공학석사학위논문

A Novel Time-Indexed Model for the Single Machine Scheduling Problem with Sequence-Dependent Setup Times

순서 의존적 작업준비시간이 존재하는 단일 기계 일정계획 문제의 새로운 시간-인덱스 모형

2025 년 2 월

서울대학교 대학원

산업공학과

조 원 우

Abstract

A Novel Time-Indexed Model for the Single Machine Scheduling Problem with Sequence-Dependent Setup Times

Wonwoo Cho
Department of Industrial Engineering
The Graduate School
Seoul National University

In this thesis, we address time-indexed models for the single machine scheduling problem with sequence-dependent setup times. While the ideal formulation of the machine's capacity constraint is known in the absence of setup times, no such formulation has been identified when sequence-dependent setup times are present. To address the computational burden caused by the large size of time-indexed models, it is essential to formulate the capacity constraint using a small number of strong constraints.

To this end, we propose a two-phase algorithm for formulating the capacity constraint. The first phase reduces the number of constraints in the time-indexed model, while the second phase constructs constraints to tighten the linear programming (LP) relaxation bound. Computational experiments demonstrate that the resulting novel time-indexed model significantly outperforms existing models in the literature, achieving much tighter LP relaxation bounds with a considerably smaller model size and in shorter solving times.

Additionally, we introduce a restricted time-indexed model that can be applied when only a subset of the decision variables is used. This model further reduces its size by identifying constraints that can be excluded while maintaining the model's validity. Computational results confirm that the restricted model effectively reduces the number of constraints.

Keywords: Single machine scheduling problem, Sequence-dependent setup time, Time-indexed model, Time-indexed formulation

Student Number: 2023-22551

Contents

Abstract **i**

Contents **v**

List of Tables **vi**

List of Figures **vii**

Chapter 1 Introduction **1**

1.1 Background 1

1.2 Literature Review 5

1.2.1 Time-Indexed Model for the Single Machine Scheduling Problem with No Setup Times 5

1.2.2 Time-Indexed Model for the Single Machine Scheduling Problem with Sequence-Dependent Setup Times 6

1.3 Motivation and Contributions 7

1.4 Organization of the Thesis 8

Chapter 2 Generic Time-Indexed Model for Single Machine Scheduling Problem with Sequence-Dependent Setup Times **10**

2.1 Formulation of Time-Indexed Model 10

2.1.1 Conflict Graph 11
2.1.2 Clique of the Conflict Graph 12
2.2 Generic Time-Indexed Model 14
2.3 Time-Indexed Models in the Literature 17

Chapter 3 A Novel Time-Indexed Model and Its Restricted Variant 20

3.1 A Novel Time-Indexed Model 20
3.1.1 Clique Template Cover Generation Algorithm - Phase 1 23
3.1.2 Clique Template Cover Generation Algorithm - Phase 2 29
3.2 Restricted Time-Indexed Model 33

Chapter 4 Computational Experiments 38

4.1 Instance Generation 38
4.1.1 Processing and Setup Times 38
4.1.2 Objective Function 39
4.1.3 Planning Horizon Length 40
4.1.4 Instance Configuration 40
4.2 Experiments on the Clique Template Cover Generation Algorithm 40
4.2.1 Clique Template Cover Generation Algorithm - Phase 1 40
4.2.2 Clique Template Cover Generation Algorithm - Phase 2 42
4.3 Comparison with Existing Time-Indexed Models 44
4.4 Experiments on the Restricted Time-Indexed Models 47

Chapter 5 Conclusion 49

Appendix 51

A.1 Processing and setup time generation 51

A.2 Due date generation 53

A.3 Planning horizon length generation 54

Bibliography **55**

국문초록 **59**

감사의 글 **60**

List of Tables

Table 2.1 Notation for the conflict graph 11

Table 2.2 Notation for clique representation 14

Table 3.1 Notation for bounds on $U_i - L_i$ 25

Table 4.1 Computation time and size of the resulting clique template cover from Phase 1 of CTCGA 41

Table 4.2 Computation time of Phase 2 of CTCGA 43

Table 4.3 Comparison between time-indexed models 45

Table 4.4 The number of constraints in restricted time-indexed models 47

List of Figures

Figure 1.1 Feasible schedule of a 3-job instance 2

Figure 2.1 Conflict graph for a two-job instance with $S_{ij} = 2$ and $S_{ji} = 3$ . . . 11

Figure 3.1 4-job example of CTCGA 22

Figure 3.2 2-job example of clique association in the restricted model 35

Chapter 1

Introduction

1.1 Background

Scheduling is the decision-making of allocating resources to jobs over a specified planning horizon. It arises in various domains, including manufacturing, production, information processing, transportation, and service systems (Pinedo, 2022). Effective scheduling can have a significant impact; for instance, after implementing a new scheduling method, Dell Inc. increased its production volume by 35% and saved over $1 million annually (Loveland et al., 2007). For comprehensive coverage of scheduling theories, algorithms, models, performance analyses and practical systems, see Leung (2004) and Pinedo (2022).

Setup time refers to the time required to prepare a resource so that it can start the next job (Allahverdi, 2015). A setup time is called *sequence-dependent* if its duration depends on both the preceding and succeeding jobs. Numerous real-world scheduling problems involve sequence-dependent setup times, such as those in production (Kress et al., 2019) and assembly (Kuo et al., 2020) systems. Consequently, sequence-dependent setup times have been explicitly studied since the mid-1960s. Studies on scheduling problems with sequence-dependent setup times conducted from the mid-1960s to the end of 2014 are summarized in comprehensive survey papers (Allahverdi, 2015; Allahverdi et al., 1999, 2008).

In this thesis, we address the single machine scheduling problem with sequence-dependent setup times. The problem is formally described as follows:

A set of jobs $J = \{1, 2, \ldots, n\}$ and a single machine are given. Once a job $i \in J$ begins processing, it occupies the machine for its entire processing time p_i. After a job i is completed, the machine requires a sequence-dependent setup time s_{ij} before starting the next job j.

The decision is to make a schedule for the single machine that meets the following two requirements:

1. **Assignment Constraint**: Every job in J must be processed by the machine exactly once.

2. **Capacity Constraint**: The machine can perform only one task—either processing or setup—at any given time.

The objective is to minimize the cost associated with the schedule. An example of a schedule that respects both the assignment and capacity constraints for the problem with $J = \{i, j, k\}$ is illustrated in Figure 1.1.

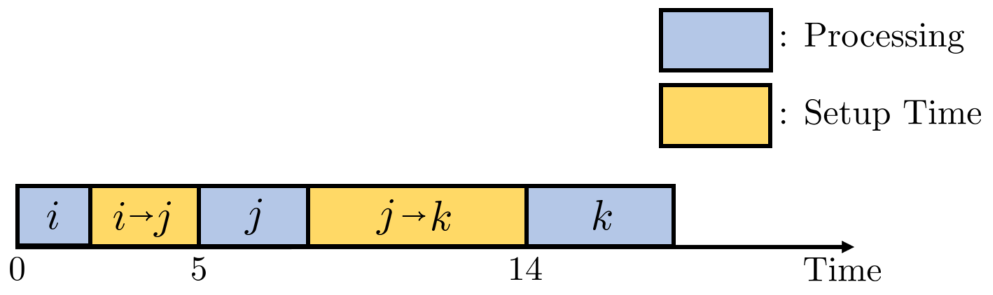

Figure 1.1: Feasible schedule of a 3-job instance

When sequence-dependent setup times are explicitly considered, even single machine scheduling problems are NP-hard for most objective functions. For example, minimizing the sum of completion times is NP-hard (Rinnooy Kan, 1976), even though it can be solved in polynomial time when sequence-dependent setup times are not considered (Smith et al., 1956). Consequently, heuristic approaches have been predominantly proposed for scheduling problems with sequence-dependent setup times (Allahverdi, 2015). However, heuristic designs often heavily exploit the specific characteristics of the problem, which can limit their applicability or effectiveness if the objective function changes or additional constraints are introduced. For example, the highly effective heuristic proposed introduced by Lin and Kernighan (1973) for the symmetric traveling salesman problem—which is equivalent to the single machine scheduling problem with symmetric setup times and a makespan objective—cannot be applied when the objective function changes or when setup times become asymmetric. Furthermore, schedules obtained using heuristics offer no guarantee regarding their solution quality.

On the other hand, mixed-integer linear programming (MILP) models offer the following strengths:

1. The optimal solutions provided by these models guarantee global optimality.

2. A single MILP model can be used to formulate various scheduling problems. Different objectives and additional specifications may be incorporated into the model by adjusting cost coefficients and adding constraints. Furthermore, parallel machine scheduling problems may be formulated based on MILP models for the single machine scheduling problems.

Given these advantages, several MILP models for the single machine scheduling problem have been proposed (Nogueira et al., 2019; Güngör, 2025). Among these MILP models, we focus on so-called *time-indexed models* in this thesis. A time-indexed model represents a schedule with binary decision variables of the form x_{it}, where $i \in J$ and t is an integer. x_{it} is equal to 1 if the job i begins at the time t, 0 otherwise. For example, the schedule shown in Figure 1.1, where jobs i, j, and k start at times 0, 5, and 14 respectively, is represented by setting the decision variables x_{i0}, x_{j5}, and x_{k14} to 1, with all other variables set to 0.

The time-indexed models offer the following strengths:

1. They can handle any objective function that involves the sum of job-wise costs depending on the completion times of jobs, by simply adjusting the objective function coefficients.

2. They can efficiently incorporate constraints such as release dates, deadlines, and fixed machine downtimes by simply excluding the corresponding decision variables from the model.

In this thesis, we make the following assumptions to ensure the validity of a time-indexed model as an optimization framework:

1. **Triangle Inequality Assumption:** A job cannot start earlier by inserting an intermediate job between itself and the previous job. Formally, this implies $s_{ij} + p_j + s_{jk} \geq s_{ik}$ for all mutually distinct job triples (i, j, k).

2. **Integer-Valued Parameters:** All parameter values are integers. This guarantees that every start time in an optimal schedule is also integer-valued, ensuring that a time-indexed model can describe an optimal solution.

3. **Job-Wise Cost Functions:** The cost function of an instance can be expressed as the sum of job-wise costs, where job i incurs a cost c_{it} if it starts at time t. Common

objective functions in the literature, such as the sum of (weighted) completion times, tardiness and earliness-tardiness, fall into this category.

1.2 Literature Review

We review previous studies dealing with the time-indexed models for the single machine scheduling problem in this section.

1.2.1 Time-Indexed Model for the Single Machine Scheduling Problem with No Setup Times

A time-indexed model for the single machine scheduling problem was first proposed by Dyer and Wolsey (1990), in a setting where no setup times exist. After the proposal, several polyhedral studies about the model were conducted. Sousa and Wolsey (1992) derived valid inequalities of the model, by relaxing the scheduling problem as a binary knapsack problem with generalized upper bounds. Crama and Spieksma (1996) studied the model where processing times of jobs are all equal. Van den Akker et al. (1999) provided all facet-defining inequalities with integral coefficients and right-hand sides of 1 or 2 for the convex hull of the set of feasible schedules. The study showed that the inequalities can further tighten the linear programming (LP) relaxation bound of the model.

Computational experiments on the model reported the following findings (Sousa and Wolsey, 1992; Van den Akker et al., 1999):

1. The model yields very strong LP relaxation bounds across various objective functions.

2. Solving the model is computationally challenging due to its large size.

To overcome the computational burden, several solution approaches based on the time-indexed model were developed. Van den Akker et al. (2000) applied Dantzig-Wolfe decomposition to the model, demonstrating that this approach can compute the LP relaxation bound significantly faster than directly solving the LP relaxation. Bigras et al. (2008) proposed an approach to obtain a dual bound more quickly by dividing the planning horizon into subperiods and allowing job processing to be interrupted. This method significantly enhances computational efficiency while incurring only a minor loss in bound quality compared to the LP relaxation bound. Avella et al. (2005) applied Lagrangian relaxation to enable fast dual bound computation.

