem $j \in \mathcal{J}$ corresponding to station k, let $\mathcal{I}^j$ be the set of tasks assigned to the station that cannot all be scheduled within the cycle time, then the j-th Benders feasibility cut based on manipulation of the key decision variables, i.e., the station assignment specified by x_i, is as follows:

$$\sum_{i \in \mathcal{I}^j} (x_i = k) \leq |\mathcal{I}^j| - 1, \ \forall k \in K. \tag{5}$$

Chu and Xia defined a valid Benders cut as a logical expression having two properties [7]:

- Property 1: The cut must exclude the current MP solution if it is not globally feasible.
- Property 2: The cut must not remove any globally feasible solutions.

Property 1 ensures finite convergence if the MP variables have finite domains. Property 2 assures optimality since the cut never removes globally feasible solutions.

▶ **Proposition 1.** *Cut (5) is valid.*

Proof. 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 Property 1. Since the solutions removed by this encoding are all infeasible globally with the set of tasks $\mathcal{I}^j$ assigned to any station, Property 2 is satisfied. ◀

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 [17]. The j-th cut based on `Count_Different` is as follows:

$$\texttt{Count\_Different}(\{x_i | i \in \mathcal{I}^j\}) \geq 2. \tag{6}$$

▶ **Proposition 2.** *Cut (6) 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 (5). Thus, Properties 1 and 2 are satisfied. ◀

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

$$\texttt{Pack}(\{w_k | k \in K\}, \{x_i | i \in \mathcal{I}^j\}, \{\mathbf{1}_i | i \in \mathcal{I}^j\}), \tag{7}$$

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

▶ **Proposition 3.** *Cut (7) 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 Property 1. Similar to the proof for Proposition 1, Property 2 is satisfied. ◀

The CP-LBBD models with cut (5), (6), and (7) are referred to as CP-LBBD$_a$, CP-LBBD$_c$, and CP-LBBD$_p$, corresponding to 'assignment', 'count', and 'pack', respectively.

4 DIDP-LBBD for SUALBP-1

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

4.1 Master Problem

As stated in Section 3.1, the master problem is equivalent to the SALBP-1. Our DIDP formulations for the master problem (with Benders cuts) of SUALBP-1 are inspired by an existing DIDP model for SALBP-1 [20], 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.

- $Assign_i = \langle U \to U \backslash \{i\} \,\wedge\, r \to r - \underline{t}_i,\ 0,\ i \in U \,\wedge\, \underline{t}_i \le r \,\wedge\, U \cap P_i^* = \emptyset,\ \perp\rangle$: assign task i to the current station.
- $Open = \langle r \to c,\ 1,\ (i \notin U \,\vee\, r < \underline{t}_i \,\vee\, U \cap P_i^* \neq \emptyset) \mid \forall i \in V,\ \perp\rangle$: 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 dominance rule, 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 [20] but presents a complication for our cut formulations (see Section 4.3.2).

Recursive function. We use $f(U, r)$ to represent the cost of a state. Let $U_1 = \{i \in U \mid r \ge \underline{t}_i \,\wedge\, U \cap P_i^* = \emptyset\}$ 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:

$$\texttt{compute } f(V, 0) \tag{8a}$$

$$f(U,r) = \begin{cases} 0 & \text{if } U = \emptyset, \quad \text{(i)} \\ 1 + f(U, c) & \text{else if } U_1 = \emptyset, \quad \text{(ii)} \\ \min_{i \in U_1} f(U \backslash \{i\}, r - \underline{t}_i) & \text{else,} \quad \text{(iii)} \end{cases} \tag{8b}$$

$$f(U,r) \le f(U,r'), \quad \text{if } r \ge r', \tag{8c}$$

$$f(U,r) \ge \max \begin{cases} \left\lceil \frac{\sum_{i \in U} \underline{t}_i - r}{c} \right\rceil, & \text{(i)} \\ \sum_{i \in U} w_i^2 + \left\lceil \sum_{i \in U} w_i'^2 - l^2 \right\rceil, & \text{(ii)} \\ \left\lceil \sum_{i \in U} w_i^3 - l^3 \right\rceil. & \text{(iii)} \end{cases} \tag{8d}$$

The term (8a) is to compute the cost of the target state. Equation (8b) is the main recursion of the DIDP model. Specifically, (8b-i) refers to the base case, while (8b-ii)

Table 2 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						

corresponds to opening a new station and (8b-iii) refers to assigning task i to the current station. Inequality (8c) formulates state domination due to the resource variable: if other variables are equal, a state with a larger remaining time dominates. (8d-i), (8d-ii), and (8d-iii) are valid dual bounds proposed by Scholl and Klein [26] with numeric constants w^2, w'^2, w^3 indexed by a task i and depending on $\underline{t}_i$, as shown in Table 2.

4.2 Feasibility Cut Encoding in DIDP-LBBD

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 $\mathcal{I}^j$ 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:

$$\sum_{i \in \mathcal{I}^j} (x_i = \overline{x}_i) \leq |\mathcal{I}^j| - 1. \tag{9}$$

This form is often formulated as a linear constraint in the MIP master problem and we call it the j-th cut.

In DIDP, however, a cut of form (9) 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.

