

Execution Flexibility in Automated Planning: A Comparative Evaluation of Deordering and Reordering Strategies

Md. Monjurul Islam¹, Sabah Binte Noor¹,
Fazlul Hasan Siddiqui¹, Gahangir Hossain^{2*}

¹Department of Computer Science and Engineering, Dhaka University of Engineering & Technology, Gazipur, Gazipur-1707, Bangladesh.

²Department of Data Science, , University of North Texas, Denton, TX, USA.

*Corresponding author(s). E-mail(s): Gahangir.Hossain@unt.edu;
Contributing authors: monjurul.islam.cs@gmail.com; sabah@duet.ac.bd;
siddiqui@duet.ac.bd;

Abstract

This study covers foundational concepts for enhancing plan-execution flexibility, including partial-order planning, the producer-consumer-threat formalism, and a range of deordering and reordering strategies. Creating a partial-order plan from a sequential one by removing unnecessary ordering constraints is a practical way to improve execution flexibility, and several methods have been proposed for this task. This study analyzes their capabilities across ordering, action handling, parameter handling, plan structure, concurrency, and complexity, and evaluates them against each other on a shared benchmark. The central finding is that block deordering-based approaches, which restructure causal dependencies through block-level grouping and subplan substitution, substantially outperform MaxSAT-based approaches despite the latter’s theoretical guarantees of minimum reordering. The reason is structural: minimum reordering optimizes within the causal structure already present in the plan, whereas block deordering-based methods change that structure, exposing orderings that would otherwise appear necessary. A further distinction is practical: block deordering-based methods are anytime algorithms that always return a valid result, while MaxSAT-based methods fail entirely on a substantial portion of plans and offer no partial solution when they do. Block substitution further extends the parallel execution by formalizing non-concurrency constraints, though its impact is limited to

domains with resource-based interactions. On efficiency, block deordering-based approaches achieve the highest flex gain per unit of computation time, while MaxSAT-based encodings incur large computational overhead.

Keywords: Automated Planning, Flexibility, Concurrency, Partial Order Planning

1 Introduction

The capability of autonomous agents to adapt to unforeseen circumstances represents a fundamental requirement for effective operation in dynamic environments. Within the domain of automated planning, multiple methodologies have been developed to enhance agent flexibility, ranging from providing agents with alternative plan selections (Graham et al. 2003) to extending the applicability of existing plans through generalization techniques (Anderson and Farley 1988). Plan generalization approaches generate partial-order plans (POPs) by relieving specific action orderings until such orderings become inevitable. This strategy of minimizing action ordering decisions has been explored through various concepts, including plan deordering (Nguyen and Kambhampati 2001; Veloso et al. 2002; Siddiqui and Haslum 2012) and reordering (Muise et al. 2016) methodologies. While deordering the plan removes inessential action orderings, reordering the plan permits arbitrary changes to the action sequence.

One way to produce a POP is through partial-order causal link (POCL) planning (Weld 1994), which constructs a plan incrementally while maintaining causal support for every action precondition. A practical alternative is to take a sequential plan from a fast heuristic planner and remove the ordering constraints that are not necessary.

Several methods have been developed for this post-processing task. Explanation-based order generalization (EOG) (Kambhampati and Kedar 1994; Veloso et al. 2002) annotates each ordering with its causal justification and removes those without one. Block deordering (BD) (Siddiqui and Haslum 2012) groups coherent operators into blocks and eliminates additional orderings by treating each block as a unit. Minimum reordering (MR) (Muise et al. 2016) encodes the deordering problem as partial weighted MaxSAT to find provably minimum reorderings and remove redundant actions; minimum reinstated reordering (MRR) (Waters et al. 2020) extends this with parameter rebinding within same-name operators. More recently, FIBS (Noor and Siddiqui 2024) replaces subplans with alternatives using block-substitution, and CIBS (Noor and Siddiqui 2025) extends FIBS to model and minimize non-concurrency constraints for parallel execution.