1.2.2 Time-Indexed Model for the Single Machine Scheduling Problem with Sequence-Dependent Setup Times

When sequence-dependent setup times are absent, the capacity constraint's *ideal* formulation—one that precisely describes the convex hull of all feasible solutions—is known. However, with the introduction of sequence-dependent setup times, the ideal formulation remains undiscovered.

This challenge has resulted in comparatively fewer studies on time-indexed models for single machine scheduling problems with sequence-dependent setup times. For example, Sun et al. (1999) and de Paula et al. (2010) employed a time-indexed model that formulates the capacity constraint using $O(n^2)$ constraints per unit time. The size is notably large compared to the case without sequence-dependent setup times, as the ideal formulation requires only a single constraint per unit time.

Worsening the situation, computational results from Nogueira et al. (2019) demonstrate that the LP relaxation bound of this $O(n^2)$-constraint model is much weaker than the LP relaxation bound of the time-indexed model without setup times. This weaker bound negates one of the primary advantages of using a time-indexed model. In response, the study proposed new valid inequalities and provided computational evidence that incorporating these inequalities effectively tightens the LP relaxation bound.

Avella et al. (2017) proposed another time-indexed model that avoided using the $O(n^2)$ constraints per unit time. Instead, their model incorporated constraints that generalized the valid inequalities proposed by Nogueira et al. (2019). The constraints to include in the model was selected greedily by solving an integer linear program. Their computational results demonstrated that their model outperformed the one proposed by Nogueira et al. (2019) on the runway scheduling problem instances they tested.

1.3 Motivation and Contributions

When sequence-dependent setup times are absent, the time-indexed model for the single machine scheduling problem is known to yield strong LP relaxation bounds. However, it remains computationally challenging to solve due to its large size. The biggest weakness becomes even worse when sequence-dependent setup times are introduced, as they further increase the model size.

To mitigate the computational burden, an effective time-indexed model should:

1. Be formulated with a small number of constraints to reduce the time required to solve the LP relaxation.

2. Yield strong LP relaxation bounds to minimize the depth of the branch-and-bound tree.

In this thesis, we propose a novel time-indexed model for the single machine scheduling problem with sequence-dependent setup times that achieves these two goals. The constraints in the model are constructed using a two-phase algorithm: the first phase addresses the first objective, while the second phase focuses on the second objective.

Through computational experiments, we demonstrate that the construction algorithm is computationally efficient for moderately sized instances. Furthermore, we show that the proposed time-indexed model improves upon previously proposed formulations in the literature with respect to the number of constraints, the computation time required to solve the LP relaxation, and the resulting LP relaxation bound.

A practical approach to reducing the model size is to exclude some decision variables from the original time-indexed model, thereby restricting the start times of jobs. To further reduce the model size, we propose a restricted time-indexed model that excludes specific constraints from the original model while preserving its validity. This approach is tailored to the time-indexed model introduced in this thesis. Computational experiments demonstrate that the proposed method effectively reduces the number of constraints in the restricted model.

1.4 Organization of the Thesis

The remainder of this thesis is organized as follows. Chapter 2 addresses the formulation of time-indexed models and derives a generic framework. This generic model includes the time-indexed models proposed in the literature as well as the novel model introduced in this thesis as specific instances. Chapter 3 presents the algorithm for constructing the novel time-indexed model and details the development of a restricted time-indexed model based on the novel model. Chapter 4 presents the computational results, including the evaluation of the construction algorithm, the performance comparison of the proposed time-indexed model with existing models in the literature, and the assessment of the restricted model. Finally, Chapter 5 provides concluding remarks and outlines directions for future research.

Chapter 2

Generic Time-Indexed Model for Single Machine Scheduling Problem with Sequence-Dependent Setup Times

2.1 Formulation of Time-Indexed Model

In a valid time-indexed model for the single machine scheduling problem with sequence-dependent setup time, the assignment constraint and the capacity constraint must be ensured by including constraints in the model. Among these two requirements, the assignment constraint is straightforwardly formulated with n constraints:

$$\sum_{t=0}^{H-p_i} x_{it} = 1, \quad \forall i \in J. \tag{2.1}$$

Each constraint in (2.1) ensures that a job in J is scheduled exactly once. Here, H represents the length of the planning horizon.

Enforcing the capacity constraint implies that for every distinct job pair i, j, job j cannot start until $p_i + s_{ij}$ time units after the start time of job i. For brevity, we define

$$S_{ij} := p_i + s_{ij}, \quad \forall i, j \in J,\, i \neq j.$$

2.1.1 Conflict Graph

To explain formulation of the capacity constraint in a time-indexed model, we introduce an undirected graph $(\mathcal{V}, \mathcal{E})$, referred to as the *conflict graph*.

Each node in $\mathcal{V}$ corresponds to a decision variable in the time-indexed model. An edge in $\mathcal{E}$ is drawn between two nodes if assigning the value 1 to both corresponding decision variables violates either the assignment constraint or the capacity constraint. We denote by $\mathcal{E}_A$ and $\mathcal{E}_C$ the subsets of $\mathcal{E}$ corresponding to violations of the assignment constraint and the capacity constraint, respectively. The symbols and their formal definitions for the conflict graph $(\mathcal{V}, \mathcal{E})$ are provided in Table 2.1.

Table 2.1: Notation for the conflict graph

Symbol	**Definition**
$\mathcal{V}$	$\{(i,t) \in J \times \mathbb{Z} \mid 0 \le t \le H - p_i\}$
$\mathcal{E}_A$	$\{((i,r),(i,s)) \mid (i,r),(i,s) \in \mathcal{V},\ r \neq s\}$
$\mathcal{E}_C$	$\{((i,r),(j,s)) \mid (i,r),(j,s) \in \mathcal{V},\ i \neq j,\ -S_{ji} < s - r < S_{ij}\}$
$\mathcal{E}$	$\mathcal{E}_A \cup \mathcal{E}_C$

Figure 2.1 illustrates the conflict graph for a 2-job instance where $S_{ij} = 2$ and $S_{ji} = 3$. Subfigure (a) highlights all edges in $\mathcal{E}$ connected to a node $(i, 3)$, with dashed arcs representing edges in $\mathcal{E}_A$ and solid arcs representing edges in $\mathcal{E}_C$. Subfigure (b) depicts the conflict graph for this instance.

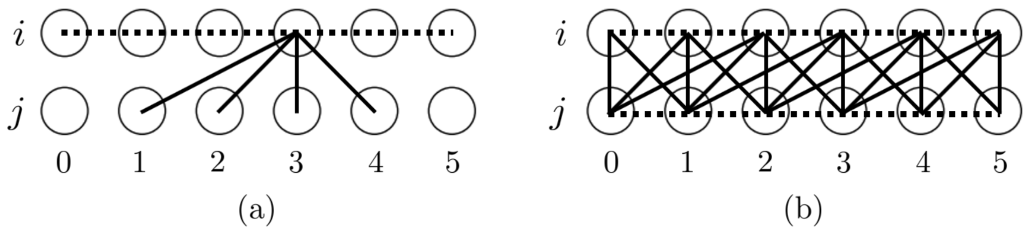

Figure 2.1: Conflict graph for a two-job instance with $S_{ij} = 2$ and $S_{ji} = 3$

2.1.2 Clique of the Conflict Graph

A key insight in formulating the capacity constraint is that a schedule derived from a 0-1 solution of a time-indexed model violates this constraint precisely when there exists an edge in $\mathcal{E}_C$ for which both corresponding decision variables are assigned the value 1.

In a time-indexed model, a constraint is said to *cover* an edge in $\mathcal{E}_C$ if it ensures that the decision variables corresponding to the endpoints of that edge cannot both be equal to 1. Therefore, enforcing the capacity constraint is equivalent to covering all edges in $\mathcal{E}_C$ with such constraints. This can be achieved by including the constraints

$$x_{ir} + x_{js} \leq 1, \quad \forall((i,r),(j,s)) \in \mathcal{E}_C,$$

where each constraint covers a single edge in $\mathcal{E}_C$. However, a more compact and stronger formulation can be obtained by employing *clique inequalities.*

A *clique* is a set of nodes in which every pair of nodes is connected by an edge. For instance, in Figure 2.1 (b), the set $\{(i,0),(i,1),(j,0),(j,1)\}$ forms a clique.

Let c be a clique of the conflict graph $(\mathcal{V},\mathcal{E})$. By definition, no two decision variables corresponding to nodes in c can both be equal to 1 in a feasible solution. Therefore, at most one decision variable x_{it} corresponding to an element $(i,t) \in c$ can be equal to 1. This leads to the following valid inequality for time-indexed models:

$$\sum_{(i,t)\in c} x_{it} \leq 1, \tag{2.2}$$

which is known as the *clique inequality* associated with c. Note that the clique inequality (2.2) covers all edges in $\mathcal{E}_C$ where both endpoints belong to c.

To derive clique inequalities for inclusion in a time-indexed model, we analyze the structure of cliques in the conflict graph $(\mathcal{V}, \mathcal{E})$ using the following two propositions.

Proposition 2.1. *Let $c \subseteq \mathcal{V}$, and define $L_i := \min\{t \mid (i,t) \in c\}$ and $U_i := \max\{t \mid (i,t) \in c\}$ for each $i \in J_c := \{i \in J \mid \exists\, t \text{ such that } (i,t) \in c\}$. Then, c is a clique in the conflict graph $(\mathcal{V}, \mathcal{E})$ if and only if $U_j - L_i < S_{ij}$ for all distinct $i, j \in J_c$.*

Proof. $(\Rightarrow)$ Consider any two distinct jobs $i, j \in J_c$. By definition, $(i, L_i), (j, U_j) \in c$. Since c is a clique, the edge $((i, L_i), (j, U_j))$ must belong to $\mathcal{E}_C$, which implies that $U_j - L_i < S_{ij}$.
$(\Leftarrow)$ Let $(i, r), (j, s)$ be any two distinct elements in c. If $i = j$, $((i,r),(j,s)) \in \mathcal{E}_A$. If $i \neq j$, $-S_{ji} < L_j - U_i \leq s - r \leq U_j - L_i < S_{ij}$ implies $((i,r),(j,s)) \in \mathcal{E}_C$. $\square$

Proposition 2.2. *Let $c_1, c_2 \subseteq \mathcal{V}$ such that $c_1 \subseteq c_2$. Then, the clique inequality associated with c_2 dominates the clique inequality associated with c_1, provided that all decision variables are nonnegative.*

Proof. Given $\sum_{(i,t)\in c_2} x_{it} \leq 1$,
$\sum_{(i,t)\in c_1} x_{it} \leq \sum_{(i,t)\in c_1} x_{it} + \sum_{(i,t)\in c_2 \setminus c_1} x_{it} = \sum_{(i,t)\in c_2} x_{it} \leq 1$. $\square$

Let c be a clique in the conflict graph $(\mathcal{V}, \mathcal{E})$, and let J_c, L_i and U_i be defined as in Proposition 2.1. According to Proposition 2.1, the node set

$$\{(i,t) \in \mathcal{V} \mid i \in J_c,\ L_i \leq t \leq U_i\} \tag{2.3}$$

forms a clique. Among cliques sharing the same J_c, L_i, U_i values, we can focus solely on the clique (2.3), as larger cliques in terms of inclusion are preferable according to Proposition 2.2. Consequently, every clique can be fully characterized by these values. Table 2.2 presents additional notation to represent cliques based on this observation.