4.2.1 Counting-based Encoding

Our first two encoding methods are based on integer numeric variables in DIDP. Let g^j be an integer numeric variable that counts the active variable-value pairs in the left-hand side (LHS) of the cut (9), i.e., $g^j = \sum_{i \in \mathcal{I}^j} (x_i = \overline{x}_i)$. In the target state, the value of 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 $\mathcal{F}^j(\tau) = |\mathcal{U}^j_\tau| - |\mathcal{D}^j_\tau|$, 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 $k \in \mathcal{D}^j_\tau$. 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 $S'[g^j] = S[g^j] + \mathcal{F}^j(\tau)$.

In practice, the implementation of $\mathcal{F}$ depends on the problem and we define the encoding for SUALBP-1 later in Section 4.3. With the LHS of cut (9) modeled, we use preconditions or state constraints to model the right-hand side (RHS).

Precondition-based Encoding. Our first method for modeling the RHS of (9) is based on preconditions. Specifically, for the cut with the form (9), we add a precondition for each transition in the DIDP model that can modify the LHS variables as follows:

$$S[g^j] + \mathcal{F}^j(\tau) \leq |\mathcal{I}^j| - 1, \tag{10}$$

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

State Constraint-based Encoding. Our second method for modeling the RHS of (9) is based on state constraints that need to be satisfied by all states. The state constraint for the j-th cut is as follows:

$$S[g^j] \leq |\mathcal{I}^j| - 1, \tag{11}$$

where S is any state. A state constraint is evaluated after a state is created but a precondition would prevent the state from being created.

4.2.2 Set-based Encoding

Our second 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 (9). More specifically, the set variable Ω^j contains an element e_i iff $x_i = \overline{x}_i$ is satisfied in a state. In the target state, $\Omega^j = \emptyset$. Let $\mathcal{O}^j$ be the function that updates the value of Ω^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 $k \in \mathcal{I}^j$, let $\mathcal{U}^j_\tau$ be the set containing all such i and $\mathcal{D}^j_\tau$ be the set containing all such k, we have $\mathcal{O}^j(\tau) = (S[\Omega^j] \cup \mathcal{U}^j_\tau) \backslash \mathcal{D}^j_\tau$. Let S be a state and $S' = S[[\tau]]$ be the state reachable from S by τ, we have $S'[\Omega^j] = \mathcal{O}^j(\tau)$. Similar to the counting-based encoding, we use preconditions or state constraints to model the RHS.

Precondition-based Encoding. For the cut (9), we add a precondition for each transition that can modify $\mathcal{O}^j(\tau)$ in the DIDP model as follows:

$$\mathcal{I}^j \nsubseteq \mathcal{O}^j(\tau), \tag{12}$$

where τ is the transition. $\mathcal{O}^j(\tau)$ 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 $\mathcal{I}^j$.

State Constraint-based Encoding. The state constraint for the j-th cut is as follows:

$$\mathcal{I}^j \nsubseteq S[\Omega^j], \tag{13}$$

where S is any state.

4.2.3 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 (9) 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 6.

4.3 Encoding DIDP-LBBD Cuts for SUALBP-1

The formulations above can be used for any cut of the form (9). Here we formally present four cut formulations for SUALBP-1.

4.3.1 Counting-based Precondition 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. 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 (8b) as follows:

$$f(U, r, \{g^j \mid \forall j \in J\}) = \begin{cases} 0 & \text{if } U = \emptyset, & \text{(i)} \\ 1 + f(U, c, \{0 \mid \forall j \in J\}) & \text{else if } U_2 = \emptyset, & \text{(ii)} \\ \min_{i \in U_2} f(U \backslash \{i\}, r - \underline{t}_i, \{g^j + \mathcal{F}^j(i) \mid \forall j \in J\}) & \text{else.} & \text{(iii)} \end{cases} \tag{14}$$

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

▶ **Proposition 4.** *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 (14-ii). As shown in (14-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 Property 1. 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 Property 2 is satisfied. ◀

4.3.2 Counting-based State Constraint Encoding

We keep the modified effects and use state constraints instead of preconditions to enforce the logic of feasibility cuts. The recursive formulation (8b) becomes:

$$f(U, r, \{g^j \mid \forall j \in J\}) = \begin{cases} 0 & \text{if } U = \emptyset, & \text{(i)} \\ 1 + f(U, c, \{0 \mid \forall j \in J\}) & \text{else if } U_2 = \emptyset, & \text{(ii)} \\ \min_{i \in U_1} f(U \backslash \{i\}, r - \underline{t}_i, \{g^j + \mathcal{F}^j(i) \mid \forall j \in J\}) & \text{else if } U_1 \neq \emptyset. & \text{(iii)} \end{cases} \tag{15}$$

In (15-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:

$$g^j \leq |\mathcal{I}^j| - 1, \forall j \in \mathcal{J}. \tag{16}$$

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 (15-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 5.** *The counting-based state constraint encoding is valid.*

Proof. Similar to the proof for Proposition 4, we have $S[g^j] \leq |\mathcal{I}^j| - 1$ at any state S of the DIDP model. Property 1 and Property 2 are hence satisfied. ◀

4.3.3 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 (8b) in the DIDP model of the master problem to address all the Benders feasibility cuts:

$$f(U, r, \{\Omega^j \mid \forall j \in J\}) = \begin{cases} 0 & \text{if } U = \emptyset, & \text{(i)} \\ 1 + f(U, c, \{\emptyset \mid \forall j \in J\}) & \text{else if } U_3 = \emptyset, & \text{(ii)} \\ \min_{i \in U_3} f(U \backslash \{i\}, r - \underline{t}_i, \{\Omega^j \cup \{i\} \mid \forall j \in J\}) & \text{else.} & \text{(iii)} \end{cases} \tag{17}$$

where $U_3 = \{i \in U \mid r \geq \underline{t}_i \wedge U \cap P_i^* = \emptyset \wedge (\forall j \in \mathcal{J}, \mathcal{I}^j \nsubseteq \Omega^j \cup \{i\})\}$.

