Mixed-Integer Linear Programming Models for Least-Commitment Partial-Order Planning

by

Buser Say

A thesis submitted in conformity with the requirements
for the degree of Master of Applied Science
Graduate Department of Mechanical and Industrial Engineering
University of Toronto

Abstract

Mixed-Integer Linear Programming Models for Least-Commitment Partial-Order Planning

Buser Say
Master of Applied Science
Graduate Department of Mechanical and Industrial Engineering
University of Toronto
2017

The central thesis of this dissertation is that the Mixed-Integer Linear Programming (MILP) technology can be effective in producing a least commitment partial-order plan (POP). A POP is a set of actions and ordering constraints between some pairs of actions such that every linear execution of actions that is consistent with the ordering constraints achieves the goals of the planning problem. In the context of Automated Planning, least commitment approach aims to generate POPs that delay the decisions of a plan to its execution time, allowing an autonomous agent to be flexible in terms of how the POP is executed. In this dissertation, we formally define the least commitment POP of a planning problem as a plan with the minimum total action cost and maximum number of linearizations. We formulate this problem as a MILP model and investigate the efficiency of solving it using the commercial MILP solvers.

Acknowledgements

I would like to express my gratitude to a number of people who have helped me throughout my M.A.Sc. program and I will take this opportunity to briefly acknowledge them.

First, I would like to thank my co-supervisors Professor J. Christopher Beck and Professor Andre A. Cire for their continuous support, guidance and patience throughout my M.A.Sc. program. Having you as my mentors have expanded my horizon as a researcher and solidified my decision to purse a career in academia. I feel privileged to have worked with both of you.

I would like to thank my committee members Professor Sheila McIlraith and Professor Scott Sanner for their time and commitment.

I would like to thank my friends at the Toronto Intelligent Decision Engineering Laboratory (TIDEL), Tony, Wen-Yang, Kyle, Margarita, Chang and Eldan, for insightful discussions, sharing ideas and many cups of coffee.

Finally, I would like to thank my mother and my father for their unconditional love, confidence and support. Without their sacrifices, I would not be where I am today.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The central thesis of this dissertation is that the Mixed-Integer Linear Programming (MILP) technology can be effective in generating least commitment partial-order plans. A partial-order plan (POP) is a set of actions and ordering constraints between some pairs of actions such that every linear execution of actions that is consistent with the ordering constraints is a solution to the planning problem. Therefore, a POP compactly represents a set of linearly executable plans i.e., linearizations. Least commitment planning aims to generate POPs that delay the decisions of a plan to its execution time, allowing an agent to be flexible in terms of how it executes the given POP. We define the least commitment flexible POP (LCFP) of a planning problem as the valid POP with the minimum total action cost and maximum number of linearizations. In this dissertation, we model the problem of finding a least commitment flexible POP as a mixed-integer linear optimization model and investigate the efficiency of solving it using the commercial MILP solvers. In particular, we show that the set of conditions that are used to validate a POP (i.e., the necessary conditions of a POP) can be encoded as a set of linear constraints and the least commitment objective can be encoded as a linear objective. In this dissertation, we address two problems:

1. Finding a LCFP given an input set of actions that form a valid plan: Actions from an initial plan can be removed and/or reordered to find a POP with minimum total action cost and maximum number of linearizations [54]. We contribute to the least commitment planning literature by developing a partial-order relaxation approach based on a MILP model.

2. Formalization and investigation of the complete least commitment flexible partial-order planning problem: Instead of finding the LCFP of an initial plan, we find the LCFP for the complete planning problem. We formalize the least commitment flexible planning problem and solve it using a logic-based Benders decomposition (LBBD), also based on the MILP technology.

1.1 Motivation

Our main motivation is to generate POPs that provide execution flexibility for an agent. Least commitment planning captures an important aspect of POP flexibility by delaying some action ordering decisions to execution time. The recent success of the MILP-based heuristic sequential planners [64, 21] provides the motivation to build a POP planner based on the MILP technology. In further detail, the motivations for the work that is presented in this dissertation are as follows.

1. **The Exploration of MILP Models for Planning** - MILP is among the state-of-the-art solution techniques

to many combinatorial optimization problems that exhibit similar characteristics to planning, such as vehicle routing [19], scheduling [49] and traveling salesman [52] problems. The extensive literature on these hard problems show that different formulations can dramatically improve both the solution quality and the computation time of the MILP models. While there has been work on MILP models for various planning problems, improving formulations via the introduction of stronger constraints and more expressive variables has received little attention from the planning community. In particular, we investigate the following:

- Utilization of non-binary decision variables, linear constraints and objective functions to reduce the size of state-of-the-art formulations in planning [56, 72],
- Representation of the necessary conditions of a plan as linear constraints to improve the performance of the existing formulations [9, 64, 41, 42, 72].

2. **The Formalization and the Optimization of the Complete Least Commitment Flexible Planning Problem** - Least commitment planning is often used either inside a partial-order planner to delay the immediate commitment decisions (i.e., do not make a planning commitment unless the commitment contributes to the validity of the POP) [51, 61, 78, 79], or as the partial-order relaxation of an initial plan [3, 26, 56, 72]. Even for the limited planning problems solved in the latter, the optimization of such least commitment objectives yields very challenging problems [1, 26, 56, 54, 72]. The literature lacks the methods for optimally solving the least commitment flexible planning problem. To our knowledge, we present the first POP planner that optimizes this complex objective.

1.2 Overview of Dissertation

The outline of the dissertation is as follows.

Chapter 2 reviews the classical planning literature with the emphasis on the different solution definitions, planning objectives and solution techniques. This chapter provides an exhaustive coverage of the use of the MILP technology in solving the classical planning problems.

In Chapter 3, we address the problem of finding a partial-order relaxation of an input sequential plan using MILP technology. We investigate three different proxy functions to optimize the flexibility of a POP. Compared to the state-of-the-art MaxSAT model [56], we empirically show that two of our MILP models result in POPs with equivalent or slightly higher solution qualities with savings of approximately one order of magnitude in computation time. This work is presented at the *Twenty-Second European Conference on Artificial Intelligence* [72].

In Chapter 4, we examine the problem of finding the least commitment flexible plan for the complete planning problem. First, we focus on finding an action cost-optimal POP to the complete planning problem using a logic-based Benders decomposition approach [39, 21]. Then, we extend this work to find the least commitment flexible POP to the complete planning problem. To our knowledge, we present the first POP planner that optimizes the least commitment flexible planning objective for the complete planning problem.

Chapter 5 concludes the content presented in this dissertation and provides areas of future work that can stem from the research that is presented in this dissertation.

1.3 Summary of Contributions

The contributions of this dissertation are as follows.

We investigate three different proxy functions to maximize the number of linearizations in a POP: two from the POP literature and a third novel function based on the temporal flexibility criteria from the scheduling literature. We optimize these functions using three MILP models. Further we strengthen these models by introducing valid linear inequalities. We show empirically that two of our strengthened MILP models results in equivalent or slightly better solution quality with savings of on average one order of magnitude in computation time, compared to the state-of-the-art MaxSAT model [56].

We formalize the least commitment flexible planning problem that consists of two criterion: the minimization of the total action cost and then the maximization of the number of linearizations in a POP. We first investigate the optimization of the total action cost of a POP using two MILP-based decompositions. Then we extend one of our planners to optimize the least commitment flexible planning problem. We show empirically that the optimization of the total action cost using our planners result in similar performance to the state-of-the-art LBBD-based sequential planner [21]. We further provide preliminary results on the optimization of the least commitment flexible planning problem. To our knowledge, we are the first to optimize this complex objective for partial-order plans.

Chapter 2

Literature Review

This chapter describes the literature of *automated planning* with the focus on the *classical planning problems*. The purpose of this literature review is threefold: (1) to present the formal definitions and the notations used for classical planning, (2) to provide a summary of the various solution techniques developed over the last 50 years, and (3) to cover and analyze the use of the *Mixed-Integer Linear Programming* (MILP) models to solve classical planning problems. In Section 2.1, classical planning is introduced in the context of the automated planning. In Section 2.2, classical planning and proposed solution approaches are formally described. In Sections 2.3-2.6, solution techniques that utilize MILP are described. Finally in Section 2.7, methods for generating a *partial-order plan* (POP) are described.

2.1 Automated Planning

Planning is the reasoning portion of acting: the process of selecting and organizing *actions* to achieve some prestated *goals* [58, 21]. Automated planning is the area of research in Artificial Intelligence that studies the theoretical and computational aspects of planning. A *model* is a conceptual device to represent the main elements of a problem. A model can be used to explain basic definitions, clarify restrictive assumptions, and can give rise to different solution techniques with respect to different objectives. A *classical planning problem* is an axiomatization of the actions and their effects, as well as the description of current state and the goals of that must be achieved in the real-world [54]. An instance of a classical planning problem, denoted by the symbol Π, is a model of the real-world planning problem. A *plan* π is a solution to Π, and corresponds to an ordered set of actions that achieve all the goals of Π. Given Π, a *planner* generates a plan π that is executed by a *controller*. The controller executes π and sends feedback to the planner. The feedback typically includes observations about the state of the real world, the execution progress of π, and external/internal changes/events that effect the real world. Given the feedback, the planner monitors the execution of π by deciding whether Π still represents the real world or not. If there are significant discrepancies found between the real world and Π, and π is no longer a solution to the real world problem, a new Π' is modeled and a new plan π' is generated. Figure 2.1 summarizes the high-level interaction between the planner and the controller. This dissertation focuses on the generation of flexible π that is robust to some of the external/internal changes/events that effect Π.

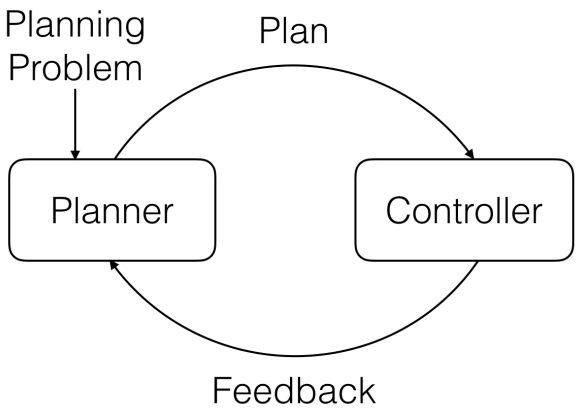

Figure 2.1: High-Level Visualization of Automated Planning.

2.2 Classical Planning

A classical planning problem is a tuple $\Pi = \langle F, A, I, G \rangle$ where F is a set of *fluent symbols*, A is a set of ground *actions*, $I \subseteq F$ is the *initial state*, and $G \subseteq F$ is the *goal state* [30, 29]. A *complete state* $s_c \subseteq F$ specifies the values of all the fluents $f \in F$ such that, $f = true$ if and only if $f \in s_c$, and $f = false$ otherwise. A *partial state* $s_p \subseteq F$ only specifies the values of a subset of fluents $f \in F$ that must be true such that, $f = true$ if $f \in s_p$, and $f = true/false$ otherwise. In classical planning, I is a complete state and G is a partial state. Each action $a \in A$ is associated with three sets of fluents, PRE_a, ADD_a, and DEL_a, representing the preconditions, add effects, and delete effects of a, respectively. An action a is executable in state $s \subseteq F$ if and only if $PRE_a \subseteq s$. The execution of an action a results in the state $s' = (s \cup ADD_a) \setminus DEL_a$. We denote by pre_f, add_f, and del_f the set of actions that have fluent f as precondition, that add f, and that delete f, respectively. A *plan* is a solution to Π and corresponds to a set of linearly executable actions $(a_1, a_2, \ldots, a_m)$ such that, when starting from the initial state I, executing each action in sequence results in a state s^* containing all fluents required in the goal G, i.e. $G \subseteq s^*$ [14]. Finally, we use $\mathcal{A}$ for set of actions since a plan can contain duplicates of an action $a \in A$ where every action a in the set $\mathcal{A}$ is uniquely named [54].

2.2.1 Model Assumptions

Modeling the real world by Π requires various restrictive assumptions. The organization and the content of the assumptions used in this dissertation closely follow [58] and are listed below:

1. Π has a finite number of states: the number of reachable states in Π is bounded by a finite number (i.e. $2^{|F|}$). The implication is that the execution of an action does not change the sets F or A.

2. All the effects of $a \in A$ are deterministic: the execution of an action $a \in A$ in state s always results in the same state $s' = (s \cup ADD_a) \setminus DEL_a$.

3. All states of Π are fully observable: all the values of fluents $f \in F$ are known in the initial state I by the planner. Together with Assumption 2, this assumption means that all the other states can be predicted with certainty.

4. All states of Π remain static with respect to internal events: the internal dynamics of the real world planning problem is not modeled by Π. Only the controller can change the state of Π and occurrence of internal events

Plan Type	**Allow Parallel Action Execution**	**Do Not Allow Parallel Action Execution**
Sequential Action Sets	Parallel Sequential Plans	Sequential Plans
Partial-order Action Sets	Parallel Partial-order Plans	Partial-order Plans

Figure 2.2: Different Solution Definitions.

that can effect the state of the world are ignored.

5. A solution to Π must be a sequentially ordered plan: the planner is restricted to generate only sequentially ordered plans.

6. The goals of Π are explicitly specified by G: A solution to Π is defined as any π that reaches the goal state G from the initial state I.

7. The execution of $a \in A$ is instantaneous: the action duration is abstracted from the set of actions A.

8. Π is static during the generation and the execution of a plan with respect to external events. This assumption is also known as offline planning and assumes that external events that might occur in the real world do not change Π. As a result, the external events are ignored.

Several interesting modifications are obtained by relaxing some of the assumptions listed above. In particular, the relaxation of Assumptions 5 and 6 allow the planner to generate plans with different solution definitions and different planning objectives, respectively.

2.2.2 Different Solution Definitions

A plan in its most general form is a structure for a set of actions. This structure may contain ordering constraints between some pairs of actions and temporal constraints on some subset of actions and goals. This dissertation assumes implicit time (i.e. assumption 7), and focuses only on the ordering structure of a plan. As summarized in Figure 2.2, depending on the ordering structure of the actions and whether a pair of unordered actions can be executed in parallel or not, plans can be grouped under sequential or partial-order plans, or parallel sequential or partial-order plans [1].

Sequential Plans

A *sequential plan* orders all pairs of actions. Formally, a sequential plan π corresponds to a sequence of executable actions $(a_1, a_2, \ldots, a_m)$ such that, when starting from the initial state I, executing each action in sequence results in a state s^* containing all fluents required in the goal G, i.e. $G \subseteq s^*$ [30, 58]. A sequential plan contains exactly one sequentially executable plan. A controller that receives a sequential plan does not have any *execution flexibility* to decide on the ordering of the actions during execution.

Partial-order Plans

A *partial order plan* (POP) corresponds to a set of actions and ordering constraints between the pairs of actions such that executing actions in any sequence consistent with the ordering constraints results in a state s^* containing all fluents required in the goal G, i.e. $G \subseteq s^*$ [71, 58, 54]. While this definition is intuitive, its validation procedure (i.e., checking whether all execution sequences correspond to a sequential plan) is not tractable. Therefore, the standard definition of a POP uses two additional concepts that are associated with its validation procedure, namely: *causal links* and *threats*.

A POP is a tuple $\pi = \langle \mathcal{A}, \mathcal{O}, K \rangle$ where $\mathcal{A}$ is the set of actions of the plan, $\mathcal{O} \subseteq \mathcal{A} \times \mathcal{A}$ is a set of ordering constraints, and K is a set of causal links without threats. Given actions $a_1, a_2 \in \mathcal{A}$ and a fluent $f \in F$, $\kappa(a_1, a_2, f)$ is a causal link for π if a_1 adds the fluent f required by a_2 in all executions of π [44]. The causal link $\kappa(a_1, a_2, f) \in K$ implies $a_1 \prec a_2 \in \mathcal{O}$, where $a_1 \prec a_2 \in \mathcal{O}$ indicates an *ordering constraint* between actions $a_1, a_2 \in \mathcal{A}$. Moreover, K is a set of causal links for π if, for every $\kappa(a_1, a_2, f) \in K$, we have $a_1 \in add_f$, $a_2 \in pre_f$, and there does not exist $\kappa(a_i, a_2, f) \in K$ for any $a_i \in add_f$ [44]. An action $a_3 \in \mathcal{A}$ is a threat to a causal link $\kappa(a_1, a_2, f) \in K$ if $a_3 \in del_f$, $a_3 \prec a_1 \notin \mathcal{O}$ and $a_2 \prec a_3 \notin \mathcal{O}$ [58].

A POP imposes only action orderings necessary for achieving a goal, as opposed to a total ordering of actions as enforced in a sequential plan. Equivalently, a POP represents a set of sequential plans, or linearizations, all including the same actions but under different orderings. POPs provide execution flexibility to the controller, which can dynamically commit to the sequence of actions during the execution of the plan [56]. Note that a POP assumes the linear execution of all actions, including the pairs that are not ordered with an ordering constraint.

Parallel Sequential Plans

A set of actions S can be executed in *parallel* if the execution of the actions result in the same state independent of whether actions $a \in S$ are executed in sequence or in parallel [1]. A *parallel sequential plan* π corresponds to a sequentially ordered sets of parallel actions $(S_1, S_2, \ldots, S_m)$, such that, when starting from the initial state I, executing each parallel action set in sequence results in a state s^* containing all fluents required in the goal G, i.e. $G \subseteq s^*$ [1]. A parallel sequential plan is also known as a *step-based plan* where each index t of the set of parallel actions S_t denotes a step of a plan and t is an element of the numerically ordered set of steps $t \in T(m) = \{1, 2, \ldots, m\}$ such that the horizon m denotes the last element of $T(m)$.

Parallel Partial-order Plans

A *parallel partial-order plan* is a tuple $\pi = \langle \mathcal{A}, \mathcal{O}, K, \# \rangle$ where $\langle \mathcal{A}, \mathcal{O}, K \rangle$ is a POP and $\# \in ((\mathcal{A} \times \mathcal{A}) - \mathcal{O})$ is a set of *non-concurrency relations* defined between pairs of unordered actions $a_1 \perp a_2 \in \#$ [1]. A non-concurrency relation $\#$ explicitly defines which pairs of unordered actions must be executed in a sequence [1]. Note that $\#$ does not contain any information about which pairs of unordered actions must be executed in parallel [1].

This dissertation assumes that a pair of unordered actions must be executed in sequence. Without relaxing the assumption on instantaneous actions (i.e. assumption 7), the parallel execution property does not add any practical information to the execution of a plan, that is, executing a pair of instantaneous unordered actions in sequence, as opposed to parallel, is equivalent in terms of the resulting states as well as the total execution time. The main reason why parallel planning techniques have received so much attention in classical planning is because it reduces the size of the search space using a method called *planning graphs* [6].

2.2.3 Different Planning Objectives

The controller may require plans with high solution quality. Therefore, it may be desirable to optimize the solution quality of a plan with respect to more complex objectives, such as plans that require the minimum cost to execute and/or plans that provide the maximum flexibility with respect to the execution order of actions.

Cost-Optimal Planning

Cost-optimal planning is finding a plan to a planning problem with minimum total action cost. Let c_a be a non-negative cost associated with each action $a \in A$. The cost of a plan $\pi = \langle \mathcal{A}, \mathcal{O} \rangle$ is the sum of all the action costs such that $c(\pi) = \sum_{a \in \mathcal{A}} c_a$. A plan is cost-optimal if and only if there does not exist another plan with lower total action cost.

Least Commitment Planning

Least commitment planning refers to delaying the ordering decisions between pairs of actions to execution time unless they contribute to the validity of the plan [51, 54]. The notion of least commitment planning is relevant to the ordering structure of a plan. Since the ordering structure depends on the set of actions, it is more natural to think about least commitment planning given a set A. Least commitment planning with respect to a set of actions is discussed in more detail in Section 2.7.2. We extend this notion in Chapters 3 and 4.

Step Optimal Planning

Step optimal planning is finding a parallel sequential plan $\pi = (S_1, S_2, \ldots, S_m)$ with the minimum number of sets of parallel actions (i.e. minimum m value). It is common to combine step optimal planning with cost-optimal planning such that, given the minimum m that allows a parallel sequential plan to be generated, the objective is to find the cost-optimal plan with respect to m. Note that finding the cost-optimal plan with respect to m does not correspond to finding the cost-optimal plan for Π. However due to the success of parallel sequential planners [47, 25, 69, 11, 70, 40], it is desirable to optimize such complex objectives.

2.3 Solution Techniques

The solution techniques used in classical planning are motivated by the inherent difficulty in generating valid plans to Π. For the classical planning problem Π, the computational complexity of *plan-existence* (i.e. whether a plan π exists for Π) is EXPSPACE-complete and *plan-length* (i.e. whether a plan π with at most m steps exists for Π) is NEXPTIME-complete [58]. As a result of these complexity results, the state-of-art planners solve Π using algorithms and procedures that show good experimental performance [58].

2.3.1 Compilation Techniques for Classical Planning

A *bounded classical planning problem* Π_m is the problem of finding a parallel sequential plan $\pi = (S_1, S_2, \ldots, S_m)$ for Π [58] with fixed horizon m. The main idea behind compilation techniques is to solve Π_m incrementally with m, until a step optimal plan to Π is found. Typically, the horizon m is initially set to 1, and incremented by one every time Π_m is proven to be infeasible, until a step optimal plan is found. The state-of-the-art parallel sequential planners solve multiple Π_m with different horizon m values in parallel, and allocate computation power that is a function of m [68].

The Constraint Satisfaction Problem (CSP) is a general paradigm for representing a problem using a set of decision variables X with their respective domains $D(X)$, and a set of constraints C such that a solution to CSP is a feasible value assignment to every decision variable in the set X from their respective domains $D(X)$ satisfying every constraint in the set C [58]. Typically, the bounded problem Π_m can be mapped into a CSP where the sets X, $D(X)$ and C have one of the following properties:

1. X is a finite set of discrete decision variables and C is a finite set of constraints. The bounded classical planning problem Π_m can be represented using two sets of decision variables; one action selection variable X_t, $t \in T(m)$ with finite domain $D(X_t) = \{v_1, \ldots v_{|A|}\}$ where each value, v_a, corresponds to the execution of action $a \in A$, and a state variable Y_t, $t \in T(m+1)$ with finite domain $D(Y_t) = \{w_1, \ldots w_k\}$ where each value w_f corresponds to the truth of $f \in F$ at step t [58]. In addition, two types of constraints are used to describe Π_m. The first type of constraint represents the initial state I and the goal state G of Π_m. The second type of constraint relates X_t to both Y_t and Y_{t+1}, by encoding the conditions under which action a is executable and its execution effects, respectively $\forall a \in A, t \in T(m)$. This structure of CSP can be efficiently solved using Constraint Programming techniques [43].

2. X is a finite set of boolean decision variables and C is a finite set of boolean constraints. This problem representation is also known as *Propositional Satisfiability Problem*. The bounded classical planning problem Π_m can be represented using two sets of boolean decision variables; one action selection variable $X_{a,t}$ such that $X_{a,t} = true$ if and only if action $a \in A$ is executed at step $t \in T(m)$, and one state-based variable $Y_{f,t}$ such that $Y_{f,t} = true$ if and only if fluent $f \in F$ is true at the beginning of step $t \in T(m)$ [58]. The equivalent linear representation of the boolean constraints that are used to describe Π_m as a Propositional Satisfiability Problem are presented in Section 2.4.1 under the state-based and the state-change formulations [77]. Propositional Satisfiability Problem can be efficiently solved using SAT techniques [23]. Notable SAT-based planners include SATPLAN [47], Blackbox [48] and others [69, 70, 40].

3. X is a finite set of real continuous or integer decision variables and C is a finite set of linear constraints. This structure is solved efficiently using MILP and is discussed in Section 2.4.

2.3.2 Heuristic Search in Classical Planning

Heuristic search [59] is a method for exploring a search space S based on a *heuristic function* $h(s)$ that approximates the distance from a state in the search space $s \in S$ to a solution $s^* \in S$. Heuristic search can be used to solve the classical planning problem by mapping the states and the solutions in the search space $s, s^* \in S$ to either:

1. the complete states $s_c \subseteq F$ that are reached by executing sequences of actions, and paths of states that start from the initial state I and end at the goal state G (also known as *heuristic state-space planning*), or

Heuristic Search Classifications	Heuristic State-Space Planning	Heuristic Plan-Space Planning
State	Complete state: reached by executing a sequence of actions	Tuple <A,O,K>
Solution	Path of states: starting from the initial state and ending at the goal state	Partial-order Plan

Figure 2.3: Organization of the heuristic search planning.

2. the tuples $\langle \mathcal{A}, \mathcal{O}, K \rangle$ such that the sets $\mathcal{A}$, $\mathcal{O}$ and/or K might be missing some elements qualify as POPs (also known as *heuristic plan-space planning*).

Figure 2.3 summarizes the differences between the heuristic state-space planning and the heuristic plan-space planning.

Heuristic State-Space Planning

In state-space planning, each state $s \in S$ corresponds to a complete state $s_c \subseteq F$ that is reached by executing a sequence of actions $seq_s = (a_1, a_2, \ldots, a_m)$, starting from the initial state I. The heuristic function $h(s)$ typically approximates either the number of steps, or the total action cost that is required to reach the goal state G from the complete state $s = s_c$ [58]. Function $g(s)$ either returns the exact number of steps, or the total action cost that is required to reach the complete state $s = s_c$ from the initial state I [58]. The heuristic state-space planners iteratively expand the state $s \in S$ with the best $f(s) = g(s) + h(s)$ value into new states $s' \in S$ by adding executable actions $a_{m+1} \in A$ at the end of $seq_s = (a_1, a_2, \ldots, a_m)$, until the goal state G is reached such that $s' = (s \cup ADD_{a_{m+1}}) \setminus DEL_{a_{m+1}}$, $seq_{s'} = (a_1, a_2, \ldots, a_m, a_{m+1})$ and $G \subseteq s'$. The majority of the state-of-the-art planners such as HSP [10], Fast Forward [37], Fast Downward [35], and YAHSP [76] are heuristic state-space planners.

Heuristic Plan-Space Planning

In plan-space planning, each state $s \in S$ corresponds to a tuple $\mathcal{T}_s = \langle \mathcal{A}, \mathcal{O}, K \rangle$ such that the sets $\mathcal{A}$, $\mathcal{O}$ and/or K might be missing some actions $a \notin A$, ordering constraints $a_1 \prec a_2 \notin \mathcal{O}$, and/or causal-links $\kappa(a_1, a_2, f) \notin K$ to form a valid POP for Π. Within the heuristic search, the heuristic function $h(s)$ typically approximates the distance between the state $s \in S$ and the goal state G using a weighted sum of the total number of additional elements required to complete the tuple $\mathcal{T}_s$ to a POP [58]. The heuristic plan-space planners iteratively add elements to the sets $\mathcal{A}$, $\mathcal{O}$ and/or K of $\mathcal{T}_s$ with the best $f(s)$ value, until a POP π to Π is found. The heuristic plan-space planners such as VHPOP [79] have not been competitive with the heuristic state-space planners because the distance to

the goal state G can be approximated much more accurately from a complete state $s_c \subseteq F$, compared to a partial state $s_p \subseteq F$ [54].

The next two sections present two powerful methods for obtaining a numerical value for $h(s)$, namely: solving the *delete relaxation* [41] and the extraction of *landmarks* of Π [65, 80, 32].

Delete Relaxation

The *delete relaxation* [13] of the planning problem $\Pi = \langle F, A, I, G \rangle$ transforms Π to Π^{r+} such that all the delete effects of $a \in A$ are removed. The delete relaxed plan π^{r+} is a plan to Π^{r+}, and finding a π^{r+} with the optimal total action cost h^+ is NP-hard [13]. In practice, various approximations to h^+ are experimentally shown to be an extremely powerful tool to guide the heuristic search of a planner for three main reasons [33, 10, 37, 35, 67, 64, 41]. The first reason is the strength of the polynomial approximations, such as h^{add} and h^{max} to the calculation of the optimal h^+ [33]. The calculation of such approximations provides accurate heuristic function $h(s)$ value estimations. The second reason is the extraction of the *useful actions* from a π^{r+}, that is the actions from a delete relaxed plan π^{r+} typically provide actions that are part of some feasible plan π to the original planning problem [37, 41]. The heuristic state-space planners select the next action to be applied to a state $s \in S$ from such useful actions from π^{r+} [37]. The third reason is the utilization of delete relaxed plans π^{r+} in other heuristic-related computations, such as the extraction of landmarks, and the computation of mutually exclusive fluents and actions [65, 37].