We analyze capabilities of different algorithms across multiple dimensions (ordering, action handling, parameter handling, plan structure, concurrency, and complexity), and evaluate on 3,345 plans from 46 IPC domains. The analysis produces a result that is not obvious from the descriptions above. MR and MRR are designed to find minimum reorderings. Yet BD and FIBS produce substantially higher flex values on the same benchmarks, with large and statistically significant effect sizes against all other methods. The reason is that minimum reordering optimizes within

the causal structure the original plan already has. Block deordering works differently: it reorganizes which operators share causal dependencies by grouping them into blocks, exposing ordering constraints that would otherwise appear necessary. Block-substitution goes further, replacing subplans with alternatives that carry different causal entailments entirely. The two families of methods are not in competition so much as they are solving different problems; restructuring causal structure turns out to be the more productive one for increasing flexibility.

2 Preliminaries

This section outlines the foundational concepts required to comprehend the methods. It begins with defining planning tasks and plans within the finite-domain representation, followed by notions of partial-order planning, causal dependencies, and threats. Finally, it outlines key concepts of plan deordering and reordering.

2.1 Planning Task

Classical planning problems are commonly represented using finite domain representation (FDR) (Helmert 2011), which describes planning tasks through state variables and their associated value domains. The following definitions establish the mathematical foundation for subsequent analysis.

Definition 1. In FDR, a planning task is denoted as $\Pi = \langle \mathcal{X}, \mathcal{O}, s_i, s_g \rangle$, where:

- $\mathcal{X}$ represents a finite non-empty set of **state variables**. A domain $\mathcal{D}_x$ with a finite number of elements is associated with each $x \in \mathcal{X}$. The pair $\langle x, d \rangle$ with $x \in \mathcal{X}$ and $d \in \mathcal{D}_x$ is a **fact**. The notion of **state** corresponds to a function $\mathbf{s}$ defined over $\mathcal{X}$, such that $s(x) \in \mathcal{D}_x$ for all $x \in \mathcal{X}$. The set of variables involved in a function is denoted as $\text{vars}(\mathbf{s})$. A function $\hat{\mathbf{s}}$ similar to a state $\mathbf{s}$, but defined over a proper subset of $\mathcal{X}$, represents a **partial state** where $\text{vars}(\hat{\mathbf{s}}) \subset \mathcal{X}$.
- $\mathcal{O}$ represents the finite non-empty set of **operators**. Every operator $o \in \mathcal{O}$ is associated with two partial states, **precondition** (pre_o) and **effect** (eff_o), along with an associated **cost** that is nonnegative ($\text{cost}_o \in \mathbb{R}_0^+$). An operator o can be applied in a state s iff $\text{pre}_o \subseteq s$. Applying the operator o within state s transforms it into state $s' = \text{apply}(s, o)$ as specified in (1).

$$s'(x) = \text{apply}(s, o) = \begin{cases} \text{eff}_o(x) & \text{if } x \in \text{vars}(\text{eff}_o) \\ s(x) & \text{otherwise} \end{cases} \quad (1)$$

- s_i corresponds to the **initial state**,
- s_g refers to a partial state that specifies the **goal conditions**.

Definition 2. Let $\pi = \langle o_1, o_2, \dots, o_k, \dots, o_n \rangle$ be a **plan**, a sequence of operators, for a planning task $\Pi = \langle \mathcal{X}, \mathcal{O}, s_i, s_g \rangle$. The plan π is **valid** iff $\text{pre}_{o_1} \subseteq s_i$, $\text{pre}_{o_{k+1}} \subseteq s_k$ for every $k \in \{1, 2, \dots, n-1\}$ where $s_k = \text{apply}(s_{k-1}, o_k)$, and $s_g \subseteq s_n$.