Table 2.2: Notation for clique representation

Symbol	**Definition**
$S(L, U)$	$\{(i, t) \in J \times \mathbb{Z} \mid L_i \leq t \leq U_i\}$
$J(L, U)$	$\{i \in J \mid L_i \leq U_i\}$
$C(L, U)$	$S(L, U) \cap \mathcal{V}$.

Throughout this thesis, L and U denote n-dimensional integer vectors, with their i-th components represented as L_i and U_i, respectively. The set $S(L, U)$ comprises all (i, t) pairs, where i is a job, and t is an integer time index satisfying $L_i \leq t \leq U_i$. $J(L, U)$ represents the set of jobs i for which $L_i \leq U_i$, ensuring the existence of at least one time index t such that $(i, t) \in S(L, U)$. The set $C(L, U)$ is the intersection of $S(L, U)$ with $\mathcal{V}$.

We say that $S(L, U)$ *defines a clique* if $U_j - L_i < S_{ij}$ for all distinct job pairs $i, j \in J(L, U)$. This is because, by Proposition 2.1, $C(L, U)$ forms a clique in the conflict graph $(\mathcal{V}, \mathcal{E})$ whenever $S(L, U)$ satisfies this condition.

2.2 Generic Time-Indexed Model

In this section, we introduce a generic time-indexed model that extends existing formulations in the literature (Dyer and Wolsey, 1990; Avella et al., 2017; Nogueira et al., 2019). The model employs a strategy of constructing clique inequalities to cover all edges in $\mathcal{E}_C$, thereby ensuring its validity. A key feature of this approach is its independence from the planning horizon length, which significantly reduces the complexity of the formulation process. This independence is achieved by leveraging the fact that every edge in the conflict graph is determined solely by job indices and the time differences between the nodes defining the edge, rather than their absolute time values. Consequently, shifting all elements of a clique by the same arbitrary amount of time results in another clique. This property is formalized in the following proposition. Throughout this thesis, we denote by $\mathbf{1}$ an n-dimensional vector with all components equal to 1.

Proposition 2.3. *If $S(L,U)$ defines a clique, then $S(L+t\mathbf{1},U+t\mathbf{1})$ also defines a clique for all $t \in \mathbb{Z}$.*

Proof. Let $S(L,U)$ define a clique and $t \in \mathbb{Z}$. By definition, $J(L,U) = J(L+t\mathbf{1},U+t\mathbf{1})$. For all distinct jobs $i,j \in J(L+t\mathbf{1},U+t\mathbf{1})$,

$$(U_j + t) - (L_i + t) = U_j - L_i < S_{ij},$$

since $S(L,U)$ defines a clique. $\square$

Based on this observation, we introduce the concept of a *clique template*, which is a collection of node sets obtained by time translations of a node set that defines a clique. The formal definition is as follows:

Definition 2.4. *Let $S(L,U)$ define a clique. Then, the clique template $T(L,U)$ is defined as:*

$$T(L,U) := \{S(L+t\mathbf{1},U+t\mathbf{1}) \mid t \in \mathbb{Z}\}.$$

Note that a clique template is invariant under time translation; that is, $T(L,U) = T(L+t\mathbf{1},U+t\mathbf{1})$ for all $t \in \mathbb{Z}$.

Assume that a set $S(L,U)$ that defines a clique contains both $(i,0)$ and (j,r). Then $S(L+t\mathbf{1},U+t\mathbf{1})$ clearly contains both (i,t) and $(j,r+t)$. This implies that every edge in $\mathcal{E}_C$ that can be expressed in the form $((i,t),(j,r+t))$ can be covered by a clique inequality induced from an element of the clique template $T(L,U)$.

Every edge in $\mathcal{E}_C$ can be represented in the form $((i,t),(j,r+t))$, where i and j are distinct jobs, and r is an integer such that $-S_{ji} < r < S_{ij}$. Therefore, if a set of clique templates includes a clique template $T(L,U)$ where $S(L,U)$ includes both $(i,0)$ and (j,r) for all such triplets (i,j,r), then every edge in $\mathcal{E}_C$ can be covered by a clique inequality induced from an element of this set. We refer to such a set as a *clique template cover*, which is formally defined as follows:

Definition 2.5. *A set of clique templates F is a clique template cover if and only if for every triplet (i,j,r), where i and j are distinct jobs and $r \in \mathbb{Z}$ satisfies $-S_{ji} < r < S_{ij}$, there exists a $T(L,U) \in F$ such that $L_j - U_i \leq r \leq U_j - L_i$.*

Based on these concepts, we present the generic time-indexed model. The generic model is instantiated with a clique template cover F. A valid time-indexed model is constructed using F by including clique inequalities induced from the clique templates in F, spanning the entire planning horizon. The model is formulated as follows:

$$
\begin{aligned}
\mathrm{TI}(F): \quad & \text{minimize} && \sum_{(i,t)\in\mathcal{V}} c_{it} x_{it} \\
& \text{subject to} && \sum_{t=0}^{H-p_i} x_{it} = 1, \quad \forall i \in J, && (2.4) \\
& && \sum_{(i,r)\in C(L+t\mathbf{1},U+t\mathbf{1})} x_{ir} \leq 1, \quad \forall (L,U,t) \in \mathcal{P}, && (2.5) \\
& && x_{it} \in \{0,1\}, \quad \forall (i,t) \in \mathcal{V}, \\
& \text{where} && \mathcal{P} := \{(L,U,t) \mid T(L,U) \in F,\ t \in \bigcup_{i\in J(L,U)} [-U_i, H-p_i-L_i] \cap \mathbb{Z}\}.
\end{aligned}
$$

In this model, the assignment constraint is ensured by (2.4), while the capacity constraint is enforced by (2.5).

We prove that TI(F) is a valid time-indexed model for the single machine scheduling problem with sequence-dependent setup times.

Proposition 2.6. *Let F be a clique template cover. Then $TI(F)$ is a valid time-indexed model.*

Proof. We only need to show that for every $((i,r),(j,s)) \in \mathcal{E}_C$, there exists a clique $C(L+t\mathbf{1}, U+t\mathbf{1})$ such that $(L,U,t) \in \mathcal{P}$ and $\{(i,r),(j,s)\} \subseteq C(L+t\mathbf{1}, U+t\mathbf{1})$. By definition of clique template cover, there exists a clique template $T(L,U) \in F$ where $L_j - U_i \leq s - r \leq U_j - L_i$. Therefore, there exist integers r^* and s^* such that $L_i \leq r^* \leq U_i$, $L_j \leq s^* \leq U_j$ and $s^* - r^* = s - r$. Since $0 \leq r \leq T - p_i$ and $L_i \leq r^* \leq U_i$, $-U_i \leq r - r^* \leq T - p_i - L_i$. Therefore, $(L, U, r - r^*) \in \mathcal{P}$ and $C(L + (r - r^*)\mathbf{1}, U + (r - r^*)\mathbf{1})$ is the desired clique. □

2.3 Time-Indexed Models in the Literature

In this section, we describe two time-indexed models proposed in the literature. Both models are instances of the generic time-indexed model introduced in Section 2.2. Therefore, it suffices to provide the clique template covers they employ.

The first model is the one introduced in Nogueira et al. (2019). The clique template cover they employ includes $n(n-1)+1$ clique templates. For every distinct job pair $i, j \in J$, clique template $T(L^{ij}, U^{ij})$ where

$$J(L^{ij}, U^{ij}) = \{i, j\}, \quad L_i^{ij} = U_i^{ij} = 0, \quad L_j^{ij} = -S_{ji} + 1, \quad U_j^{ij} = S_{ij} - 1$$

is included in the clique template cover. Since every edge in $\mathcal{E}_C$ connecting nodes corresponding to jobs i and j is covered by including clique inequalities derived from $T(L^{ij}, U^{ij})$, the set

$$\left\{T(L^{ij}, U^{ij}) \mid i, j \in J,\, i \neq j\right\} \tag{2.6}$$

forms a clique template cover.

In addition to the $n(n-1)$ clique templates, a single clique template $T(L, U)$ defined as

$$(L_i, U_i) = \left(\max_{j \in J \setminus \{i\}} \{-S_{ij} + 1\}, 0\right), \quad \forall i \in J \tag{2.7}$$

is included in the clique template cover. This clique template ensures that $J(L, U) = J$ and that all components of U are equal. The components of L are chosen to be as small as possible while satisfying the clique condition. This clique template is included in the clique template cover (2.6). We refer to the model based on this clique template cover as *Nogueira*, named after the first author of Nogueira et al. (2019).

In the second model, introduced by Avella et al. (2017), each element in the clique template cover is completely characterized by a subset of jobs $J^* \subseteq J$. For a given $J^* \subseteq J$, the L and U values in the corresponding clique template $T(L, U)$ are defined as:

$$J(L, U) = J^*, \quad (L_i, U_i) = \left(\max_{j \in J^* \setminus \{i\}} \{-S_{ij} + 1\}, 0\right), \forall i \in J^*.$$

The L and U values are constructed similarly to (2.7), with all U values set equal and L values chosen to be as small as possible while satisfying the clique condition.

The subset J^* is sequentially selected by solving an integer program, and the corresponding clique template is added to the set of clique templates. This process is repeated until the set of clique templates forms a clique template cover. Additionally, a clique-enlarging algorithm proposed by Kopf and Ruhe (1987) is applied to each element in the clique template cover. Specifically, a clique template $T(L, U)$ is replaced by $T(L^*, U^*)$, where $S(L, U) \subseteq S(L^*, U^*)$. This model is referred to as *Avella*, named after the first author of Avella et al. (2017).

Chapter 3

A Novel Time-Indexed Model and Its Restricted Variant

3.1 A Novel Time-Indexed Model

In this section, we propose a novel time-indexed model that is an instance of the generic time-indexed model. Consequently, the modeling task is reduced to constructing a clique template cover. Before explaining the construction process, we introduce the core idea behind the characteristics of the clique templates included in the clique template cover.

Definition 2.5 indicates that constructing a clique template cover requires considering all (i, j, r) triplets, where i and j are distinct jobs, and the integer r representing the time difference ranges from $-S_{ji} + 1$ to $S_{ij} - 1$. Since the range of r is pseudo-polynomial, exhaustively considering all such triplets can be computationally intensive, especially when processing and setup times are large.

To address this issue, we construct a clique template cover such that for every unordered distinct job pair (i, j), all edges in $\mathcal{E}_C$ connecting job i and j are covered by clique inequalities derived from a single clique template. We say that such a clique template *covers* the job pair (i, j). The formal definition, expressed in terms of L and U, is as follows:

Definition 3.1. *Let i and j be distinct jobs. A clique template $T(L, U)$ is said to cover* *the job pair (i, j) if (L, U) satisfies the followings:*

- $\{i, j\} \subseteq J(L, U)$,
- $U_j - L_i = S_{ij} - 1$,
- $U_i - L_j = S_{ji} - 1$.

Two primary objectives are considered in the construction process of the clique template cover:

1. The resulting time-indexed model includes a small number of constraints, to reduce the computational time required to solve the LP relaxation.
2. The resulting time-indexed model yields a strong LP relaxation bound, to minimize the depth of the branch-and-bound tree.

To achieve these objectives, we propose a two-phase algorithm for constructing a clique template cover, referred to as the *clique template cover generation algorithm* (CTCGA). Each phase of the algorithm is briefly explained below, accompanied by an illustrative example in Figure 3.1.