Landmarks

A *causal landmark* is a fluent $f \in F$ that must be true in at least one state of any valid plan π [65, 80]. Similarly, an *action landmark* is an action $a \in A$ that must be included at least once in any valid plan π [32]. The extraction of a causal landmark for fluent $f \in F$ can be achieved by proving that the planning problem $\Pi = \langle F, A \backslash pre_f, I \backslash f, G \rangle$ is infeasible. Similarly, the extraction of action landmark for action $a \in A$ can be achieved by proving that the planning problem $\Pi = \langle F, A \backslash a, I, G \rangle$ is infeasible. The parameters used for landmarks are as follows.

- L^F denotes the set of causal landmarks.
- L^A denotes the set of action landmarks.

Complete and sound landmark extraction is known to be PSPACE-complete [65]. Therefore heuristic landmark extraction is usually performed on a relaxation of the original planning problem Π, such as the relaxed-planning graphs [38] and the domain transition graphs of multi-valued state variables [66]. After extraction, the landmarks provide a very powerful tool to guide the heuristic search of a planner for two main reasons [45]. The first reason is the additional information that can be obtained from the ordering of the landmarks [38]. For example, if a fluent is a causal landmark $f \in L^F$ such that there exists an action landmark $a \in (L^A \cup pre_f)$, it must be true that the fluent f must be achieved before the action a is achieved in every plan π. This information is used in the state-of-the-art heuristic planners to decompose the original planning problem Π into much smaller subproblems, such that the goals of the subproblems correspond to the achievement of the ordered set of landmarks [38]. The second reason is the accurate heuristic function $h(s)$ value estimation from different states $s \in S$ of Π [45]. For example, a heuristic estimate on the cost-optimal plan π to Π is obtained from adding up the action costs of all action landmarks plus the minimum cost of adding every causal landmark such that $\sum_{a \in L^A} c_a + \sum_{f \in L^F} min_{a \in add_f}\{c_a\}$ [45]. The state-of-the-art cost-optimal heuristic planners use the landmark extraction and the heuristic function $f(s)$ calculation technique *LM-CUT* [36] to guide their heuristic search [63].

2.3.3 Decomposition Techniques for Classical Planning

Decomposition techniques partition an original problem P into subproblems with substantially smaller complexities than that of P. Chen et al. [15] discuss how the heuristic search and the compilation techniques can be viewed as *search space partitioning*. Heuristic search recursively partitions the search space S into independent subproblems until a feasible solution is found. In contrast, the compilation techniques use solvers (e.g. MILP and SAT solvers) that employ search space partitioning. *Constraint partitioning* decomposes the constraints of the original problem into a conjunction of subproblems where constraints are disjoint, but search spaces may overlap. The two most common methods of decomposing Π into subproblems using constraint partitioning are:

1. *Hierarchical Task Network* (HTN) Planning [28],
2. Partitioning the goals $g \in G$ of Π into separate subproblems (e.g. SGPlan [15], Fragment-based planner [22], etc.).

HTN [28] is a specification for recursively decomposing Π into smaller subtasks until all the subtasks correspond to an action $a \in A$ of Π. Analogous to an action $a \in A$, each subtask has preconditions and effects, and consists of partially ordered actions $a \in A$ and smaller subtasks. HTN-based planners, such as NOAH [71] and NONLIN [74] are among the first planners that have successfully utilized constraint partitioning to solve Π.

An example of a planner that use constraint partitioning on the goals $g \in G$ of Π is Subgoal Partitioning and Resolution in Planning (SGPlan) [15]. SGPlan [15] partitions the goals of Π into independent subproblems, where each subproblem is a smaller planning problem with one subgoal $g \in G$. The master problem of SGPlan evaluates the subplans returned by each subproblem; if the subplans can be executed in parallel, SGPlan terminates with a feasible plan π to Π, otherwise the conflicts between the subplans are resolved through the introduction of the globally violated constraints into the subproblems, and the subproblems are resolved. The globally violated constraints penalize the inconsistencies from the previous solutions and the penalties are incremented until all the inconsistencies are resolved. The experimental results have shown that SGPlan is successful in solving the benchmarks from the International Planning Competitions in 2003 and 2004 [15].

2.4 Mixed-Integer Linear Programming as a Compilation Technique

A MILP represents a combinatorial optimization problem using linear constraints over some finite set of integer and continuous decision variables. This dissertation uses various MILP techniques to solve Π. The three main problem representations that leverage MILP to solve Π are using MILP as compilation, for heuristic search and in decomposition.

The works described below, as presented in Figure 2.4, are grouped with respect to their planning formalism and representation of state transitions:

1. Planning formalism: two representations that are used to describe the classical planning problem: the *propositional representation* (or STRIPS formalism [29]) and *multi-valued representation* (or SAS+ formalism [3]). The propositional representation describes the real world using binary atoms. The multi-valued representation describes the real world using multi-valued, discrete *state variables*.
2. Representation of the state transitions: the state transitions between the steps t and $t + 1$ can be implicit in the constraints of the *state-based formulation*, or can be explicitly represented using a decision variable in the *state-change formulation*.

Organization Of MIP As Compilation Techniques	Propositional Representation	Multi-Valued Representation
State-Based Formulation	Bockmayr et al. [7,8], Vossen et al. [72], Kautz et al. [47]	
State-Change Formulation	Vossen et al. [72], van den Briel et al. [11]	van den Briel et al. [11]

Figure 2.4: Organization of different MILP models that are used in the literature to solve the planning problem as a compilation technique.

2.4.1 Propositional Representation

State-Based Formulations

Bockmayr et al. [7] present the earliest work that uses MILP as a compilation technique. This work and its extension [8] use domain-specific knowledge to find step optimal plans to Π. Given the horizon m, the MILP model uses a state-based formulation to explicitly represent the complete state s_c at the beginning of every step $t \in T(m+1)$ with a set of binary decision variables $y_{f,t}$ for each fluent $f \in F$ such that $y_{f,t} = 1$ if and only if $f \in s_c$. In addition, a set of binary decision variables is used to represent the execution of actions $a \in A$ at step $t \in T(m)$ such that $x_{a,t} = 1$ if and only if $a \in S_t$. The MILP model [7] uses linear constraints to represent the state transitions between steps t and $t+1$ $\forall t \in T(m)$. The objective function of the MILP model maximizes the number of goals met at step $m+1$.

Bockmayr et al. [7, 8] do not present a complete model to solve the classical planning problem. Instead, the authors state that their core MILP model resembles the SAT formulation of the SAT-based planner [47]. The linear constraints and the objective functions encode the domain knowledge of specific problem instances. For example, in an instance of the blocks world domain, the total capacity of the table is limited to 7. The constraint $\sum_{b \in blocks} y_{ontable(b),t} \leq 7 \quad \forall t \in T(m+1)$ represents the fact that over all blocks $b \in block$, at most 7 of them can be simultaneously on the table, where the binary fluent $y_{ontable(b),t} = 1$ if and only if the block $b \in blocks$ is on the table at step $t \in T(m+1)$.

Vossen et al. [77] present the earliest work that uses MILP as a domain-independent compilation technique. Given the horizon m, the work presented two domain-independent MILP formulations: a state-based formulation that is similar to Bockmayr et al. [7, 8], and an alternative state-change formulation that explicitly represents the state transitions between steps t and $t+1$ $\forall t \in T(m)$ with a set of binary decision variables. The state-based formulation uses the same sets of binary decision variables as the domain-dependent state-based formulation

[7, 8]. The objective function and the linear constraints are presented below:

$$\text{minimize} \sum_{a \in A} \sum_{t \in T(m)} x_{a,t}$$

$$y_{f,1} = 1 \qquad \forall f \in I \qquad (2.1)$$

$$y_{f,1} = 0 \qquad \forall f \notin I \qquad (2.2)$$

$$y_{f,m+1} = 1 \qquad \forall f \in G \qquad (2.3)$$

$$x_{a,t} + x_{a,t}^{maintain} \leq y_{f,t} \qquad \forall a \in pre_f, t \in T(m) \qquad (2.4)$$

$$y_{f,t+1} \leq \sum_{a \in add_f} x_{a,t} + x_{a,t}^{maintain} \qquad \forall f \in F, t \in T(m) \qquad (2.5)$$

$$x_{a,t} + x_{a',t} \leq 1 \qquad \forall a, a' \in A, \exists f \in (DEL_a \cap (ADD_f \cup PRE_a)), t \in T(m) \qquad (2.6)$$

Constraints (2.1)-(2.2) describe the initial state I. Constraint (2.3) describes the goal state G. Constraint (2.4) states that if action $a \in A$ is executed at step t, all its preconditions $f \in PRE_a$ must be true in the beginning of step t. Constraint (2.5) states that if fluent $f \in F$ is true at step $t+1$, an action that adds fluent $f \in ADD_a$ must be executed at step t. Note that the formulation includes a dummy 'no-op' decision variable for each step t, $x_{a,t}^{maintain}$, which propagates fluent $f \in F$ between steps t and $t+1$ where $ADD_a = PRE_a = f$. Constraint (2.6) ensures that two *conflicting actions*, that is if one action deletes the precondition or the add effect of the other, cannot be executed in the same time step. The objective function minimizes the total number of non-dummy actions executed in the plan.

State-Change Formulations

Vossen et al. [77] present the first state-change formulation that replaces the state variables with the following set of binary state-change variables:

$$\text{Let } y_{f,t}^{pre-add} = \begin{cases} 1, & \text{if action } a \in pre_f \backslash del_f \text{ is executed at step } t \\ 0, & \text{otherwise.} \end{cases} \qquad \forall f \in F, t \in T(m) \cup \{0\}$$

$$\text{Let } y_{f,t}^{pre-del} = \begin{cases} 1, & \text{if action } a \in pre_f \cap del_f \text{ is executed at step } t \\ 0, & \text{otherwise.} \end{cases} \qquad \forall f \in F, t \in T(m) \cup \{0\}$$

$$\text{Let } y_{f,t}^{add} = \begin{cases} 1, & \text{if action } a \in add_f \backslash pre_f \text{ is executed at step } t \\ 0, & \text{otherwise.} \end{cases} \qquad \forall f \in F, t \in T(m) \cup \{0\}$$

$$\text{Let } y_{f,t}^{maintain} = \begin{cases} 1, & \text{if no action } a \text{ that effects fluent } f \text{ is executed at step } t \\ & \text{such that } a \in add_f \cup del_f \cup pre_f \\ 0, & \text{otherwise.} \end{cases} \qquad \forall f \in F, t \in T(m) \cup \{0\}$$

The objective function and the linear constraints used in the state-change formulation are presented below:

$$\text{minimize} \sum_{a \in A} \sum_{t \in T(m)} x_{a,t}$$

$$y_{f,0}^{add} = 1 \qquad \forall f \in I \qquad (2.7)$$

$$y_{f,0}^{add} = 0 \qquad \forall f \notin I \qquad (2.8)$$

$$y_{f,m}^{add} + y_{f,m}^{pre-add} + y_{f,m}^{maintain} \geq 1 \qquad \forall f \in G \qquad (2.9)$$

$$\sum_{a \in pre_f \backslash del_f} x_{a,t} \geq y_{f,t}^{pre-add} \qquad \forall t \in T(m) \qquad (2.10)$$

$$\sum_{a \in add_f \backslash pre_f} x_{a,t} \geq y_{f,t}^{add} \qquad \forall t \in T(m) \qquad (2.11)$$

$$\sum_{a \in pre_f \cap del_f} x_{a,t} = y_{f,t}^{pre-del} \qquad \forall t \in T(m) \qquad (2.12)$$

$$x_{a,t} \leq y_{f,t}^{pre-add} \qquad \forall a \in pre_f \backslash del_f, t \in T(m) \qquad (2.13)$$

$$x_{a,t} \leq y_{f,t}^{add} \qquad \forall a \in add_f \backslash pre_f, t \in T(m) \qquad (2.14)$$

$$y_{f,t}^{add} + y_{f,t}^{maintain} + y_{f,t}^{pre-del} \leq 1 \qquad \forall f \in F, t \in T(m) \qquad (2.15)$$

$$y_{f,t}^{pre-add} + y_{f,t}^{maintain} + y_{f,t}^{pre-del} \leq 1 \qquad \forall f \in F, t \in T(m) \qquad (2.16)$$

$$y_{f,t}^{pre-add} + y_{f,t}^{maintain} + y_{f,t}^{pre-del} \leq y_{f,t-1}^{add} + y_{f,t-1}^{pre-add} + y_{f,t-1}^{maintain} \qquad \forall f \in F, t \in T(m) \qquad (2.17)$$

Constraints (2.7)-(2.8) describe the initial state I. Constraint (2.9) describes the goal state G. Constraints (2.10) - (2.12) ensure that for each type of state-change on fluent $f \in F$ at step t, at least one action with the corresponding effect and precondition is executed. Constraints (2.13)-(2.14) ensure that if an action is executed at step t, its effects and preconditions are described correctly by the state-change variables. Constraints (2.15)-(2.16) ensure that two conflicting actions cannot be executed in the same step. Constraint (2.17) propagates fluent $f \in F$ between steps t and $t+1$. The objective function minimizes the total number of actions executed in the plan.

The comparison of the run times, the number of problem instances solved, and the number of nodes explored all show significant performance improvement by using the state-change formulation instead of the state-based formulation [77]. Across 13 problem instances tested, the state-based formulation can solve only 4 instances whereas the state-change formulation can solve all of them. In terms of the number of nodes explored and the solution times of the 4 instances that were solved by both formulations, one to two orders of magnitude improvement was observed by using the state-change formulation over the state formulation. The results further show that the linear relaxation values for the state-change formulation are consistently higher compared to the linear relaxation values for the state-based formulation [77]. A pair of fluents are said to be *mutually exclusive* if they cannot be both true in the same state s. It has been further shown experimentally [24] that the exploitation of the domain structure through the analysis of mutually exclusive fluents improves the run time of the state-change formulation.

Resource Extension to Domain-Independent State-Based Formulation

Kautz et al. [48] presented extensions to the core state-based formulation, in terms of set of linear constraints and an alternative objective function, to model the domains with resource requirements [48]. We denote this extended formulation as the state-resource formulation. The additional parameters used in the state-resource formulation are as follows:

- $Prod$ denotes the set of actions that produce the resource.
- $Cons$ denotes the set of actions that consume the resource.
- k_a denotes the resource usage of action $a \in Prod \cup Cons$.
- UB^r denotes the maximum bound on the resource.
- LB^r denotes the minimum bound on the resource.

The additional decision variables used in the state-resource formulation are as follows:

Let $s_t = \begin{cases} 1, & \text{if the resource provider resets the resource to its maximum value } UB^r \text{ at step } t \\ 0, & \text{otherwise.} \end{cases}$ $\forall t \in T(m)$

Let r_t denote the resource level at step t $\forall t \in T(m)$

Let v_t denote the amount of resource created by the resource provider at step t $\forall t \in T(m)$

The objective function and the linear constraints used in the state-resource formulation are presented below:

minimize $r_1 - r_m \sum_{t \in T(m)} v_t$

Constraints $(2.1) - (2.6)$

$$r_{t+1} = r_t + v_t - \sum_{a \in Cons} k_a x_{a,t} + \sum_{a \in Prod} k_a x_{a,t} \qquad \forall t \in T(m-1) \qquad (2.18)$$

$$r_{t+1} \geq UB^r s_t \qquad \forall t \in T(m-1) \qquad (2.19)$$

$$UB^r s_t \geq v_t \qquad \forall t \in T(m) \qquad (2.20)$$

$$x_{a,t} + s_t \leq 1 \qquad \forall a \in A, t \in T(m) \qquad (2.21)$$

$$LB^r \leq r_t - \sum_{a \in Cons} k_a x_{a,t} \qquad \forall t \in T(m) \qquad (2.22)$$

$$UB^r \geq r_t + \sum_{a \in Prod} k_a x_{a,t} \qquad \forall t \in T(m) \qquad (2.23)$$

$$LB^r \leq r_t \leq UB^r \qquad \forall t \in T(m) \qquad (2.24)$$

$$LB^r \leq k_t \leq UB^r \qquad \forall t \in T(m) \qquad (2.25)$$

Constraint (2.18) propagates the value of resource variable between two consecutive steps t and $t+1$ according to the resource usage of each action executed at step t and the resource obtained from the resource provider. Constraint (2.19) ensures that the resource provider resets the resource level to its maximum value UB^r. Constraint (2.20) ensures that no resource is created from the resource provider if the resource provider is not used. Constraint (2.21) states that either the resource provider can be used or an action can be executed at any step. Constraints (2.22)-(2.23) ensure that after the execution of actions, the resource levels are limited by the minimum and the maximum resource bounds respectively. The objective function minimizes the total amount of resource created by the resource provider plus the difference between the resource levels at the first and the last steps.

The experimental results presented by Kautz et al. [48] showed that the state-resource formulation finds plans with better solution quality in terms of minimum resource usage, compared to both state-of-art parallel sequential and sequential planners. The superior solution quality of the state-resource formulation is not surprising as the

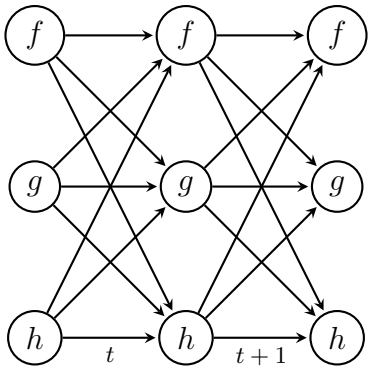

Figure 2.5: Network representation of a state variable $c \in C$ with the domain $D(c) = \{f, g, h\}$ during steps t and $t + 1$ for the Single State-Change Formulation [11].

other planners do not optimize the resource usage in a plan. Kautz et al. [48] further reported that the linear relaxation of the state-resource formulation was not very useful to guide the search of the MILP solver. This result is in-line with the experimental findings of Vossen et al. [77], who demonstrated the state-based formulations perform significantly worse than state-change formulations. In the light of these experimental results, a promising area of future research can be extending the state-change formulations to handle problem domains with resource constraints and objectives.

2.4.2 Multi-valued Representation

A state s in the classical planning problem can be represented using a finite set of multi-valued, discrete state variables C. Each state variable $c \in C$ has a finite domain $D(c) = \{v_1, \ldots, v_k\}$ where each value $v_i \in D(c)$ represents the truth of at most one unique fluent $f \in F$ in the state s. I_c denotes the value of state variable c in the initial state I. $C^* \subseteq C$ denotes the set of state variables whose domain contains a value that represents a goal $g \in G$, and G_c denotes the value of state variable $c \in C^*$ in the goal state G.

The multi-valued representation has two interesting properties. The first property is the representation of mutually exclusive fluents in the domain of each state variable. A value assignment $D(c) = v$ of a state variable $c \in C$ corresponds to the truth of at most one fluent in the state s, and the falsehood of the remaining fluents that are represented by the values in the variable domain $D(c) \backslash \{v\}$. The second property is the explicit representation of the action condition *prevail* on some fluent that must hold true prior and during the execution of that action [3].

The *Single State-Change (1SC) Formulation* [11] represents the planning problem as a set of loosely-coupled network flow problems where each network represents one state variable. As shown in Figure 2.5, each node represents a state variable's value at a given step, t, of the plan, and each arc between the nodes represent the value transition between steps t and $t + 1$ $\forall t \in T(m - 1)$. The goal is to find a step-based plan such that when all the networks are merged, the plan is cost-optimal with respect to m. The $1SC$ formulation uses the following parameters to represent the relationship between the actions and the state variables:

- $SC_{c:f \to g}$ denotes the set of actions that change the value of the state variable c from f to g $\forall f, g \in D(c), c \in C$.
- $Prev_{c:g}$ denotes the set of actions that maintain the value g of the state variable c $\forall g \in D(c), c \in C$.

Similar to the propositional state-change formulation that is presented in Section 2.4.1 [77], the $1SC$ formulation uses binary variables to represent the state transitions between the steps t and $t + 1$, $t \in T(m - 1)$. The $1SC$ formulation uses one set of binary decision variables $y^c_{f,g,t}$ for each state variable $c \in C$ to represent the value

transition of c from $f \in D(c)$ to $g \in D(c)$ between steps t and $t+1$. The objective function and the linear constraints used in the $1SC$ formulation are presented below:

$$\text{minimize} \sum_{a \in A} \sum_{t \in T(m)} x_{a,t}$$

$$\sum_{g \in D(c)} y^c_{I_c,g,1} = 1 \qquad \forall c \in C \tag{2.26}$$

$$\sum_{f \in D(c)} y^c_{f,G_c,m} = 1 \qquad \forall c \in C^* \tag{2.27}$$

$$\sum_{f \in D(c)} y^c_{f,g,t} = \sum_{h \in D(c)} Y^c_{g,h,t+1} \qquad \forall f, g \in D(c), c \in C, t \in T(m-1) \tag{2.28}$$

$$\sum_{a \in SC_{c:f \to g}} x_{a,t} = y^c_{f,g,t} \qquad \forall f \neq g \in D(c), c \in C, t \in T(m) \tag{2.29}$$

$$x_{a,t} \leq y^c_{g,g,t} \qquad \forall g \in D(c), c \in C, a \in Prev_{c:g}, t \in T(m) \tag{2.30}$$

Constraint (2.26) describes the initial state I by ensuring that every state variable $c \in C$ takes the correct value from their domain $D(c) = I_c$ in the initial state I. Constraint (2.27) describes the goal state G by ensuring that the state variables $c \in C^*$ that describe the goal state G take the correct values from their domain $D(c) = G_c$. Constraint (2.28) preserves the network-flow on every value $f \in D_c$ of each state variable $c \in C$ for each step $t \in T(m)$. Constraint (2.29) ensures that if there is a state transition from $f \in D_c$ to $g \in D_c$ at step $t \in T(m)$, an action with the corresponding precondition and effect must be executed. Constraint (2.30) ensures that if an action with a prevail condition $a \in Prev_{c:g}$ is executed at step $t \in T(m)$, the value $g \in D(c)$ must propagate between steps t and $t+1$ $t \in T(m-1)$. The objective function minimizes the total number of actions executed in the plan.

In addition to the $1SC$ formulation, two different formulations that use more relaxed notions of parallel actions have been presented [11]. The $1SC$ formulation uses the notion of parallel actions, that is, a set of actions can be executed in parallel if any linearization of the actions is a valid subplan that leads to the same complete state s_c. Given the set of state variables C, the $1SC$ formulation allows for either $a_1 \in SC_{c:f \to g}$ or $a_2 \in Prev_{c:h}$ to be executed for each state variable $c \in C$ per step. The generalized one-state-change, $G1SC$, formulation relaxes the notion of parallelism to 'there exists some feasible linearization of parallel actions', and allows for actions a_1 and a_2 to be executed in the same step if $h = f$ or $h = g$. Finally the k-state-change, kSC, formulation allows each value of a state variable to transition from one unique value to another unique value at most k times per step. Both the $G1SC$ and kSC formulations can produce plans with cyclic state transitions. The authors use a branch-and-cut algorithm to solve the planning problem where cuts are dynamically added to remove cyclic solutions.

Experimental results have shown that the $1SC$ formulation performed competitively with the winner of the International Planning Competition in 2004 [46] in terms of run time and plan horizon comparisons. The overall comparison of $1SC$, $G1SC$ and kSC formulations shows that run times are inversely correlated with both the plan horizon required to find a feasible solution, and the number of cycle elimination constraints that must be generated for cyclic solutions. The experimental results showed that the $G1SC$ formulation performed the best overall, and can be viewed as a compromise between the $1SC$ and kSC formulations [11].

2.5 Mixed-Integer Linear Programming for Heuristic Search

In heuristic search, MILP has been used to obtain a numerical value for $h(s)$ and to select which state $s \in S$ to explore next. This section describes the MILP formulations that are used to calculate $h(s)$.

2.5.1 Step Optimal Heuristic Planning

The earliest use of MILP to solve planning problems is in heuristic search [14]. Bylander [14] introduced a step-based model to calculate an admissible heuristic for step optimal plans. In addition to the action selection $x_{a,t}$ and state-based $y_{f,t}$ variables, a set of binary decision variables $x^f_{a,t}$ is introduced for each precondition $f \in PRE_a$ of every action $a \in A$ at every step $t \in T(m)$ such that $x^f_{a,t} = 1$ if the effect of the action $a \in A$ is already satisfied before its execution, and $x^f_{a,t} = 0$ otherwise. The objective function and the linear constraints that are used [14] to solve Π_m are as follows:

$$\text{maximize} \sum_{f \in G} y_{f,m+1}$$

$$y_{f,1} = 1 \qquad \forall f \in I \quad (2.31)$$

$$y_{f,1} = 0 \qquad \forall f \notin I \quad (2.32)$$

$$\sum_{a \in A} x_{a,t} \leq 1 \qquad \forall t \in T(m) \quad (2.33)$$

$$y_{f,t+1} = y_{f,t} + \sum_{a \in add_f} x_{a,t} - \sum_{a \in add_f} x^f_{a,t} - \sum_{a \in del_f} x_{a,t} + \sum_{a \in del_f} x^f_{a,t} \qquad \forall f \in F, t \in T(m) \quad (2.34)$$

$$x^f_{a,t} \leq x_{a,t} \qquad \forall f \in PRE_a, a \in A, t \in T(m) \quad (2.35)$$

$$y_{f,t} \geq \sum_{a \in pre_f} x_{a,t} + \sum_{a \in add_f} x^f_{a,t} - \sum_{a \in del_f} x^f_{a,t} \qquad \forall f \in F, t \in T(m) \quad (2.36)$$

$$1 - y_{f,t} \geq \sum_{a \in add_f} x_{a,t} - \sum_{a \in add_f} x^f_{a,t} + \sum_{a \in del_f} x^f_{a,t} \qquad \forall f \in F, t \in T(m) \quad (2.37)$$

Constraints (2.31)-(2.32) describe the initial state I. Constraint (2.33) ensures that at most one action can be executed at a step. Constraint (2.34) ensures the transition of the fluent values between steps t and $t+1$. Constraint (2.35) ensures that the decision variable $x^f_{a,t}$ can only take the value of 1 only if the action selection variable $x_{a,t}$ has the value of 1. Constraints (2.36)-(2.37) ensure that the preconditions of an action are true at the step an action is executed. Finally the objective function maximizes the number of goals satisfied by the MILP formulation.

Bylander implements a modification of this MILP model that allows for actions $a \in A$ to be partially-ordered. Unfortunately the MILP formulation that is presented [14] is specific to a problem instance, and does not provide sufficient information on how it can be generalized to solve any Π_m. This implementation solves the linear-relaxation of the modified MILP model at every state $s \in S$ of the heuristic search. The results show good approximation to the optimal plan horizon m, however also show that it is computationally expensive to solve the linear relaxation.

2.5.2 Cost-Optimal Heuristic Planning

Operator counting constraints [64] count the number of actions in some relaxed representation of a plan using one continuous decision variable z_a for every action $a \in A$. A number of different planning heuristics, including

landmarks [65, 80, 32], the delete relaxation [41], and net state change constraints [9] have been incorporated into this heuristic.

Delete Relaxation and Landmarks

Imai and Fukunaga [41] introduced a MILP model that finds a cost optimal delete-relaxed plan to Π^{r+}, denoted as H^+. The parameters used in H^+ are as follows.

- $I_f = 1$ if fluent $f \in I$, $I_f = 0$ otherwise.

- inv_a is the set of inverse actions of action $a \in A$, such that $a' \in inv_a$ if and only if $ADD_{a'} \subseteq PRE_a$ and $ADD_a \subseteq PRE_{a'}$.

H^+ assigns actions and fluents to steps in order to represent the causal links between two actions. The decision variables used in H^+ are as follows.

$$\text{Let } u_f = \begin{cases} 1, & \text{if fluent } f \text{ is added by } \pi^{r+} \\ 0, & \text{otherwise.} \end{cases} \qquad \forall f \in F$$

$$\text{Let } u_a = \begin{cases} 1, & \text{if action } a \text{ is used in } \pi^{r+} \\ 0, & \text{otherwise.} \end{cases} \qquad \forall a \in A$$

Let T_a denote the step at which action a is executed. $T_a \in T(|A| - 1) \quad \forall a \in A$.