▶ **Proposition 6.** *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. With transition *Open*, the current station changes to the next station and $\Omega^j = \emptyset$, as shown in (17-ii). As shown in (17-iii), with transition $Assign_i$ for any i, since the effects never remove any element from Ω^j and $\mathcal{I}^j \nsubseteq \Omega^j \cup \{i\}$ is the precondition stated in U_3, we have $\mathcal{I}^j \nsubseteq S[\Omega^j]$ 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 Property 1. Similar to the proof for Proposition 4, Property 2 is satisfied. ◀

4.3.4 Set-based State Constraint Encoding

The recursive formulation (8b) becomes:

$$f(U, r, \{\Omega^j \mid \forall j \in J\}) = \begin{cases} 0 & \text{if } U = \emptyset, & \text{(i)} \\ 1 + f(U, c, \{\emptyset \mid \forall j \in J\}) & \text{else if } U_3 = \emptyset, & \text{(ii)} \\ \min_{i \in U_1} f(U \backslash \{i\}, r - \underline{t}_i, \{\Omega^j \cup \{i\} \mid \forall j \in J\}) & \text{else if } U_1 \neq \emptyset. & \text{(iii)} \end{cases} \tag{18}$$

The added state constraint is:

$$\mathcal{I}^j \nsubseteq \Omega^j, \forall j \in \mathcal{J}. \tag{19}$$

Similar to (15), we maintain the dominance specified by the preconditions of the transition *Open* by inserting the case violating state constraints (19) into the preconditions (the new preconditions become $U_3 = \emptyset$).

▶ **Proposition 7.** *The set-based state constraint encoding is valid.*

Proof. Similar to the proof for Proposition 6, Property 1 and Property 2 are satisfied. ◀

The DIDP-LBBD models with recursive formulation (14), (15), (17), and (18) replacing (8b) 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'.

5 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 [9] (see Appendix A) on the 788 instances of the SBF2 data set [25].[1]

5.1 MIP-LBBD Master Problem

We use a MIP-LBBD model as the baseline LBBD approach. For the master problem, instead of a simplified MIP formulation proposed by Akpinar et. al [1] 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

[1] `https://assembly-line-balancing.de/sualbsp/data-set-of-scholl-et-al-2013/`

relaxation. For the Benders cuts, linear constraints [1] 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:

$$\sum_{i \in \mathcal{I}^j} x_{ik} \leq |\mathcal{I}^j| - 1, \; \forall k \in K, \tag{20}$$

where x_{ik} is the decision variable used in the MP MIP formulation and $x_{ik} = 1$ if task i is assigned to station k and 0 otherwise.

5.2 Solving the Subproblem

In the LBBD framework for SUALBP-1, the MP solution assigns tasks to each station. 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 Travelling Salesman Problem (TSP) with precedence constraints. For this constrained TSP variant, our preliminary investigations showed that DIDP outperforms CP and MIP 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 $U_4 = \{i \in \mathcal{I}^j \mid \mathcal{I}^j \cap P_i^{j*} = \emptyset\}$.

$$\texttt{compute } \mathcal{V}(I^j, d_s, d_s) \tag{21a}$$

$$\mathcal{V}(U, s, f) = \begin{cases} 0 & \text{if } U = \emptyset \;\wedge\; s = d_s, & \text{(i)} \\ \mu_{sf} + \mathcal{V}(U, d_s, d_s) & \text{else if } U = \emptyset \;\wedge\; s \neq d_s, & \text{(ii)} \\ \mu_{si} + \min_{i \in U_4} \mathcal{V}(U \backslash \{i\}, i, f) & \text{else if } U_4 \neq \emptyset \;\wedge\; s \neq d_s, & \text{(iii)} \\ \min_{i \in U_4} \mathcal{V}(U \backslash \{i\}, i, i) & \text{else,} & \text{(iv)} \end{cases} \tag{21b}$$

$$\mathcal{V}(U, s, f) \geq \max \begin{cases} \underline{\mu}_f + \sum_{i \in U} \underline{\tau}_i, & \text{if } s = d_s, & \text{(i)} \\ 0, & \text{else.} & \text{(ii)} \end{cases} \tag{21c}$$

Case (21b-i) refers to the base case, while (21b-iv) corresponds to assigning the first task to the current empty station. Case (21b-iii) represents assigning the next task to the current station and adding the corresponding setup time. (21b-ii) represents closing the station and adding the setup time to the first task. (21c) is the dual bound [20].

Although this DIDP model is designed for optimization problems, since some DIDP solvers support anytime solving [21], by setting a primal bound, the search can be stopped after a solution satisfying all the constraints and having a total cost no greater than the cycle time minus the total processing time is found.