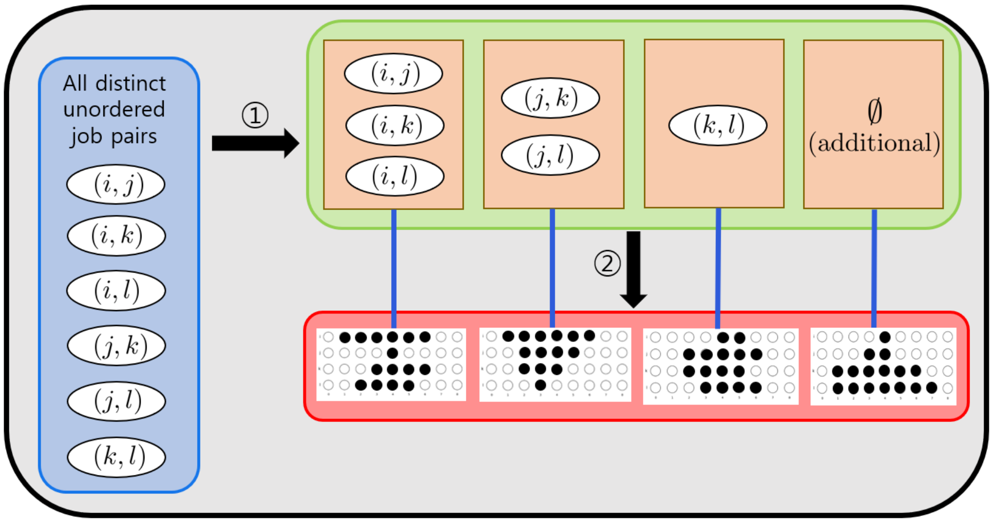

Figure 3.1: 4-job example of CTCGA

- **Phase 1: Partitioning job pairs**

  In the first phase, the set of all distinct unordered job pairs is partitioned. For instance, in the case of 4 jobs, there are $6 = \binom{4}{2}$ distinct unordered job pairs, which are divided into three partitions. Each partition corresponds to a clique template that covers all the job pairs it contains. Since reducing the number of constraints in the generic time-indexed model is equivalent to reducing the size of the clique template cover, the first phase aims to minimize the number of partitions by including as many job pairs as possible in each partition. The inclusion is performed while ensuring the existence of a clique template that covers all the job pairs in the partition.

- **Phase 2: Constructing clique templates**

  In the second phase, a clique template is constructed for each partition created in the first phase. Additionally, a single clique template, not required to cover any specific job pair, is generated to enhance the LP relaxation bound. During the construction of clique templates, the algorithm aims to maximize clique sizes while satisfying coverage conditions to tighten the LP relaxation bound.

3.1.1 Clique Template Cover Generation Algorithm - Phase 1

In the first phase of CTCGA, the algorithm determines which job pairs will be covered by the same clique template. These job pairs are selected such that a single job consistently appears in every pair. For instance, in the example depicted in Figure 3.1, the first partition $\{(i,j),(i,k),(i,l)\}$ includes a job i in every pair. As a result, deciding which job pairs are covered by the same clique template reduces to selecting a common job and a set of jobs to pair with it. For example, the set $\{(i,j),(i,k),(i,l)\}$ can be equivalently represented as $(i,\{j,k,l\})$. Using this notation, for a job i and a set of jobs $J^* \subseteq J \setminus \{i\}$, we say that a clique template *covers* (i,J^*) if it covers (i,j) for every job $j \in J^*$.

For the remainder of this section, we fix a job i and aim to determine a set of jobs $J^* \subseteq J \setminus \{i\}$ to pair with i. While selecting a larger J^* is desirable for minimizing the size of the clique template cover, J^* cannot be chosen arbitrarily, as a clique template that covers (i,J^*) may not exist.

The key idea in detecting the existence of a clique template $T(L,U)$ that covers (i,J^*) is that the jobs in J^* impose both lower and upper bounds on the value of $U_i - L_i$.

First, we address the upper bound of $U_i - L_i$ imposed by a job in J^*.

Proposition 3.2. *If $T(L,U)$ covers (i,j), then $U_i - L_i \le S_{ij} + S_{ji} - 2$.*

Proof. Since $T(L,U)$ covers (i,j), we have:

$$U_j - L_i = S_{ij} - 1, \quad U_i - L_j = S_{ji} - 1, \quad \text{and} \quad U_j - L_j \ge 0.$$

Thus:

$$U_i - L_i = (L_j + S_{ji} - 1) - (U_j - S_{ij} + 1) = -(U_j - L_j) + (S_{ij} + S_{ji} - 2) \leq S_{ij} + S_{ji} - 2.$$

□

Next, we derive the lower bound of $U_i - L_i$. Unlike the upper bound, it is imposed by two jobs in J^*.

Proposition 3.3. *Let i, k, l be mutually distinct jobs. If $T(L, U)$ covers $(i, \{k, l\})$, then*

$$U_i - L_i \geq \max\{S_{ki} + S_{il} - S_{kl} - 1,\, S_{li} + S_{ik} - S_{lk} - 1\}.$$

Proof. Since $T(L, U)$ covers both (i, k) and (i, l), we have

$$U_i - L_i = (L_k + S_{ki} - 1) - (U_l - S_{il} + 1) = -(U_l - L_k) + (S_{ki} + S_{il} - 2).$$

Since $U_l - L_k \leq S_{kl} - 1$, it follows:

$$U_i - L_i \geq S_{ki} + S_{il} - S_{kl} - 1.$$

By symmetry, swapping k and l, we also get

$$U_i - L_i \geq S_{li} + S_{ik} - S_{lk} - 1.$$

□

If the upper bound on $U_i - L_i$ imposed by a job j is smaller than the lower bound imposed by jobs k and l, it is impossible for any clique template to cover $(i, \{j, k, l\})$. To resolve this, either k or l is excluded from J^*, prioritizing the exclusion of the job that imposes higher upper bound on $U_i - L_i$.

The following table summarizes the relevant bounds and provides corresponding notation that aligns with this strategy.

Table 3.1: Notation for bounds on $U_i - L_i$

Symbol	**Definition**
$UB_i(j)$	$S_{ij} + S_{ji} - 2$
J_i	The list of elements in $J \setminus \{i\}$, sorted in ascending order of $UB_i(\cdot)$
$LB_i(k)$	$\max\left\{S_{ki} + S_{il} - S_{kl} - 1,\ S_{li} + S_{ik} - S_{lk} - 1 \mid l \text{ precedes } k \text{ in } J_i\right\}$

Here, $UB_i(j)$ denotes the upper bound on $U_i - L_i$ due to job j, while $LB_i(k)$ indicates the lower bound due to job k. The exclusion strategy ensures that job k is excluded from J^* if $UB_i(j) < LB_i(k)$ for a job $j \in J^*$.

We provide a pseudo-code of the first phase of CTCGA as follows:

Algorithm 1 Clique template cover generation algorithm - phase 1

```
1: $P \leftarrow \emptyset$.
2: Sort $J$ in the ascending order of $\sum_{j\in J\setminus\{i\}}(S_{ij} + S_{ji})$ for each $i \in J$.
3: for $i \in J$ do
4:     Construct $J_i$.
5: end for
6: for $i \in J$ do
7:     for $a = 2, \dots, n-1$ do
8:         $k \leftarrow a$-th element of $J_i$.
9:         $LB_i(k) \leftarrow 0$.
10:        for $b = 1, \dots, a-1$ do
11:            $l \leftarrow b$-th element of $J_i$.
12:            $LB_i(k) \leftarrow \max\{LB_i(k), S_{ki} + S_{il} - S_{kl} - 1, S_{li} + S_{ik} - S_{lk} - 1\}$.
13:        end for
14:    end for
15: end for
16: for $i \in J$ do
17:    while $J_i \neq \emptyset$ do
18:        $j \leftarrow$ first element of $J_i$.
19:        Delete $j$ from $J_i$.
20:        Delete $i$ from $J_j$.
21:        $J^* \leftarrow \{j\}$.
22:        for $k \in J_i$ do
23:            if $LB_i(k) \leq UB_i(j)$ then
24:                $J^* \leftarrow J^* \cup \{k\}$.
25:                Delete $k$ from $J_i$.
26:                Delete $i$ from $J_k$.
27:            end if
28:        end for
29:        $P \leftarrow P \cup \{(i, J^*)\}$.
30:    end while
31: end for
32: return $P$.
```

Phase 1 performs two main tasks:

1. **Preparation (lines 2–15):**

    - **Line 2:** Determines a job sequence for the partitioning task.

- **Lines 3–5:** Constructs the list J_i for every job $i \in J$.
- **Lines 6–15:** Calculates the lower bound $LB_i(k)$ for every distinct job pair (i, k).

2. **Partitioning (lines 16–31):**

- In the partitioning task, a partition of all distinct unordered job pairs is constructed sequentially. During the task, for each job i, every job k included in J_i indicates that the job pair (i, k) has not yet been assigned to any partition.
- **Lines 16–21:** A single job i is fixed, and the first element j of J_i is included in J^*. Since J_i is sorted in the ascending order of $UB_i(\cdot)$, $UB_i(j)$ is the smallest upper bound on $U_i - L_i$.
- **Lines 22–28:** Iterate through the remaining jobs $k \in J_i$. If the lower bound $LB_i(k)$ is less than or equal to $UB_i(j)$, include k in J^*.

We prove the existence of a clique template that covers all job pairs in each partition returned by the algorithm. The existence ensures the algorithm's validity.

Proposition 3.4. *Let P be the returned set of the first phase of CTCGA. Then, for every $(i, J^*) \in P$, there exists a clique template that covers (i, J^*) .*

Proof. We prove the proposition by constructing vectors $L, U \in \mathbb{Z}^n$ such that $T(L, U)$ is a clique template that covers (i, J^*). Specifically, we define
$J(L, U) = J^* \cup \{i\}$, $L_i = 0$, $U_i = UB_i(j)$, and

$$(L_k, U_k) = (U_i - S_{ki} + 1, L_i + S_{ik} - 1), \quad \forall k \in J^*,$$

where j is the first element of J_i in the iteration that generated (i, J^*).

First, we show that $J^* \cup \{i\} \subseteq J(L, U)$. It is clear that $U_i - L_i = UB_i(j) \geq 0$, and for each $k \in J^*$:

$$\begin{aligned} U_k - L_k =&(L_i + S_{ik} - 1) - (U_i - S_{ki} + 1) = (0 + S_{ik} - 1) - (UB_i(j) - S_{ki} + 1) \\ =&S_{ik} + S_{ki} - UB_i(j) - 2 = UB_i(k) - UB_i(j) \geq 0, \end{aligned}$$

since j is the first element in J_i, ensuring $UB_i(k) \geq UB_i(j)$.

Next, we prove that $S(L, U)$ defines a clique by showing that for every distinct job pair (k, l) in $J(L, U) = J^* \cup \{i\}$, $U_l - L_k \leq S_{kl} - 1$ holds. For distinct $k, l \in J(L, U)$, if both k, l are elements of J^*,

$$\begin{aligned} U_l - L_k &= (L_i + S_{il} - 1) - (U_i - S_{ki} + 1) \\ &= (S_{ki} + S_{il} - S_{kl} - 1) - UB_i(j) + (S_{kl} - 1) \\ &\leq \max\{LB_i(k), LB_i(l)\} - UB_i(j) + (S_{kl} - 1) \\ &\leq UB_i(j) - UB_i(j) + S_{kl} - 1 = S_{kl} - 1. \end{aligned}$$

If $k = i$ or $l = i$, $U_l - L_k \leq S_{kl} - 1$ holds by definition of L, U values. Therefore, $S(L, U)$ defines a clique.

Finally, $T(L, U)$ covers (i, J^*) by definition of L_k, U_k values for every k in J^*. $\square$

The time complexity of Algorithm 1 is $O(n^3)$. Sorting jobs (line 2) takes $O(n^2)$ time. Constructing J_i for all $i \in J$ (lines 3–5) takes $O(n^2 \log n)$ time. Calculating $LB_i(k)$ for all distinct job pairs (i, k) (lines 6–15) requires $O(n^3)$ time. Lastly, the partitioning process (lines 16–31) also operates in $O(n^3)$ time.