When $T_a = |A| - 1$, action a is not in π^{r+}.

Let T_f denote the step at which fluent f is added. $T_f \in T(|A|) \quad \forall f \in F$.

When $T_f = 0$, the fluent f is not added by π^{r+}.

$$\text{Let } x_{a,f} = \begin{cases} 1, & \text{if action } a \text{ adds fluent } f \text{ that is required by} \\ & \text{the goal state } G, \text{ or by another action} \\ 0, & \text{otherwise.} \end{cases} \qquad \forall f \in add_f, a \in A$$

The constraints and the objective function used in H^+ are as follows.

$$\text{minimize} \sum_{a \in A} c_a z_a$$

$$u_f = 1 \qquad \forall f \in G \tag{2.38}$$

$$u_f \geq u_a \qquad \forall f \in PRE_a, a \in A \tag{2.39}$$

$$u_a \geq x_{a,f} \qquad \forall f \in ADD_a, a \in A \tag{2.40}$$

$$I_f + \sum_{a \in add_f} x_{a,f} = u_f \qquad \forall f \in F \tag{2.41}$$

$$T_f \leq T_a \qquad \forall f \in PRE_a, a \in A \tag{2.42}$$

$$T_a + 1 \leq T_f + |A + 1|(1 - x_{a,f}) \qquad \forall f \in ADD_a, a \in A \tag{2.43}$$

$$u_f - \sum_{a' \in add_f \cap inv_a} x_{a',f} \geq u_a \qquad \forall f \in PRE_a, a \in A \tag{2.44}$$

$$u_f = 1 \qquad \forall f \in L^F \tag{2.45}$$

$$u_a = 1 \qquad \forall a \in L^A \tag{2.46}$$

$$u_a \leq z_a \qquad \forall a \in A \tag{2.47}$$

Constraint (2.38) ensures that all goal fluents are added by π^{r+}. Constraint (2.39) ensures that all preconditions of all the actions are satisfied. Constraint (2.40) states that if action a adds fluent f required by another action or the goal state, action a has to be in π^{r+}. Constraint (2.41) states that if a fluent is added by π^{r+}, it must either be true in the initial state, or added by some action. Constraint (2.42) ensures that every precondition of action a is true by the step it is executed in. Constraint (2.43) ensures that when an action adds a fluent to the plan, the fluent is true after the execution step of the action. Constraint (2.44) is the inverse action constraint, which states that action a cannot add fluent f required by another action a' if a' adds fluent g required by a. Constraints (2.45) and (2.46) are causal and action landmark constraints. Constraint (2.47) incorporates H^+ into the operator counting heuristic [64] by ensuring that if an action is used in π^{r+}, the decision variable z_a takes the value of at least 1. The objective function minimizes the total action cost of π^{r+}.

Imai and Fukunaga [41, 42] compare the performance of the H^+ formulation, to the state-of-the-art delete relaxation planner (HST) [34] for the computation of the optimal total action cost h^+ over 1,300 International Planning Competition problem instances. Given a 15-minute time limit, the H^+ formulation solves more delete relaxed problems in less computation time. Imai and Fukunaga further implement the H^+ formulation in an A*-search planner to solve cost-optimal planning problems. The experimental results show that solving the linear relaxation of the H^+ formulation performs significantly better than solving the original H^+ formulation.

Net Change Constraints

Net change constraints [9] are a set of linear constraints that bound the number of times a set of actions can be used in any valid plan. These constraints group actions $a \in A$ into four different sets, namely: *always produce*, *sometimes produce*, *always consume* and *sometimes consume*, for each fluent $f \in F$. The parameters used in the net change constraints are as follows.

- M_f denotes the set of fluents that are mutually exclusive with fluent $f \in F$.
- AP_f denotes the set of actions that always produce fluent $f \in F$ such that action $a \in AP_f$ if and only if

$f \in ADD_a$ and there exists a fluent $g \in PRE_a \cap M_f$.

- SP_f denotes the set of actions that sometimes produce fluent $f \in F$ such that action $a \in SP_f$ if and only if $f \in ADD_a$ and there does not exist a fluent $g \in PRE_a \cap M_f$.

- AC_f denotes the set of actions that always consume fluent $f \in F$ such that action $a \in AC_f$ if and only if $f \in DEL_a \cap PRE_a$.

- SC_f denotes the set of actions that sometimes consume fluent $f \in F$ such that action $a \in SC_f$ if and only if $f \in DEL_a \backslash PRE_a$.

$$\bullet\langle L_f, U_f\rangle = \begin{cases} \langle -1,-1\rangle, & \text{if fluent } f \in I \text{ and there exists fluent } g \in G \cap M_f. \\ \langle -1,0\rangle, & \text{if fluent } f \in I \backslash G \text{ and there does not exist fluent } g \in G \cap M_f. \\ \langle 1,1\rangle, & \text{if fluent } f \in G \backslash I. \\ \langle 0,1\rangle, & \text{if fluent } f \notin I \cup G. \\ \langle 0,0\rangle, & \text{otherwise.} \end{cases} \qquad \forall f \in F$$

The linear constraints used in the net change constraints are as follows.

$$\sum_{a \in AP_f} z_a + \sum_{a \in SP_f} z_a - \sum_{a \in AC_f} z_a \geq L_f \qquad \forall f \in F \tag{2.48}$$

$$\sum_{a \in AP_f} z_a - \sum_{a \in AC_f} z_a - \sum_{a \in SC_f} z_a \leq U_f \qquad \forall f \in F \tag{2.49}$$

Constraints (2.48)-(2.49) reason about the number of times the fluent $f \in F$ is added and deleted by the plan π. Constraint (2.48) ensures that for each action $a_2 \in AC_f$, there must exists a unique action $a_1 \in add_f$ for the causal-link $\kappa(a_1, a_2, f) \in K$. Constraint (2.49) ensures that for each action $a \in AP_f$, there must exists a unique action that deletes f because a has at least one precondition $g \in PRE_a$ that is mutually exclusive with f such that $g \in M_f$.

Pommerening et al. [64] incorporated both individual and different combinations of operator counting constraints into the A*-search configuration of Fast Downward [35]. The performance was tested over 1300 problem instances from the International Planning Competition domains of 2008 and 2011. Pommerening et al. reported that the best performing combination of the linear constraints are the net state change constraints and the landmark constraints. This configuration is reported to solve 25 more problem instances than the state-of-art heuristic planner that use LM-Cut.

2.6 Mixed-Integer Linear Programming in Decomposition Techniques

This section of the thesis describes the use of MILP in different decomposition techniques. The works described in the next section decompose Π with respect to its goals $g \in G$ [22] using *column generation*, or decompose the procedure for finding π to Π by separating the selection and the ordering of actions $\mathcal{A}$ using *logic-based Benders decomposition* [21].

2.6.1 Column Generation

Davies et al. [22] introduced a column generation decomposition to find a step optimal plan to Π. Given a horizon m, the master problem selects a set of *subplans* that consist of a set of actions executed across different steps $t \in T(m)$; each satisfying at least one goal $g \in G$ of Π, with minimum total action cost. Each subproblem (i.e. one subproblem for each $g \in G$) generates a subplan that minimizes the cost of achieving $g \in G$. Davies et al. [22] use column generation to solve a train scheduling problem, and give a brief introduction on how it can be generalized for solving the classical planning problems.

Master Problem

The master problem is a step-based MILP formulation that selects subplans to satisfy all goals $g \in G$ of Π, with minimum total action cost. The parameters used in the master problem are as follows.

- P denotes the set of subplans.
- P_g denotes the set of subplans that satisfy goal $g \quad \forall g \in G$.
- c_p denotes the total action cost of subplan $p \quad \forall p \in P$.
- M denotes a large numerical constant.
- $u_{p,f,t} = \{0, 1\}$ denotes the usage of fluent f at step t by subplan $p \quad \forall p \in P, f \in F, t \in T(m) \cup \{0\}$.

Davies et al. [22] do not provide the exact definition of the parameter $u_{p,f,t}$ for solving the classical planning problem Π, which is used to ensure that a set of subplans can be executed in parallel. The decision variables used in the master problem are presented below:

$$\text{Let } x_p = \begin{cases} 1, & \text{if subplan } p \text{ is executed.} \\ 0, & \text{otherwise.} \end{cases} \qquad \forall p \in P$$

$$\text{Let } v_g = \begin{cases} 1, & \text{if goal } g \text{ is not satisfied.} \\ 0, & \text{otherwise.} \end{cases} \qquad \forall g \in G$$

The objective function and the linear constraints used in the master problem are presented below:

$$\text{minimize} \sum_{p \in P} c_p x_p + \sum_{g \in G} M v_g$$

$$\sum_{p \in P} u_{p,f,0} x_p \leq I_f \qquad \forall f \in F \tag{2.50}$$

$$\sum_{p \in P} u_{p,f,m} x_p \leq -1 \qquad \forall f \in G \tag{2.51}$$

$$\sum_{p \in P} u_{p,f,m} x_p \leq 0 \qquad \forall f \notin G \tag{2.52}$$

$$\sum_{p \in P} u_{p,f,t} x_p \leq 0 \qquad \forall f \in F, t \in T(m-1) \tag{2.53}$$

$$v_g + \sum_{p \in P_g} x_p = 1 \qquad \forall g \in G \tag{2.54}$$

Constraints (2.50)-(2.53) ensure that subplans that are selected are executable in parallel at each step $t \in T(m) \cup \{0\}$. Constraint (2.54) ensures that either each goal $g \in G$ is satisfied or the dummy variable v_g takes the value of 1. The objective function first maximizes the number of goals $g \in G$ satisfied, and then minimizes the total action cost of the selected subplans.

Subproblem

Each subproblem generates a subplan that satisfies at least one goal $g \in G$ with minimum total action cost. The action cost of each action at each step is updated after solving the master problem with respect to the *dual prices* of Constraints (2.50)-(2.53), that correspond to the optimal values of the decision variables in the equivalent dual problem. Davies et al. [22] solve the subproblems using a temporal Golog implementation [50].

Davies et al. [22] report preliminary results on the multi-agent version of the *Blocksworld domain* (i.e. there are as many as grippers as there are blocks). Experimental results show that the column generation performed competitively with other temporal planners in terms of finding the first feasible plan, and the best in terms of proving the optimality.

2.6.2 Logic-based Benders decomposition

Davies et al. [21] introduced a logic-based Benders decomposition (LBBD) to find a cost optimal sequential plan to Π. The master problem of the LBBD is a MILP model that combines several linear constraints from the operator counting constraints [64] to generate a set of actions $\mathcal{A}$ with minimum total action cost. Given $\mathcal{A}$, the subproblem is a SAT encoding that either finds a sequential plan π if the subproblem is feasible, or returns a cut to the master problem if the subproblem is infeasible. The overall LBBD presented by Davies et al. [21] is as follows:

Algorithm 1 The algorithmic description of the LBBD presented by Davies et al. [21]

```
Set Lower Bound Value LB = 0
Set Upper Bound Value UB = ∞
Set Incumbent POP π* = ∅
while LB < UB do
    Solve the master problem
    if The master problem is infeasible or |A| > 2^|F| then
        Terminate
    else
        Solve the subproblem
        if The subproblem is infeasible then
            Add the generalized-landmark constraint associated with A to master problem
        else
            Let π = ⟨A*, O, K⟩ denote the POP returned by the subproblem.
            Set π* = π
            Set LB = UB = c(π)
            Terminate
        end if
    end if
end while
```

Master Problem

The master problem is a MILP formulation that is based on three operator counting constraints, namely H^+ [41], net state change constraints [9], and landmark constraints. Similar to the operator counting constraints [64], this formulation uses one continuous decision variable z_a for every action $a \in A$. In addition, the master problem incorporates cuts (or *generalized landmarks*) returned from the subproblem if the subproblem is infeasible. The generalized landmarks are in the form of: $\sum_{a \in A}[z_a \geq C(a) + 1] \geq 1$ and consist of a set of bounds on every action $a \in A$ that are in the form of: $[z_a \geq C(a)+1]$, where $C(a)$ is the value of the decision variable z_a from the last optimal solution of the master problem. Intuitively, each generalized landmark is feasible by the satisfaction of at least one of its bounds, meaning that a plan requires at least one more action to be returned by the master problem. Instead of adding a generalized landmark with the complete set of bounds on every action $a \in A$, Davies et al. [21] use Conflict-Directed Clause Learning, which is a conflict analysis method for finding the minimal set of bounds that have caused the infeasibility of the subproblem [27].

Subproblem

The subproblem is a SAT encoding of a state-based sequential planner that either returns a sequential plan using the operator counts $C_a \; \forall a \in A$ from the solution of the master problem, or a new set of generalized landmarks. The SAT model uses the multi-valued representation with the addition of the following parameters:

- $Pre_{a:c} \in D(c)$ denotes the precondition value of action a on state variable $c \quad \forall a \in A, c \in C$.
- $Post_{a:c} \in D(c)$ denotes the execution effect of action a on state variable $c \quad \forall a \in A, c \in C$.

In addition to the binary action selection variable $x_{a,t}$, $\forall a \in A, t \in T(|\mathcal{A}|)$, the SAT model uses the binary decision variable $y^c_{f,t}$, $\forall f \in D(c), c \in C, t \in T(|\mathcal{A}|)$ that is true if the state variable $c \in C$ takes the value $f \in D(c)$ at step $t \in T(|\mathcal{A}|)$, and is false otherwise. We follow the same notation that has been used by Davies et al. [21] to represent the at-most-k constraint, that is denoted as $\leq_k$. Given a set of literals, this constraint enforces that at most k literals can be simultaneously true. The core SAT model used by Davies et al. [21] is as follows:

$$(\{x_{a,t} | a \in A\}) \leq_1 \qquad \forall t \in T(|\mathcal{A}|) \tag{2.55}$$

$$(\{y^c_{f,t} | f \in D(c)\}) \leq_1 \qquad \forall c \in C, t \in T(|\mathcal{A}|) \tag{2.56}$$

$$y^c_{I_c,1} \qquad \forall c \in C \tag{2.57}$$

$$\bigwedge_{Pre_{a:c} \in D(c), c \in C} (\neg x_{a,t+1} \vee y^c_{Pre_{a:c},t}) \qquad \forall a \in A, t \in T(|\mathcal{A}| - 1) \tag{2.58}$$

$$\bigwedge_{Post_{a:c} \in D(c), c \in C} (\neg x_{a,t} \vee y^c_{Post_{a:c},t}) \qquad \forall a \in A, t \in T(|\mathcal{A}|) \tag{2.59}$$

$$y^c_{f,t+1} \rightarrow y^c_{f,t} \bigvee_{a \in SC_{c:g \rightarrow f} \forall g \neq f \in D(c)} x_{a,t} \qquad \forall c \in C, t \in T(|\mathcal{A}| - 1) \tag{2.60}$$

$$y^c_{G_c,|\mathcal{A}|} \vee [\sum_{a \in A} C(a) \geq |\mathcal{A}| + 1] \qquad \forall c \in C^* \tag{2.61}$$

$$(\{x_{a,t} | t \in T(|\mathcal{A}|)\}) \leq_{C(a)} \vee \, [z_a \geq C(a) + 1] \qquad \forall a \in A, t \in T(|\mathcal{A}|) \tag{2.62}$$

Clause (2.55) ensures that at each step, at most one action can be executed. Clause (2.56) ensures that each state variable can take at most one value from its domain at each step. Clause (2.57) ensures that initial state is set correctly for every state variable. Clause (2.58) and (2.59) make sure that if an action is executed, its preconditions

and effects are set, respectively. Clause (2.60) either propagates the value of a state variable, or ensures that an action with the proper effect is executed in the previous step. Clause (2.61) states that all goals of Π are satisfied. Clause (2.62) states that for each action $a \in A$, the total number of actions of a executed across all the steps $t \in T(|\mathcal{A}|)$ is less than or equal to the bound $C(a)$. Additionally, Davies et al. [21] use assumptions that are the negation of the bounds $\neg[\sum_{a \in A} z_a \geq \sum_{a \in A} C(a) + 1]$, and $\neg[z_a \geq C(a) + 1]$. These assumptions are used in the conflict analysis to find whether the infeasibility of the SAT model is due to the lack of additional steps or additional actions.

Davies et al. [21] reported the results on a modification of their LBBD that use Branch-and-Check $OpSeq$ instead of LBBD. The three main differences between the LBBD and $OpSeq$ are as follows. First, $OpSeq$ solves the linear relaxation of the master problem. Every time an optimal solution is found to the LP, the set of operator counts are rounded-up to the nearest integer values. Second, the subproblem is solved only if the sum of rounded-up operator counts are within 20% of the sum of linear operator counts. $OpSeq$ is proposed by Davies et al. [21] to reduce the large amount of time spent in solving the master problem of LBBD to optimality. The analysis of the experimental results show that $OpSeq$ is successful in finding high quality dual scores (i.e., good quality lower bounds on the total action cost of a plan). However, the inspection of the number of problems solved show that $OpSeq$ is not competitive with the with the winner of the International Planning Competition in 2014 for the Optimal Track $SymBA^*$-2 [75].

2.7 Partial-Order Planning

A POP is a plan that imposes only action orderings necessary for achieving a goal, as opposed to a total ordering of actions as enforced in sequential planning. Equivalently, a POP represents a set of linearizations all including the same actions but under different orderings. POPs provide flexibility to agents, who can dynamically commit to the sequence of actions during the real-time execution of the plan [56]. Least commitment planning aims to generate POPs that do not impose unnecessary ordering constraints between the pairs of actions [51]. Least commitment planning is used to generate POPs inside:

1. A POP planner that adds an ordering constraint between a pair of actions only if the ordering constraint removes a threat from the tuple $\langle \mathcal{A}, \mathcal{O}, K \rangle$ [61, 79], or

2. A post-processing model that generates a POP given an initial plan, an approach that is also known as *partial-order relaxation* [1, 44, 56].

This section presents these two methods of generating POPs using the least commitment planning.

2.7.1 Partial-Order Planners

The traditional method for generating a POP is called *partial-order causal link* (POCL) planning [78]. The original POCL planner, UCPOP [61], performs complete search in plan-space. Unlike in heuristic search, the search systematically enumerates all states $s \in S$ instead of evaluating and selecting the states with the best heuristic function $h(s)$ value. The integration of the heuristic function $h(s)$ into POCL planning produced a more efficient POP planner, VHPOP [79]. Neither UCPOP [61] nor VHPOP [79] are competitive with the state-of-the-art heuristic state-space planners [10, 37, 35, 17, 76].

An alternative method for generating a POP is using the state-space heuristic calculations in POP planning. POPF [17] is a POP planner that restricts the allowable expansion on the current state $s \in S$ to adding new actions

ordered at the end of the plan with causal links. Unlike the state-space planners, POPF is not required to order all pairs of actions. When a new action $a_1 \in A$ is added to the tuple $\langle \mathcal{A}, \mathcal{O}, K \rangle$, POPF ensures that all preconditions of a_1, $f \in PRE_{a_1}$, are supported with the causal links $\kappa(a_2, a_1, f)$ such that there exists actions $a_2 \in \mathcal{A}$ that add the preconditions of a_1 i.e., $f \in PRE_{a_1}$. In addition, POPF introduces an ordering constraint between a_1 and $a_3 \in \mathcal{A}$ only if $a_3 \prec a_1$ removes a threat from the updated tuple $\langle \mathcal{A} \cup \{a_1\}, \mathcal{O}, K \rangle$. As a result, POPF pertains a partially-ordered set of actions that can be easily linearized so that a strong heuristic function $h(s)$ value from the complete state $s = s_c$ can be calculated [54].

2.7.2 Partial-Order Relaxation

A least commitment POP introduces an ordering constraint between a pair of actions only if the ordering constraint contributes to the validity of the POP. Since the ordering constraints are defined between the pairs of actions, it is common to generate POPs according to the least commitment planning principle given a set $\mathcal{A}$ from an initial plan $\pi = \langle \mathcal{A}, \mathcal{O} \rangle$ [1, 44, 26, 56, 57] (i.e., a partial-order relaxation). This section presents different objectives and the MaxSAT models [56, 57] for partial-order relaxation given $\mathcal{A}$.

The earliest work on producing plans using partial-order relaxation investigates the theoretical complexity of removing ordering constraints from an initial plan, and reorganizing the ordering structure of an initial plan, namely *optimal deordering* and *optimal reordering* of a plan [1]. This work assumes that the action precedence graph is transitively closed, that is $a_1 \prec a_3 \in \mathcal{O}$ if $a_1 \prec a_2, a_2 \prec a_3 \in \mathcal{O}$.

Definition 1. *(Optimal Deordering of a Plan [1]). Let $P = \langle \mathcal{A}, \mathcal{O}, K \rangle$ and $P' = \langle \mathcal{A}, \mathcal{O}', K' \rangle$ be two valid POPs for a planning problem Π. P' is a optimal deordering of a plan P if and only if*

(i) $\mathcal{O}' \subseteq \mathcal{O}$; and

(ii) For all valid POPs $P'' = \langle \mathcal{A}, \mathcal{O}'', K'' \rangle$, the number of ordering constraints of P' is less than or equal to the number of ordering constraints of P'' such that $|\mathcal{O}'| \leq |\mathcal{O}''|$.

Definition 2. *(Optimal Reordering of a Plan [1]). Let $P = \langle \mathcal{A}, \mathcal{O}, K \rangle$ and $P' = \langle \mathcal{A}, \mathcal{O}', K' \rangle$ be two valid POPs for a planning problem Π. P' is a optimal reordering of a plan P if and only if for all valid POPs $P'' = \langle \mathcal{A}, \mathcal{O}'', K'' \rangle$, the number of ordering constraints of P' is less than or equal to the number of ordering constraints of P'' such that $|\mathcal{O}'| \leq |\mathcal{O}''|$.*

Both definitions aim to decrease the number of ordering constraints in a plan. The difference between finding the optimal deordering and optimal reordering of a plan is that deordering only allows the removal of ordering constraints from $\mathcal{O}$ while reordering can introduce new ordering constraints to $\mathcal{O}$. The theoretical complexity of finding the optimal deordering or optimal reordering of a plan is NP-hard [1].

In temporal planning, a MILP model [26] has been introduced to relax the ordering structure of an initial plan. In the problem that is being solved, the initial plan is a temporal plan with its actions fixed to specific time points, and the objective either minimizes the total number of ordering constraints, or maximizes a measure of temporal flexibility of the resulting *order constrained plan*. The definition of order constrained plan is similar to a parallel POP except that an order constrained plan orders a pair of non-concurrent actions $a_1, a_2 \in \mathcal{A}$ with an ordering constraint $a_1 \prec a_2 \in \mathcal{O}$ or $a_2 \prec a_1 \in \mathcal{O}$ instead of the non-concurrency relation $a_1 \perp a_2 \in \#$, and, the action precedence graph of an order constrained plan is not transitively closed. The experimental results have demonstrated that on average, the makespan of an initial temporal plan can be decreased by 40% as a result of producing an order constrained plan with the objective of minimizing its makespan [26].

In partial order planning, a MaxSAT model [56, 57] has been introduced to generate *Minimum Cost Least Commitment POPs*, a weighted combination of cost optimality and optimal reordering objectives.

Definition 3. *(Minimum Cost Least Commitment POP (MCLCP) [56]). Let $P = \langle \mathcal{A}, \mathcal{O}, K \rangle$ and $P' = \langle \mathcal{A}', \mathcal{O}', K' \rangle$ be two valid POPs for a planning problem Π where $\mathcal{A}' \subseteq \mathcal{A}$. Moreover, let c_a be a non-negative cost associated with each action $a \in \mathcal{A}$. P' is a least commitment flexible POP (MCLCP) of a plan P iff*

(i) For all valid POPs with action set $\mathcal{A}'' \subseteq \mathcal{A}$, we have $\sum_{a \in \mathcal{A}''} c_a \geq \sum_{a \in \mathcal{A}'} c_a$; and

(ii) For all valid POPs P'' with action set $\mathcal{A}'' \subseteq \mathcal{A}$ and $\sum_{a \in \mathcal{A}''} c_a = \sum_{a \in \mathcal{A}'} c_a$, $|\mathcal{O}'| \leq |\mathcal{O}''|$.

Given a set of actions $\mathcal{A}$, Muise et al. [56] introduced a MaxSAT model that selects and orders actions to find a MCLCP. The parameters used in the MaxSAT model are as follows.

- a_I is a special action that represents the initial state I such that $ADD_{a_I} = I$.
- a_G is a special action that represents the goal state G such that $PRE_{a_G} = G$.

The decision variables used in the MaxSAT model are as follows, where 1 represents true and 0 represents false.

$$\text{Let } X^f_{a_i,a_j} = \begin{cases} 1, & \text{if action } a_i \text{ supports fluent } f \text{ for} \\ & \text{action } a_j \text{with the causal link} \\ & \kappa(a_i, a_j, f) \in K \\ 0, & \text{otherwise.} \end{cases} \qquad \forall a_i, a_j \in \mathcal{A}, f \in ADD_{a_i} \cap PRE_{a_j}$$

$$\text{Let } O_{a_i,a_j} = \begin{cases} 1, & \text{if action } a_i \text{ is ordered before action} \\ & a_j \text{ with an ordering constraint} \\ 0, & \text{otherwise.} \end{cases} \qquad \forall a_i, a_j \in \mathcal{A}$$

$$\text{Let } Z_a = \begin{cases} 1, & \text{if action } a \text{ is selected} \\ 0, & \text{otherwise.} \end{cases} \qquad \forall a \in \mathcal{A}$$

The hard clauses used in the MaxSAT model are as follows.

$$Z_{a_j} \rightarrow \bigwedge_{f \in PRE_{a_j}} \bigvee_{a_i \in add_f} X^f_{a_i,a_j} \wedge O_{a_i,a_j} \qquad \forall a_j \in \mathcal{A} \tag{2.63}$$

$$X^f_{a_i,a_j} \rightarrow \bigwedge_{a_d \in del_f} (Z_{a_d} \rightarrow O_{a_d,a_i} \vee O_{a_j,a_d}) \qquad \forall a_i, a_j \in \mathcal{A}, f \in PRE_{a_j} \cap ADD_{a_i} \tag{2.64}$$

$$O_{a_i,a_j} \rightarrow Z_{a_i} \wedge Z_{a_j} \qquad \forall a_i, a_j \in \mathcal{A} \tag{2.65}$$

$$Z_a \rightarrow O_{a_I,a} \wedge O_{a_I,a} \qquad \forall a \in \mathcal{A} \tag{2.66}$$

$$Z_{a_I} \wedge Z_{a_G} \tag{2.67}$$

$$\neg O_{a,a} \qquad \forall a \in \mathcal{A} \tag{2.68}$$

$$O_{a_i,a_j} \wedge O_{a_j,a_k} \rightarrow O_{a_i,a_k} \qquad \forall a_i, a_j, a_k \in \mathcal{A} \tag{2.69}$$

Hard clause (2.63) ensures that if an action is selected, all of its preconditions are met at least once. Hard clause (2.64) orders the threatening actions either before or after the causal-link. Hard clause (2.65) makes sure that the actions in an enforced ordering constraint are included in the plan. Hard clause (2.66) constrains all

the included actions to be between the initial and the goal actions. Hard clause (2.67) includes the initial and goal actions. Hard clause (2.68) disallows self-loops for ordering constraints. Hard clauses (2.68)-(2.69) together forbid cycles in the action precedence graph. The model grows cubically with the size of the action set $\mathcal{A}$ due to hard clause (2.69). In addition to the hard clauses (2.63)-(2.69), the soft clauses used in the MaxSAT model are as follows.

$$w(\neg Z_a) = c_a + |\mathcal{A}| \qquad \forall a \in \mathcal{A} \tag{2.70}$$

$$w(\neg O_{a_i,a_j}) = 1 \qquad \forall a_i, a_j \in \mathcal{A} \tag{2.71}$$

Soft clauses (2.70)-(2.71) are weighted such that the model first minimizes the total action cost, then the total number of ordering constraints in a cost optimal POP w.r.t. the action set $\mathcal{A}$.