Definition 3. The $\mathbf{prod}_o$, $\mathbf{cons}_o$, and $\mathbf{del}_o$ represent the facts that an operator o produces, consumes, and deletes, respectively. The fact—

- $\langle x, d \rangle \in \mathbf{prod}_o$ iff $\langle x, d \rangle \in \mathbf{eff}_o$.
- $\langle x, d \rangle \in \mathbf{cons}_o$ iff $\langle x, d \rangle \in \mathbf{pre}_o$.
- $\langle x, d \rangle \in \mathbf{del}_o$ iff—
 - i. either $x \notin \mathbf{vars}(\mathbf{cons}_o)$ or $\mathbf{cons}_o(x) = d$, and
 - ii. $\mathbf{eff}_o(x) = d'$ such that $d' \in (\mathcal{D}_x \setminus \{d\})$.

2.2 Partial-Order Planning

Operators can be executed in any possible sequence using the partial-order plan (POP) framework while imposing partial orderings over them. It is assumed that each operator is individually identifiable, even if an operator may appear multiple times within a POP.

Definition 4. A **POP** is represented as $\pi_{pop} = \langle \mathcal{O}, \prec \rangle$, regarding a planning task $\Pi = \langle \mathcal{X}, \mathcal{O}, s_i, s_g \rangle$, where:

- $\mathcal{O}$ denotes the set of operators, and
- $\prec$ represents ordering constraints over $\mathcal{O}$. The notation $\mathbf{o}_a \prec \mathbf{o}_b$ represents a **ordering constraint** between two operators $o_a, o_b \in \mathcal{O}$, which specifies that the operator o_a needs to be executed at any time before the execution of operator o_b . $\prec$ exhibits transitive property, which means that if $o_a \prec o_b$ and $o_b \prec o_c$, then $o_a \prec o_c$. Basic orderings represent constraints that are not transitively inferred by other constraints in the set.

Plan flexibility quantification uses the metric *flex* (Siddiqui and Haslum 2012), which is the proportion of the pairs of operators that do not have fundamental or transitive ordering to all operator pairs. Higher *flex* values indicate greater execution flexibility, though the relationship between flexibility and practical adaptability depends on domain-specific factors.

Definition 5. Let $\pi_{pop} = \langle \mathcal{O}, \prec \rangle$ be a POP. The **flex** value of the plan π_{pop} is $\mathbf{flex}(\pi_{pop})$ and is calculated as specified in (2).

$$\mathbf{flex}(\pi_{pop}) = 1 - \frac{|\prec|}{\sum_{i=1}^{|\mathcal{O}|-1} i} \quad (2)$$

where $\sum_{i=1}^{|\mathcal{O}|-1} i$ is the maximum number of pairings that might be constructed with a collection of $|\mathcal{O}|$ items.

The framework of producer-consumer-threat (PCT) (Bäckström 1998) establishes ordering structures by identifying operator relationships regarding fact production,

consumption, and deletion. This formalism introduces causal links that map operator preconditions to effect providers.

Definition 6. A **causal link** $o_p \xrightarrow{\langle x, d \rangle} o_c$ between o_p and o_c specifies that $o_p \prec o_c$ and the operator o_p produces $\langle x, d \rangle$ for the operator o_c , where $\langle x, d \rangle \in (\text{prod}_{o_p} \cap \text{cons}_{o_c})$. A **threat** is defined as a conflict between a causal link $o_p \xrightarrow{\langle x, d \rangle} o_c$ and the effect of an operator o_d , where o_d deletes $\langle x, d \rangle$ and can be ordered in between o_p and o_c .

A **promotion**, which adds the ordering $o_d \prec o_p$, or a **demotion**, which adds the ordering $o_c \prec o_d$, can resolve a threat between a causal link $o_p \xrightarrow{\langle x, d \rangle} o_c$ and an operator o_d . A POP is valid if causal relationships support each operator precondition without any threat (Weld 1994). Three labels, *PC*, *CT*, and *TP*, are introduced by Siddiqui and Haslum (2012) to outline a POP's ordering constraints.