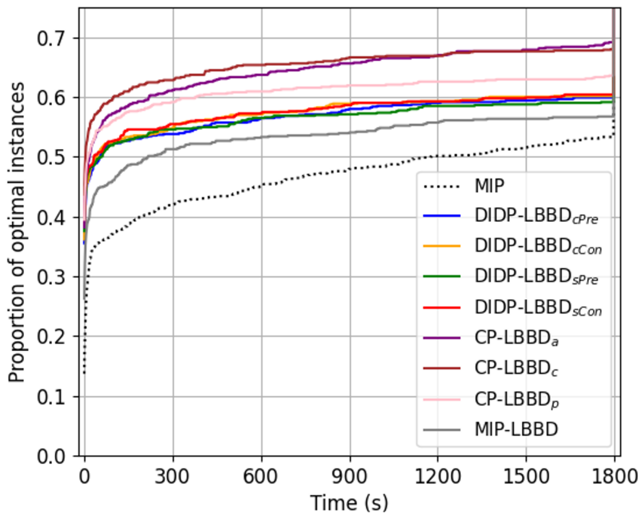


Figure 2 Ratio of instances solved and proved optimal over time for SUALBP-1.

5.3 Experiment Setting

We use the SBF2 data set proposed by Zohali et al. [31] 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 [21] in didp-rs v0.7.3.[2] For the CP models, we use CP Optimizer 22.1.1 [17]. For the MIP models, we use Gurobi 11.0.1 [12]. 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.

5.4 Experiment Results

The results on SUALBP-1 are shown in Fig. 2.[3] Better performance is indicated by curves closer to the top left corner of the graph. First note that all of our proposed techniques outperform the current state of the art. CP-LBBD$_a$ achieves the best performance at the time limit with 69% of instances proved to optimality. CP-LBBD$_c$ performs best before 1500 seconds. In particular, CP-LBBD$_c$ achieves 63% in 300 seconds while CP-LBBD$_a$ is two times 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 significantly, performs better than DIDP-LBBD, MIP-LBBD, and MIP approaches. These results imply that direct manipulation of core decision variables x_i

2 https://didp.ai/
3 Disaggregated results for datasets A, B, C, and D are presented in Fig. 7-10 in Appendix B.

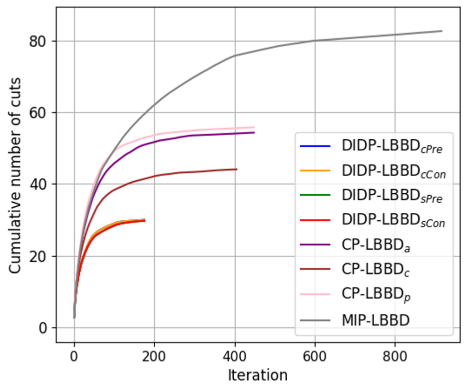


Figure 3 Mean cumulative number of cuts added over iterations.

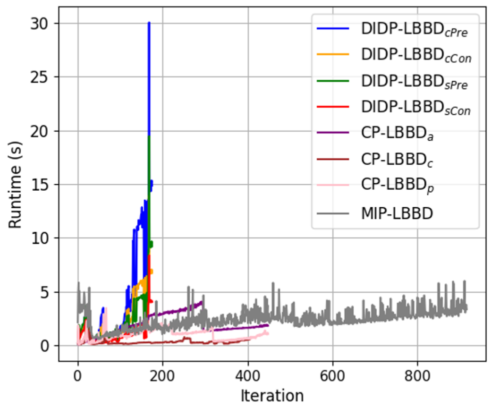


Figure 4 Mean MP runtime over iterations.

in the CP model is advantageous compared to global constraints, especially when using a global constraint requires extra variables such as w_k in the `Pack` constraint.

The DIDP-LBBD models find and prove optimal solutions for more instances in a shorter computation time than MIP-LBBD and MIP. In 60 seconds, all four DIDP-LBBD models find and prove optimality on 50% of the instances. MIP cannot achieve the same performance in 1100 seconds. At 1800 seconds, DIDP-LBBD has found and proved optimality for around 60% of the problem instances compared to 57% and 54% for MIP-LBBD and MIP, respectively.

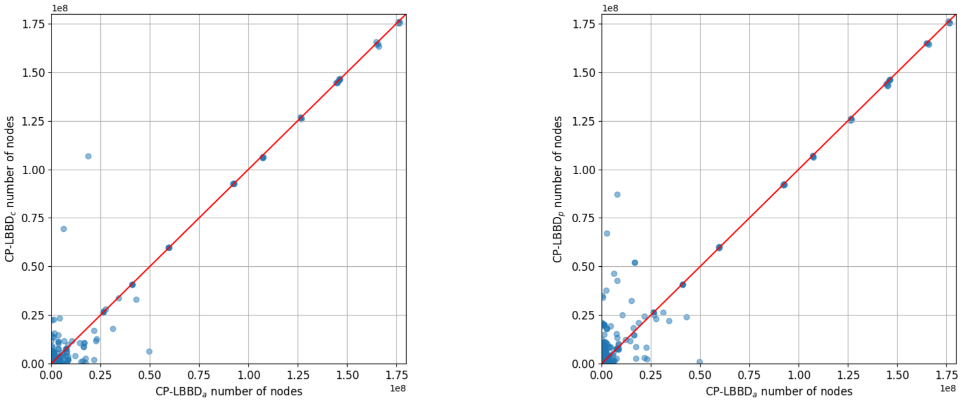
Focusing on the LBBD models, the relative rankings are: CP-LBBD, DIDP-LBBD, and MIP-LBBD, which demonstrates the promise of CP-LBBD and DIDP-LBBD. Though the three CP-LBBD variants differ substantially in Fig. 2, there is no significant performance difference among the four DIDP-LBBD variants. Note that the subproblem solve time is very short, e.g., 0.001s.