The MaxSAT model [55, 56, 57] is compared against a polynomial-time deordering algorithm KK [44] on six different domains from the International Planning Competition in 2003. The initial sequential plans are obtained using the Fast Forward planner [37]. Given the actions $a \in A$ from these sequential plans π, Muise et al. first reported the solution qualities obtained from finding the optimal deordering of π using both the MaxSAT model and the KK algorithm. In terms of solution quality, both models performed equally. Muise et al. then compared the solution qualities obtained from the calculation of the optimal reordering of π, and the MCLP given A using the MaxSAT model. The comparison of the POPs produced by the optimal reordering versus the optimal deordering of π show consistent reductions in the number of ordering constraints of the resulting POPs. Similarly, the POPs produced using the MCLP criteria show signification reductions in the number of actions compared to the size of the set A. Finally, Muise et al. compared the number of linearizations produced by the optimization of deordering versus reordering criterion [54]. The analysis of the results show that finding the optimal reordering of a plan produce up to 10^6 times more linearizations in some benchmark instances compared to the POPs produced by finding the optimal deordering of π.

The next chapter of this thesis investigates different MILP formulations to find POPs with the optimal number of linearizations, given an initial sequential plan π. The fourth chapter of this thesis investigates a decomposition approach for finding a least commitment flexible POP to Π.

Chapter 3

Mixed-Integer Linear Programming Models for Optimizing Partial-Order Plan Flexibility

A partial-order plan (POP) compactly encodes a set of sequential plans that can be dynamically chosen by an agent at execution time. One natural measure of the quality of a POP is its flexibility, which is defined to be the total number of sequential plans it embodies (i.e., its linearizations) [56, 54]. As this criteria is hard to optimize, existing work (see Section 2.7.2) has instead optimized proxy functions that are correlated with the number of linearizations. In this chapter, we develop and strengthen MILP models for three proxy functions: two from the POP literature and a third novel function based on the *temporal flexibility* criteria from the scheduling literature. We show theoretically and empirically that none of the three proxy measures dominate the others in terms of number of sequential plans. Compared to the state-of-the-art MaxSAT model [56] for the problem, we empirically demonstrate that two of our MILP models result in equivalent or slightly better solution quality with savings of approximately one order of magnitude in computation time. The work presented in this chapter is published in the *Twenty-Second European Conference on Artificial Intelligence* [72].

3.1 Introduction

Least commitment planning aims to generate POPs that give agents the discretion of how to execute their plans [54]. A POP generated according to least commitment approach does not commit to actions or ordering decisions that do not contribute to the validity of the plan (see Section 2.2.3). Given a POP, an agent can dynamically commit to these sequencing decisions during the execution of the plan. As discussed in Section 2.7, the POCL-based planners are not competitive with the state-of-the-art sequential planners in terms of their computation effort. Alternatively, a sequential plan can be efficiently generated using a sequential planner and its solution quality can be optimized by partial-order relaxation (see Section 2.7.2).

Naturally, a key question in this area concerns the criterion which best reflects the quality of a plan. Several different objectives have been proposed in the literature, such as the *makespan* of the plan (i.e., longest path from the initial state to the goal), total number of unordered actions, existence of possible action reorderings, among others [73, 60, 2]. This work extends recent research [56, 54] that combines two distinct criteria: a *cost* per

action and the flexibility of a plan, here measured as the total number of linearizations of the POP. This objective naturally incorporates the least commitment principle of first executing as few (costly) actions as possible, and then improving the robustness of the system by placing as many sequential plans as possible at the disposal of the agent.

While total action cost has been traditionally tackled in sequential planning, enhancing the flexibility of a plan poses a much more challenging problem. Specifically, optimizing the number of linearizations of a POP is equivalent to maximizing the number of Hamiltonian paths in a directed acyclic graph, which is computationally impractical in general [53, 5]. To address this issue, Muise et al. [56, 54] optimize the number of ordering constraints in a POP, a metric that is correlated to the number of linearizations. Such a metric function is more computationally tractable and can be efficiently handled, e.g., by MaxSAT solvers.

Building on the work of Muise et al. and previous literature in planning and scheduling [20, 26], we address the problem of converting a valid sequential plan into a valid POP with minimum action cost and maximum number of linearizations. In particular, we consider the notion of *temporal flexibility* from the scheduling literature as a novel proxy function for the number of linearizations of a POP. We show that there is no dominance relation between our proxy function and two previous proxy functions in the literature: depending on the problem instance, the optimization of any of the proxy functions may lead to a greater number of linearizations than either of the others.

Nonetheless, a central benefit of the temporal flexibility criteria is the scaling of model size that is quadratic in the number of actions, rather than cubic as in Muise's model. Further, the linear relationships inherent in the temporal flexibility are amenable to mathematical programming techniques. We exploit this advantage and propose three MILP models for minimizing action cost and maximizing flexibility: a novel model of temporal flexibility, a model that linearizes the existing MaxSAT formulation [56], and a model that adapts an existing MILP formulation for temporal planning to POPs [26]. Going further, we derive a number of valid linear inequalities that can also be applied to the MILP models, substantially decreasing their solution times.

We compare our three MILP formulations to the current state-of-the-art MaxSAT model by Muise et al. [56]. Our empirical evaluation suggests that optimizing any of the three proxy functions results in equivalent solution quality, consistent with our theoretical results. Furthermore, the strengthened MILP models achieve approximately one order of magnitude speedup compared to the state-of-the-art MaxSAT model, solving significantly more problem instances to optimality. These results hold both when minimizing total action cost and maximizing flexibility and when only maximizing flexibility with a fixed set of actions.

Contributions. We present temporal flexibility as a novel proxy function for maximizing the number of linearizations in a partial-order plan and provide a novel mixed-integer linear program to optimize this criterion. We derive new valid linear inequalities that can be applied to the new and existing POP MILP formulations. Finally, we show that the modified MILP models achieve substantially better run-time performance than the current state-of-the-art without sacrificing solution quality.

3.2 Least Commitment Flexible POPs

Given a sequential plan π to a STRIPS problem Π, our aim is to derive a POP P from π that is ideally optimal both in terms of its action cost and its flexibility. These two characteristics are embodied in the notion of least commitment planning [51].

The least commitment planning evaluates POPs according to two metrics: the *total action cost* and the *number of linearizations* of the plan. The first is a natural and common objective in planning as agents would like to incur the minimum total action cost possible to achieve a goal. The second metric intuitively gives us a notion of how

flexible the plan is, since more linearizations indicate more alternative ways to execute the actions to achieve the goal [56].

Equipped with these two notions, we formally define the structure of an optimal POP in our context.

Definition 4. *(Least Commitment Flexible POP (LCFP) of a Plan). Let $P = \langle \mathcal{A}, \mathcal{O}, K \rangle$ and $P' = \langle \mathcal{A}', \mathcal{O}', K' \rangle$ be two valid POPs for a planning problem Π where $\mathcal{A}' \subseteq \mathcal{A}$. Moreover, let c_a be a non-negative cost associated with each action $a \in \mathcal{A}$. P' is a least commitment flexible POP (LCFP) of plan P iff*

(i) For all valid POPs with action set $\mathcal{A}'' \subseteq \mathcal{A}$, we have $\sum_{a \in \mathcal{A}''} c_a \geq \sum_{a \in \mathcal{A}'} c_a$; and

(ii) For all valid POPs P'' with action set $\mathcal{A}'' \subseteq \mathcal{A}$ and $\sum_{a \in \mathcal{A}''} c_a = \sum_{a \in \mathcal{A}'} c_a$, the number of linearizations of P' is greater or equal to the number of linearizations of P''.

This definition differs from that of Muise et al. [56] as it contains a slightly more general action cost structure and explicitly incorporates the number of linearizations, as opposed to the number of ordering constraints.

As in previous work, Definition 4 does not include solving the traditional POP planning problem of finding the set of actions and the ordering constraints that achieve a valid POP. Rather, we are concerned with the optimization of action cost and flexibility, given a valid POP. Thus, our primary goal in this work is to find the LCFP of a sequential plan, which is itself a POP. Since counting linearizations is computationally challenging, we investigate the optimization of alternative proxy functions that correlate with the number of linearizations of a POP.

3.3 Proxy Measures of POP Flexibility

We study three proxy functions for the number of linearizations of a POP, two of which are extracted from previous works, and one novel to this work: the minimization of the number of *open ordering* constraints [26], the minimization of the number of *closed ordering* constraints [56], and the maximization of temporal flexibility adapted from the scheduling literature [20].

3.3.1 Order Flexibility

Muise et al. [56] optimized what we term the *order flexibility* of a POP: the total number of ordering constraints in the plan. The more ordering constraints, the less order flexibility a POP has. We investigate two definitions of an ordering constraint.

Definition 5. *(Open Ordering Constraint): Given the set of actions $\mathcal{A}$ and the set of causal links K, the open ordering constraint $a_1 \prec a_2$ belongs to the set $\mathcal{O}$ for some POP $P = \langle \mathcal{A}, \mathcal{O}, K \rangle$ to a planning problem Π if:*

1. *There exists a causal link $\kappa(a_1, a_2, f) \in K$ from action a_1 to action a_2 on some fluent $f \in F$, or*
2. *There exists a causal link $\kappa(a_2, a_3, f) \in K$ from action a_2 to action a_3 on some fluent $f \in F$ and action $a_1 \in del_f$ is ordered before action a_2 to resolve the threat, or*
3. *There exists a causal link $\kappa(a_3, a_1, f) \in K$ from action a_3 to action a_1 on some fluent $f \in F$ and action $a_2 \in del_f$ is ordered after action a_1 to resolve the threat.*

Do and Kambhampati [26] minimize the equivalent of the number of open ordering constraints in *order constrained plans*, temporal plans which have a partial-order structure but allow concurrent execution of non-interfering actions. Concurrent execution semantics create a subtle but relevant difference in the meaning of

open ordering constraints. In particular, Definition 2 relaxes Do and Kambhampati's definition by assuming a sequential execution of a POP.

We can now characterize the ordering constraint definition used by Muise et al. [56] as specifically the *transitive closure* of the open ordering constraints.

Definition 6. *(Closed Ordering Constraint): Given the set of actions $\mathcal{A}$ and the set of causal links K, the closed ordering constraint $a_1 \prec a_2$ belongs to the set $\mathcal{O}$ for some POP $P = \langle \mathcal{A}, \mathcal{O}, K \rangle$ to a planning problem Π if:*

1. *There exists an open ordering constraint between actions a_1 and a_2, or*
2. *There exists some other action a_3 such that: $a_1 \prec a_3 \in \mathcal{O}$ and $a_3 \prec a_2 \in \mathcal{O}$.*

We will refer to the set of open and closed ordering constraints as $\mathcal{O}_O$ and $\mathcal{O}_C$, respectively.

3.3.2 Temporal Flexibility

In scheduling, temporal flexibility refers to a schedule's ability to absorb temporal variation during execution [62]. We exploit the same property to define an analogous version of temporal flexibility for planning problems.

Given a POP $P = \langle \mathcal{A}, \mathcal{O}, K \rangle$ for Π and a duration d_a of each action $a \in A$, the *horizon* of P, H_P, is the sum of the action durations. Intuitively, since the actions will be executed in sequence, H_P is the (temporal) length of the plan. The *earliest start time* of an action a is $est_a = \max_{a' \prec a \in \mathcal{O}} (est_{a'} + d_{a'})$ and the *latest finish time of action* a is $lft_a = \min_{a \prec a' \in \mathcal{O}} (lft_{a'} - d_{a'})$, where $est_{a_I} = 0$ and $lft_{a_G} = H_P$ for the unique first and last actions. Let the *action slack* of a, T_a, be such that $T_a = lft_a - est_a - d_a$ [20]. The *temporal flexibility*, T, of P is given by $T = \sum_{a \in \mathcal{A}} T_a$. For classical (i.e. non-temporal) planning, we assume that all actions have a duration of one time unit.

3.3.3 Dominance Relations Among Proxy Functions

A proxy function obj_1 *dominates* another proxy function obj_2 if, for any planning problem instance Π, any POP P that optimizes obj_1 has at least as many linearizations as any POP P' that optimizes obj_2 and, for at least one instance Π^*, the number of linearizations in P is strictly greater than that of P'. We have the following result.

Proposition 7. *There are no dominance relations among the open ordering, closed ordering, and temporal flexibility proxy functions.*

Proof. For each pair of objective functions (obj_i, obj_j), $i \neq j$, it suffices to show the existence of two problem instances Π_1, Π_2, where, for Π_1, a POP P that optimizes obj_i has more linearizations than a POP P' that optimizes obj_j and vice versa for Π_2. The counter-examples are simple but tedious to verify and are presented in the next section. □

To formally prove Proposition 7, Figures 3.1 to 3.4 present counter-examples showing that there does not exist a dominance relation between any pair of the three proxy functions in terms of the resulting number of linearizations in a POP. That is, it is not the case that optimizing one proxy function will always lead to more POP linearizations than optimizing one of the other two.

Each counter-example presents an abstracted planning problem with fluents indexed by integers and two graphs presenting the POP found by optimizing two of the objective functions, respectively. The nodes represent the actions and arcs represent the ordering constraints under the $\mathcal{O}_O$ definition. For clarity, we do not show the causal links, the action variables a_I and a_G, or the transitive closure. We use the notation $obj_1 \not\Rightarrow obj_2$ to

represent the non-dominance of obj_1 over obj_2 with respect to the number of linearizations, where obj_1 and obj_2 represent a pair of proxy objectives. In order to show $obj_1 \not\Rightarrow obj_2$, we separately optimize the set of actions with respect to both obj_1 and obj_2, and show that $L_1 < L_2$, where L_i is the number of linearizations of objective i counted through exhaustive enumeration. For each POP example, we report the proxy objective functions optimized, and the resulting $|\mathcal{O}_O|$, $|\mathcal{O}_C|$, T and L values.

For clarity, we demonstrate how the counter-examples work using the example presented in Figure 3.1. The table in Figure 3.1 presents the planning problem. From left to right, the columns represent the action names, the preconditions, add effects and delete effect of actions $a \in \mathcal{A}$. Each numeric value represents a unique fluent $f \in F$. Each row of the table represents an action $a \in \mathcal{A}$. For example, action a_1 has a precondition on fluent $0 \in F$, adds fluents $1, 7 \in F$ and has no delete effects. In total, the set $\mathcal{A}$ has eight actions, including the dummy actions a_I and a_G. The items in the caption represent the proxy objectives under which the optimal LCFPs are generated. The action precedence graph of the POP on the left is an optimal LCFP with respect to the objective $min.\ |\mathcal{O}_C|$, and the POP on the right is an optimal LCFP with respect to the objective $max.\ T$. The notation $min.\ |\mathcal{O}_C| \not\Rightarrow max.\ T$ indicates that Figure 3.1 is a counter-example to demonstrate the non-dominance of the minimization of $|\mathcal{O}_C|$ over the maximization of T. The remaining items in the caption report the proxy objectives that are optimized, and the numerical values of every proxy objective of the produced POP. For example, the POP presented on the right maximizes temporal flexibility and has 5 open ordering constraints, 8 closed ordering constraints, 18 units of temporal flexibility and 16 linearizations. Since the example on the left minimizes $|\mathcal{O}_C|$ and has 15 linearizations, we conclude that minimizing $|\mathcal{O}_C|$ does not dominate the maximization of T.

Actions	Pre	Add	Del
a_I	-	0	-
a_1	0	1,7	-
a_2	1	2,3,8	-
a_3	0	6,9	4
a_4	6	4,5,10	-
a_5	4	2,3,11	-
a_6	2,3,5	12	-
a_G	7,8,9,10,11,12	-	-

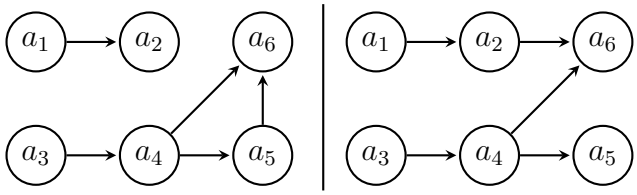

Figure 3.1: $min.\ |\mathcal{O}_C| \not\Rightarrow max.\ T$. **Left:** min. $|\mathcal{O}_C|$, $|\mathcal{O}_O| = 5$, $|\mathcal{O}_C| = 7$, $T = 16$, $L = 15$. **Right:** max. T, $|\mathcal{O}_O| = 5$, $|\mathcal{O}_C| = 8$, $T = 18$, $L = 16$.

Actions	Pre	Add	Del
a_I	-	8,9,10	-
a_1	8	0,3	-
a_2	0	1,2,4	-
a_3	0,1,2	5	-
a_4	9	1,6	-
a_5	10	2,7	-
a_G	3,4,5,6,7	-	-

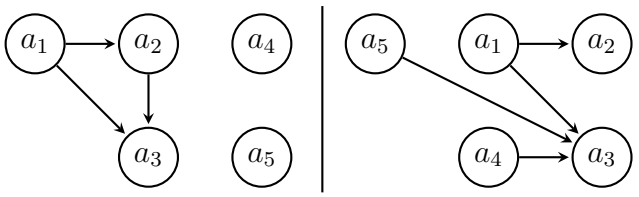

Figure 3.2: $max.\ T \not\Rightarrow min.\ |\mathcal{O}_C|$ and $max.\ T \not\Rightarrow min.\ |\mathcal{O}_O|$. **Left:** min. $|\mathcal{O}_C|$ or min. $|\mathcal{O}_O|$, $|\mathcal{O}_O| = 3$, $|\mathcal{O}_C| = 3$, $T = 14$, $L = 20$. **Right:** max. T, $|\mathcal{O}_O| = 4$, $|\mathcal{O}_C| = 4$, $T = 15$, $L = 18$.

Actions	Pre	Add	Del
a_I	-	7	-
a_1	7	2,3	0
a_2	7	0,1,4	-
a_3	0,7	1,2,5	-
a_4	1,2,7	6	-
a_G	3,4,5,6	-	-

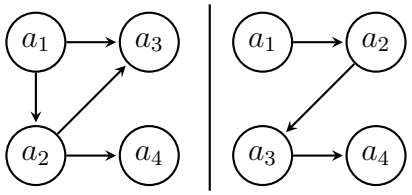

Figure 3.3: $min.\ |\mathcal{O}_O| \not\Rightarrow min.\ |\mathcal{O}_C|$ and $min.\ |\mathcal{O}_O| \not\Rightarrow max.\ T$. **Left:** min. $|\mathcal{O}_C|$ or max. T, $|\mathcal{O}_O| = 4$, $|\mathcal{O}_C| = 5$, $T = 4$, $L = 2$. **Right:** min. $|\mathcal{O}_O|$, $|\mathcal{O}_O| = 3$, $|\mathcal{O}_C| = 6$, $T = 0$, $L = 1$.

Actions	Pre	Add	Del
a_I	-	10	-
a_1	10	3,4,5	-
a_2	3,4,10	0,1,2,6	-
a_3	10	1,4,7	3
a_4	1,10	2,8	-
a_5	2,10	3,9	-
a_G	5,6,7,8,9	-	-

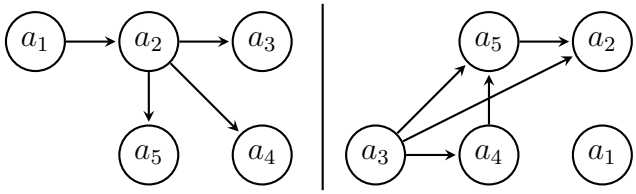

Figure 3.4: $min.\ |\mathcal{O}_C| \not\Rightarrow min.\ |\mathcal{O}_O|$. **Left:** min. $|\mathcal{O}_O|$, $|\mathcal{O}_O| = 4$, $|\mathcal{O}_C| = 7$, $T = 10$, $L = 6$. **Right:** min. $|\mathcal{O}_C|$, $|\mathcal{O}_O| = 5$; $|\mathcal{O}_C| = 6$, $T = 8$, $L = 5$.

Given this result of non-dominance, we now turn to the investigation of different MILP formulations to optimize the proxy objectives of an input set of actions $\mathcal{A}$.

3.4 The $\mathcal{O}_C^{\text{MILP}}$ Model

Muise [54] empirically investigated equivalent MILP and MaxSAT models for the problem of minimum reordering: finding a POP with the minimum number of closed ordering constraints without removing actions from the initial set of actions. $\mathcal{O}_C^{\text{MILP}}$ is a minor extension of Muise's MILP encoding for the LCFP problem under the closed constraint proxy function. $\mathcal{O}_C^{\text{MILP}}$ does not use the input ordering of the initial sequential plan.

The parameters used in $\mathcal{O}_C^{\text{MILP}}$ are as follows.

- $\mathcal{A}' = \mathcal{A} \backslash \{a_I, a_G\}$ is the set of non-dummy actions.

- $G_f = 1$ if dummy action a_G requires fluent f or equivalently if f is required to be true in the goal state.

The objective function and the linear constraints used in $\mathcal{O}_C^{\text{MILP}}$ are as follows.

$$\text{minimize } (|\mathcal{A}'|^2 + 1) \sum_{a \in \mathcal{A}'} c_a Z_a + \sum_{a_i, a_j \in \mathcal{A}'} O_{a_i, a_j}$$

$$\sum_{a_i \in add_f} X^f_{a_i, a_j} = Z_{a_j} \qquad \forall a_j \in \mathcal{A}, f \in PRE_{a_j} \tag{3.1}$$

$$(1 - X^f_{a_i, a_j}) + (O_{a_d, a_i} + O_{a_j, a_d}) \geq Z_{a_d}$$
$$\forall a_i, a_d, a_j \in \mathcal{A}, f \in PRE_{a_j} \cap DEL_{a_d} \cap ADD_{a_i} \tag{3.2}$$

$$O_{a_i, a_j} \geq X^f_{a_i, a_j} \qquad \forall a_i, a_j \in \mathcal{A}, f \in ADD_{a_i} \cap PRE_{a_j} \tag{3.3}$$

$$O_{a_I, a} = Z_a \qquad \forall a \in \mathcal{A} \backslash \{a_I\} \tag{3.4}$$

$$O_{a, a_G} = Z_a \qquad \forall a \in \mathcal{A} \backslash \{a_G\} \tag{3.5}$$

$$O_{a_i, a_j} + O_{a_j, a_i} \leq \frac{Z_{a_i} + Z_{a_j}}{2} \qquad \forall a_i, a_j \in \mathcal{A} \tag{3.6}$$

$$Z_{a_I} = Z_{a_G} = 1 \tag{3.7}$$

$$O_{a,a} = 0 \qquad \forall a \in \mathcal{A} \tag{3.8}$$

$$(1 - O_{a_i, a_j}) + (1 - O_{a_j, a_k}) + O_{a_i, a_k} \geq 1 \qquad \forall a_i, a_j, a_k \in \mathcal{A} \tag{3.9}$$

The objective first minimizes the sum of action costs, then minimizes the number of closed ordering constraints. The first coefficient of the objective function, $|\mathcal{A}'|^2 + 1$, guarantees this property when all the actions have costs greater or equal to 1 unit. When this is not the case, we need to normalize the action costs so that all the actions have costs greater or equal to 1 unit. Constraint (3.1) ensures that if an action is selected, all of its preconditions are met exactly once. Constraint (3.2) orders the threatening actions either before or after the causal link. Constraint (3.3) is an order implication constraint that states that if action $a_1 \in \mathcal{A}$ supports some other action $a_2 \in A$ with some fluent $f \in F$, a_1 must be ordered before a_2. Constraints (3.4)-(3.5) restrict all the included actions to be between the initial and the goal actions. Constraint (3.6) makes sure that the actions in an enforced ordering constraint are included in the plan. Constraint (3.7) includes the initial and goal actions. Constraint (3.8) disallows self-loops for ordering constraints. Constraint (3.9) produces a transitively-closed POP. Constraints (3.8)-(3.9) together forbid cycles in the action precedence graph.

Note that due to the ternary arity of Constraint (3.9), $\mathcal{O}_C^{\text{MILP}}$ grows cubically with the number of actions.

3.5 The $\mathcal{O}_O^{\text{MILP}}$ and $\mathcal{T}^{\text{MILP}}$ Models

Both $\mathcal{O}_O^{\text{MILP}}$ and $\mathcal{T}^{\text{MILP}}$ build on Do and Kambhampati's MILP model [26]. The main differences are that $\mathcal{O}_O^{\text{MILP}}$ does not enforce an ordering between all pairs of interfering actions, while also allowing for actions to be excluded when they are not relevant to the POP's validity.

$\mathcal{O}_O^{\text{MILP}}$ and $\mathcal{T}^{\text{MILP}}$ are identical aside from their objectives. The key difference with $\mathcal{O}_C^{\text{MILP}}$ is the use of start time variables to represent ordering constraints. With the addition of a linear number of variables, the models grow quadratically with the number of actions.

Compared to the formulation of $\mathcal{O}_C^{\text{MILP}}$, $\mathcal{O}_O^{\text{MILP}}$ and $\mathcal{T}^{\text{MILP}}$ introduce three additional variables: $Est_a = \max_{a' \prec a \in \mathcal{O}} (Est_{a'} + d_{a'})$ is the earliest start time of action a; $Lft_a = \min_{a \prec a' \in \mathcal{O}} (Lft_{a'} - d_{a'})$ is the latest finish time of a; and finally $T_a = Lft_a - Est_a - d_a$ is the slack of a.

$$\text{minimize } (|\mathcal{A}'| \sum_{a \in \mathcal{A}'} d_a + 1) \sum_{a \in \mathcal{A}'} c_a Z_a - \sum_{a \in \mathcal{A}'} T_a$$

Constraints (3.1)-(3.8)

$$Est_{a_i} + d_{a_i} O_{a_i,a_j} \leq \quad Est_{a_j} + \sum_{a \in \mathcal{A}} d_a (1 - O_{a_i,a_j}) \qquad \forall a_i, a_j \in \mathcal{A} \tag{3.10}$$

$$Lft_{a_i} + d_{a_j} O_{a_i,a_j} \leq \quad Lft_{a_j} + \sum_{a \in \mathcal{A}} d_a (1 - O_{a_i,a_j}) \qquad \forall a_i, a_j \in \mathcal{A} \tag{3.11}$$

$$Est_a + d_a Z_a + T_a = Lft_a \qquad \forall a \in \mathcal{A} \tag{3.12}$$

$$Est_{a_I} = 0 \tag{3.13}$$

$$Lft_{a_G} = \sum_{a \in \mathcal{A}} d_a Z_a \tag{3.14}$$

As $\mathcal{O}_O^{\text{MILP}}$ uses the same objective function as $\mathcal{O}_C^{\text{MILP}}$ and the only difference between $\mathcal{T}^{\text{MILP}}$ and $\mathcal{O}_O^{\text{MILP}}$ are the objectives, we present the model for $\mathcal{T}^{\text{MILP}}$. Analogously to $\mathcal{O}_C^{\text{MILP}}$, in $\mathcal{T}^{\text{MILP}}$ the objective first minimizes the sum of action cost, then maximizes the sum of temporal slack of all the actions. Constraints (3.10)-(3.11) make sure that, if some action a_i is ordered before some other action a_j, the earliest start time and the latest finish time of a_j are not before the earliest start time and the latest finish time of a_i, respectively. Constraint (3.12) defines the temporal slack of an action. Constraints (3.13)-(3.14) set the plan horizon.

3.6 Valid Inequalities

We now present valid linear inequalities to strengthen the MILP models. Unless noted, the constraints can be added to all formulations.

Mutual Threat Constraints As illustrated in Figure 3.5, a cycle is formed when action a_i supports fluent f for action a_j and fluent g for action a_j, where a_j and a_k threaten the causal links $X^g_{a_i,a_k}$ and $X^f_{a_i,a_j}$, respectively. Since a cycle is not allowed in the action precedence graph, $X^g_{a_i,a_k}$ and $X^f_{a_i,a_j}$ are mutually exclusive in any POP and are removed via Constraint (3.15). When $f = g$, all $X^f_{a_i,a_j}$ are mutually exclusive and are also removed by Constraint (3.15).