3.1.2 Clique Template Cover Generation Algorithm - Phase 2

The second phase of CTCGA constructs clique templates that satisfy the coverage conditions defined in the first phase. Additionally, this phase generates a single, additional clique template specifically designed to improve the LP relaxation bound, which is included in the final clique template cover.

To construct a clique template $T(L, U)$ that satisfies the coverage conditions with large clique size $|S(L, U)|$, a linear program is solved. Given a job i and a job set J^*, the solution of this linear program provides a clique template that covers (i, J^*). This program is referred to as the *clique template generating linear program* (CTGLP). The formulation is as follows:

$$\begin{aligned}
\text{CTGLP}(i, J^*): \quad & \text{maximize} \quad && \sum_{j \in J}(u_j - l_j) && (3.1)\\
& \text{subject to} \quad && u_k - l_j \leq S_{jk} - 1, \quad \forall j, k \in J,\, j \neq k, && (3.2)\\
& && u_i - l_i \geq 0, && (3.3)\\
& && u_j - l_j \geq 0, \quad \forall j \in J^*, && (3.4)\\
& && u_j - l_i = S_{ij} - 1, \quad \forall j \in J^*, && (3.5)\\
& && u_i - l_j = S_{ji} - 1, \quad \forall j \in J^*. && (3.6)
\end{aligned}$$

The linear program CTGLP(i, J^*) has $2n$ decision variables: l_j and u_j for each job j in J. Let (l^*, u^*) be a feasible integral solution of the linear program. The constraints (3.2) enforce that $S(l^*, u^*)$ defines a clique. The constraints (3.3) and (3.4) ensure that $\{i\} \cup J^* \subseteq J(l^*, u^*)$. The constraints (3.5) and (3.6) ensure that $T(l^*, u^*)$ covers (i, J^*).

The objective (3.1) is to maximize the size of $S(l^*, u^*)$.

We show that an integral optimal solution to CTGLP(i, J^*) can be obtained by solving the linear program.

Proposition 3.5. *Let $i \in J$ and $J^* \subseteq J \setminus \{i\}$. Then there exists an integral optimal solution of CTGLP(i, J^*), provided it is feasible.*

Proof. We first show that the constraint matrix defined by (3.2) - (3.6) is totally unimodular. Each row in the constraint matrix contains at most two nonzero elements, and these elements are either -1 or 1. Importantly, both -1 and 1 do not appear twice in any row. Such matrices are known to be totally unimodular (Ghoulia-Houri, 1962). Specifically, the total unimodularity is ensured because, for any collection of columns selected from this matrix, the sum of those columns yields a vector with each component equal to -1, 0 or 1. This property confirms that the constraint matrix is totally unimodular.

Since the constraint matrix is totally unimodular and every right-hand side value is integral, the polyhedron defined by (3.2) - (3.6) is integral. Therefore, the linear program admits an integral optimal solution if it is feasible. $\square$

By Proposition 2.2, including clique inequalities associated with maximal cliques in the conflict graph is preferable to tighten the LP relaxation bound. For the purpose, we introduce the concept of *maximality* for clique templates.

Definition 3.6. *A clique template $T(L, U)$ is maximal if and only if the following condition is satisfied:*
If $S(L^*, U^*)$ defines a clique and $S(L^*, U^*) \supseteq S(L, U)$, then $S(L^*, U^*) = S(L, U)$.

A key property of maximal clique templates is that if a clique template $T(L, U)$ is maximal and $S(L + t\mathbf{1}, U + t\mathbf{1}) \subseteq \mathcal{V}$, then $C(L + t\mathbf{1}, U + t\mathbf{1})$ is a maximal clique in the conflict graph $(\mathcal{V}, \mathcal{E})$ for every $t \in \mathbb{Z}$. Therefore, incorporating maximal clique inequalities into a time-indexed model is equivalent to including maximal clique templates in the clique template cover.

Let (l^*, u^*) be an integral optimal solution of CTGLP(i, J^*). If $J(l^*, u^*) = J$, then $T(l^*, u^*)$ is a maximal clique template, as any counterexample would contradict the optimality of (l^*, u^*). However, $T(l^*, u^*)$ is not guaranteed to be maximal if $J(l^*, u^*) \neq J$. In such cases, a maximal clique template can be obtained by enlarging $S(l^*, u^*)$. Note that the enlarged clique template still covers (i, J^*). The enlarging algorithm is referred to as the *maximal clique template generation algorithm* (MCTGA). The pseudo-code of the algorithm is as follows:

Algorithm 2 Maximal clique template generation algorithm

```
procedure MCTGA($(l^*, u^*)$)
    $L \leftarrow l^*$, $U \leftarrow u^*$.
    for $i \in J(l^*, u^*)$ do
        $U_i \leftarrow \min_{j \in J(l^*, u^*) \setminus \{i\}} \{L_j + S_{ji} - 1\}$.
    end for
    for $i \in J(l^*, u^*)$ do
        $L_i \leftarrow \min_{j \in J(l^*, u^*) \setminus \{i\}} \{U_j - S_{ij} + 1\}$.
    end for
    return $T(L, U)$.
end procedure
```

We prove that the algorithm returns a maximal clique template if the input to the algorithm is an integral optimal solution of CTGLP(i, J^*).

Proposition 3.7. *Let $i \in J$ and $J^* \subseteq J \setminus \{i\}$. If (l^*, u^*) is an integral optimal solution to CTGLP(i, J^*), MCTGA(l^*, u^*) returns a maximal clique template.*

Proof. Let (l^*, u^*) be an integral optimal solution to CTGLP(i, J^*), and let $T(L, U)$ be the clique template returned by MCTGA(l^*, u^*). Note that for each job $j \in J$, there exists a job $k \in J \setminus \{j\}$ such that $u_j^* = l_k^* + S_{kj} - 1$; otherwise, u_j could be increased while maintaining the feasibility, contradicting the optimality of (l^*, u^*). Similarly, for each job $j \in J$, there exists a job $k \in J \setminus \{j\}$ such that $l_j^* = u_k^* - S_{jk} + 1$.

We show that any set $S(L^*, U^*)$ that strictly includes $S(L, U)$ does not define a clique. Without loss of generality, let $(j, t) \in J \times \mathbb{Z}$ exists such that $(j, t) \notin S(L, U)$ and $S(L, U) \cup \{(j, t)\} \subseteq S(L^*, U^*)$.

Case (i): $j \in J \setminus J(l^*, u^*)$

Let $j^1 \in J \setminus \{j\}$ be a job such that $u_{j^1}^* = l_j^* + S_{jj^1} - 1$. If $j^1 \in J(l^*, u^*)$, $(j^1, u_{j^1}^*) \in S(l^*, u^*)$ leads $t \geq l_j^*$. Otherwise, there exists a job j^2 such that $u_{j^2}^* = l_{j^1}^* + S_{j^1 j^2} - 1$. This process continues iteratively until an index m such that $j^m \in J(l^*, u^*)$ is found. Such m exists since no job can appear twice in $j^1, \ldots, j^m$, as $l_{j^1}^* < \ldots < l_{j^m}^*$ holds. Let $j^0 := j$. Then $u_{j^m}^* - l_j^* = \sum_{k=1}^{m} (u_{j^k}^* - l_{j^{k-1}}^*) + \sum_{k=1}^{m-1} (l_k^* - u_k^*) \geq \sum_{k=1}^{m} (S_{j^{k-1} j^k} - 1) + (m - 1) \geq S_{jj^m} - 1$. By the constraint $u_{j^m} - l_j \leq S_{jj^m} - 1$ of CTGLP(i, J^*), $u_{j^m}^* - l_j^*$ must equal $S_{jj^m} - 1$. Therefore, it follows that $t \geq l_j^*$. Similarly, it can be shown that $t \leq u_j^*$. No such t exists, since $j \in J \setminus J(l^*, u^*)$ implies $l_j^* > u_j^*$.

Case (ii): $j \in J(l^*, u^*)$

Since $j \in J(l^*, u^*)$, t must either be smaller than l_j^* or greater than u_j^*. Consider the case where t is smaller than l_j^*. By the lines 6–8 of the Algorithm 2, there exists a job $k \in J(l^*, u^*)$ such that $u_k^* = l_j^* + S_{jk} - 1$. Therefore, $S(L^*, U^*)$ cannot define a clique, as $U_k^* - L_j^* \geq u_k^* - t > u_k^* - l_j^* = S_{jk} - 1$. The case where $t > u_j^*$ can be proven similarly. □

We denote by CTGLP$(\emptyset)$ the linear program obtained by relaxing the constraints (3.3)–(3.6) from CTGLP(i, J^*). The linear program returns a maximum-sized set $S(l^*, u^*)$ that defines a clique if $J(l^*, u^*)$ equals J. Assuming that larger cliques provide a stronger LP relaxation bound, we include the clique template obtained from the solution of CTGLP$(\emptyset)$

in the clique template cover. Propositions 3.5 and 3.7, which address CTGLP(i, J^*), also hold for CTGLP($\emptyset$) because the same proofs apply.

The pseudo-code for the second phase of CTCGA is as follows:

Algorithm 3 Clique template cover generation algorithm - phase 2

```
1: $F \leftarrow \emptyset$.
2: $P \leftarrow$ Set of (job, job set) pairs returned by the first phase of CTCGA.
3: $P \leftarrow P \cup \{\emptyset\}$.
4: for $elem \in P$ do
5:     $(l^*, u^*) \leftarrow$ Optimal integral solution of CTGLP($elem$).
6:     $T(L, U) \leftarrow MCTGA(l^*, u^*)$.
7:     $F \leftarrow F \cup \{T(L, U)\}$.
8: end for
9: return $F$.
```

The second phase repeatedly solves a linear program and enlarges a clique template. As a result, a clique template cover composed of maximal clique templates is constructed.

The second-phase algorithm is valid, as every linear program CTGLP(i, J^*) solved in the second phase is feasible. The feasibility is guaranteed by Proposition 3.4, which ensures the existence of a clique template $T(L, U)$ that covers (i, J^*). The L, U values of this clique template provide a feasible solution to CTGLP(i, J^*).

Finally, an instance of the generic time-indexed model provided in Section 2.2 is constructed with the clique template cover returned by CTCGA. The time-indexed model is referred to as the model *Proposed*.

3.2 Restricted Time-Indexed Model

The number of decision variables and constraints in any instance of the generic time-indexed model, including our model proposed in Section 3.1, is pseudo-polynomial. This large size may render solving the integer program—or even its LP relaxation— computationally challenging. Reducing the model's size by excluding some decision variables is a viable strategy to address this challenge. Specifically, in this approach job i can start processing at time t, where x_{it} remains in the *restricted time-indexed model*. The restricted model sacrifices optimality because the optimal solution may no longer be representable with the included decision variables. However, the potential loss in solution quality may be acceptable given the computational gains achieved by using the restricted model.

In this section, we propose a restricted time-indexed model based on the novel time-indexed model presented in Section 3.1. Let $\{x_{it} \mid (i,t) \in \mathcal{V}^R\}$ be the set of decision variables in the restricted model, where $\mathcal{V}^R$ is an arbitrary subset of $\mathcal{V}$. Formulating the capacity constraint in the restricted model involves covering all edges in $\mathcal{E}_C$ where both endpoints belong to $\mathcal{V}^R$. Consequently, the reduced number of edges allows these to be covered with fewer clique inequalities compared to covering all edges in $\mathcal{E}_C$. Leveraging the properties of the novel model, we develop a method to identify which clique inequalities from the original model can be excluded in the restricted model while maintaining its validity. This exclusion enables a further reduction in the model size.