5.5 Algorithm Analysis

For the SBF2 data set, 394 of the 788 instances are proved optimal by each of the eight LBBD models. The mean cumulative numbers of cuts added for the 394 instances are shown in Fig. 3.[4] We can see that DIDP-LBBD models have significantly fewer iterations and cuts than CP-LBBD and MIP-LBBD. We believe that this difference is due to the existence of multiple optimal solutions of the master problem: different models find different optimal solutions and different Benders cuts, leading to different numbers of MP runs. While CP and DIDP models require many fewer iterations on average, we found no evidence that this is a systematic difference but rather the arbitrary impact of which optimal solutions are found.

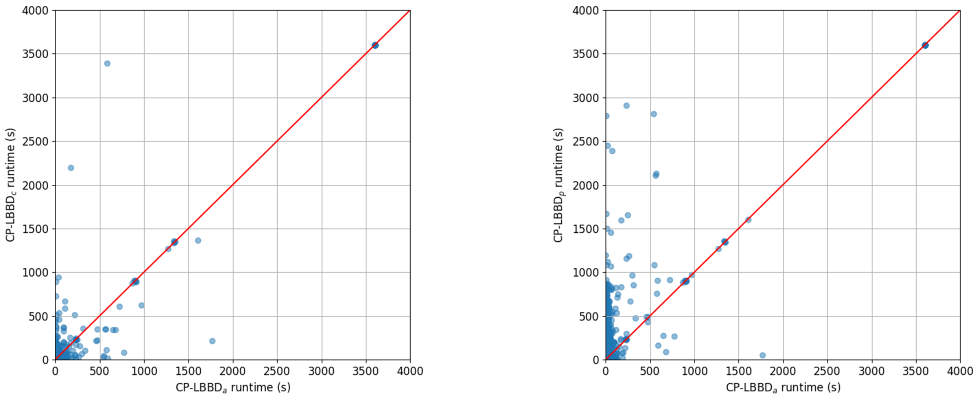
The mean MP runtimes of the 394 instances over iterations for all the eight LBBD models are shown in Fig. 4. CP-LBBD and MIP-LBBD have relatively consistent MP runtime across different iterations. For DIDP-LBBD models, although starting from small magnitude, the MP runtimes increase drastically as the iterations increase. As discussed in Section 4.2.3, with more state variables added to the DIDP model of the master problem, the state space of the model is enlarged and needs more search effort to find and prove optimality, hence the MPs become more time-consuming to solve. This performance degradation can partially

[4] The behaviors of DIDP-LBBD$_{cPre}$ and DIDP-LBBD$_{cCon}$ are exactly the same in terms of cuts added. The behaviors of DIDP-LBBD$_{sPre}$ and DIDP-LBBD$_{sCon}$ are the same, too. Thus, their plots overlap.



(a) CP-LBBD$_c$ and CP-LBBD$_a$. **(b)** CP-LBBD$_p$ and CP-LBBD$_a$.

Figure 5 Number of nodes of the MPs in CP-LBBD models for the SBF2 dataset.



(a) CP-LBBD$_c$ and CP-LBBD$_a$. **(b)** CP-LBBD$_p$ and CP-LBBD$_a$.

Figure 6 Runtime of the MPs in CP-LBBD models for the SBF2 dataset.

explain the worse results of DIDP-LBBD compared to 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 the corresponding cut forms, with a time limit of 3600 seconds. Thus for each MP iteration, the three models solve identical problems except for the differences in the form of the cuts.

Fig. 5 and 6 show scatter plots for the number of nodes and the runtimes. 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 2. 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, 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.

A different perspective on the results in Fig. 5 and 6, is shown by the runtime vs. number of nodes of the MPs in three CP-LBBD models in Fig. 11 in Appendix C. Since the three models solve identical problems except for cut forms, the results reflect the runtime each of the three CP-LBBD models needs for exploring the same number of nodes and also coincide with the performance rankings of CP-LBBD models from a regression perspective.

6 Discussion

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 [24]. 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 CP-LBBD$_c$ manipulate the main decision variables more directly while not inducing much larger constraint models.

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 encoding 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 [16, 3].

In a parallel work, a monolithic DIDP model for SUALBP-1 performs better than all the LBBD models presented here [30]. This is a surprising result as the state of the art for similar problems with sequence-dependent setup times is typically based on decomposition [31, 28]. Further research is required to understand why DIDP models for SUALBP-1 do not follow this pattern. We speculate that the relaxation of the setup time in the MP hurts performance compared to the monolithic DIDP model because setup time can be directly accounted for in the transitions.

7 Conclusions

In this paper, we proposed novel logic-based Benders decomposition (LBBD) models with master problems modeled and solved with constraint programming (CP) and domain-independent dynamic programming (DIDP), using simple assembly line balancing problem with sequence-dependent setup times type-1 (SUALBP-1) as a testbed. We developed three CP-based master problem formulations with Benders feasibility cuts formulated as key variable manipulation, constraint-based expressions, and global constraints. In the state transition system of DIDP, we proposed four encoding methods for Benders feasibility cuts by exploiting the integer or set variables and preconditions or state constraints. 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. This work demonstrates the promise of decomposition-based approaches employing CP and DIDP approaches.