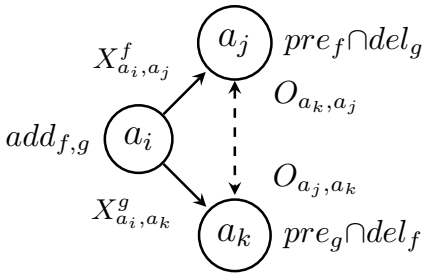

Figure 3.5: Mutual threat constraint example.

$$\max \left\{ X^g_{a_i,a_k} + X^f_{a_i,a_j}, \sum_{a_j \in (pre_f \cap del_f)} X^f_{a_i,a_j} \right\} \leq Z_{a_i} \quad \forall f \in ADD_{a_i}, a_i \in \mathcal{A} \tag{3.15}$$

Action Relevance Constraint Constraint (3.16) states that a selected action must support at least one causal link. If there does not exist a causal link $\kappa(a_1, a_2, f) \in K$ for fluent $f \in F$ and action $a_2 \in pre_f$, action $a_1 \in add_f$ can be removed without effecting the validity of the POP. If the action has a positive action cost $c_{a_i} > 0$, the removal of a_1 will decrement the total action cost by c_{a_i}. This means the original POP that has action a_1 cannot be a LCFP of the initial plan.

$$\sum_{a_j \in pre_f, f \in ADD_{a_i}} X^f_{a_i,a_j} \geq Z_{a_i} \quad \forall c_{a_i} > 0, a_i \in \mathcal{A}'. \tag{3.16}$$

Minimal Interference Constraint When an action $a_1 \in \mathcal{A}$ with positive action cost $c_{a_1} > 0$ adds only one fluent $f \in F$, there must exist another action $a_2 \in pre_f$ and the causal link $\kappa(a_1, a_2, f) \in K$ (due to Constraint 3.16). The decision variable that corresponds to the causal link $\kappa(a_1, a_2, f) \in K$ must take the value of 1 i.e., $X^f_{a_1,a_2} = 1$. Further, if there exists a third action $a_3 \in del_f$, action a_1 must be ordered with respect to it, due to Constraint (3.2) i.e., $O_{a_3,a_1} = 1$ or $O_{a_1,a_3} = 1$ because $X^f_{a_1,a_2} \rightarrow O_{a_1,a_2}$ and $O_{a_1,a_2} \wedge O_{a_2,a_3} \rightarrow O_{a_1,a_3}$ due to Constraint (3.9). Constraint (3.17)[1] enforces ordering constraints on all the pairs of actions, $a_1, a_2 \in \mathcal{A}$, if, given $|ADD_{a_2}| = 1$, either $a_1 \in del_f$ and $a_2 \in (add_f \cup pre_f)$ or $a_2 \in del_f$ and $a_1 \in (add_f \cup pre_f)$.

$$\begin{aligned} &O_{a_i,a_j} + O_{a_j,a_i} \geq Z_{a_i} + Z_{a_j} - 1 \\ &\quad \forall a_i, a_j \in \mathcal{A}, \exists f \in (PRE_{a_i} \cup ADD_{a_i}) \cap DEL_{a_j}, \text{and } |ADD_{a_i}| = 1. \end{aligned} \tag{3.17}$$

Counting Constraints Inspired by operator counting constraints [64] (see Section 2.5.2), we observe the following: since all actions a_i that delete and require fluent f (i.e., $f \in F, a_i, a_j \in del_f \cap pre_f$) are sequentially ordered due to interference on f, there must exist at least one action a_k that adds fluent f (i.e., $a_k \in add_f$) between each a_i. This observation gives rise to three constraints: Constraint (3.18) counts the total number of occurrences of both a_i and a_k while Constraints (3.19) and (3.20) each split the plan into two for each a_i and ensure there are more actions a_k than a_j succeeding and preceding a_i, respectively.

[1] Based on the interference constraints of Do and Kambhampati [26].

$$\sum_{a_k \in add_f} Z_{a_k} + I_f \geq \sum_{a_j \in (del_f \cap pre_f)} Z_{a_j} + G_f \qquad \forall f \in F \tag{3.18}$$

$$\sum_{a_k \in add_f} O_{a_i,a_k} \geq \sum_{a_j \in (del_f \cap pre_f)} O_{a_i,a_j} + G_f Z_{a_i} \qquad \forall a_i \in \mathcal{A}, f \in PRE_{a_i} \cap DEL_{a_i} \tag{3.19}$$

$$\sum_{a_k \in add_f} O_{a_k,a_i} + I_f Z_{a_i} \geq \sum_{a_j \in (del_f \cap pre_f)} O_{a_j,a_i} \qquad \forall a_i \in \mathcal{A}, f \in PRE_{a_i} \cap DEL_{a_i} \tag{3.20}$$

Symmetry Breaking Constraints Two actions that are equivalent in their preconditions, add effects, and delete effects (denoted $a_i \equiv a_j$) introduce symmetrically identical solutions (and non-solutions) into the search space. We can break this symmetry by enforcing a lexicographical ordering as shown below. Constraints (3.21)-(3.23) disallow the inclusion, ordering, and start of the action with lower index value before its equivalent action with higher index value, respectively. Constraint (3.24) ensures that two equivalent actions, a_i, a_j, that delete and require fluent f (i.e. $f \in F, a_i, a_j \in del_f \cap pre_f$) are ordered with respect to their index values.

$$Z_{a_i} \leq Z_{a_j} \qquad \forall c_{a_i} > 0, i < j, a_i \equiv a_j \in \mathcal{A} \tag{3.21}$$

$$O_{a_i,a_j} = 0 \qquad \forall i < j, a_i \equiv a_j \in \mathcal{A} \tag{3.22}$$

$$Est_{a_j} \leq Est_{a_i} + \sum_{a \in A} d_a(1 - Z_{a_i}) \qquad \forall i < j, a_i \equiv a_j \in \mathcal{A} \tag{3.23}$$

$$O_{a_j,a_i} = Z_{a_i} \quad \forall i < j, a_i \equiv a_j \in \mathcal{A}, \qquad \exists f \in F, a_i, a_j \in del_f \cap pre_f \tag{3.24}$$

Constraints (3.17), (3.19), (3.20), and (3.24) do not introduce any ordering constraints that are not relevant to the validity of a POP. However, their addition to $\mathcal{O}_O^{\text{MILP}}$ can change the optimal solution because the orderings due to threats may now be replaced with explicit ordering constraints and therefore result in a higher number of open ordering constraints. For example, given the causal link $\kappa(a_1, a_2, f) \in K$ between actions $a_1 \in add_f$ and $a_2 \in pre_f$ on some fluent $f \in F$, the optimal LCFP generated by $\mathcal{O}_O^{\text{MILP}}$ orders actions a_2 and $a_3 \in del_f$ with the ordering constraint $a_2 \prec a_3$. The addition of Constraint (3.17) explicitly orders a_1 and a_3 with the ordering constraint $a_1 \prec a_3$, which can change the optimal value or the optimal solution. We add Constraints (3.15)-(3.22) and (3.24) to $\mathcal{O}_C^{\text{MILP}}$, $\mathcal{O}_O^{\text{MILP}}$ and $\mathcal{T}^{\text{MILP}}$, and add Constraint (3.23) only to $\mathcal{O}_O^{\text{MILP}}$ and $\mathcal{T}^{\text{MILP}}$. We refer to these strengthened models as $\mathcal{O}_C^{\text{MILP+S}}$, $\mathcal{O}_O^{\text{MILP+S}}$ and $\mathcal{T}^{\text{MILP+S}}$, respectively.

3.7 Computational Results

In this section, we present the results of two computational experiments. In Experiment 1, we investigate the empirical behaviour of the three proxy functions. While we demonstrated in Section 3.3.3 that theoretically none of the proxy functions dominate the others, this result does not speak to the average empirical behaviour: it is possible that, in practice, a proxy function often results in more linearizations than others. To test this possibility, we focus on the *minimum reordering problem*: a version of LCFP where the number of actions is fixed. This restriction ensures that we are comparing the proxy functions while controlling for the complicating factor of different action sets that arises in the LCFP models. We show that, generally, there is also no empirical domination among the proxy functions, with our only significant comparison being that $\mathcal{T}^{\text{MILP+S}}$ achieves a statistically significant higher mean logarithmic number of linearizations than $\mathcal{O}_C^{\text{maxSAT}}$. We also provide empirical

evidence that the MILP models for temporal flexibility and open ordering constraints are substantially faster than the state-of-the-art MaxSAT model on the minimum reordering problems.

In Experiment 2, we solve the full LCFP problems. Our findings show that the proposed models $\mathcal{O}_O^{\text{MILP+S}}$ and $\mathcal{T}^{\text{MILP+S}}$ can be solved to optimality faster on the majority of the tested instances, and scale better with the initial number of actions compared to $\mathcal{O}_C^{\text{maxSAT}}$. The solution quality across proxy functions is similar with no significant differences in the action cost or mean number of linearizations but with $\mathcal{T}^{\text{MILP+S}}$ and $\mathcal{O}_C^{\text{maxSAT}}$ finding a statistically significantly higher mean logarithmic number of linearizations than $\mathcal{O}_O^{\text{MILP+S}}$.

Experimental Details For both experiments, the initial plans are generated using the *Fast-Forward* planning system [37]. For the first experiment, we use eight domains from the International Planning Competition: `Depots`, `Driverlog`, `Freecell`, `Gripper`, `Logistics`, `Rovers`, `Tpp`, and `Zenotravel`, giving in total 144 instances. For the second experiment, we use the use the same experimental setup as Muise et al. [56] including the same domains: `Depots`, `Driverlog`, `Logistics`, `Rovers`, `Tpp`, and `Zenotravel`, giving in total 138 instances. The experiments ran on a MacBookPro computer with 2.66 GHz Intel Core i7. The MILP models were solved using IBM ILOG CPLEX 12.6.2 with 1 thread. For $\mathcal{O}_C^{\text{maxSAT}}$, we use the *SAT4j* MaxSAT 2.3.5 solver with a memory limit of 2GB. A time limit of 1,800 seconds was imposed on all models.

3.7.1 Experiment 1: Comparing Proxy Functions

To observe the effect of the optimization of each proxy objective on the number of linearizations in a POP, we fix the set of actions (i.e., $Z_a = 1, \forall a \in \mathcal{A}$) and optimize the proxy objective functions.

Solution Quality In Table 3.1, we report the mean logarithmic number of linearizations considering the instances for which all three models return an optimal solution and for which we successfully count the number of linearizations: 99 instances. Values in boldface represent the maximum in each row. The number of linearizations is found through a simple depth-first search with a time limit of 30 minutes per instance. We note that the mean logarithmic number of linearizations is equivalent to the *geometric mean* of such numbers, which is more appropriate than reporting their means when comparing large-magnitude numbers (as the number of linearizations grows exponentially large with the number of actions). This measure is largely used in the optimization literature (see, e.g., [4, 12]).

We performed bootstrap paired t-tests [16] using two statistics: number of linearizations and logarithmic number of linearizations (base 10). Our results indicated that there are no statistically significant differences in the mean number of linearizations while for the mean logarithmic statistic the only significant difference ($p \leq 0.01$) is that $\mathcal{T}^{\text{MILP+S}}$ finds a higher mean than $\mathcal{O}_C^{\text{maxSAT}}$. Among the two models that optimized the closed ordering constraints, $\mathcal{O}_C^{\text{maxSAT}}$ consistently dominated $\mathcal{O}_C^{\text{MILP+S}}$. Therefore the results for $\mathcal{O}_C^{\text{MILP+S}}$ are not presented.

In Figure 3.6, we plot the number of linearizations for which optimal solutions were found for both closed ordering flexibility and temporal flexibility and for which we successfully counted the number of linearizations. For 83% of the instances, optimization of both proxy objective functions resulted in POPs with the same number of linearizations.

	Mean Log_{10} Number of Linearizations		
Dom	$\mathcal{T}^{\text{MILP+S}}$	$\mathcal{O}_O^{\text{MILP+S}}$	$\mathcal{O}_C^{\text{maxSAT}}$
Dep	**7.09**	6.97	6.93
Dri	5.84	**5.95**	5.83
Fre	7.55	6.80	**7.57**
Gri	**4.34**	**4.34**	**4.34**
Log	**11.91**	11.42	11.38
Rov	**8.53**	8.15	8.41
Tpp	**3.71**	**3.71**	**3.71**
Zen	**7.69**	7.55	**7.69**
Mean	**7.76**	7.51	7.63

Table 3.1: Solution quality in terms of linearizations (logarithmic) in Experiment 1.

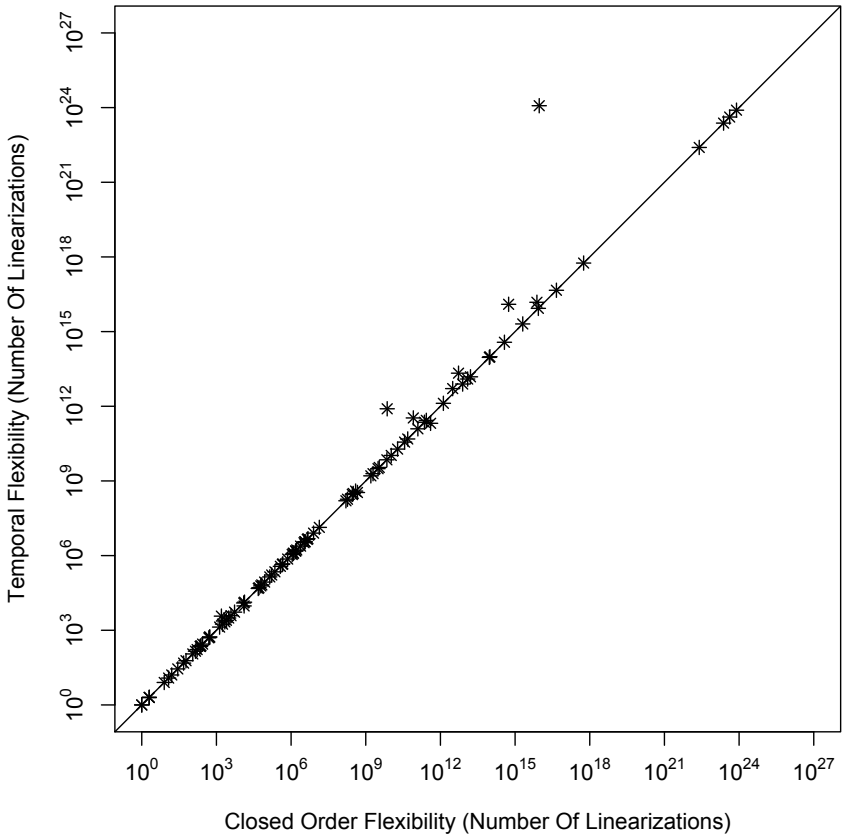

Figure 3.6: Number of linearizations between temporal flexibility and closed ordering flexibility (in logarithmic scale) in Experiment 1.

Computational Effort We now compare the models with respect to the effort to optimize each proxy objective. Figure 3.7 shows a performance profile depicting the number of instances solved to optimality over the 30-minute time limit. Optimization of open order, temporal and closed ordering flexibility using $\mathcal{O}_O^{\text{MILP+S}}$, $\mathcal{T}^{\text{MILP+S}}$ and

$\mathcal{O}_C^{\text{maxSAT}}$ models solve 126, 122 and 118 problem instances to optimality within 30 minutes, respectively. It can be observed that both MILP models outperform $\mathcal{O}_C^{\text{maxSAT}}$, while $\mathcal{O}_O^{\text{MILP+S}}$ is also superior to $\mathcal{T}^{\text{MILP+S}}$ across all time points.

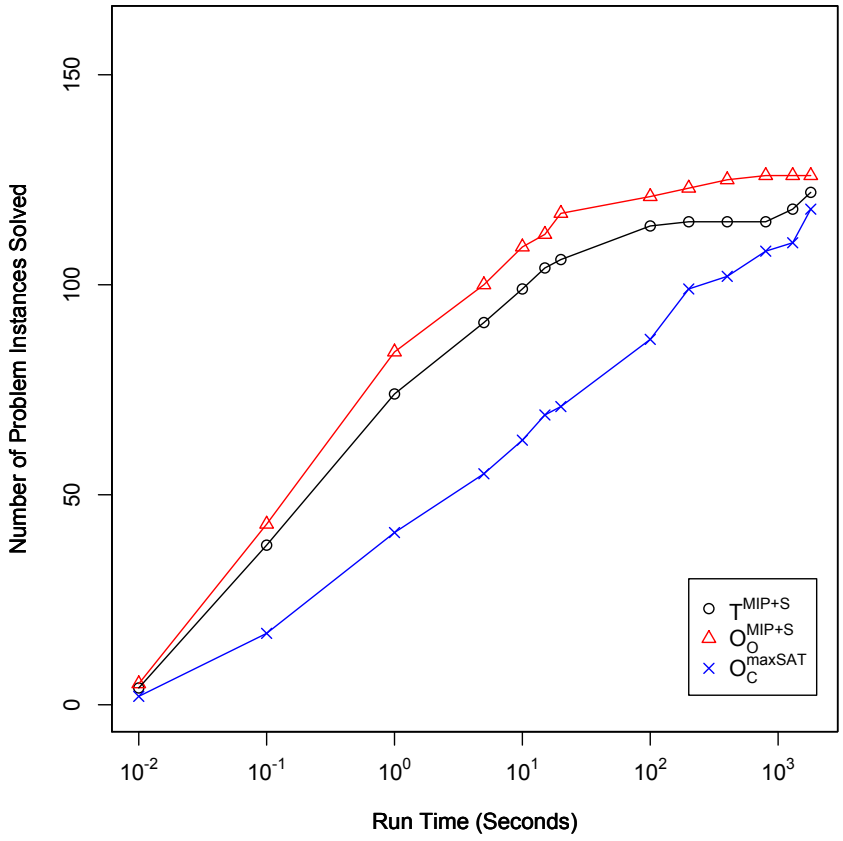

Figure 3.7: Performance profile (in log scale) for Experiment 1.

3.7.2 Experiment 2: Solving LCFPs

Turning to LCFPs, we present results on three issues in this subsection: the quality of LCFPs produced by each model, the computational effort for each model, and the impact of the strengthening constraints that we introduced above.

Solution Quality In Table 4.1, we report the mean action cost for each planning domain. The table includes results from the 131 problems instances for which at least one of the approaches found a feasible solution. For instances for which an approach found no feasible POP but another one did, we use the action cost of the input sequential plan for the former approach. Values in boldface are the minimum for each row. All four models perform similarly, a result reflected in the bootstrap paired t-tests showing no significant differences. The `Tpp` domain is the only one that appears to have variation, an observation that we further explore below (see Figure 3.11).

	Average Total Action Cost			
Dom	$\mathcal{O}_O^{\text{MILP+S}}$	$\mathcal{O}_C^{\text{MILP+S}}$	$\mathcal{T}^{\text{MILP+S}}$	$\mathcal{O}_C^{\text{maxSAT}}$
Dep	**42.30**	43.86	**42.30**	**42.30**
Dri	**26.80**	**26.80**	**26.80**	**26.80**
Log	**91.11**	91.94	**91.11**	91.14
Rov	**35.2**	**35.2**	**35.2**	**35.2**
Tpp	91.5	94.05	91.82	**85.95**
Zen	**33.1**	**33.1**	**33.1**	**33.1**
Mean	59.81	60.74	59.87	**58.95**

Table 3.2: Solution quality in terms of total action cost.

	Mean Log_{10} Number of Linearizations			
Dom	$\mathcal{O}_O^{\text{MILP+S}}$	$\mathcal{O}_C^{\text{MILP+S}}$	$\mathcal{T}^{\text{MILP+S}}$	$\mathcal{O}_C^{\text{maxSAT}}$
Dep	9.24	8.55	**11.17**	10.95
Dri	6.66	**6.81**	6.79	**6.81**
Log	18.01	**18.76**	18.70	**18.76**
Rov	14.82	**15.05**	**15.05**	**15.05**
Tpp	9.75	8.06	10.53	**10.58**
Zen	7.64	**7.84**	7.82	**7.84**
Mean	11.06	10.97	**11.75**	11.72

Table 3.3: Solution quality in terms of linearizations (logarithmic).

In Table 3.3 we report the average logarithmic number of linearizations considering instances for which all four models return a feasible solution with the same action cost and for which we successfully count the number of linearizations: 93 instances in total. Values in boldface represent the maximum in each row. As in Experiment 1, we generate the linearizations through a depth-first search with a 30-minute time limit per instance. $\mathcal{T}^{\text{MILP+S}}$ performs at the same level as the weighted MaxSAT model $\mathcal{O}_C^{\text{maxSAT}}$, while $\mathcal{O}_O^{\text{MILP+S}}$ and $\mathcal{O}_C^{\text{MILP+S}}$ trail substantially. Bootstrap paired t-tests indicate no significant differences between any pair in terms of the mean number of linearizations while reflecting the pattern in the table in terms of the mean of the logarithm of the number of linearizations: both $\mathcal{T}^{\text{MILP+S}}$ and $\mathcal{O}_C^{\text{maxSAT}}$ have a significantly higher log mean number of linearizations than $\mathcal{O}_O^{\text{MILP+S}}$ ($p \leq 0.01$).

Computational Effort We compare the models with respect to solution times, focusing on the strengthened formulations – we evaluate the direct effect of the strengthening below (see Figure 3.12). Figure 3.8 shows a performance profile depicting the number of instances solved to optimality over time. $\mathcal{O}_O^{\text{MILP+S}}$ and $\mathcal{T}^{\text{MILP+S}}$ each solved 119 instances out of 138, while $\mathcal{O}_C^{\text{MILP+S}}$ and $\mathcal{O}_C^{\text{maxSAT}}$ each solved 111 instances, all of which were also solved by the other methods.

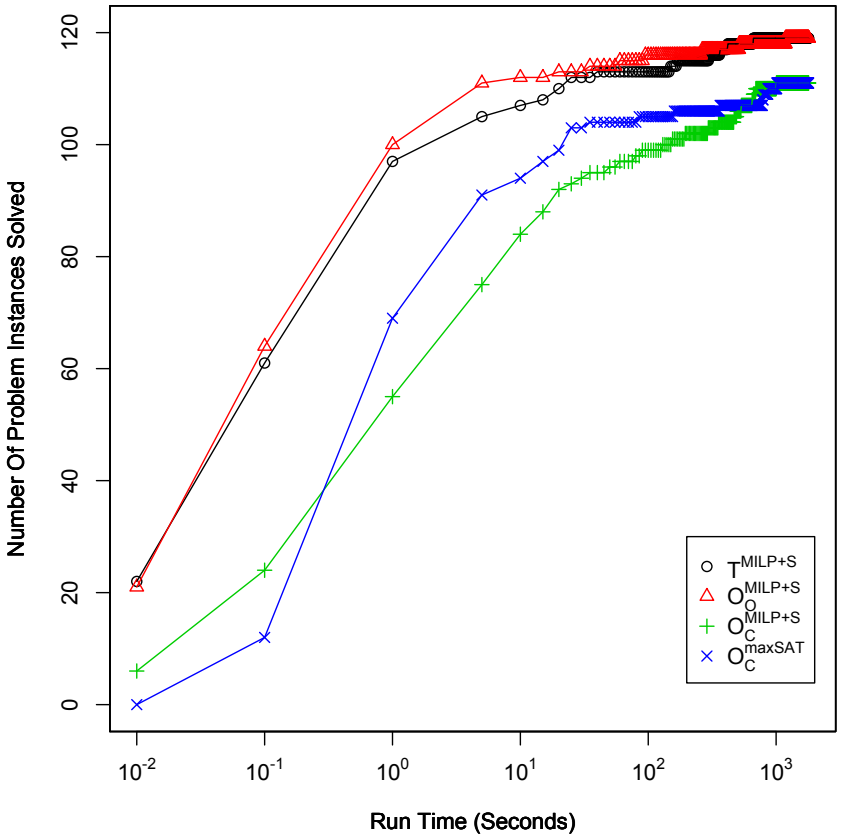

Figure 3.8: Performance profile (in log scale) for Experiment 2.

For the 111 instances solved by all methods, $\mathcal{O}_O^{\text{MILP+S}}$ and $\mathcal{T}^{\text{MILP+S}}$ were faster than the other approaches in all but one case. On average, $\mathcal{O}_O^{\text{MILP+S}}$ and $\mathcal{T}^{\text{MILP+S}}$ were approximately 27 times and 20 times faster than $\mathcal{O}_C^{\text{maxSAT}}$, respectively. A scatter plot comparing the run times of $\mathcal{T}^{\text{MILP+S}}$ and $\mathcal{O}_C^{\text{maxSAT}}$ for all instances is depicted in Figure 3.9. The plot comparing $\mathcal{O}_O^{\text{MILP+S}}$ and $\mathcal{O}_C^{\text{maxSAT}}$ on the same basis is similar. The speedups obtained both by $\mathcal{O}_O^{\text{MILP+S}}$ and $\mathcal{T}^{\text{MILP+S}}$ are likely due to a smaller formulation when compared to other models. As noted above, the number of constraints in $\mathcal{O}_C^{\text{MILP+S}}$ (which is derived directly from $\mathcal{O}_C^{\text{maxSAT}}$) grows cubically with the number of actions, while in $\mathcal{O}_O^{\text{MILP+S}}$ and $\mathcal{T}^{\text{MILP+S}}$ the growth is quadratic. This explanation is supported by Figure 3.10, which indicates that the difference in run times between $\mathcal{T}^{\text{MILP+S}}$ and $\mathcal{O}_C^{\text{maxSAT}}$ is positively correlated with the number of actions in the original plan.

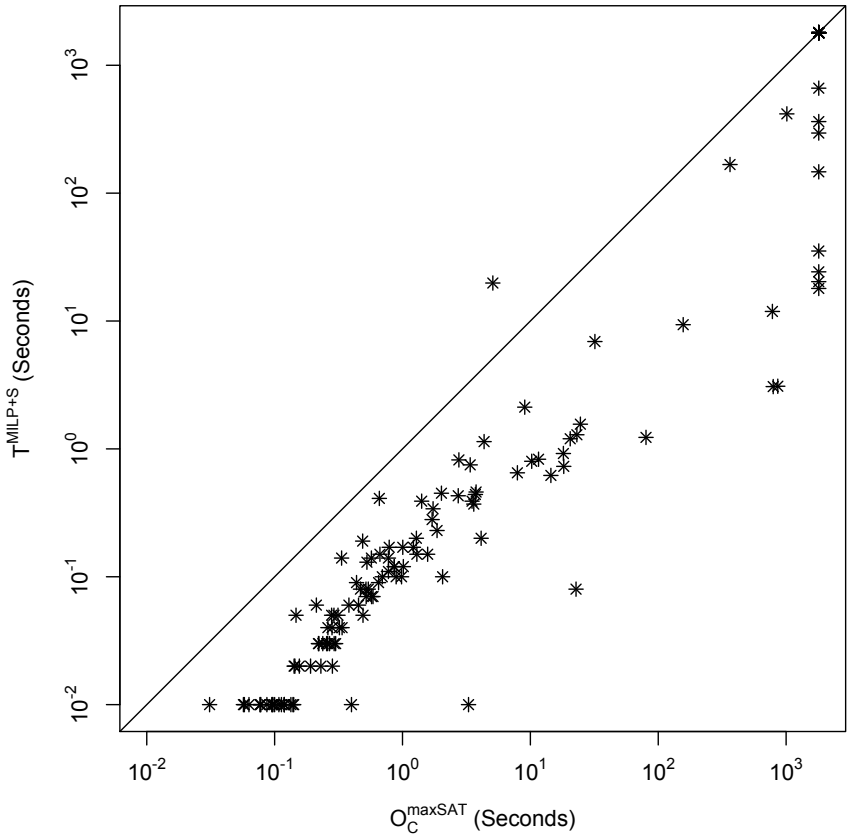

Figure 3.9: Run time comparison between $\mathcal{T}^{\text{MILP+S}}$ and $\mathcal{O}_C^{\text{maxSAT}}$ (in logarithmic scale).