Definition 7. Let $\pi_{\text{pop}} = \langle \mathcal{O}, \prec \rangle$ be a POP and $\text{Re}(o_a \prec o_b)$ represents the ordering reasons for an ordering constraint $o_a \prec o_b$. An ordering restriction $o_a \prec o_b$ might form for three reasons:

- $\text{PC}(\langle x, d \rangle) \in \text{Re}(o_a \prec o_b)$ is the **producer-consumer** of a fact $\langle x, d \rangle$, where o_a produces $\langle x, d \rangle$ and o_b consumes $\langle x, d \rangle$. The fact $\langle x, d \rangle$ may be produced and consumed by multiple operators. One operator o_a is assigned via a causal link to obtain $\langle x, d \rangle$ for o_b .
- $\text{CT}(\langle x, d \rangle) \in \text{Re}(o_a \prec o_b)$ is the **consumer-threat** of a fact $\langle x, d \rangle$, where operator o_a consumes $\langle x, d \rangle$ and o_b deletes $\langle x, d \rangle$.
- $\text{TP}(\langle x, d \rangle) \in \text{Re}(o_a \prec o_b)$ is the **threat-producer** of a fact $\langle x, d \rangle$, where operator o_a deletes the fact $\langle x, d \rangle$ and there exist one or more causal link $o_b \xrightarrow{\langle x, d \rangle} o_c$ for some $o_c \in \mathcal{O}$.

Causal links of a POP are denoted by the label *PC*, and the demotion and promotion ordering constraints of the plan are denoted by *CT* and *TP*, respectively. These labels assist in identifying and monitoring the rationale behind the orderings in a POP.

2.3 Plan Deordering and Reordering

Plan *deordering* and *reordering* are two key concepts for attaining plan execution flexibility. Formal definitions of these notions are given in Bäckström (1998). A POP is assumed to be transitively closed in the Definition 8.

Definition 8. Let $\pi_p = \langle \mathcal{O}, \prec_p \rangle$ and $\pi_q = \langle \mathcal{O}, \prec_q \rangle$ be two different POPs with respect to Π (a planning task). Therefore:

- π_q is a **deordering** of π_p with respect to Π iff both π_p and π_q are valid POP and $\prec_q \subseteq \prec_p$.

- π_q is a **proper deordering** of π_p with respect to Π iff π_q is a deordering of π_p and $\prec_q \subset \prec_p$.
- π_q is a **reordering** of π_p with respect to Π iff both π_p and π_q are valid POP.
- π_q is a **proper reordering** of π_p with respect to Π iff π_q is a reordering of π_p with $\prec_q \neq \prec_p$.
- π_q is a **minimum deordering** of π_p with respect to Π iff—
 1. π_q is a deordering of π_p , and
 2. there exists no partial-order plan $\pi_r = \langle \mathcal{O}, \prec_r \rangle$ such that π_r is a deordering of π_p with $|\prec_r| < |\prec_q|$.
- π_q is a **minimum reordering** of π_p with respect to Π iff—
 1. π_q is a reordering of π_p , and
 2. there exists no partial-order plan $\pi_r = \langle \mathcal{O}, \prec_r \rangle$ such that π_r is a reordering of π_p and $|\prec_r| < |\prec_q|$.

3 Partial-Order Causal Link-based Approach

The partial-order causal link (POCL) (Weld 1994) planning framework serves as the foundation for the traditional method of creating a partial-order plan (POP). An initial POP is iteratively refined through modifications involving operators, causal links, and ordering constraints in this paradigm. Such a preliminary plan typically begins with an initial operator and a goal operator. Refinement steps may introduce a new operator, insert a causal link connecting two operators, or establish an ordering constraint between them. A POP is considered completed once each operator’s precondition is protected by a threat-free causal link. The two most popular POCL-based partial-order planners are VHPOP (Simmons and Younes 2011) and UCPOP (Penberthy and Weld 1992). Furthermore, the POCL strategy has been adapted to find temporal plans through state-based forward search techniques (Coles et al. 2021), and it also forms the foundation for numerous hierarchical planning approaches (Bercher et al. 2017, 2016; Bit-Monnot et al. 2020, 2016).