References

1. Sener Akpinar, Atabak Elmi, and Tolga Bektaş. Combinatorial benders cuts for assembly line balancing problems with setups. *European Journal of Operational Research*, 259(2):527–537, 2017.
2. Carlos Andres, Cristobal Miralles, and Rafael Pastor. Balancing and scheduling tasks in assembly lines with sequence-dependent setup times. *European Journal of Operational Research*, 187(3):1212–1223, 2008.
3. Jorge A Baier, Fahiem Bacchus, and Sheila A McIlraith. A heuristic search approach to planning with temporally extended preferences. *Artificial Intelligence*, 173(5-6):593–618, 2009.
4. Ilker Baybars. A survey of exact algorithms for the simple assembly line balancing problem. *Management science*, 32(8):909–932, 1986.
5. Christian Becker and Armin Scholl. A survey on problems and methods in generalized assembly line balancing. *European journal of operational research*, 168(3):694–715, 2006.
6. Yossi Bukchin and Tal Raviv. Constraint programming for solving various assembly line balancing problems. *Omega*, 78:57–68, 2018.
7. Yingyi Chu and Quanshi Xia. Generating benders cuts for a general class of integer programming problems. In Jean-Charles Régin and Michel Rueher, editors, *Proceedings of the First International Conference on the Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems (CPAIOR 2004)*, volume 3011, pages 127–136. Springer, Berlin Heidelberg, 2004.
8. Maryam Daryalal, Hamed Pouya, and Marc Antoine DeSantis. Network migration problem: A hybrid logic-based benders decomposition approach. *INFORMS Journal on Computing*, 2023.
9. Rasul Esmaeilbeigi, Bahman Naderi, and Parisa Charkhgard. New formulations for the setup assembly line balancing and scheduling problem. *OR spectrum*, 38:493–518, 2016.
10. Michael Forbes, Mitchell Harris, Marijn Jansen, Femke van der Schoot, and Thomas Taimre. Combining optimisation and simulation using logic-based benders decomposition. *arXiv preprint arXiv:2107.08390*, 2021.
11. Cheng Guo, Merve Bodur, Dionne M Aleman, and David R Urbach. 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.
12. LLC Gurobi Optimization. Gurobi optimizer reference manual, 2021. Accessed on 2024-04-10. URL: `http://www.gurobi.com`.
13. John Hooker. *Logic-Based Methods for Optimization: Combining Optimization and Constraint Satisfaction*. John Wiley & Sons, Inc., New York, 2000.
14. John N Hooker. Planning and scheduling by logic-based benders decomposition. *Operations research*, 55(3):588–602, 2007.
15. John N Hooker and Greger Ottosson. Logic-based benders decomposition. *Mathematical Programming*, 96(1):33–60, 2003.
16. Chih-Wei Hsu, Benjamin W Wah, Ruoyun Huang, and Yixin Chen. Constraint partitioning for solving planning problems with trajectory constraints and goal preferences. In *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI 2007)*, pages 1924–1929, 2007.
17. IBM. IBM ILOG CPLEX Optimizer. Accessed on 2024-04-20. URL: `https://www.ibm.com/products/ilog-cplex-optimization-studio/cplex-cp-optimizer`.
18. Naveen Kumar and Dalgobind Mahto. Assembly line balancing: a review of developments and trends in approach to industrial application. *Global Journal of Researches in Engineering Industrial Engineering*, 13(2):29–50, 2013.
19. Ryo Kuroiwa and J. C. Beck. Domain-independent dynamic programming. *arXiv preprint arXiv:2401.13883*, 2024.
20. Ryo Kuroiwa and J Christopher Beck. Domain-independent dynamic programming: Generic state space search for combinatorial optimization. In *the 33rd International Conference on Automated Planning and Scheduling (ICAPS), 236–244.*, 2023.

21 Ryo Kuroiwa and J Christopher Beck. Solving domain-independent dynamic programming problems with anytime heuristic search. In *the 33rd International Conference on Automated Planning and Scheduling (ICAPS), 245–253.*, 2023.

22 Florin Leutwiler and Francesco Corman. A logic-based benders decomposition for microscopic railway timetable planning. *European Journal of Operational Research*, 303(2):525–540, 2022.

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

24 Francesca Rossi, Peter Van Beek, and Toby Walsh. Constraint programming. *Foundations of Artificial Intelligence*, 3:181–211, 2008.

25 Armin Scholl, Nils Boysen, and Malte Fliedner. 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.

26 Armin Scholl and Robert Klein. Salome: A bidirectional branch-and-bound procedure for assembly line balancing. *INFORMS journal on Computing*, 9(4):319–334, 1997.

27 Paul Shaw. A constraint for bin packing. In *International conference on principles and practice of constraint programming*, pages 648–662. Springer, 2004.

28 Tony T Tran, Arthur Araujo, and J Christopher Beck. Decomposition methods for the parallel machine scheduling problem with setups. *INFORMS Journal on Computing*, 28(1):83–95, 2016.

29 Tony T Tran and J Christopher Beck. Logic-based benders decomposition for alternative resource scheduling with sequence dependent setups. In *ECAI 2012*, pages 774–779. IOS Press, 2012.

30 Jiachen Zhang and J. C. Beck. Domain-independent dynamic programming and constraint programming approaches for assembly line balancing problems with setups. *arXiv preprint arXiv:2403.06780*, 2024.