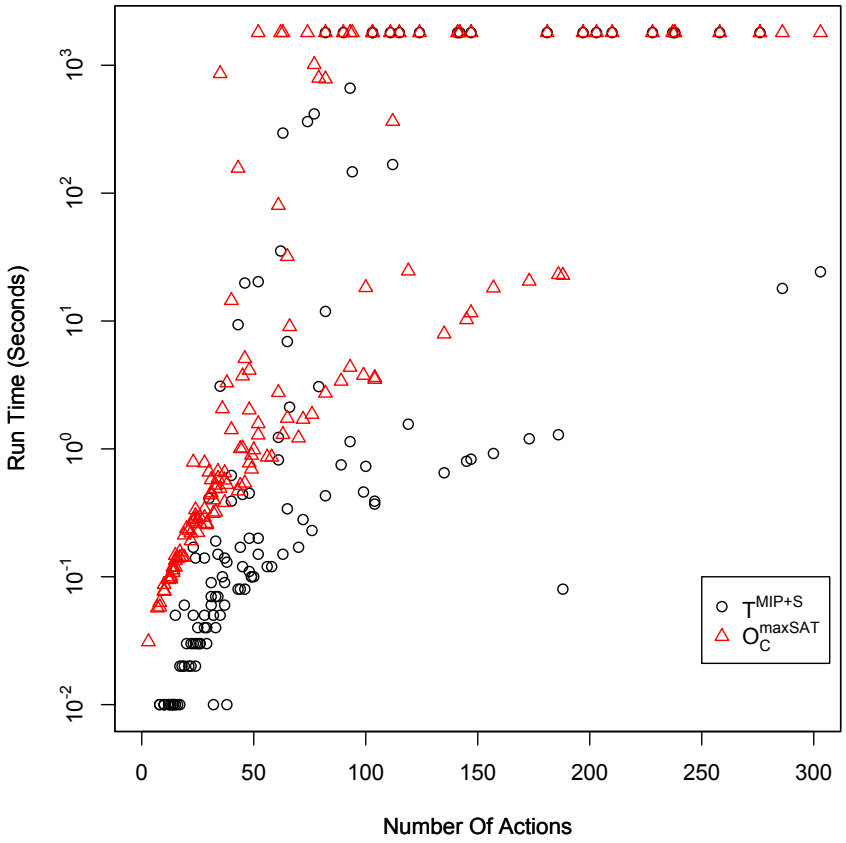

Figure 3.10: Run times of $\mathcal{O}_C^{\text{maxSAT}}$ and $\mathcal{T}^{\text{MILP+S}}$ and number of actions in the original plan (in logarithmic scale).

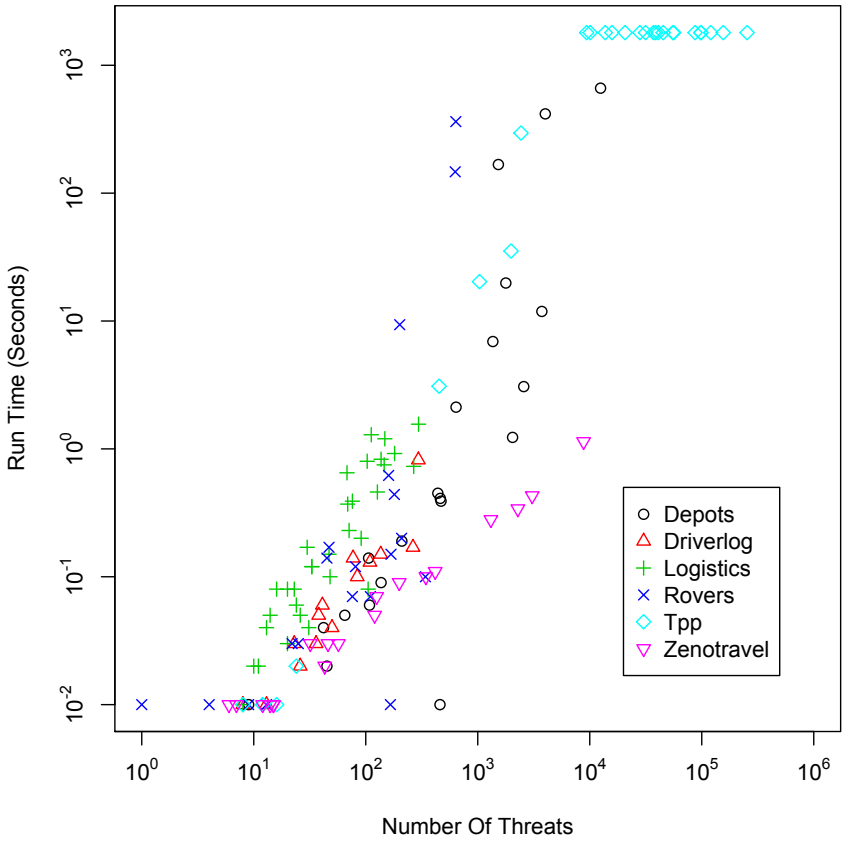

Figure 3.11: Run time performance of $\mathcal{T}^{\text{MILP+S}}$ and number of threat ordering constraints (in logarithmic scale).

All models grow linearly with the number of threats in the plan, here encoded by Constraints (3.2), which now becomes more relevant to the size of both $\mathcal{O}_O^{\text{MILP+S}}$ and $\mathcal{T}^{\text{MILP+S}}$. Figure 3.11 depicts the run time of $\mathcal{T}^{\text{MILP+S}}$ as a function of the number of threat ordering constraints for each domain, and strongly suggests a direct correlation. The 19 instances that were unsolved by $\mathcal{O}_O^{\text{MILP+S}}$ and $\mathcal{T}^{\text{MILP+S}}$ are from the `Tpp` domain and have more than 10,000 threats.

The Effect of the Strengthening Constraints In Figure 3.12 we plot run time comparisons between the pairs of base and strengthened models. The effect is significant for the instances that take longer than one second to solve. On average, the strengthened models are one order of magnitude faster than their corresponding base models.

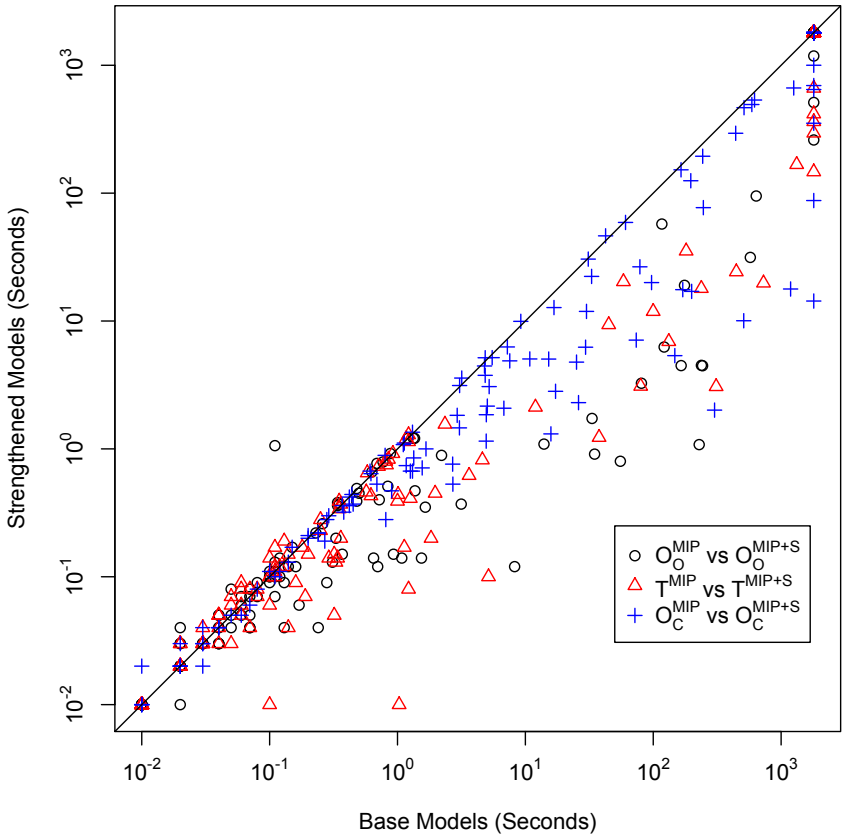

Figure 3.12: Effect of Constraints (3.15)-(3.24) on base models.

Summary of Results The performance profiles in Figures 3.7 and 3.8 clearly show the superior performance of $\mathcal{T}^{\text{MILP+S}}$ and $\mathcal{O}_O^{\text{MILP+S}}$ over $\mathcal{O}_C^{\text{maxSAT}}$ in terms of problem solving efficiency. Figure 3.12 demonstrates that the run time advantage is largely a result of the valid inequalities that we derived. The solution quality results are more nuanced, showing no significant differences in action cost in Experiment 2 or in mean number of linearizations in either experiment. The mean logarithmic number of linearizations does show superiority for $\mathcal{T}^{\text{MILP+S}}$ over $\mathcal{O}_C^{\text{maxSAT}}$ in Experiment 1 and for both $\mathcal{T}^{\text{MILP+S}}$ and $\mathcal{O}_C^{\text{maxSAT}}$ over $\mathcal{O}_O^{\text{MILP+S}}$ in Experiment 2.

3.8 Conclusion

We presented three MILP models for converting a sequential plan to a POP by optimizing a combination of action cost and one of three different proxy functions: the number of open ordering constraints [26], the number of closed ordering constraints [56] and a novel proxy function, temporal flexibility. We proved, through a set of counter-examples, that none of these functions dominates the others in terms of the number of linearizations in the resulting POPs. We then added valid strengthening constraints to these models, resulting in approximately an order of magnitude improvement in performance. Finally, we demonstrated that the two MILP models based on open ordering constraints and temporal flexibility achieve solution quality equal to that of the previous state-of-the-art MaxSAT approach [56] with a decrease in run time of approximately one order of magnitude.

An obvious direction for future work is to investigate the improvement of the MaxSAT model through the encoding of temporal variables rather than closed ordering constraints. The ideas of Crawford & Baker [18] and Frausto-Solis & Cruz-Chavez [31] may be useful here. We would also like to investigate other proxy functions and their relationship to POP flexibility.

In the next chapter, we introduce a decomposition method to find a LCFP to the complete planning problem Π. Our logic-based Benders decomposition (LBBD) consists of a master problem that selects and updates the set of actions that are to be ordered by a subproblem that performs a partial-order relaxation on the given set of actions. Our master problem is solved using a MILP model that is based on the operator counting constraints [64] that are described in Section 2.5.2. Our subproblem is solved using using a modified version of $\mathcal{O}_O^{\text{MILP}}$. We build on the LBBD introduced by Davies et al. [21] (see Section 2.6.2) to produce an optimal LCFP to Π.

Chapter 4

Mixed-Integer Linear Programming Models for Least Commitment Flexible Partial-Order Planning

The least commitment flexible planning criteria evaluates a partial-order plan with respect to two objectives: its total action cost and the number of linearizations it contains [72]. The first objective, which is equivalent to determining a cost-optimal plan to the planning problem Π, is typically optimized efficiently using sequential planners. The second objective, i.e., finding the maximum number of linearizations contained in a plan, is optimized using an approach denoted by partial-order relaxation. In this chapter we investigate solution methods to find a least commitment flexible POP (LCFP) to the original planning problem Π in an integrated way. We first focus on finding a cost-optimal POP to Π using a logic-based Benders decomposition (LBBD) that is similar to the approach by Davies et al. [21]. In particular, we show experimentally that the overall computational time needed for generating a cost-optimal POP is similar to that of generating a cost-optimal sequential plan. We then extend this work to find a LCFP to Π. To our knowledge, this is the first work that globally optimizes this objective. Our results show that our LCFP planner solves fewer problems to optimality using more computation time compared to our cost-optimal LBBD POP planner.

4.1 Introduction

Least commitment flexible planning aims to find a POP with the minimum total action cost and the maximum number of linearizations [56, 72]. In the previous chapter, we solved the problem of finding a least commitment flexible POP (LCFP) of a given plan. However, the solution quality of the resulting POP depends substantially on the set of actions contained in the initial plan, and previous works [1, 44, 26, 56, 72] do not address the question of how to find the set of actions that yields an optimal LCFP to the original planning problem.

We start investigating this issue by building on the recent planner that iteratively *selects* and *orders* actions to find a cost-optimal sequential plan [21]. Similar to Davies et al. [21], we decompose the planning problem into a master problem and a subproblem in a logic-based Benders decomposition (LBBD) framework. In our master problem, actions are selected based on a MILP model that incorporates operator counting constraints. In the subproblem, we find a POP with the minimum total action cost, restricted to the actions that are selected

previously by the master problem. To solve our subproblem, we use a modified version of $\mathcal{O}_O^{\text{MILP}}$ from Section 3.6 that only minimizes the total action cost of a POP. We denote this LBBD by $\mathcal{POP}^{\text{LBBD}}$. We compare our cost-optimal POP planner $\mathcal{POP}^{\text{LBBD}}$ to the cost-optimal sequential planner $OpSeq$ by Davies et al. [21] over 200 problem instances from the 2011 International Planning Competition. We compare the planners in terms of the total number of problems solved given a time limit and show the performance of $\mathcal{POP}^{\text{LBBD}}$ is almost as good as $OpSeq$, without enforcing the action from the final plan to be totally-ordered.

Next, we modify $\mathcal{POP}^{\text{LBBD}}$ to find a least commitment flexible POP to Π without the need of an initial plan. We call this model $\mathcal{POP}^{\text{LBBD}}_{\text{LCFP}}$, and compare it to $\mathcal{POP}^{\text{LBBD}}$ in terms of both the number of problems solved given a time limit and the run time performance. To our knowledge, $\mathcal{POP}^{\text{LBBD}}_{\text{LCFP}}$ is the first planner that generates POPs with minimum total action cost and maximum number of linearizations to the original planning problem.

Contributions. We present the first decomposition model to generate POPs with the minimum total action cost and the maximum number of linearizations to the complete planning problem without the need of an initial plan. We derive new operator counting constraints to strengthen both the master and the subproblems of our decomposition models. Finally, we experimentally show that over 200 problem instances across the International Planning Competition 2011 sequential optimal track benchmark, our least commitment flexible planner solves 67 to optimality.

4.2 Cost-Optimal Planning

The primary objective of the least commitment flexible planning is to generate a POP with the minimum total action cost. In this section we find a cost-optimal POP to the planning problem Π using a logic-lased Benders decomposition (LBBD) that is similar to the LBBD presented by Davies et al. [21]. We will denote the LBBD that is presented in this section as $\mathcal{POP}^{\text{LBBD}}$.

$\mathcal{POP}^{\text{LBBD}}$ is composed of a master problem $\mathcal{M}$ and a subproblem $\mathcal{S}(\mathcal{A})$. The master problem $\mathcal{M}$ is an MILP model that is similar to the master problem of Davies et al. [21]. The set of bounds on the number of times action $a \in A$ appears in the plan $C(a)$ is obtained from the master problem of $\mathcal{POP}^{\text{LBBD}}$ and then used to add actions to the set $\mathcal{A}$. Given a set of actions $\mathcal{A}$, the subproblem $\mathcal{S}(\mathcal{A})$ is a MILP model that checks the existence of a cost-optimal POP $\pi = \langle \mathcal{A}^*, \mathcal{O}, K \rangle$ with respect to the set of actions $\mathcal{A}$ such that $\mathcal{A}^* \subseteq \mathcal{A}$. To solve our subproblem, we use a modified version of $\mathcal{O}_O^{\text{MILP}}$ that only minimizes the total action cost of a POP since $\mathcal{O}_O^{\text{MILP}}$ has a smaller encoding size compared to $\mathcal{T}^{\text{MILP}}$ and $\mathcal{O}_C^{\text{MILP}}$. In the next three sections we describe the master MILP problem $\mathcal{M}$, the subproblem $\mathcal{S}(\mathcal{A})$, and the modified generalized landmark constraints that consists of a set of bounds $C(a)$ on the number of times action $a \in A$ appears in the plan, respectively.

4.2.1 Master Problem

The master problem $\mathcal{M}$ is a relaxation of the original planning problem Π that selects a set of actions with minimum total action cost. Similar to Davies et al. [21], we formulate $\mathcal{M}$ as a MILP model. Our formulation builds on the H^+ MILP formulation [42] and incorporates a set of operator counting constraints that count the number of actions in a relaxed representation of a plan [64] (see Section 2.5.2).

The formulation for $\mathcal{M}$ are as follows.

minimize $\sum_{a \in A} c_a z_a$

Constraints $(2.38) - (2.47)$

Constraints $(2.48) - (2.49)$

$$T_a + 1 \leq T_f + |A|(1 - x_{a,f}) \qquad \forall f \in ADD_a, a \in A \tag{4.1}$$

$$z_a \leq M_a u_f \qquad \forall f \in PRE_a, a \in A \backslash L^A \tag{4.2}$$

$$\sum_{a \in A} z_a \leq 2^{|F|} \tag{4.3}$$

Modified Generalized Landmark Constraints

where M_a is an arbitrarily large constant that is a minimum upper bound on the set of bounds $C(a)$ on the number of times each action $a \in A$ appears in a cost-optimal POP. If the number of actions required for a cost-optimal plan is not known, we choose a large constant, i.e. 100, that is experimentally observed to be an upper bound to the 2011 International Comptetition benchmark. Otherwise, if the total action cost of a valid POP π is known, the constant M_a can be set equal to the expression $\frac{c(\pi)}{c_a}$, since the total action cost of an incumbent POP $c(\pi)$ is guaranteed to be greater or equal to the action cost c_a times the cardinality of the action $a \in A$ in a cost-optimal POP. The constant M_a is always updated using the lowest known total action cost of a POP.

Constraint (4.1) is a modified version of Constraint (2.43). The constant $|A+1|$ can be reduced to $|A|$ because the maximum value T_a can take is equal to $|A - 1|$, that is, $T_a = |A - 1|$, the maximum value $T_a + 1$ can take is $|A|$. When $T_f = 0$ and $x_{a,f} = 0$, the minimum value required to satisfy this constraint is equal to $|A|$. Constraint (4.2) ensures that all the preconditions of all the actions are added. Without Constraint (4.2), the selected actions $a \in A$ can have preconditions $f \in PRE_a$ that are never added by another action such that $\forall f \in PRE_a \nexists a' \in \mathcal{A} \cap add_f$. Constraint (4.3) bounds the sum of all actions in any feasible plan by the maximum number of states, i.e. $2^{|F|}$, of the planning problem Π [21]. The final set of constraints are modified versions of generalized landmark constraints [21] that will be discussed in Section 4.3.2.

4.2.2 Subproblem

Given a set of actions $\mathcal{A}$, the subproblem $\mathcal{S}(\mathcal{A})$ is a MILP model that finds a cost-optimal POP $\pi = \langle \mathcal{A}^*, \mathcal{O}, K \rangle$ such that $\mathcal{A}^* \subseteq \mathcal{A}$. The subproblem described below is a simplified version of the MILP model $\mathcal{O}_O^{\text{MILP}}$ that was introduced in Chapter 3 with the addition of net state change constraints [9], landmark constraints [65, 80, 32]. The addition of valid inequalities that are introduced in section 3.6 are omitted since we have not investigated their interaction with the net state change and landmark constraints.

The objective function of the MILP model only minimizes the total action cost, as opposed to first minimizing the total action cost and then minimizing the total number of open ordering constraints $\mathcal{O}_O$ in a POP. Instead of using two sets of decision variables to encode the earliest start time and the latest finish time of an action, the MILP model uses an integer decision variable S_a for all $a \in \mathcal{A}$ to denote the step at which action a is executed.

The additional parameters used in $\mathcal{S}(\mathcal{A})$ are as follows.

- $\mathcal{L}^{\mathcal{A}}$ denotes the action landmark set that is extracted for the restricted planning problem $\Pi = \langle F, \mathcal{A}, I \backslash a, G \rangle$.
- $\mathcal{L}^{\mathcal{F}}$ denotes the causal landmark set that is extracted for the restricted planning problem. $\Pi = \langle F, \mathcal{A} \backslash pre_f, I \backslash f, G \rangle$.

The objective function and the linear constraints used in $\mathcal{S}(\mathcal{A})$ are as follows.

$$\text{minimize} \sum_{a \in \mathcal{A}} c_a Z_a$$

Constraints $(3.1) - (3.8)$

$$S_{a_I} = 0 \tag{4.4}$$

$$S_{a_G} = \sum_{a \in \mathcal{A}} Z_a + 1 \tag{4.5}$$

$$S_{a_i} + O_{a_i,a_j} \leq S_{a_j} + (|\mathcal{A}| - 1)(1 - O_{a_i,a_j}) \qquad \forall a_i, a_j \in \mathcal{A} \tag{4.6}$$

$$\sum_{a_i \equiv a, a_i \in \mathcal{A}} Z_{a_i} \geq 1 \qquad \forall a \in \mathcal{L}^{\mathcal{A}} \tag{4.7}$$

$$\sum_{a \in add_f} Z_a \geq 1 \qquad \forall f \in \mathcal{L}^{\mathcal{F}} \tag{4.8}$$

$$\sum_{a \in AP_f} Z_a + \sum_{a \in SP_f} Z_a - \sum_{a \in AC_f} Z_a \geq L_f \qquad \forall f \in F \tag{4.9}$$

$$\sum_{a \in AP_f} Z_a - \sum_{a \in AC_f} Z_a - \sum_{a \in SC_f} Z_a \leq U_f \qquad \forall f \in F \tag{4.10}$$

$$Z_{a_j} \leq Z_{a_i} \qquad \forall c_{a_i} > 0, a_i \equiv a_j, i < j, a_i, a_j \in \mathcal{A} \tag{4.11}$$

$$Z_{a_j} = O_{a_i,a_j} \qquad \forall c_{a_i} > 0, a_i \equiv a_j, i < j, a_i, a_j \in \mathcal{A} \tag{4.12}$$

Constraints (4.4)-(4.6) ensure that the action precedence graph does not contain cycles. Constraints (4.7)-(4.8) denote the action and causal landmark constraints, stating that the set of actions and the set of fluents that are identified as landmarks must be added by the POP, respectively. Constraints (4.9)-(4.10) denote the net state change constraints, which reason about the number of times the fluent $f \in F$ is added and deleted by the POP using the two pairs of sets AP_f, SP_f and AC_f, SC_f, respectively (see Section 2.5.2). A pair of actions $a_i, a_j \in \mathcal{A}$ are equivalent (denoted $a_i \equiv a_j$) if their preconditions, add effects, and delete effects are identical. Constraints (4.11) -(4.12) break symmetry between all pairs of equivalent actions to remove symmetrically identical solutions (and non-solutions) from the search space (see Section 3.7). The objective function minimizes the total action cost of a POP.

Next, we formally show that in every cost-optimal POP, the symmetry between two equivalent actions $a_i \equiv a_j \in \mathcal{A}$ can always be broken by introducing the ordering constraint $a_i \prec a_j \in \mathcal{O}$ for all $i < j$.

Lemma 8. *Let $\pi^* = \langle \mathcal{A}^*, \mathcal{O}^*, K^* \rangle$ be a cost-optimal POP. All pairs of actions $a_1, a_2 \in \mathcal{A}^*$ can be ordered with respect to each other with the ordering constraint $a_1 \prec a_2 \in \mathcal{O}$ if $c_{a_1} > 0$ and $a_1 \equiv a_2$.*

Proof. Let $\pi^* = \langle \mathcal{A}^*, \mathcal{O}^*, K^* \rangle$ be a cost-optimal POP. Further, suppose actions $a_1, a_2 \in \mathcal{A}^*$ are equivalent, i.e., $a_1 \equiv a_2$, and are not ordered with respect to each other. This implies that there exists a linear execution of actions $\mathcal{A}^*$ such that a_2 is executed immediately after a_1. Since the execution of a_2 does not change the state it is executed in (as the effects and preconditions of a_1 and a_2 are equivalent), the execution order of a_1 and a_2 can be reversed without impacting the validity of the POP. □

4.2.3 Benders Cuts: Modified Generalized Landmark Constraints

The modified generalized landmark constraints consist of a set of cardinality bounds on every action $a \in A$ (see Section 2.6.2). For each action $a \in A$, the generalized landmark constraints use one binary decision variable e_a that is associated with the bound $C(a)$. In addition, $e_a = 1$ only if the condition $z_a \geq C(a) + 1$ is satisfied for each action $a \in A$.

The linear constraints used in the modified generalized landmark constraints are as follows.

$$z_a \geq (C(a) + 1)e_a \qquad \forall a \in A \tag{4.13}$$

$$\sum_{a \in A} e_a = 1 \tag{4.14}$$

Constraints (4.13)-(4.14) ensure that exactly one bound on action $a \in A$ is satisfied. When $e_a = 1$, the decision variable z_a must take a value that is at least one greater than the action bound $C(a)$ such that $z_a \geq C(a)+1$ for exactly one action $a \in A$. From a planning perspective, the modified generalized landmark constraints represent the fact that the set of actions $\mathcal{A}$ require the addition of at least one more action $a \in A$ to contain a cost-optimal POP.

4.2.4 Updating the set of Actions $\mathcal{A}$

Before the subproblem $\mathcal{S}(\mathcal{A})$ is solved, the set of actions $\mathcal{A}$ must be updated. The modified generalized landmark constraints (4.13)- (4.14) ensure that the set of actions $\mathcal{A}'$ selected by the master problem $\mathcal{M}$ contains at least one new action $a \in \mathcal{A}'$ such that $\mathcal{A}' \nsubseteq \mathcal{A}$. We add the new actions $\mathcal{A}_{new} = \mathcal{A}' \backslash \mathcal{A}$ to the set $\mathcal{A}$ such that $\mathcal{A} = \mathcal{A} \cup \mathcal{A}_{new}$. This is equivalent to updating the action bounds by the maximum of the current action bound $C(a)$ and the optimal value of the decision variable z_a such that $C(a) \leftarrow \max\{z_a^*, C(a)\}$, where z_a^* denotes the optimal value of the decision variable z_a, for action $a \in A$. Note that the Constraints (4.13)-(4.14) are modified by updating the coefficient matrix associated with the decision variable e_a in constraint (4.13) of the MILP model.

4.2.5 Using the Incumbent Information

The information from an incumbent POP π can be used to prune the search space of the master problem $\mathcal{M}$. Given the total action cost of an incumbent POP $c(\pi)$, the constraint added to the master problem $\mathcal{M}$ is as follows:

$$\sum_{a \in A} c_a z_a < c(\pi) \tag{4.15}$$

Constraint 4.15 ensures that once an incumbent POP π is found, every action count with the total action cost $\sum_{a \in A} c_a z_a \geq c(\pi)$ is removed from the search space of the master problem $\mathcal{M}$.

4.2.6 The Cost-Optimal Logic-Based Benders Decomposition: $\mathcal{POP}^{\text{LBBD}}$

Given the master and the subproblems, $\mathcal{POP}^{\text{LBBD}}$ starts with an empty set of actions $\mathcal{A} = \emptyset$. At every iteration, at least one action $a \in A$ is added to the set $\mathcal{A}$. $\mathcal{POP}^{\text{LBBD}}$ iteratively adds actions to $\mathcal{A}$ until either $\mathcal{A}$ is guaranteed to contain a cost-optimal POP $\pi^* = \langle \mathcal{A}^*, \mathcal{O}, K \rangle$ to Π such that $\mathcal{A}^* \subseteq \mathcal{A}$, or Π is proven to be infeasible. The algorithmic description of $\mathcal{POP}^{\text{LBBD}}$ is as follows:

Algorithm 2 The algorithmic description of $\mathcal{POP}^{\text{LBBD}}$

Set Lower Bound Value $LB = 0$
Set Upper Bound Value $UB = \infty$
Set Incumbent POP $\pi_{inc} = \emptyset$
while $LB < UB$ **do**
  Solve the master problem $\mathcal{M}$
  if $\mathcal{M}$ is infeasible **then**
    Terminate
  else
    Let $\mathcal{A}'$ denote the set of actions returned by the master problem.
    if $\sum_{a \in \mathcal{A}'} c_a \geq UB$ **then**
      Terminate
    else
      Set $LB = \sum_{a \in \mathcal{A}'} c_a$
      Update $\mathcal{A} = \mathcal{A} \cup \mathcal{A}_{new}$ where $\mathcal{A}_{new} = \mathcal{A}' \backslash \mathcal{A}$
      Solve the subproblem $\mathcal{S}(\mathcal{A})$
      if $\mathcal{S}(\mathcal{A})$ is feasible **then**
        Let $\pi = \langle \mathcal{A}^*, \mathcal{O}, K \rangle$ denote the POP returned by the subproblem.
        if $c(\pi) < UB$ **then**
          Set $UB = c(\pi)$
          Set $\pi_{inc} = \pi$
        end if
      end if
      if $\mathcal{S}(\mathcal{A})$ is infeasible **or** $LB < UB$ **then**
        Update the modified generalized landmark constraint in $\mathcal{M}$.
      end if
    end if
  end if
end while

4.2.7 Comparison to Previous Logic-Based Benders Decompositions

The main differences between the LBBD described by Davies et al. [21] and $\mathcal{POP}^{\text{LBBD}}$ are as follows. First, $\mathcal{POP}^{\text{LBBD}}$ produces a POP instead of a sequential plan. Second, $\mathcal{POP}^{\text{LBBD}}$ maintains one set of modified generalized landmark constraints, i.e., Constraint (4.13), instead of adding a unique generalized landmark constraint after every iteration of the LBBD. Therefore the size of $\mathcal{M}$ stays constant, instead of growing linearly with the number of iterations of the LBBD. Third, $\mathcal{POP}^{\text{LBBD}}$ uses only one binary decision variable per action $a \in A$, instead of pre-allocating a binary decision variable for every unique integer value of the action bound $C(a)$. Therefore, $\mathcal{POP}^{\text{LBBD}}$ solves the complete planning problem and the LBBD described by Davies et al. [21] iteratively solves the bounded planning problem.