In addition to POCL, other techniques have been developed to generate POPs. One such method is *Petri net unfolding* (Hickmott et al. 2007), which uses iterative unfolding of a specifically constructed Petri net to simulate the execution of a forward planning system. Another prominent approach is *Graphplan* (Blum and Furst 1997), which leverages a compact structure called a *planning graph* to derive optimal POPs. Over time, this planning graph has been used as a preprocessing mechanism for diverse planning systems, including STAN (Fox and Long 2001), IPP (Koehler 1999), and Blackbox (Kautz and Selman 1998).

4 Partial Weighted MaxSAT-based Approach

Transforming sequential plans into partial-order plans through deordering or reordering is a prominent approach for generating POPs. Early strategies for deordering involved generalizing and storing sequential plans in triangle tables (Regnier and Fade 1991; Fikes and Nilsson 1971), primarily to support plan modification and reuse. Triangle tables were later utilized as a preprocessing technique to take a sequential plan

with conditional effects and extract partial ordering (Winner and Veloso 2002). More recent work has applied partial weighted MaxSAT encodings to enhance execution flexibility by reducing the orderings in a plan (Muise et al. 2016). Building on this, the *action reinstatement* extends the MaxSAT formulation with additional constraints, allowing operator parameter reassignment (Waters et al. 2018, 2020).

The partial weighted MaxSAT problem extends the traditional satisfiability (SAT) problem by introducing two categories of clauses: *hard* and *soft*. Hard clauses function like those in the standard SAT problem and must always be satisfied. Soft clauses are not mandatory; instead, each is assigned a weight indicating its relative importance. Finding an assignment that satisfies all hard clauses and maximizes the cumulative weight of completely satisfied soft clauses is the goal.

4.1 MR Encodings

The task of achieving a plan’s minimum reordering (MR) as a partial weighted MaxSAT problem instance is formulated by Muise et al. (2016). The solution of the instance corresponds to a POP, which they denote as the *target* POP. For a given POP or sequential plan $\pi = \langle \mathcal{O}, \prec \rangle$, the encoding employs three categories of propositional variables—

- x_o : Indicates the presence of operator o in the *target* POP for each $o \in \mathcal{O}$.
- $\kappa(o_a, o_b)$: Indicates that the *target* POP includes the ordering constraint $o_a \prec o_b$ for each operator pair $o_a, o_b \in \mathcal{O}$.
- $\Upsilon(o_p, \langle x, d \rangle, o_c)$: Indicates that a causal link $o_p \xrightarrow{\langle x, d \rangle} o_c$ is present in the *target* POP, for a pair of operator $o_p, o_c \in \mathcal{O}$, specifies that the operator o_p produces $\langle x, d \rangle$ for the operator o_c , where $\langle x, d \rangle \in (prod_{o_p} \cap cons_{o_c})$.

The encoding procedure starts with defining hard clauses as Boolean formulae, which are then converted into conjunctive normal form (CNF), followed by the introduction of soft clauses and their associated weights. A soft clause with weight k is represented by the syntax $\overset{k}{(\dots)}$, whereas no weight indication represents a hard clause. To guarantee correctness, the encoding defines formulae (3) and (4) that enforce acyclicity in the target POP, where (4) ensures that the ordering constraints include the transitive closure. Formula (5) includes the initial and goal operators. Operators are treated as universally quantified, and in the case of formula (6), it is assumed that $o_I \neq o_i \neq o_G$. Formula (7) guarantees that every causal link remains threat-free by enforcing the required promotion or demotion orderings, while formula (8) ensures that each precondition is backed by an appropriate causal link.