31 Hassan Zohali, Bahman Naderi, and Vahid Roshanaei. Solving the type-2 assembly line balancing with setups using logic-based benders decomposition. *INFORMS Journal on Computing*, 34(1):315–332, 2022.

A Monolithic MIP Model of SUALBP-1

Table 3 Additional parameters for SUALBP-1 [9].

Notation	Definition
$\mathcal{E}$	set of all precedence relations
E_i	earliest station for task $i \in V$, e.g., $E_i = \lceil \frac{t_i + \sum_{j \in P_i^*} t_j}{\bar{c}} \rceil$
L_i	latest station for task $i \in V$, e.g., $L_i = \overline{m} + 1 - \lceil \frac{t_i + \sum_{j \in F_i^*} t_j}{\bar{c}} \rceil$
$KD(KP)$	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., $FT_k = \{i \in V \mid k \in FS_i\}$
A_i	set of tasks that cannot be assigned to the station to which task i is assigned, e.g., $A_i = \{j \in V \mid FS_j \cap FS_i = \emptyset\}$
$F_i^F(P_i^F)$	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 $P_i^F = \{j \in V \mid i \in F_j^F\}$
$F_i^B(P_i^B)$	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 $P_i^B = \{j \in V \mid i \in F_j^B\}$

To present the monolithic MIP model of SUALBP-1, additional parameters are required, as shown in Table 3. Since the SSBF model can be adapted to both SUALBP-1 and SUALBP-2, we name it SSBF-1 [9]. 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 task $i \in V$ 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 immediately 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. [9] is as follows.

$$\min \sum_{k \in KP} u_k + \underline{m} \tag{22a}$$

$$\text{s.t.} \sum_{k \in FS_i} x_{ik} = 1, \quad \forall i \in V, \tag{22b}$$

$$\sum_{k \in FS_i} k \cdot x_{ik} = z_i, \quad \forall i \in V, \tag{22c}$$

$$\sum_{i \in FT_k \cap F_i^F} g_{ijk} + \sum_{i \in FT_k \cap F_i^B} h_{ijk} = x_{ik}, \quad \forall i \in V, \forall k \in FS_i, \tag{22d}$$

$$\sum_{i \in FT_k \cap P_j^F} g_{ijk} + \sum_{i \in FT_k \cap P_j^B} h_{ijk} = x_{jk}, \quad \forall j \in V, \forall k \in FS_j, \tag{22e}$$

$$\sum_{i \in FT_k} \sum_{j \in (FT_k \cap F_i^B)} h_{ijk} = 1, \quad \forall k \in KD, \tag{22f}$$

$$\sum_{i \in FT_k} \sum_{j \in (FT_k \cap F_i^B)} h_{ijk} = u_k, \quad \forall k \in KP, \tag{22g}$$

$$r_i + 1 + (n - |F_i^*| - |P_j^*|) \cdot (\sum_{k \in (FS_i \cap FS_j)} g_{ijk} - 1) \le r_j, \quad \forall i \in V, \forall j \in F_i^F, \tag{22h}$$

$$r_i + 1 \le r_j, \quad \forall (i,j) \in \mathcal{E}, \tag{22i}$$

$$z_i \le z_j, \quad \forall (i,j) \in \mathcal{E}, \tag{22j}$$

$$\sum_{i \in FT_k} t_i x_{ik} + \sum_{i \in FT_k} \sum_{j \in (FT_k \cap F_i^F)} \tau_{ij} g_{ijk} + \sum_{i \in FT_k \cap P_i^B} \mu_{ij} h_{ijk} \le c, \quad \forall k \in KD, \tag{22k}$$

$$\sum_{i \in FT_k} t_i x_{ik} + \sum_{i \in FT_k} \sum_{j \in (FT_k \cap F_i^F)} \tau_{ij} g_{ijk} + \sum_{i \in FT_k \cap P_i^B} \mu_{ij} h_{ijk} \le c \cdot u_k, \quad \forall k \in KP, \tag{22l}$$

$$\sum_{i \in FT_k \setminus \{j\}} x_{ik} \le (n - \underline{m} + 1) \cdot (1 - h_{jjk}), \quad \forall k \in K, \forall j \in FT_k, \tag{22m}$$

$$u_{k+1} \le u_k, \quad \forall k \in KP \setminus \{\overline{m}\}. \tag{22n}$$

$$g_{ijk} \in \{0,1\}, \quad \forall k \in K, \forall i \in FT_k, \forall j \in (FT_k \cap F_i^F), \tag{22o}$$

$$h_{ijk} \in \{0,1\}, \quad \forall k \in K, \forall i \in FT_k, \forall j \in (FT_k \cap F_i^B), \tag{22p}$$

$$|P_i^*| + 1 \le r_i \le n - |F_i^*|, \quad \forall i \in V, \tag{22q}$$

$$x_{ik} \in \{0,1\}, \quad \forall i \in V, \forall k \in FS_i, \tag{22r}$$

$$r_i, z_i \in \mathbb{Z}^+, \quad \forall i \in V, \tag{22s}$$

The objective (22a) minimizes the number of stations. Constraint (22b) ensures that a task is assigned to a station. Constraint (22c) links x_{ik} and z_i. Constraints (22d) and (22e) 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 (22f) and (22g), in each cycle exactly one of the relations is a backward setup. Constraints (22h) and (22i) establish the precedence relations among the tasks within each station. Note that the constraint (22h) is inactive if tasks i and j are assigned to different stations. We add the constraint (22j) to make sure that the precedence relations among the tasks of different stations are satisfied. Knapsack constraints (22k) and (22l) ensure that no station time exceeds the cycle time. Constraint (22m) guarantees that only task j is allocated to station k when $h_{jjk} = 1$. Constraint (22n) guarantees that stations are used in the correct order and no empty station is in the middle of used stations. Constraints (22o) to (22s) specify the domain of the variables.