4.2.8 Proof of Correctness

In this section, we present theoretical results on the completeness and soundness of $\mathcal{POP}^{\text{LBBD}}$. First, we prove that given a set of actions $\mathcal{A}$, the subproblem $\mathcal{S}(\mathcal{A})$ always finds a cost-optimal POP. To do so, we prove that our temporal constraints (4.4)-(4.6) and the symmetry breaking constraints (4.11)-(4.12) do not remove any unique optimal solutions from the search space of $\mathcal{S}(\mathcal{A})$. Then, we prove the completeness and soundness of $\mathcal{POP}^{\text{LBBD}}$

Lemma 9. *Let $\pi = \langle \mathcal{A}, \mathcal{O}, K \rangle$ be a POP for the planning problem Π. The action precedence graph $\mathcal{O}$ does not contain cycles if and only if Constraints (4.4)-(4.6) assign each action $a \in \mathcal{A}$ to a step S_a.*

Proof. First, we show by contradiction that if a feasible solution is returned by the subproblem, its action precedence graph $\mathcal{O}$ does not contain cycles. Suppose the action precedence graph $\mathcal{O}$ contain cycles, and let there be a feasible step assignment S_a to each action $a \in \mathcal{A}$ with respect to Constraints (4.4)-(4.6). This implies that there exists a set of ordering constraints such that $a_1 \prec a_2, a_2 \prec a_3, a_3 \prec a_1 \in \mathcal{O}$. There does not exist a feasible set of step assignments $S_{a_1}, S_{a_2}, S_{a_3}$ to each action such that at most two out of three clauses can be satisfied in the formula $(S_{a_1} + 1 \leq S_{a_2}) \wedge (S_{a_2} + 1 \leq S_{a_3}) \wedge (S_{a_3} + 1 \leq S_{a_1})$ due to Constraint (4.6), which is a contradiction.

Next, we show by contradiction that if there exists a POP $\pi = \langle \mathcal{A}, \mathcal{O}, K \rangle$, the subproblem returns a valid POP with a feasible step assignment. Suppose the action precedence graph $\mathcal{O}$ contains no cycles, and there does not exist a feasible step assignment S_a to each action $a \in \mathcal{A}$ with respect to Constraints (4.4)-(4.6). This implies that, for any linear execution of the actions $a \in \mathcal{A}$, there does not exist a feasible step assignment with respect to Constraints (4.4)-(4.6). The actions a_I and a_G will always be assigned to the steps 0 and $|\mathcal{A}| + 1$, respectively. These assignments are feasible with respect to Constraints (4.4)-(4.6). The number of steps required for a linear execution of $a \in \mathcal{A}$ is equal to $|\mathcal{A}|$. Let $L = [a_1, a_2, ...a_n]$ be a linear execution of $a \in \mathcal{A}$ (w.r.t. $\mathcal{O}$). Given L, each action $a \in \mathcal{A}$ can be assigned to a feasible step with respect to Constraint (4.6) such that $S_{a_i} = i$, which is a contradiction. □

Theorem 10. *Let $\Pi = \langle F, A, I, G \rangle$ be a planning problem. In finite number of iterations $i \in I$, $\mathcal{POP}^{LBBD}$ returns a cost-optimal POP π^* if and only if Π is feasible.*

Proof. We will first prove by induction that the following properties are valid at each iteration of $\mathcal{POP}^{\text{LBBD}}$ if the planning problem Π is feasible,

1. the optimal objective function value of the master problem $\mathcal{M}$ is a lower bound LB on the cost of an optimal POP π^* such that $LB \leq c(\pi^*)$, and

2. at least one of the two must hold:

   (a) there exists at least one action cost-optimal POP $\pi^* = \langle \mathcal{A}^*, \mathcal{O}, K \rangle$ with the action set $\mathcal{A}^*$ that is either feasible to the subproblem such that $\mathcal{A}^* \subseteq \mathcal{A}$, or

   (b) there exists at least one action count z_a^* for all $a \in A$ that is feasible to the master problem, that is, z_a^* denotes a feasible value assignment to the action count decision variables z_a for all actions $a \in A$.

Base Case: In the first iteration $i = 1$, the action set $\mathcal{A}$ and the incumbent POP π_{inc} are empty, the action bounds $C(a)$ on every action $a \in A$ are set to 0, and the lower and upper bounds are set to $LB = 0$ and $UB = \infty$, respectively. Since $C(a) = 0$ for all $a \in A$, solving the master problem $\mathcal{M}$ in the base case is equivalent to solving the relaxation of Π with operator counting heuristics [64]. Property 1 holds in the base case since solving the delete relaxation of Π is guaranteed to generate a total action that is less than or equal to the optimal action cost of Π.

Property 2 holds in the base case since every action cost-optimal POP has a corresponding action count z_a^* for all actions $a \in A$ that is feasible to the master problem.

Induction Hypothesis: Assume that up to iteration $i < k$, Properties 1 and 2 hold. This implies that LB is a lower bound on the optimal action cost of the planning problem Π, and either the incumbent POP π_{inc} is updated with a cost-optimal POP $\pi^* = \langle \mathcal{A}^*, \mathcal{O}, K \rangle$ such that $\pi_{inc} = \pi^*$ (i.e., $\mathcal{A}^* \subseteq \mathcal{A}$), or there exists an action cost-optimal POP π^* whose action count z_a^*, $a \in A$ is feasible to the master problem $\mathcal{M}$.

Induction Step: Let $i = k$ be the next iteration of $\mathcal{POP}^{\text{LBBD}}$. Given the action set $\mathcal{A}$ and the incumbent plan π_{inc}, solving the master problem $\mathcal{M}$ either returns an optimal action count z_a^*, $a \in A$ that corresponds to a cost-optimal set $\mathcal{A}'$, or infeasibility.

If the master problem $\mathcal{M}$ is infeasible and π_{inc} is empty, $\mathcal{POP}^{\text{LBBD}}$ proves the infeasibility of the original planning problem and terminates because there does not exist a relaxed plan with the action set $\mathcal{A}^* \nsubseteq \mathcal{A}$, and hence there does not exist a POP $\pi^* = \langle \mathcal{A}^*, \mathcal{O}, K \rangle$ such that $\mathcal{A}^* \subseteq \mathcal{A}$ since $\pi_{inc} = \emptyset$.

If the master problem $\mathcal{M}$ is infeasible and π_{inc} is not empty, $\mathcal{POP}^{\text{LBBD}}$ proves that π_{inc} is cost-optimal and terminates. Property 1 is satisfied since the master problem $\mathcal{M}$ must be infeasible due to Constraint (4.15) and therefore the lower bound is less than or equal to the upper bound $LB \leq UB$. Property 2 is satisfied since the infeasibility of the master problem $\mathcal{M}$ implies that there does not exist a set of feasible action counts z_a^* to the master problem $\mathcal{M}$ for all actions $a \in A$ with the total action cost less than the total action cost of the incumbent POP π_{inc} such that $\sum_{a \in A} c_a z_a^* < UB = c(\pi_{inc})$. Further, the non-empty incumbent POP $\pi_{inc} = \langle \mathcal{A}^*, \mathcal{O}, K \rangle$ implies that $\mathcal{A}^* \subseteq \mathcal{A}$.

If the master problem $\mathcal{M}$ is feasible and the set $\mathcal{A}'$ is not empty, the action set $\mathcal{A}$ is updated with the addition of new actions from the set $\mathcal{A}'$ as described in Section 4.2.4. If solving the subproblem $\mathcal{S}(\mathcal{A})$ returns a POP π where $c(\pi) < UB$, both π_{inc} and UB are updated such that $\pi_{inc} = \pi$ and $UB = c(\pi)$. Property 1 is satisfied since solving the master problem $\mathcal{M}$ returned a relaxed plan with the action set $\mathcal{A}'$ and the total action cost $\sum_{a \in \mathcal{A}'} c_a \leq c(\pi^*)$. Property 2 is satisfied because either a cost-optimal POP $\pi^* = \langle \mathcal{A}^*, \mathcal{O}, K \rangle$ is found by solving the subproblem $\mathcal{S}(\mathcal{A})$ such that $\pi_{inc} = \pi^*$ where $\mathcal{A}^* \subseteq \mathcal{A}$, or there exists a set of action counts z_a^*, $a \in A$ that correspond to the set $\mathcal{A}' \nsubseteq \mathcal{A}$ with the total action cost of $\sum_{a \in \mathcal{A}'}$. The total action cost $\sum_{a \in \mathcal{A}'}$ is equal to the optimal action cost of the planning problem Π such that $\sum_{a \in \mathcal{A}'} = c(\pi^*)$. Given the action bounds $C(a)$ for all actions $a \in A$, there must exist a set of action counts z_a^* for all actions $a \in A$ that is a feasible solution to the master problem $\mathcal{M}$ because $\mathcal{A}' \nsubseteq \mathcal{A}$ implies the existence of an action $a_i \notin \mathcal{A}$ with the corresponding action count $z_a^* \geq C(a) + 1$ for at least one action $a_i \equiv a \in A$.

There are two possible termination conditions: $\mathcal{POP}^{\text{LBBD}}$ either terminates with a cost-optimal POP π^* if Π is feasible, or proves the infeasibility of Π if Π is infeasible. If Π is feasible, $\mathcal{POP}^{\text{LBBD}}$ will find a cost-optimal POP π^* at some iteration k and prove its optimality since either $LB = UB$, or by ensuring that there does not exist any relaxed plan with the action set $\mathcal{A}'$ and total action cost $\sum_{a \in \mathcal{A}'} c_a$ that is lower than the total action cost of the incumbent POP due to Constraint (4.15) such that $\sum_{a \in \mathcal{A}'} c_a < UB = \pi_{inc}$. If Π is infeasible, $\mathcal{POP}^{\text{LBBD}}$ proves the infeasibility of Π by showing that there does not exist a cost-optimal POP $\pi^* = \langle \mathcal{A}^*, \mathcal{O}, K \rangle$ such that $\mathcal{A}^* \subseteq \mathcal{A}$ since $\mathcal{A}$ is bounded by $|\mathcal{A}| \leq 2^{|F|} \times |A|$.

□

In the next section, we demonstrate how $\mathcal{POP}^{\text{LBBD}}$ works on a small example.

4.2.9 Example: Solving a Simple Planning Problem with $\mathcal{POP}^{\text{LBBD}}$

In this example we demonstrate how $\mathcal{POP}^{\text{LBBD}}$ solves a simple planning problem. Let us consider a planning problem represented by the tuple $\Pi = \langle F, A, I, G \rangle$, where $F = \{f, g\}$, $A = \{a_I, a_1, a_2, a_3, a_G\}$, $I = ADD_{a_I} = \emptyset$, $G = PRE_{a_G} = \{f, g\}$, $ADD_{a_1} = f$, $DEL_{a_1} = g$, $ADD_{a_2} = g$, $DEL_{a_2} = f$, $ADD_{a_3} = \{f, g\}$. Further, let each action be associated with the actions costs $c_{a_1} = c_{a_2} = 1$, $c_{a_3} = 3$. $\pi^* = \langle \mathcal{A}^*, \mathcal{O}, K \rangle$ is a cost optimal POP to Π where $\mathcal{A}^* = \{a_I, a_3, a_G\}$, $\mathcal{O} = \{a_I \prec a_3, a_3 \prec a_G\}$ and $K = \{\kappa(a_3, a_G, f), \kappa(a_3, a_G, g)\}$ with the total action cost of 3. In Figure 4.1, the table summarizes the example planning problem Π and the graph with the directed arcs represents an action cost-optimal POP to Π where the nodes represent the actions $a \in \mathcal{A}$ and the arcs represent the ordering constraints $a_i \prec a_j \in \mathcal{O}$ between the actions $a_i, a_j \in \mathcal{A}$.

Actions	Pre	Add	Del	Cost
a_I	-	-	-	0
a_1	-	f	g	1
a_2	-	g	f	1
a_3	-	f,g	-	3
a_G	f,g	-	-	0

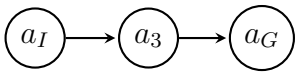

Figure 4.1: On the left: The planning problem represented by the table. On the right: The action-cost optimal POP for the example planning problem.

The maximum number of unique states that can be reached by any plan is $2^{|F|} = 4$. Similar to Davies et al. [21], we use this information to bound (or prove the infeasibility of) Π. If the number of the actions is more than $2^{|F|}$, i.e., $\mathcal{A}' > 2^{|F|} = 4$, Π must be infeasible.

$\mathcal{POP}^{\text{LBBD}}$ starts with an empty action set $\mathcal{A} = \emptyset$. In its first iteration, $\mathcal{M}$ selects the actions a_1 and a_2 because the delete effects of actions a_1 and a_2 are ignored. The set of actions and the lower bound are updated such that $\mathcal{A} = \{a_1, a_2\}$ and $LB = 2$. Given the set $\mathcal{A}$, the subproblem searches for a cost-optimal POP $\pi^* = \langle \mathcal{A}^*, \mathcal{O}, K \rangle$ such that $\mathcal{A}^* \subseteq \mathcal{A}$. Since there does not exist a POP given the set $\mathcal{A}$, $\mathcal{S}(\mathcal{A})$ returns infeasibility and the modified generalized landmark constraints are updated.

In the second iteration, $\mathcal{M}$ can select either: (1) a_1 twice and a_2 once, or (2) a_2 once and a_2 twice, or (3) a_3 once. Note that all three options are equivalent cost-optimal relaxed plans for $\mathcal{M}$, with the total action cost of 3. Let us assume that $\mathcal{M}$ select option (1) as the cost-optimal relaxed plan. The set of actions and the lower bound are updated such that $\mathcal{A} = \{a_1, a_1', a_2\}$ and $LB = 3$. Given the set $\mathcal{A}$, $\mathcal{S}(\mathcal{A})$ will again return infeasibility.

In the third iteration, let us assume $\mathcal{M}$ selects option (2) as the cost-optimal relaxed plan. The set of actions is updated such that $\mathcal{A} = \{a_1, a_1', a_2, a_2'\}$. The $\mathcal{S}(\mathcal{A})$ returns infeasibility and the modified generalized landmark constraints are updated.

In the fourth iteration, $\mathcal{M}$ can only select option (3) as the cost-optimal relaxed plan. The set of actions is updated such that $\mathcal{A} = \{a_1, a_1', a_2, a_2', a_3\}$. Given the set of actions $\mathcal{A}$, $\mathcal{S}(\mathcal{A})$ returns the cost-optimal POP π^*. The incumbent plan and the upper bound are updated such that $\pi = \pi^*$ and $UB = 3$, respectively. Since $LB \geq UB$, and $\mathcal{POP}^{\text{LBBD}}$ terminates with the cost-optimal POP π^* to Π with the total action cost of 3.

With this example, we have demonstrated that $\mathcal{POP}^{\text{LBBD}}$ performs a complete search to find a set of actions $\mathcal{A}$ that contains a cost-optimal POP $\pi^* = \langle \mathcal{A}^*, \mathcal{O}, K \rangle$ to Π such that $\mathcal{A}^* \subseteq \mathcal{A}$. We further showed how $\mathcal{POP}^{\text{LBBD}}$ uses the duality information, i.e., the lower and upper bounds, to prove the optimality of its incumbent plan and terminate its search.

In the next section we modify $\mathcal{POP}^{\text{LBBD}}$ to find the least commitment flexible POP to the planning problem Π.

4.3 Optimal Least Commitment Flexible Planning

The main motivation for generating a POP instead of a sequential plan is the execution flexibility it provides to an agent (see Section 3.1). Unlike a sequential plan, a POP compactly represents a set of plans, i.e., linearizations, from which an agent can dynamically choose from during execution. While there are methods, such as partial-order relaxation, that optimize the flexibility of an input plan (see Section 2.7.2), the problem of finding a Least Commitment Flexible POP (LCFP) to the complete planning problem has not been investigated. In this section, we formally define the least commitment flexible POP of the planning problem.

Definition 11. *(Least commitment flexible POP of the planning problem). Let $P = \langle \mathcal{A}, \mathcal{O}, K \rangle$ be a valid POP for the planning problem Π. Moreover, let c_a be a non-negative cost associated with each action $a \in \mathcal{A}$. P is a least commitment flexible POP (LCFP) of the planning problem Π iff*

(i) For all valid POPs $P' = \langle \mathcal{A}', \mathcal{O}', K' \rangle$ to Π, we have $\sum_{a \in \mathcal{A}'} c_a \geq \sum_{a \in \mathcal{A}} c_a$; and

(ii) For all valid POPs $P' = \langle \mathcal{A}', \mathcal{O}', K' \rangle$ to Π and $\sum_{a \in \mathcal{A}'} c_a = \sum_{a \in \mathcal{A}} c_a$, the number of actions with zero action cost of P is less than or equal to the number of actions with zero action cost of P' (i.e., $\sum_{c_a=0, a \in \mathcal{A}} 1 \leq \sum_{c_a=0, a \in \mathcal{A}'} 1$); and

(iii) For all valid POPs $P' = \langle \mathcal{A}', \mathcal{O}', K' \rangle$ to Π and $\sum_{c_a=0, a \in \mathcal{A}} 1 = \sum_{c_a=0, a \in \mathcal{A}'} 1$, the number of linearizations of P is greater or equal to the number of linearizations of P'.

Definition 11 differs from the LCFP definition introduced in Chapter 3 for two reasons. The first difference is that the scope of Definition 11 is the complete planning problem, as opposed to that of Definition 4, which is restricted to finding the LCFP of an input plan. The second difference is the tie-breaking rule for POPs with equal action costs (i.e., condition (ii)). When there are two POPs with equal action costs, we consider the POP with the lower number of actions with zero action cost. The main reason for this tie-breaking rule is that the minimization of the number of actions in a POP competes with the maximization of the linearizations in a POP. Without this tie-breaking rule, the number of linearizations in a POP can be trivially optimized by adding infinite number of zero-cost actions to the POP.

We modify $\mathcal{POP}^{\text{LBBD}}$ to find a LCFP to Π. We denote our least commitment flexible planner as $\mathcal{POP}^{\text{LBBD}}_{\text{LCFP}}$. $\mathcal{POP}^{\text{LBBD}}_{\text{LCFP}}$ uses the same master problem and the subproblem as $\mathcal{POP}^{\text{LBBD}}$, except the objective function of its subproblem. The subproblem of $\mathcal{POP}^{\text{LBBD}}_{\text{LCFP}}$ first minimizes the total action cost, then minimizes the total number of zero-cost actions, and then minimizes the total number of open ordering constraints in a POP. We optimize the total open ordering constraints in a POP because it is the fastest proxy objective to optimize and has the equivalent solution quality compared to the other proxy objectives, as demonstrated empirically in Chapter 3.

In the next section we compare the computational efficiency of our POP planners against the sequential planner $OpSeq$.

4.4 Computational Results

We evaluate the computational efficiency of four different cost-optimal planners on the International Planning Competition 2011 benchmark. We compare $\mathcal{POP}^{\text{LBBD}}$ as it is described in Section 4.2 to $OpSeq$, to another

LBBD model $\mathcal{POP}_2^{\text{LBBD}}$ and to $\mathcal{POP}_{\text{LCFP}}^{\text{LBBD}}$.

The only difference between $\mathcal{POP}^{\text{LBBD}}$ and $\mathcal{POP}_2^{\text{LBBD}}$ is as follows. In the master problem of $\mathcal{POP}_2^{\text{LBBD}}$, we use the generalized landmarks described in Davies et al. [21] without using the conflict analysis method. As a result, the input set of actions $\mathcal{A}$ for the subproblem of $\mathcal{POP}_2^{\text{LBBD}}$ is always equivalent to the set $\mathcal{A}'$ that is selected by the master problem such that $\mathcal{A} = \mathcal{A}'$.

Note that $\mathcal{POP}_{\text{LCFP}}^{\text{LBBD}}$ is also a cost-optimal planner with secondary and tertiary objectives. We run $\mathcal{POP}^{\text{LBBD}}$, $\mathcal{POP}_2^{\text{LBBD}}$ and $\mathcal{POP}_{\text{LCFP}}^{\text{LBBD}}$ on MacPro computer with 3.5 GHz 6-Core Intel Xeon E5, with 1-hour time limit for each problem instance. The MILP models are solved using IBM ILOG CPLEX 12.6.3 with 1 thread. Due to a bug in the source code of $OpSeq$, we report the results presented by Davies et al. [21].

	Number of Problems Solved to Optimality			
Domain	$\mathcal{POP}_{\text{LCFP}}^{\text{LBBD}}$	$\mathcal{POP}_2^{\text{LBBD}}$	$\mathcal{POP}^{\text{LBBD}}$	$OpSeq$
barman	0	0	0	0
elevators	0	0	0	**11**
nomystery	2	3	4	**5**
openstacks	0	0	0	0
parcprinter	**20**	**20**	**20**	**20**
pegsol	6	7	**8**	2
scananalyzer	0	**2**	1	1
sokoban	1	**2**	**2**	0
transport	0	0	0	**5**
visitall	18	**20**	**20**	14
woodworking	**20**	**20**	**20**	**20**
Total	67	74	75	78

Table 4.1: Coverage of problem instances across International Planning Competition 2011 sequential optimal track benchmarks.

Table 1 summarizes the total number of problem instances solved to optimality across International Planning Competition 2011 sequential optimal track benchmarks. In comparison to $OpSeq$, the POP planners i.e., $\mathcal{POP}_2^{\text{LBBD}}$, $\mathcal{POP}^{\text{LBBD}}$ and $\mathcal{POP}_{\text{LCFP}}^{\text{LBBD}}$, solved 4, 3 and 11 fewer problem instances within the 1-hour time limit, respectively. The analysis of Table 1 shows the performance trade-off across different domains. In *elevators* and *transport* domains, $OpSeq$ performed significantly better compared to the POP planners. In contrast, our POP planners performed better compared to $OpSeq$ in *pegsol* and *visitall* domains. The pairwise comparison of $\mathcal{POP}^{\text{LBBD}}$ and $\mathcal{POP}_2^{\text{LBBD}}$ show that the performance of the two POP planners are almost identical across different domains. The pairwise comparison of $\mathcal{POP}^{\text{LBBD}}$ and $\mathcal{POP}_{\text{LCFP}}^{\text{LBBD}}$ show that our least commitment flexible planning objective takes significantly more time to optimize. In fact, in all 8 instances for which $\mathcal{POP}^{\text{LBBD}}$ finds an optimal solution and $\mathcal{POP}_{\text{LCFP}}^{\text{LBBD}}$ does not, $\mathcal{POP}_{\text{LCFP}}^{\text{LBBD}}$ terminates with an incumbent POP that is cost optimal.

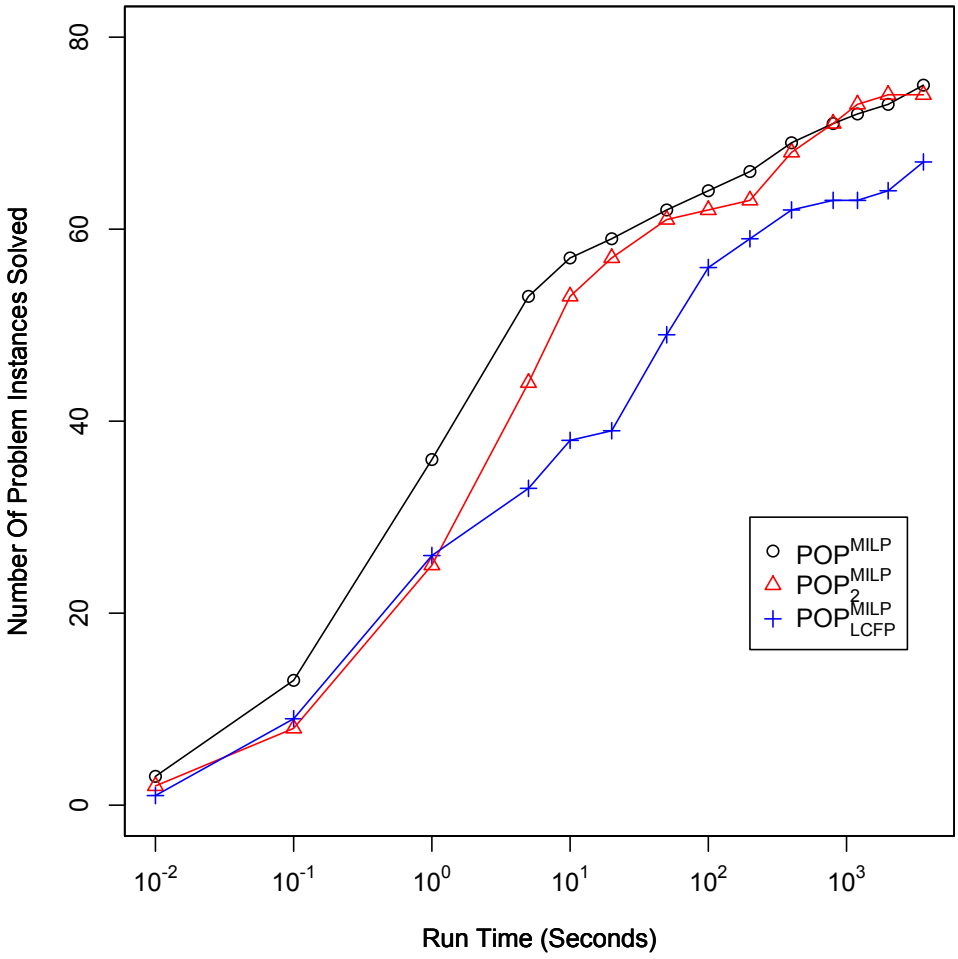

Figure 4.2: Performance profile (in log scale) for different decomposition models.

Figure 4.2 shows a performance profile depicting the number of instances solved to optimality over time. The comparison of the cost optimal POP planners (i.e., $\mathcal{POP}^{\text{LBBD}}$ and $\mathcal{POP}^{\text{LBBD}}_{2}$), against $\mathcal{POP}^{\text{LBBD}}_{\text{LCFP}}$ shows that the least commitment flexible planning objective takes significantly more time to optimize. It can be observed that both cost-optimal POP planners outperform $\mathcal{POP}^{\text{LBBD}}_{\text{LCFP}}$ across all time points that are greater than a second to solve.