We achieve the identification by associating a set of clique inequalities from the original model with each decision variable in the original model. This set is constructed to ensure that excluding a clique inequality preserves the validity of the restricted model, provided that no decision variable in the restricted model is associated with it.

Let F be the clique template cover returned by CTCGA, and let x_{ir} be a decision variable in the original model. The set of clique inequalities derived from F to be associated with x_{ir} is constructed as follows:

For every job j in $J \setminus \{i\}$, there exists a clique template $T(L^{ij}, U^{ij})$ in F that covers the job pair (i,j). We compare the values of $U_i^{ij} - L_i^{ij}$ and $U_j^{ij} - L_j^{ij}$:

- **Case 1:** If $U_i^{ij} - L_i^{ij} \leq U_j^{ij} - L_j^{ij}$, associate two cliques derived from $T(L^{ij}, U^{ij})$ with x_{ir}:

  1. **Clique 1:** Set $L_i = r$.
  2. **Clique 2:** Set $U_i = r$.

  These cliques are uniquely obtained through appropriate time translations of $S(L^{ij}, U^{ij})$.

- **Case 2:** If $U_i^{ij} - L_i^{ij} > U_j^{ij} - L_j^{ij}$, no clique derived from $T(L^{ij}, U^{ij})$ is associated with x_{ir}.

An example of this inclusion is shown in Figure 3.2, which shows two cliques associated with $(i, 3)$, derived from the clique template that covers (i, j).

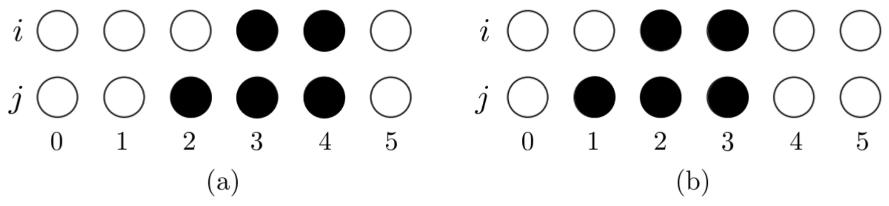

Figure 3.2: 2-job example of clique association in the restricted model

We present the restricted time-indexed model based on the exclusion strategy is as follows:

$$
\begin{aligned}
\mathrm{TI}(F,\mathcal{V}^R): \quad & \text{minimize} && \sum_{(i,t)\in\mathcal{V}^R} c_{it}x_{it} \\
& \text{subject to} && \sum_{t:(i,t)\in\mathcal{V}^R} x_{it} = 1, \quad \forall i \in J, && (3.7)\\
& && \sum_{(i,t)\in c\cap\mathcal{V}^R} x_{it} \le 1, \quad \forall c \in \bigcup_{(i,r)\in\mathcal{V}^R} C(i,r), && (3.8)\\
& && x_{it}\in\{0,1\}, \quad \forall (i,t)\in\mathcal{V}^R, \\
& \text{where} && C(i,r) := \bigcup_{j\in J\setminus\{i\}} \{C(L^{ij}+t\mathbf{1}, U^{ij}+t\mathbf{1}) \mid \\
& && \qquad U_i^{ij}-L_i^{ij} \le U_j^{ij}-L_j^{ij},\ t\in\{r-L_i^{ij},\ r-U_i^{ij}\}\}.
\end{aligned}
$$

Here, F is the clique template cover returned by CTCGA, $\mathcal{V}^R$ is a subset of $\mathcal{V}$, and $T(L^{ij},U^{ij})$ is an element of F that covers (i,j). Constraints (3.7) and (3.8) ensure the assignment constraint and the capacity constraint, respectively.

We conclude this section by demonstrating the validity of $\mathrm{TI}(F,\mathcal{V}^R)$.

Proposition 3.8. *Let F be the clique template cover returned by CTCGA and $\mathcal{V}^R$ be a subset of $\mathcal{V}$. Then $TI(F,\mathcal{V}^R)$ is a valid restricted time-indexed model.*

Proof. Let $\{(i,r),(j,s)\}\subseteq\mathcal{V}^R$, where $((i,r),(j,s))\in\mathcal{E}_C$. For the clique template $T(L^{ij},U^{ij})$, either $U_i^{ij}-L_i^{ij}\le U_j^{ij}-L_j^{ij}$ or $U_i^{ij}-L_i^{ij}\ge U_j^{ij}-L_j^{ij}$ holds. Without loss of generality, assume that $U_i^{ij}-L_i^{ij}\le U_j^{ij}-L_j^{ij}$. In this case, $\{C(L^{ij}+(r-L_i^{ij})\mathbf{1},U^{ij}+(r-L_i^{ij})\mathbf{1}),C(L^{ij}+(r-U_i^{ij})\mathbf{1},U^{ij}+(r-U_i^{ij})\mathbf{1})\}\subseteq C(i,r)$.
Since $((i,r),(j,s))\in\mathcal{E}_C$, $-S_{ji}+1\le s-r\le S_{ij}-1$ holds. Therefore, $(s-r)$ is included in at least one of the intervals $[-S_{ji}+1,U_j^{ij}-U_i^{ij}]$ or $[L_j^{ij}-L_i^{ij},S_{ij}-1]$. If $(s-r)$ lies within in the interval $[-S_{ji}+1,U_j^{ij}-U_i^{ij}]$, $\{(i,r),(j,s)\}\subseteq C(L^{ij}+(r-L_i^{ij})\mathbf{1},U^{ij}+(r-L_i^{ij})\mathbf{1})$.

Otherwise, $\{(i,r),(j,s)\} \subseteq C(L^{ij} + (r - U_i^{ij})\mathbf{1}, U^{ij} + (r - U_i^{ij})\mathbf{1})$. $\square$

Chapter 4

Computational Experiments

This chapter presents the results of the computational experiments conducted to evaluate the proposed models and algorithms. All algorithms were implemented in C++, and FICO Xpress v9.2.2 was utilized as the LP/MILP solver. Linear programs were solved using the barrier algorithm provided by the solver. The experiments were performed on a machine running Windows 10, equipped with a 64-bit Intel(R) Core(TM) i7-4770S CPU (3.10 GHz) and 16 GB of RAM.

4.1 Instance Generation

Synthetic instances were generated and tested across all experiments. This section explains the instance generation process.

4.1.1 Processing and Setup Times

Processing and setup times were generated by an algorithm that takes four parameters: the number of jobs n, minimum processing time $p_{\min}$, maximum processing time $p_{\max}$, and maximum setup time $s_{\max}$. In the algorithm, for each job $i \in J$, processing time p_i is randomly generated from the uniform distribution $U[p_{\min}, p_{\max}]$. Once all processing times are fixed, setup times s_{ij} for every distinct job pair (i, j) are assigned sequentially.

Initially, each setup time has a lower bound of 0 and an upper bound of $s_{\max}$. Setup times are selected randomly, one at a time, and fixed to a value within their current bounds, chosen uniformly at random.

To ensure the triangle inequality assumption, the algorithm adjusts the bounds of unfixed setup times immediately after a setup time is assigned. For instance, suppose that S_{ij} has already been fixed to 10. If S_{jk} is fixed to 15, by the triangle inequality assumption, $S_{ik} \leq S_{ij} + S_{jk} = 25$ must hold and therefore s_{ik} must be less than or equal to $25 - p_i$. Therefore, if the upper bound of s_{ik} exceeds $25 - p_i$, it is updated to $25 - p_i$ at the point S_{jk} is just fixed to 15. The pseudo-code of the instance generation algorithm is provided in Appendix A.1.

4.1.2 Objective Function

The sum of weighted tardiness was used as the objective function for all problem instances. The objective function requires two parameter values for each job i to set the objective function coefficient c_{it}: due date $d_i \in \mathbb{Z}_+$ and weight $w_i \in \mathbb{Z}_+$. A job i started at time t incurs a cost of $w_i \max\{0, t + p_i - d_i\}$. Therefore, the objective function is expressed as:

$$\sum_{i \in J} \sum_{t=0}^{H-p_i} w_i \max\{0, t + p_i - d_i\} x_{it}$$

in a time-indexed model. The due date parameter values were generated using the method proposed by Lee et al. (1997), with the detailed information provided in Appendix A.2. All job weights were drawn from the uniform distribution $U[1, 10]$.

4.1.3 Planning Horizon Length

The planning horizon length H was set sufficiently large to ensure that every feasible schedule without idle time could be represented within the model, thereby guaranteeing its optimality. Detailed methodology for determining H is provided in Appendix A.3. However, it is important to note that the computational performance reported in this thesis could be further improved by appropriately reducing the planning horizon length. Currently, H is significantly larger than the anticipated makespan of an optimal schedule, which unnecessarily increases the size of the model.

4.1.4 Instance Configuration

In all subsequent experiments, synthetic instances were generated using fixed values for $p_{\min}$ and $p_{\max}$, while varying the parameters n and $s_{\max}$. For every combination of $(n, s_{\max})$, 10 instances were generated and tested.

4.2 Experiments on the Clique Template Cover Generation Algorithm

We present the experimental results of the clique template cover generation algorithm. Since the algorithm consists of two phases, experiments were conducted separately for each phase.

4.2.1 Clique Template Cover Generation Algorithm - Phase 1

We evaluated the performance of the first phase of CTCGA by measuring its computation time and the size of the resulting clique template cover. Although the clique templates are not explicitly constructed in the first phase, the size of the cover is determined during this phase.

In this experiment, we tested combinations of $n \in \{25, 50, 100, 200, 400\}$ and $s_{\max} \in \{25, 50, 75, 100\}$, with the parameters $p_{\min}$ and $p_{\max}$ fixed to 10 and 50, respectively. The results are presented in Table 4.1.

Table 4.1: Computation time and size of the resulting clique template cover from Phase 1 of CTCGA

		Time (s)		Size	
n	s_{max}	Avg.	Std.	Avg.	Std.
25	25	0.0011	0.0008	37.7	1.900
	50	0.0011	0.0006	56.2	1.887
	75	0.0009	0.0004	63.8	3.341
	100	0.0008	0.0001	68.6	3.555
50	25	0.0054	0.0023	86.7	6.165
	50	0.0057	0.0027	141.4	3.292
	75	0.0059	0.0024	166.0	6.403
	100	0.0041	0.0003	179.2	4.833
100	25	0.0289	0.0025	208.2	10.619
	50	0.0338	0.0052	344.9	8.455
	75	0.0324	0.0048	418.7	7.577
	100	0.0281	0.0022	454.6	11.629
200	25	0.2204	0.0294	480.1	18.625
	50	0.2157	0.0144	825.6	21.143
	75	0.2115	0.0160	1013.0	19.422
	100	0.2233	0.0153	1142.0	22.454
400	25	1.7888	0.0753	1059.2	31.799
	50	1.7644	0.0476	1940.5	28.654
	75	1.7627	0.0387	2455.3	31.922
	100	1.7183	0.0376	2836.1	25.225

The computation time remains minimal even with up to 400 jobs, terminating within 2 seconds for all instances. This confirms that the first phase of CTCGA does not impose a computational burden. As expected from the algorithm's time complexity $O(n^3)$, the computation time increases with the number of jobs. However, variation in setup times has a negligible impact on the computation time.

The size of resulting clique template cover exhibits a clear growth trend. It generally increases with the number of jobs and the value of $s_{\max}$. Additionally, the ratio of the size to the number of jobs shows a gradual increase as the number of jobs grows, indicating that larger instances require proportionally more clique templates.

4.2.2 Clique Template Cover Generation Algorithm - Phase 2

We report the computation time of the second phase of CTCGA. Since this phase sequentially constructs maximal clique templates, we present both the total computation time and the average computation time per clique template.