Note that the decision variables r_i and z_i are set to continuous in [9]. However, doing so results in infeasible solutions being labeled as feasible for some problem instances. In addition to the MIP model, Esmaeilbeigi et al. [9] developed pre-processing techniques to reduce the number of variables and constraints. We implement all these techniques, as well.

B Approach Performances for Separate Datasets

The performance of each approach on datasets A, B, C, and D separately are presented in Fig. 7 - 10, respectively. As shown in Fig. 7, all approaches except MIP solve all problems in dataset A to proved optimality in a few seconds. For dataset B (Fig. 8), all approaches, including MIP, are competitive and behave similarly. For dataset C, MIP-LBBD has the worst performance while surprisingly it outperforms all DIDP-LBBD approaches and MIP for dataset D, as shown in Fig. 9 and 10. We can also see the performance degradation of DIDP-LBBD when solving larger problems.

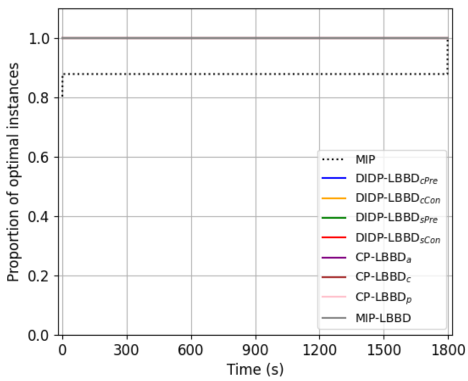


Figure 7 Ratio of instances solved and proved optimal over time for dataset A.

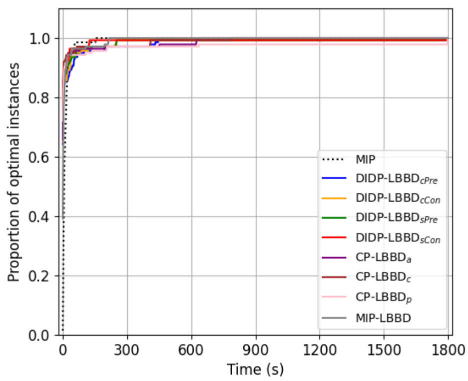


Figure 8 Ratio of instances solved and proved optimal over time for dataset B.

C Analysis of CP-LBBD

In Section 5.5, 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 the corresponding cut forms, with a time limit of 3600 seconds. The runtime over the number of nodes of the MPs in CP-LBBD

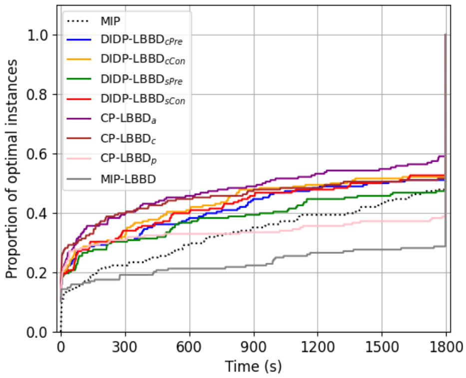


Figure 9 Ratio of instances solved and proved optimal over time for dataset C.

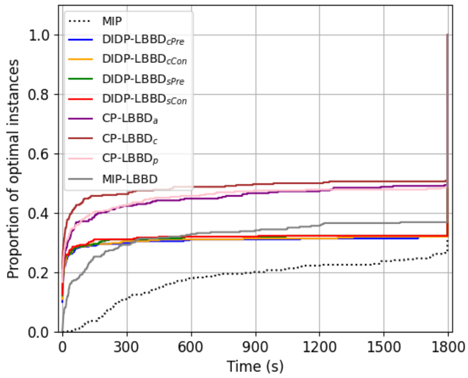


Figure 10 Ratio of instances solved and proved optimal over time for dataset D.

models for the SBF2 dataset is shown in Fig. 11. The regression lines demonstrate the performance rankings of the three CP-LBBD models in terms of the runtime required to explore the same number of nodes.

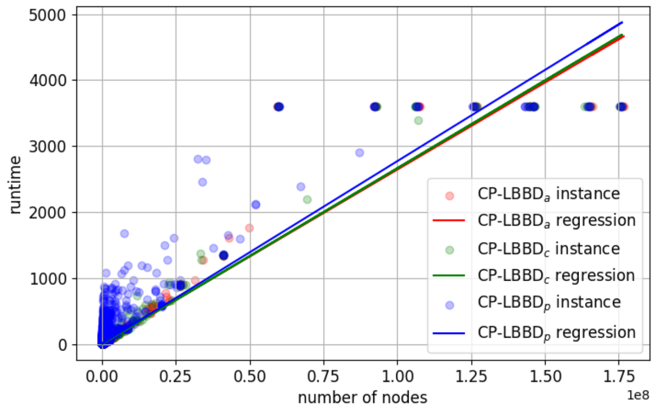


Figure 11 Runtime vs. number of nodes of the MPs in CP-LBBD models for the SBF2 dataset.