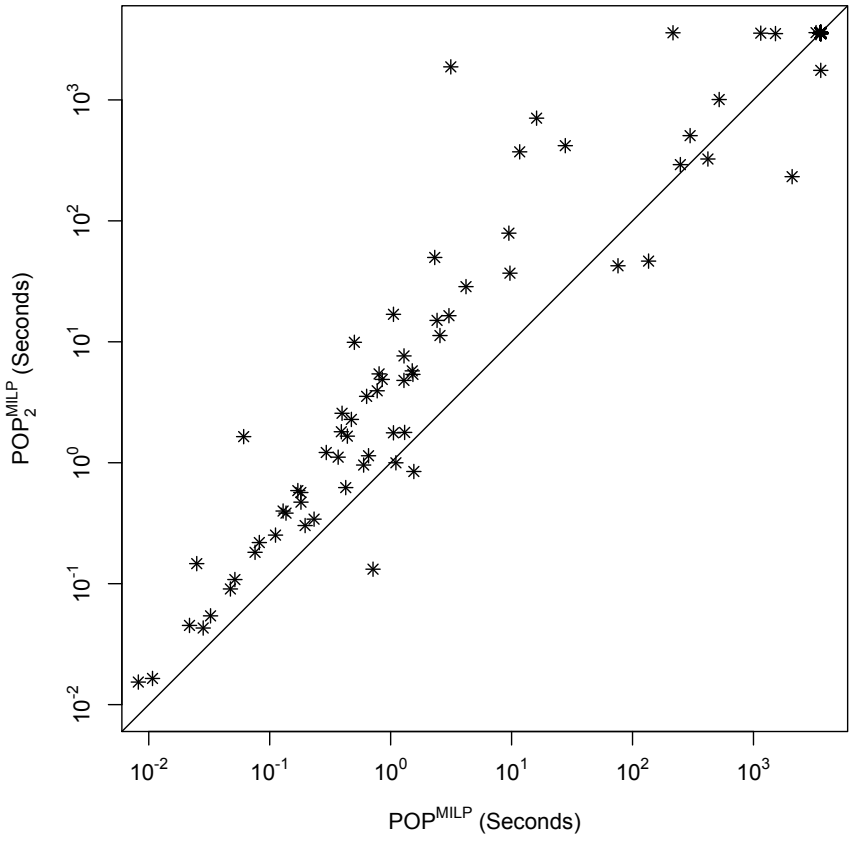

Figure 4.3: Run time comparison between $\mathcal{POP}^{\text{LBBD}}$ and $\mathcal{POP}_2^{\text{LBBD}}$ (in logarithmic scale).

Figure 4.3 is a scatter plot that compares the run times of the two cost-optimal POP planners. The pairwise comparison of $\mathcal{POP}^{\text{LBBD}}$ and $\mathcal{POP}_2^{\text{LBBD}}$ shows that for the problem instances that take less than 100 seconds to optimize, $\mathcal{POP}^{\text{LBBD}}$ outperforms $\mathcal{POP}_2^{\text{LBBD}}$. For the problem instances that take longer than 100 seconds to optimize, $\mathcal{POP}^{\text{LBBD}}$ performs slightly better than $\mathcal{POP}_2^{\text{LBBD}}$.

4.5 Discussion And Future Work

Davies et al. [21] report that most of the problem instances in the International Planning Competition 2011 sequential optimal track benchmarks can be solved to optimality without using more than two equivalent actions. Therefore in the implementation of $OpSeq$, Davies et al. [21] pre-allocate decision variables up to 2 for every action $a \in A$. In theory, $OpSeq$ or $\mathcal{POP}_2^{\text{LBBD}}$ can perform worse on the problem instances that only have cost-optimal plans with at least three equivalent actions.

The selection of the best master problem is still an open question. The master problem of $\mathcal{POP}_{\text{LCFP}}^{\text{LBBD}}$ does not provide an informative lower bound with respect to its secondary objective. Therefore, $\mathcal{POP}_{\text{LCFP}}^{\text{LBBD}}$ naively searches through all the cost-optimal POPs, and returns the one with the minimum number of open ordering constraints. A MILP formulation that represents some relaxation of the ordering constraints in a plan can reduce the number of iterations $\mathcal{POP}_{\text{LCFP}}^{\text{LBBD}}$ goes through to exhaustively search all the cost-optimal POPs.

Inside of $OpSeq$, Davies et al. [21] use Conflict-Directed Clause Learning to produce sets of generalized landmarks with significantly smaller sizes. Similar to SAT planners, MILP solvers also include minimal conflict analysis methods which can be utilized to produce stronger modified generalized landmarks and this could be investigated as future work to improve the performance of $\mathcal{POP}_2^{\text{LBBD}}$.

Different utilization of the conflict analysis methods inside the causal link-based POP formulations is another interesting area of future research. Specifically, understading how conflict analysis methods can be utilized to explain the infeasibility of the subproblem $\mathcal{S}(\mathcal{A})$ with missing causal links (instead of missing actions as it is done in $OpSeq$) can produce more informative modified generalized landmarks.

4.6 Conclusion

Overall performance of both $\mathcal{POP}^{\text{LBBD}}$ and $\mathcal{POP}_2^{\text{LBBD}}$ show promising results in terms of generating cost-optimal POPs using a LBBD. We showed experimentally that the overall computational time for generating a cost-optimal POP using a causal link-based subproblem is similar to that of generating a cost-optimal sequential plan. Given this result, we extended our best performing cost-optimal POP planner to solve the least commitment flexible planning problem. We experimentally showed that over 90% of the problem instances for which the best performing cost-optimal POP planner i.e., $\mathcal{POP}^{\text{LBBD}}$, finds a cost-optimal POP, the LCFP planner i.e., $\mathcal{POP}_{\text{LCFP}}^{\text{LBBD}}$, also finds a LCFP. To our knowledge, $\mathcal{POP}_{\text{LCFP}}^{\text{LBBD}}$ is the first of its kind for generating a LCFP to Π.

Chapter 5

Conclusions and Future Work

The central thesis of this dissertation is that the Mixed-Integer Linear Programming (MILP) technology can be effective in generating least commitment partial-order plans. In this chapter, we first summarize the work that is presented in the previous chapters and re-state our contributions. We conclude this dissertation by presenting some insight and intuition for future work.

5.1 Summary and Contributions

In this section, we summarize the work presented in the previous chapters and re-state our contributions.

5.1.1 Mixed-Integer Linear Programming Models for Optimal Least Commitment Flexible Partial-Order Plan of an Initial Plan

Summary In Chapter 3, we addressed the problem of finding a least commitment flexible partial-order plan (LCFP) given an input set of actions that form a valid plan. We formally defined LFCP as a partial-order plan (POP) with minimum total action cost and maximum number of linearizations. As the latter objective is computationally impractical, we investigated three proxy objective functions to maximize the number of linearizations in a POP. Two of the proxy objective functions are from the partial-order relaxation literature [26, 56] while the third is adapted from the scheduling literature [20] and applied to POP planning for the first time here. We formally demonstrated through counter-examples that none of the proxy objective functions dominate the other two with respect to the number of linearizations they produce. We then presented three base MILP models to optimize our proxy objective functions and introduced new linear constraints to strengthen the base models. Experimentally, we showed that two of our MILP models result in equivalent or slightly better solution quality with savings of approximately one order of magnitude in computation time compared to the state-of-the-art MaxSAT model [56].

Contributions We showed that MILP models are more effective in finding the LCFP of an input plan, compared to the state-of-the-art weighted SAT model [56]. We presented two base MILP models with significantly smaller encoding sizes compared to the propositional representation of the problem. We strengthened the base MILP models by introducing linear constraints. Finally, we demonstrated that the optimization of the strengthened MILP models generate POPs with higher solution quality in less computation time, compared the MaxSAT model [56]. Overall, we demonstrated that MILP technology can be effective in generating LCFP of an input plan.

5.1.2 A Logic-Based Benders Decomposition for Finding A Least Commitment Flexible Partial-Order Plan

Summary In Chapter 4, we addressed the problem of finding a LCFP to the complete planning problem, that is, without assuming an input set of actions that form a valid plan. We showed how partial-order relaxation approaches can be utilized inside a logic-based Benders decomposition (LBBD) to find a LCFP. First, we investigated two LBBDs to find an action cost-optimal POP to the complete planning problem. We derived a new symmetry breaking constraints to strengthen our decomposition models. Experimentally, we showed that both of our cost-optimal MILP-based POP planners are comparable to the sequential planner of Davies et al. [21]. Finally, we extended our best performing LBBD model to find a LCFP to the complete planning problem. Our preliminary empirical results show promising results for using a MILP-based LBBD to optimize this complex objective.

Contributions We showed that MILP models are effective in finding a LCFP to the complete planning problem. We formalized the least commitment flexible planning problem and presented a MILP-based LBBD to optimize it. To our knowledge, both the formalization and the MILP-based solution method of the problem are novel. All three MILP-based LBBDs presented in Chapter 4 use variations of a causal link-based MILP model that is presented in Chapter 3 to solve their subproblems. Unlike the sequential planner of Davies et al. [21], our LBBDs are not restricted to generate sequential plans, which allow our planners to optimize more complex objectives than the total action cost of a plan. Experimentally, we showed that the overall computational expense for generating a cost-optimal POP using a causal link-based subproblem is similar to that of generating a cost-optimal sequential plan. We then showed that the MILP-based LBBD can successfully optimize the least commitment flexible planning problem. Experimentally, we showed that over 90% of the problem instances for which the best performing cost-optimal POP planner finds a cost-optimal POP, the LCFP planner also finds a LCFP. Overall, we showed that MILP technology can be effective in generating a LCFP to the complete planning problem.

5.2 Future Work

In this section, we present possible directions for future work.

5.2.1 Improvement of the SAT Model

The main intuition behind the work in Chapter 3 that improved on the state-of-the-art MaxSAT formulation [56] for partial-order relaxation problem was the adoption of temporal variables and temporal constraints from the scheduling literature. In particular, we observed that the transitive closure of ordering constraints (i.e., Constraint (2.69) from Section 2.7.2) grows cubically with the number of actions. By introducing the temporal variables Est_a, Lft_a for every action a and replacing Constraint (2.69) with Constraints (3.14)-(3.10), we reduced the size of our MILP formulations significantly. In the light of our experimental results, one promising area of research is to explore SAT formulations that utilize temporal variables [18] i.e., $S_{a,t} = 1$ if and only if action a start at step t, and reduce the size of the encoding by eliminating redundant clauses [31]. If there is sufficient performance improvement, this new SAT model may be competitive with the MILP-based subproblem of the LBBD that is presented in Chapter 4.

5.2.2 Pre-Processing for Partial-Order Relaxation Problem

In Chapter 3, we introduced new linear constraints that represent the necessary conditions of a valid POP. The linear constraints that we identified do not require any pre-processing of the planning problem. One possible future direction of research is the exploration of pre-processing algorithms to identify stronger necessary conditions for the problem of partial-order relaxation. Pommerening et al. [64] demonstrated that strong operator counting constraints can be derived with polynomial-time pre-processing algorithms for the problem of finding a cost-optimal plan. In Chapter 4, we investigated the addition of operator counting constraints [64] to our base MILP models that are introduced in Chapter 3. It would be an interesting area of research to explore how Constraints (3.15)-(3.24) (see Section 3.7) interact with the state-of-the-art operator counting constraints [64]. Further, it would be interesting to explore pre-processing algorithms for partial-order relaxation problem to identify more complex necessary conditions of a POP in order to derive stronger linear constraints.

5.2.3 Investigation of Order Counting Heuristics

In Chapter 4, we introduced a logic-based Benders decomposition to find a least commitment flexible POP (LCFP). Our LBBD uses the same master problem as Davies et al. [21], based on operator counting constraints [64], since the primary objective of the LFCP problem is the minimization of the total action cost of a POP. The master problem of our LBBD does not include a lower bound on the proxy objective of the LFCP problem. As a result, our LBBD exhaustively searches all cost-optimal POPs. An interesting area of future research would be to investigate different formulations that represent valid lower bounds for the proxy functions used to approximate the number of linearizations in a POP. The addition of such a lower bound in the master problem of our LBBD can limit the number of times the subproblem is solved as it provides a stronger dual bound on the LFCP problem.

5.2.4 Conflict-Directed Clause Learning for MILP-based Partial-Order Formulations

Building on the ideas presented in Chapter 4, the overall performance of our LBBD can be improved by the incorporation of Conflict-Directed Clause Learning methods. Davies et al. [21] use Conflict-Directed Clause Learning to find the minimum set of missing actions to explain the infeasibility of their subproblem. Some current MILP solvers (e.g. SCIP) have conflict analysis methods that are similar to that of SAT solvers. An interesting area of future research would be the exploitation of such conflict analysis methods in our subproblems.

5.3 Conclusion

The central thesis of this dissertation is that the MILP technology can be effective in least commitment flexible partial-order planning. We demonstrated this thesis by introducing MILP models for partial-order relaxation problem and a MILP-based logic-based Benders decomposition (LBBD) for the least commitment flexible partial-ordering problem. We experimentally showed the effectiveness of MILP technology in finding a least commitment partial-order plan. We have contributed to the partial-order literature with our MILP models, and formally introduced the least commitment flexible partial-order planning problem and a MILP-based solution method.

Bibliography

[1] Christer Bäckström. Finding least constrained plans and optimal parallel executions is harder than we thought. In *Proceedings Second European Workshop On Planning*, pages 46–59. IOS Press, 1994.

[2] Christer Bäckström. Computational aspects of reordering plans. *Journal of Artificial Intelligence Research*, pages 99–137, 1998.

[3] Christer Bäckström and Bernhard Nebel. Complexity results for SAS+ planning. *Computational Intelligence*, 11:625–655, 1995.

[4] Dimitris Bertsimas, Iain Dunning, and Miles Lubin. Reformulation versus cutting-planes for robust optimization. *Computational Management Science*, 13(2):195–217, 2016.

[5] Sudip Biswas, Stephane Durocher, Debajyoti Mondal, and RahnumaIslam Nishat. Hamiltonian paths and cycles in planar graphs. In Guohui Lin, editor, *Combinatorial Optimization And Applications*, volume 7402 of *Lecture Notes in Computer Science*, pages 83–94. Springer Berlin Heidelberg, 2012.

[6] Avrim L. Blum and Merrick L. Furst. Fast planning through planning graph analysis. *Artificial Intelligence*, 90(1):1636–1642, 1995.

[7] Alexander Bockmayr and Yannis Dimopoulos. Mixed integer programming models for planning problems. In Jeremy Frank and Mihaela Sabin, editors, *Proceedings of the Workshop on Constraint Problem Reformulation (CP 1998)*, pages 1–6, Pisa, Italy, 1998. NASA Ames Research Center.

[8] Alexander Bockmayr and Yannis Dimopoulos. Integer programs and valid inequalities for planning problems. In *Proceedings of the Fifth European Conference on Planning, (ECP 1999)*, pages 239–251, September 1999.

[9] Blai Bonet. An admissible heuristic for SAS+ planning obtained from the state equation. In *Proceedings of the Twenty Third International Joint Conference on Artificial Intelligence (IJCAI 2013)*, 2013.

[10] Blai Bonet and Héctor Geffner. Planning as heuristic search. In *Entry at AIPS 1998 Planning Competition*, 2000.

[11] Menkes Van Den Briel. Reviving integer programming approaches for AI planning: A branch-and-cut framework. In *Proceedings of the Fifteenth International Conference on Automated Planning and Scheduling (ICAPS 2005)*, pages 310–319. AAAI Press, 2005.

[12] Christina Büsing, Arie M. C. A. Koster, and Manuel Kutschka. Recoverable robust knapsacks: The discrete scenario case. *Optimization Letters*, 5(3):379–392, 2011.

[13] Tom Bylander. The computational complexity of propositional STRIPS planning. *Artificial Intelligence*, 69:165–204, 1994.

[14] Tom Bylander. A linear programming heuristic for optimal planning. In *AAAI*. AAAI Press, 1997.

[15] Yixin Chen, Benjamin W. Wah, and Chih-Wei Hsu. Temporal planning using subgoal partitioning and resolution in SGPlan. In *Journal of Artificial Intelligence Research*, pages 323–369, 2006.

[16] Paul R. Cohen. *Empirical Methods For Artificial Intelligence*. MIT Press, Cambridge, MA, USA, 1995.

[17] Andrew Coles, Amanda Coles, Maria Fox, and Derek Long. Forward-chaining partial-order planning. In *Proceedings of the Twentieth International Conference on Automated Planning and Scheduling (ICAPS 2010)*, 2010.

[18] James M. Crawford and Andrew B. Baker. Experimental results on the application of satisfiability algorithms to scheduling problems. In *Proceedings of the Twelfth National Conference on Artificial Intelligence (Vol. 2)*, AAAI'94, pages 1092–1097, Menlo Park, CA, USA, 1994. American Association for Artificial Intelligence.

[19] George Bernard Dantzig and John H. Ramser. The truck dispatching problem. *Management Science*, 6(1):80–91, October 1959.

[20] Andrew J. Davenport, Christophe Gefflot, and J. Christopher Beck. Slack-based techniques for robust schedules. In *Proceedings of the Sixth European Conference on Planning (ECP 2001)*, 2001.

[21] Toby Davies, Adrian R. Pearce, Peter Stuckey, and Nir Lipovetzky. Sequencing operator counts. In *Proceedings of the Twenty Fifth International Conference on Automated Planning and Scheduling (ICAPS 2015)*, 2015.

[22] Toby Davies, Adrian R. Pearce, Peter Stuckey, and Harald Sndergaard. Fragment-based planning using column generation. In *Proceedings of the Twenty Fourth International Conference on Automated Planning and Scheduling (ICAPS 2014)*, 2014.

[23] Martin Davis and Hilary Putnam. A computing procedure for quantification theory. *Journal of the ACM*, 7(3):201–215, July 1960.

[24] Yannis Dimopoulos. Improved integer programming models and heuristic search for AI planning. In *Proceedings of the European Conference on Planning (ECP 2001)*, pages 301–313. Springer-Verlag, 2001.

[25] Yannis Dimopoulos, Bernhard Nebel, and Jana Koehler. Encoding planning problems in nonmonotonic logic programs. In *Proceedings of the Fourth European Conference on Planning*, pages 169–181. Springer-Verlag, 1997.

[26] Minh B. Do and Subbarao Kambhampati. Improving the temporal flexibility of position constrained metric temporal plans. In *Proceedings of the Thirteenth International Conference on Automated Planning and Scheduling (ICAPS 2003)*, 2003.

[27] Niklas Eën and Niklas Sörensson. An extensible sat-solver. In *Theory And Applications Of Satisfiability Testing*, page 502518. Springer, 2004.

[28] Kutluhan Erol, James Hendler, and Dana S. Nau. HTN planning: Complexity and expressivity. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, volume 2 of *AAAI 1994*, pages 1123–1128, Menlo Park, CA, USA, 1994. American Association for Artificial Intelligence.

[29] Richard E. Fikes, Peter E. Hart, and Nils J. Nilsson. Learning and executing generalized robot plans. *Artificial Intelligence*, 3:251–288, 1972.

[30] Richard E. Fikes and Nils J. Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. In *Proceedings of the Second International Joint Conference on Artificial Intelligence*, IJCAI 1971, pages 608–620, San Francisco, CA, USA, 1971. Morgan Kaufmann Publishers Inc.

[31] Juan Frausto-Solis and Marco A. Cruz-Chavez. A reduced codification for the logical representation of job shop scheduling problems. In Antonio Lagan, MarinaL. Gavrilova, Vipin Kumar, Youngsong Mun, C.J.Kenneth Tan, and Osvaldo Gervasi, editors, *Computational Science And Its Applications ICCSA 2004*, volume 3046 of *Lecture Notes in Computer Science*, pages 553–562. Springer Berlin Heidelberg, 2004.

[32] Avitan Gefen and Ronen I. Brafman. Pruning methods for optimal delete-free planning. In *Proceedings of the Twenty Second International Conference on Automated Planning and Scheduling (ICAPS 2012)*, 2012.

[33] Patrik Haslum and Héctor Geffner. Admissible heuristics for optimal planning. pages 140–149. AAAI Press, 2000.

[34] Patrik Haslum, John Slaney, and Sylvie Thiébaux. Minimal landmarks for optimal delete-free planning. In *Proceedings of the Twenty Second International Conference on Automated Planning and Scheduling (ICAPS 2012)*, 2012.

[35] Malte Helmert. The fast downward planning system. *Journal of Artifical Intelligence Research*, pages 191–246, 2006.

[36] Malte Helmert and Carmel Domshlak. Landmarks, critical paths and abstractions: Whats the difference anyway? In *Proceedings of the Nineteenth International Conference on Automated Planning and Scheduling (ICAPS 2009)*, pages 162–169, 2009.

[37] Jörg Hoffmann and Bernhard Nebel. The FF planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research*, 14(1):253–302, May 2001.

[38] Jörg Hoffmann, Julie Porteous, and Laura Sebastia. Ordered landmarks in planning. *Journal of Artificial Intelligence Research*, 22:215–278, November 2004.

[39] John N. Hooker. Logic-based benders decomposition. *Mathematical Programming*, 96:2003, 1995.

[40] Ruoyun Huang, Yixin Chen, and Weixiong Zhang. SAS+ planning as satisfiability. *Journal of Artificial Intelligence Research*, pages 293–328, March 2012.

[41] Tatsuya Imai and Alex Fukunaga. A practical, integer-linear programming model for the delete-relaxation in cost-optimal planning. In *Proceedings of Twenty First European Conference on Artificial Intelligence (ECAI 2014)*, 2014.

[42] Tatsuya Imai and Alex Fukunaga. On a practical, integer-linear programming model for delete-free tasks and its use as a heuristic for cost-optimal planning. *Journal of Artificial Intelligence Research*, 54:631–677, December 2015.

[43] Joxan Jaffar and Jean L. Lassez. Constraint logic programming. In *Proceedings of the Fourteenth ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages*, POPL 1987, pages 111–119, New York, NY, USA, 1987. ACM.

[44] Subbarao Kambhampati and Smadar Kedar. A unified framework for explanation-based generalization of partially ordered and partially instantiated plans. *Artificial Intelligence*, 67(1):29–70, 1994.

[45] Erez Karpas and Carmel Domshlak. Cost-optimal planning with landmarks. In *Proceedings of the Twenty First International Joint Conference on Artificial Intelligence*, IJCAI'09, pages 1728–1733, San Francisco, CA, USA, 2009. Morgan Kaufmann Publishers Inc.

[46] Henry Kautz. SATPLAN04: Planning as satisfiability. In *Working Notes on the International Planning Competition (IPC 2004)*, pages 44–45, 2004.

[47] Henry Kautz and Bart Selman. Pushing the envelope: Planning, propositional logic, and stochastic search. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, volume 2 of *AAAI'96*, pages 1194–1201. AAAI Press, 1996.

[48] Henry Kautz and Joachim P. Walser. State-space planning by integer optimization. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence*, pages 526–533. AAAI Press, 1999.

[49] James E. Kelley, Jr and Morgan R. Walker. Critical-path planning and scheduling. In *Papers Presented at the December 1-3, 1959, Eastern Joint IRE-AIEE-ACM Computer Conference*, IRE-AIEE-ACM 1959 (Eastern), pages 160–173, New York, NY, USA, 1959. ACM.

[50] Ryan F. Kelly and Adrian R. Pearce. Towards high level programming for distributed problem solving. In *IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT 2006)*, pages 490–497. IEEE Computer Society, 2006.

[51] David McAllester and David Rosenblitt. Systematic nonlinear planning. In *Proceedings of the Ninth National Conference on Artificial Intelligence*, volume 2 of *AAAI'91*, pages 634–639. AAAI Press, 1991.

[52] Karl Menger. Das botenproblem. 2:11–12, 1932.

[53] Steven Minton, John Bresina, and Mark Drummond. Total-order and partial-order planning: A comparative analysis. *Journal of Artificial Intelligence Research*, 2(1):227–262, August 1995.

[54] Christian Muise. *Exploiting Relevance To Improve Robustness And Flexibility In Plan Generation And Execution*. PhD thesis, Graduate Department of Computer Science, University of Toronto, 2014.

[55] Christian Muise, Sheila A. McIlraith, and J. Christopher Beck. Optimization of partial-order plans via MAXSAT. In *Proceedings of the Twenty First International Conference on Automated Planning and Scheduling (ICAPS 2011): Workshop on Constraint Satisfaction Techniques for Planning and Scheduling Problems (COPLAS 2011)*, pages 31–38, 2011.

[56] Christian Muise, Sheila A. McIlraith, and J. Christopher Beck. Optimally relaxing partial-order plans with maxSAT. In *Proceedings of the Twenty Second International Conference on Automated Planning and Scheduling (ICAPS 2012)*, pages 358–362, 2012.

[57] Christian Muise, Sheila A. McIlraith, and J. Christopher Beck. Optimal partial-order plan relaxation via maxSAT. *Journal of Artificial Intelligence Research*, 57:113–149, 2016.

[58] Dana Nau, Malik Ghallab, and Paolo Traverso. *Automated Planning: Theory & Practice*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2004.

[59] Allen Newell and Herbert A. Simon. Computer science as empirical inquiry: Symbols and search. *Communications of the ACM*, 19(3):113–126, March 1976.

[60] XuanLong Nguyen and Subbarao Kambhampati. Reviving partial order planning. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI2001)*, pages 459–464, 2001.

[61] J. Scott Penberthy and Daniel S. Weld. UCPOP: A sound, complete, partial order planner for ADL. pages 103–114. Morgan Kaufmann, 1992.

[62] Nicola Policella, Stephen F. Smith, Amedeo Cesta, and Angelo Oddi. Generating robust schedules through temporal flexibility. In *Proceedings of the Fourteenth International Conference on Automated Planning and Scheduling (ICAPS 2004)*, 2004.

[63] Florian Pommerening and Malte Helmert. Incremental LM-Cut. In *Proceedings of the Twenty Third International Conference on Automated Planning and Scheduling (ICAPS 2013)*, 2013.

[64] Florian Pommerening, Gabriele Roger, Malte Helmert, and Blai Bonet. LP-based heuristics for cost-optimal planning. In *Proceedings of the Twenty Fourth International Conference on Automated Planning and Scheduling (ICAPS 2014)*, 2014.

[65] Julie Porteous, Laura Sebastia, and Jörg Hoffmann. On the extraction, ordering, and usage of landmarks in planning. In *Proceedings of the Sixth European Conference on Planning (ECP 2001)*, 2001.

[66] Silvia Richter, Malte Helmert, and Matthias Westphal. Landmarks revisited. In *Proceedings of the Twenty Third National Conference on Artificial Intelligence*, volume 2 of *AAAI'08*, pages 975–982. AAAI Press, 2008.

[67] Silvia Richter and Matthias Westphal. The LAMA planner: Guiding cost-based anytime planning with landmarks. *Journal of Artificial Intelligence Research*, 39(1):127–177, September 2010.

[68] Jussi Rintanen. Evaluation strategies for planning as satisfiability. In *Proceedings of Sixteenth European Conference on Artificial Intelligence (ECAI 2004)*, pages 682–687. IOS Press, 2004.

[69] Jussi Rintanen, Keijo Heljanko, and Ilkka Niemel. Planning as satisfiability: Parallel plans and algorithms for plan search. *Artificial Intelligence*, 180:2006, 2005.

[70] Nathan Robinson, Charles Gretton, Duc-Nghia Pham, and Abdul Sattar. SAT-based parallel planning using a split representation of actions. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence*, pages 526–533. AAAI Pres, 2009.

[71] Earl D. Sacerdoti. The nonlinear nature of plans. In *Proceedings of the Fourth International Joint Conference on Artificial Intelligence*, volume 1 of *IJCAI'75*, pages 206–214, San Francisco, CA, USA, 1975. Morgan Kaufmann Publishers Inc.

[72] Buser Say, Andre A. Cire, and J. Christopher Beck. Mathematical programming models for optimizing partial-order plan flexibility. In *Proceedings of Twenty Second European Conference on Artificial Intelligence (ECAI 2016)*, 2016.

[73] Fazlul Hasan Siddiqui and Patrik Haslum. Block-structured plan deordering. In *Proceedings of the Twenty Fifth Australasian Joint Conference on Artificial Intelligence*, pages 803–814, 2012.

[74] Austin Tate. Generating project networks. In *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, volume 2 of *IJCAI'77*, pages 888–893, San Francisco, CA, USA, 1977. Morgan Kaufmann Publishers Inc.

[75] Alvaro Torralba, Vidal Alcazar, Daniel Borrajo, Peter Kissmann, and Stefan Edelkamp. SymBA*: A symbolic bidirectional a planner. In *International Planning Competition (IPC 2014)*, 2014.

[76] Vincent Vidal. The YAHSP planning system: Forward heuristic search with lookahead plan analysis. In *Booklet of the International Planning Competition (IPC-2004)*, 2004.

[77] Thomas Vossen, Michael Ball, and Robert H. Smith. On the use of integer programming models in AI planning. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, pages 304–309. Morgan Kaufmann, 1999.

[78] Daniel S. Weld. An introduction to least commitment planning. *AI Magazine*, 1994.

[79] Håkan L. S. Younes and Reid G. Simmons. VHPOP: Versatile heuristic partial order planner. volume 20, pages 405–430, 2003.

[80] Lin Zhu and Robert Givan. Landmark extraction via planning graph propagation. In *ICAPS Doctoral Consortium 2003 (ICAPS 2003)*, pages 156–160, 2003.