In this experiment, the parameters $p_{\min}$ and $p_{\max}$ were fixed to 10 and 50, respectively. The number of jobs n varied among $\{25, 50, 100, 200\}$, and the maximum setup time $s_{\max}$ varied among $\{25, 50, 75, 100\}$. The results are summarized in Table 4.2.

Table 4.2: Computation time of Phase 2 of CTCGA

n	s_{max}	Total Time (s)	Average Time (s)
25	25	0.079	0.0022
	50	0.126	0.0023
	75	0.156	0.0024
	100	0.159	0.0023
50	25	0.746	0.0082
	50	1.482	0.0106
	75	1.664	0.0103
	100	1.913	0.0107
100	25	8.076	0.0394
	50	14.476	0.0427
	75	17.444	0.0420
	100	18.940	0.0411
200	25	63.582	0.1340
	50	123.548	0.1498
	75	155.634	0.1539
	100	176.579	0.1555

The average time to construct a maximal clique template increases significantly with the number of jobs. This escalation is primarily due to the growing number of decision variables and constraints within CTGLP. However, the construction time appears to be independent of the $s_{\max}$ value, as the setup times only affect the parameter values of the linear program and not its size.

The total computation time increases sharply with the number of jobs, driven by the growing number of maximal clique templates to construct and the longer time required for each construction. Although the results indicate that the computation time is not scalable to the number of jobs, they demonstrate that the approach remains computationally acceptable for moderate-sized instances.

4.3 Comparison with Existing Time-Indexed Models

We compared three instances of the generic time-indexed model—*Proposed*, *Nogueira*, and *Avella*—to demonstrate the effectiveness of our novel time-indexed model relative to those in the literature. The comparison was conducted using the following three metrics:

1. Size of the clique template cover,
2. Time required to solve the LP relaxation,
3. LP relaxation bound.

In the experiment, if the LP relaxation of a model could not be solved within 3,600 seconds, the solving time and the LP relaxation bound were reported as 3,600 seconds and 0, respectively.

Instances were tested with the following parameters: $p_{min} = 10$, $p_{max} = 50$, $n \in \{8, 12, 16, 20\}$, and $s_{max} \in \{25, 50\}$. The results are summarized in Table 4.3.

Table 4.3: Comparison between time-indexed models

		Clique Template Cover Size		LP Solve Time (s)		LP Relaxation Bound	
(n, s_{max})	Model	Avg.	Std.	Avg.	Std.	Avg.	Std.
(8, 25)	Proposed	8.8	0.600	2.9	0.70	427.560	437.400
	Avella	26.7	1.900	3.8	1.54	332.325	401.695
	Nogueira	57.0	0.000	5.3	2.00	350.535	379.968
(8, 50)	Proposed	11.3	1.418	10.1	3.45	495.459	339.810
	Avella	27.6	1.356	8.2	1.72	329.355	282.835
	Nogueira	57.0	0.000	13.3	4.24	338.384	278.595
(12, 25)	Proposed	15.1	0.943	18.9	4.16	381.234	269.439
	Avella	55.1	2.385	31.1	7.02	200.051	184.086
	Nogueira	133.0	0.000	48.9	12.23	291.938	243.374
(12, 50)	Proposed	19.8	0.980	81.2	22.59	413.271	206.328
	Avella	60.4	2.245	104.4	87.60	160.974	153.794
	Nogueira	133.0	0.000	122.8	62.83	213.054	146.186
(16, 25)	Proposed	22.1	2.427	82.7	38.42	424.031	310.158
	Avella	89.6	2.332	161.4	59.58	181.769	266.449
	Nogueira	241.0	0.000	620.1	902.96	352.505	271.436
(16, 50)	Proposed	30.0	1.844	380.8	29.82	396.183	195.661
	Avella	100.4	3.072	692.6	975.15	123.077	101.605
	Nogueira	241.0	0.000	522.7	206.97	173.895	142.308
(20, 25)	Proposed	29.5	2.156	284.3	212.73	438.112	247.324
	Avella	134.1	5.467	919.1	912.50	213.509	185.259
	Nogueira	381.0	0.000	1568.0	1050.27	186.156	129.926
(20, 50)	Proposed	40.4	2.332	1744.6	350.83	874.589	407.376
	Avella	150.9	4.182	2165.2	635.79	318.019	308.609
	Nogueira	381.0	0.000	3116.3	601.58	278.217	324.355

The results for the size of the clique template covers demonstrate that CTCGA constructs a clique template cover with significantly fewer clique templates compared to the models proposed by Avella et al. (2017) and Nogueira et al. (2019). This efficiency is attributed to the manner in which restrictions are imposed within the clique templates.

In our approach, a single clique template is encouraged to contribute to the sufficient condition for forming a valid clique template cover, by covering multiple job pairs simultaneously. To achieve this, the first phase of CTCGA imposes no unnecessary restrictions on L, U values, except for the necessary condition that $S(L, U)$ must define a clique. In contrast, the existing models from the literature adopt stricter structural conditions for their clique templates, thereby limiting their flexibility and effectiveness.

Specifically, in the model *Avella*, all clique templates must have identical U_i values for every job i in $J(L, U)$. This restriction diminishes the number of conditions that a single clique template can satisfy, resulting in a larger clique template cover. Similarly, the model *Nogueira* includes $n(n-1)$ clique templates in its clique template cover, with each template designed to cover only a single job pair. This approach leads to a significantly larger clique template cover compared to our method.

In terms of the time required to solve the LP relaxation, the computational results demonstrate that our proposed model outperforms the other models, especially as the number of jobs increases. The results underscore the relationship between the size of the clique template cover and the time taken to solve the LP relaxation.

The results of the LP relaxation bound are particularly impressive, as the proposed model achieves a much stronger LP relaxation bound compared to the existing models in the literature, despite having significantly fewer constraints. This superior bound can be attributed to the fact that all clique templates in our model are derived from the solution of a linear program aimed at finding the largest clique that satisfies predefined conditions. Consequently, generating clique inequalities through this approach is highly effective for constructing a time-indexed model that yields a strong LP relaxation bound.

4.4 Experiments on the Restricted Time-Indexed Models

We evaluated the effectiveness of the proposed restricted time-indexed model by generating multiple restricted models of the same instance, each differing in the set of included decision variables. Specifically, each restricted model incorporated only those decision variables whose time indices are multiples of a constant value, referred to as *stride*.

In this experiment, we evaluated four values of n from the set $\{8, 12, 16, 20\}$, while keeping other parameters fixed: $s_{\max} = 30$, $p_{\min} = 5$, and $p_{\max} = 30$. For each value of n, we formulated four restricted models using stride values of 1, 2, 4, and 8. The experimental results include the number of constraints in each model and the percentage of constraints retained relative to the original model (stride = 1). These results are summarized in Table 4.4.

Table 4.4: The number of constraints in restricted time-indexed models

		Number of Constraints		Retained Constraints (%)	
n	Stride Value	Avg.	Std.	Avg.	Std.
8	1	3357.5	442.14	1.0000	0.0000
	2	2973.2	380.49	0.8867	0.0379
	4	2234.7	191.94	0.6704	0.0508
	8	1514.5	92.58	0.4574	0.0597
12	1	10 450.0	1022.02	1.0000	0.0000
	2	9621.9	851.86	0.9218	0.0253
	4	7731.2	497.30	0.7426	0.0407
	8	5254.6	317.50	0.5047	0.0233
16	1	20 666.1	1342.73	1.0000	0.0000
	2	19 712.7	1389.08	0.9538	0.0201
	4	16 518.6	799.41	0.8005	0.0309
	8	12 210.2	513.14	0.5924	0.0349
20	1	36 871.5	1896.40	1.0000	0.0000
	2	34 527.8	1631.10	0.9368	0.0231
	4	29 494.7	1330.05	0.8004	0.0209
	8	22 433.5	989.91	0.6089	0.0217

The computational results confirm that the restricted model effectively reduces the number of constraints included in the model while maintaining its validity.

Chapter 5

Conclusion

In this thesis, we investigated time-indexed models for the single machine scheduling problem with sequence-dependent setup times. We proposed a novel time-indexed model based on the concept of a *clique template cover*, a concept also adopted by other models in the literature. The clique template cover that characterizes our model was constructed through a two-phase algorithm. The first phase focused on reducing the number of constraints included in the model, while the second phase aimed to identify strong valid inequalities to tighten the LP relaxation bound. Computational experiments demonstrated that our model outperformed models in the literature, in terms of the number of constraints, the time required to solve the LP relaxation, and the LP relaxation bound.

Additionally, we proposed a restricted time-indexed model, a restricted version of the novel model, that reduces the number of constraints when only a subset of decision variables is employed. Computational results showed that the restricted model effectively reduces the number of constraints while preserving the model's validity.

Several extensions of this research can be explored. First, investigating the problem of finding a minimum-sized clique template cover is a potential research direction. Enhancing the two-phase clique template cover generation algorithm is another avenue for exploration. For instance, in the first phase, constructing a clique template cover without relying on the notion of covering a job pair could lead to alternative and potentially more effective approaches. In the second phase, exploring methods for constructing clique templates that do not require solving a linear program could significantly reduce computational overhead. Lastly, theoretical or empirical studies on the impact of individual clique templates on the LP relaxation bound could provide valuable insights.

Appendix

A.1 Processing and setup time generation

The pseudo-code for generating processing and setup times for the synthetic instances is provided in Algorithm 4. Additionally, the pseudo-code for the functions that update the lower and upper bounds of unfixed setup times after a setup assignment is presented in Algorithm 5 and 6, respectively.

Algorithm 4 Processing and Setup Time Generation Algorithm

```
1: procedure PROCESSING AND SETUP TIME GENERATION($n, p_{min}, p_{max}, s_{max}$)
2:     $J \leftarrow \{1, \ldots, n\}$.
3:     for $i \in J$ do
4:         $p_i \sim U[p_{min}, p_{max}]$.
5:     end for
6:     for $i \in J$ do
7:         for $j \in J$ do
8:             $S_{lb}[i][j] \leftarrow p_i$.
9:             $S_{ub}[i][j] \leftarrow p_i + s_{max}$.
10:        end for
11:    end for
12:    $JobPairs \leftarrow$ list of all distinct job pairs, randomly sorted.
13:    for $(i, j)$ in JobPairs do
14:        $S_{ij} \sim U[S_{lb}[i][j], S_{ub}[i][j]]$.
15:        if $S_{ij} > S_{lb}[i][j]$ then
16:            $S_{lb}[i][j] \leftarrow S_{ij}$.
17:            UpdateLB($i, j, S_{lb}, S_{ub}$).
18:        end if
19:        if $S_{ij} < S_{ub}[i][j]$ then
20:            $S_{ub}[i][j] \leftarrow S_{ij}$.
21:            UpdateUB($i, j, S_{lb}, S_{ub}$).
22:        end if
23:    end for
24: end procedure
```

Algorithm 5 Updating lower bound

```
1: procedure UPDATELB($i, j, S_{lb}, S_{ub}$)
2:     for $k \in J \setminus \{i, j\}$ do
3:         if $S_{lb}[i][j] - S_{ub}[k][j] > S_{lb}[i][k]$ then
4:             $S_{lb}[i][k] \leftarrow S_{lb}[i][j] - S_{ub}[k][j]$.
5:             UpdateLB($i, k, S_{lb}, S_{ub}$).
6:         end if
7:         if $S_{lb}[i][j] - S_{ub}[i][k] > S_{lb}[k][j]$ then
8:             $S_{lb}[k][j] \leftarrow S_{lb}[i][j] - S_{ub}[i][k]$.
9:             UpdateLB($k, j, S_{lb}, S_{ub}$).
10:        end if
11:    end for
12: end procedure
```

Algorithm 6 Updating upper bound

```
1: procedure UPDATEUB($i, j, S_{lb}, S_{ub}$)
2:     for $k \in J \setminus \{i, j\}$ do
3:         if $S_{ub}[i][j] + S_{ub}[j][k] < S_{ub}[i][k]$ then
4:             $S_{ub}[i][k] \leftarrow S_{ub}[i][j] + S_{ub}[j][k]$.
5:             UpdateUB($i, k, S_{lb}, S_{ub}$).
6:         end if
7:         if $S_{lb}[i][k] - S_{ub}[i][j] > S_{lb}[j][k]$ then
8:             $S_{lb}[j][k] \leftarrow S_{lb}[i][k] - S_{ub}[i][j]$.
9:             UpdateLB($j, k, S_{lb}, S_{ub}$).
10:        end if
11:        if $S_{lb}[k][j] - S_{ub}[i][j] > S_{lb}[k][i]$ then
12:            $S_{lb}[k][i] \leftarrow S_{lb}[k][j] - S_{ub}[i][j]$.
13:            UpdateLB($k, i, S_{lb}, S_{ub}$).
14:        end if
15:        if $S_{ub}[k][i] + S_{ub}[i][j] < S_{ub}[k][j]$ then
16:            $S_{ub}[k][j] \leftarrow S_{ub}[k][i] + S_{ub}[i][j]$.
17:            UpdateUB($k, j, S_{lb}, S_{ub}$).
18:        end if
19:    end for
20: end procedure
```

A.2 Due date generation

The distribution of due dates is determined by two parameters: the due date tightness factor τ and the due date range factor R. For each job i, the due date d_i is generated as follows:

$$d_i \sim \begin{cases} U[(1-R) \cdot d_{\text{avg}}, d_{\text{avg}}] & \text{with probability } \tau, \\ U[d_{\text{avg}}, (1 + R \cdot \dfrac{\tau}{1-\tau}) \cdot d_{\text{avg}}] & \text{with probability } 1-\tau, \end{cases}$$

where $d_{avg} = (1-\tau) \cdot n \cdot \dfrac{(p_{min} + p_{max}) + 0.3 \cdot s_{max}}{2}$. For all instances, $\tau = 0.3$ and $R = 0.3$ were applied.

A.3 Planning horizon length generation

The maximum makespan of feasible schedules without idle time can be obtained by solving an asymmetric traveling salesman problem (ATSP). The set of cities for the ATSP is defined as $J_0 := J \cup \{0\}$, where 0 represents a dummy job that precedes the first job and follows the last job, enabling the construction of a tour from a schedule. For each job $i \in J$, we define $S_{0i} = 0$ and $S_{i0} = p_i$.

The decision variable y_{ij} is defined for all distinct $i, j \in J_0$ and takes the value 1 if job j immediately follows job i and 0 otherwise. The planning horizon length H was set to an upper bound on the optimal cost of the ATSP, which is obtained by relaxing the subtour elimination constraints. The model is formulated as follows:

$$\begin{aligned} H = \text{maximize} \quad & \sum_{i,j \in J_0, i \neq j} S_{ij} y_{ij} \\ \text{subject to} \quad & \sum_{j \in J_0 \setminus \{i\}} y_{ij} = 1, \quad i \in J_0, \\ & \sum_{i \in J_0 \setminus \{j\}} y_{ij} = 1, \quad j \in J_0, \\ & y_{ij} \in \{0, 1\}, \quad i, j \in J_0,\, i \neq j. \end{aligned}$$

Bibliography

A. Allahverdi, "The third comprehensive survey on scheduling problems with setup times/costs," *European Journal of Operational Research*, vol. 246, no. 2, pp. 345–378, 2015.

A. Allahverdi, J. N. Gupta, and T. Aldowaisan, "A review of scheduling research involving setup considerations," *Omega*, vol. 27, no. 2, pp. 219–239, 1999.

A. Allahverdi, C. T. Ng, T. E. Cheng, and M. Y. Kovalyov, "A survey of scheduling problems with setup times or costs," *European Journal of Operational Research*, vol. 187, no. 3, pp. 985–1032, 2008.

P. Avella, M. Boccia, and B. D'Auria, "Near-optimal solutions of large-scale single-machine scheduling problems," *INFORMS Journal on Computing*, vol. 17, no. 2, pp. 183–191, 2005.

P. Avella, M. Boccia, C. Mannino, and I. Vasilyev, "Time-indexed formulations for the runway scheduling problem," *Transportation Science*, vol. 51, no. 4, pp. 1196–1209, 2017.

L.-P. Bigras, M. Gamache, and G. Savard, "Time-indexed formulations and the total weighted tardiness problem," *INFORMS Journal on Computing*, vol. 20, no. 1, pp. 133–142, 2008.

Y. Crama and F. C. Spieksma, “Scheduling jobs of equal length: complexity, facets and computational results,” *Mathematical Programming*, vol. 72, pp. 207–227, 1996.

M. R. de Paula, G. R. Mateus, and M. G. Ravetti, “A non-delayed relax-and-cut algorithm for scheduling problems with parallel machines, due dates and sequence-dependent setup times,” *Computers & Operations Research*, vol. 37, no. 5, pp. 938–949, 2010.

M. E. Dyer and L. A. Wolsey, “Formulating the single machine sequencing problem with release dates as a mixed integer program,” *Discrete Applied Mathematics*, vol. 26, no. 2-3, pp. 255–270, 1990.

A. Ghoulia-Houri, “Characterisation des matrices totalement unimodulaires,” *CR Acad/Sci. Paris*, vol. 254, pp. 1192–1194, 1962.

M. Güngör, “Classification and comparison of integer programming formulations for the single-machine sequencing problem,” *Computers & Operations Research*, vol. 173, p. 106844, 2025.

R. Kopf and G. Ruhe, “A computational study of the weighted independent set problem for general graphs.” *Found. Control Eng.*, vol. 12, no. 4, pp. 167–180, 1987.

D. Kress, D. Müller, and J. Nossack, “A worker constrained flexible job shop scheduling problem with sequence-dependent setup times,” *OR Spectrum*, vol. 41, pp. 179–217, 2019.

Y. Kuo, S.-I. Chen, and Y.-H. Yeh, “Single machine scheduling with sequence-dependent setup times and delayed precedence constraints,” *Operational Research*, vol. 20, pp. 927–942, 2020.

Y. H. Lee, K. Bhaskaran, and M. Pinedo, "A heuristic to minimize the total weighted tardiness with sequence-dependent setups," *IIE transactions*, vol. 29, no. 1, pp. 45–52, 1997.

J. Y. Leung, *Handbook of scheduling: algorithms, models, and performance analysis.* Chapman and Hall/CRC, 2004.

S. Lin and B. W. Kernighan, "An effective heuristic algorithm for the traveling-salesman problem," *Operations Research*, vol. 21, no. 2, pp. 498–516, 1973.

J. L. Loveland, S. K. Monkman, and D. J. Morrice, "Dell uses a new production-scheduling algorithm to accommodate increased product variety," *Interfaces*, vol. 37, no. 3, pp. 209–219, 2007.

T. H. Nogueira, C. R. V. d. Carvalho, M. G. Ravetti, and M. C. d. Souza, "Analysis of mixed integer programming formulations for single machine scheduling problems with sequence dependent setup times and release dates," *Pesquisa Operacional*, vol. 39, pp. 109–154, 2019.

M. L. Pinedo, *Scheduling: Theory, Algorithms, and Systems*, 6th ed. Springer, 2022.

A. H. Rinnooy Kan, "Machine scheduling problems: classification, complexity and computations," *(No Title)*, 1976.

W. E. Smith *et al.*, "Various optimizers for single-stage production," *Naval Research Logistics Quarterly*, vol. 3, no. 1-2, pp. 59–66, 1956.

J. P. Sousa and L. A. Wolsey, "A time indexed formulation of non-preemptive single machine scheduling problems," *Mathematical Programming*, vol. 54, pp. 353–367, 1992.

X. Sun, J. S. Noble, and C. M. Klein, “Single-machine scheduling with sequence dependent setup to minimize total weighted squared tardiness,” *IIE Transactions*, vol. 31, no. 2, pp. 113–124, 1999.

J. Van den Akker, C. Van Hoesel, and M. W. Savelsbergh, “A polyhedral approach to single-machine scheduling problems,” *Mathematical Programming*, vol. 85, pp. 541–572, 1999.

J. Van den Akker, C. A. Hurkens, and M. W. Savelsbergh, “Time-indexed formulations for machine scheduling problems: Column generation,” *INFORMS Journal on Computing*, vol. 12, no. 2, pp. 111–124, 2000.

국문초록

본 논문에서는 순서 의존적 작업준비시간을 가지는 단일 기계 일정계획 문제를 푸는 시간-인덱스 모형을 다룬다. 작업준비시간이 없는 경우에는 설비의 용량 제약에 대한 이상적인 모형화가 알려져 있는데, 순서 의존적 작업준비시간이 존재하는 경우에는 이러한 모형화가 아직 알려져 있지 않다. 시간-인덱스 모형의 큰 크기로 인한 계산 부담을 해결하기 위해서는 적은 수의 강한 제약식들로 설비의 용량 제약을 모형화하는 것이 필수적이다.

이를 위해 용량 제약을 모형화하기 위한 두 단계 알고리즘을 제안한다. 첫 번째 단계에서는 시간-인덱스 모형에 포함될 제약식 수를 줄이고, 두 번째 단계에서는 모형의 선형완화경계값을 강화시키기 위한 제약식들을 구축한다. 계산 실험을 통해 제안된 새로운 시간-인덱스 모형이 기존 문헌의 모형들보다 훨씬 더 작은 모델 크기를 가지며 훨씬 더 좋은 선형완화경계값을 더 빠르게 얻을 수 있음을 확인하였다.

추가적으로, 결정 변수의 부분 집합만을 사용할 때 적용 가능한 제한된 시간-인덱스 모형을 소개한다. 이 모형은 모형의 유효성을 유지하면서 제외할 수 있는 제약식을 식별하여 모형의 크기를 더욱 줄인다. 이 제한된 모형이 제약식의 수를 효과적으로 줄임을 실험적으로 확인하였다.

주요어: 단일 기계 일정계획 문제, 순서 의존적 작업준비시간, 시간-인덱스 모형, 시간-인덱스 모형화

학번: 2023-22551

감사의 글

이 학위논문을 마무리하며, 이 논문을 완성할 수 있도록 도와주신 분들께 깊은 감사를 전하고자 합니다.

먼저, 학위 과정 동안 지도와 격려를 아끼지 않으신 이경식 교수님께 진심으로 감사드립니다. 교수님의 가르침이 연구자로서, 또 인간으로서 성장하는 데 큰 도움이 되었습니다. 특히, 학업과 연구에 온전히 집중할 수 있는 환경을 마련해 주신 교수님의 배려와 헌신에 깊이 감사드립니다. 이 은혜는 반드시 보답하겠습니다.

또한, 바쁜 일정 중에도 이 논문을 심사해 주시고 귀중한 조언을 아끼지 않으신 문일경 교수님과 홍성필 교수님께도 진심으로 감사의 말씀을 드립니다.

연구실에서 함께한 종헌 선배님, 준영 선배님, 세영 선배님, 성원 선배님, 호진이 형, 학용 선배님, 영주 선배님, 우석이, 예빈이, 민규에게도 고마움을 전합니다. 여러분과 함께하며 많은 것을 배우고 성장할 수 있었던 것은 제게 큰 행운이었습니다. 긴 시간이 흘러도 여러분들의 동료로 남아있고 싶습니다.

끝으로, 언제나 아낌없는 사랑과 응원을 보내주신 부모님께 깊은 감사의 마음을 전합니다. 지금까지 베풀어 주신 사랑의 절반만이라도 되돌려드릴 수 있도록 부단히 노력하겠습니다. 존경하고 사랑합니다, 아버지, 어머니.