$$(\neg \kappa(o, o)) \tag{3}$$

$$\kappa(o_p, o_q) \wedge \kappa(o_q, o_r) \rightarrow \kappa(o_p, o_r) \tag{4}$$

$$(x_{o_I}) \wedge (x_{o_G}) \tag{5}$$

$$x_{o_i} \rightarrow \kappa(o_I, o_i) \wedge \kappa(o_i, o_G) \tag{6}$$

$$\Upsilon(o_p, \langle x, d \rangle, o_c) \rightarrow \bigwedge_{o_d: \langle x, d \rangle \in del_{o_d}} x_{o_d} \rightarrow \kappa(o_d, o_p) \vee \kappa(o_c, o_d) \quad (7)$$

$$x_{o_c} \rightarrow \bigwedge_{\langle x, d \rangle \in cons_{o_c}} \bigvee_{o_p: \langle x, d \rangle \in prod_{o_p}} \kappa(o_p, o_c) \wedge \Upsilon(o_p, \langle x, d \rangle, o_c) \quad (8)$$

For each operator and ordering variable in the encoding, a soft unit clause is introduced in formulae (9) and (10) containing the negation of that variable. If any of these unit clauses is violated, it indicates that the operator or the ordering constraint associated with the variable of the violated clause is incorporated into the solution. As a result, solutions with higher weights in the encoding correspond to target POPs that contain fewer ordering constraints.

$$\frac{cost_o + |\mathcal{O}|^2 + 1}{(\neg x_o)} \quad , \quad \forall o \in (\mathcal{O} \setminus \{o_I, o_G\}) \quad (9)$$

$$\frac{1}{(\neg \kappa(o_i, o_j))} \quad , \quad \forall o_i, o_j \in \mathcal{O} \quad (10)$$

4.2 MRR Encodings

The partial weighted MaxSAT encoding for achieving a plan's minimum reordering is extended by Waters et al. (2020) through allowing modifications to both operator parameters and ordering constraints to achieve greater flexibility, defined as minimum reinstantiated reordering (MRR). Variables are denoted by symbols like x , y , and z , constants by c , and terms by t , u , and v . The definition of a term t is an ordered series of items, where $t[i]$ is the i -th element in the sequence. A mapping of terms with variables is designated as substitution Ψ ; for instance, $\Psi = \{x_1/t_1, x_2/t_2, \dots, x_n/t_n\}$ maps x_i to the corresponding term t_i for all $i \in \{1, 2, \dots, n\}$. The sets of variables and constants that appear in a given structure η are denoted by $vars(\eta)$ and $consts(\eta)$. In this approach, a POP is represented as $\pi = \langle \mathcal{O}, \Psi, \prec \rangle$, where the set of operators is represented by $\mathcal{O}$, $\prec$ is a strict partial order over $\mathcal{O}$ that is transitively closed, and Ψ is complete with respect to $\mathcal{O}$.

Definition 9. Let $\pi_p = \langle \mathcal{O}, \Psi, \prec \rangle$ and $\pi_q = \langle \mathcal{O}, \Psi', \prec' \rangle$ be two different POPs, then:

- π_q is a **reinstantiated reordering** of π_p if π_p and π_q both are valid.
- π_q is a **minimum reinstantiated reordering** of π_p if π_q is a reinstantiated reordering of π_p and there does not exist a POP $\pi_r = \langle \mathcal{O}, \Psi'', \prec'' \rangle$ such that π_r is a reinstantiated reordering of π_p with $|\prec''| < |\prec'|$.

An operator is defined as a tuple $o = \langle vars_o, pre_o, eff_o \rangle$, where $vars_o$ denotes the set of variables, and both eff_o and pre_o are the finite sets of facts containing $vars_o$'s variables. If both pre_o and eff_o are made up entirely of ground facts, then o is considered as a ground operator. $\langle o_p, q(\vec{t}), o_c, q(\vec{u}) \rangle$ represents a causal link, where $q(\vec{u})$ and $q(\vec{t})$ are literals, with $q(\vec{t}) \in prod_o$ and $q(\vec{u}) \in cons_o$. The MRR encoding adds two additional categories of propositional variables in addition to the x and κ utilized in the MR encoding:

- $\varepsilon(u, v)$: Encodes that in the final POP the condition $\Psi(t) = \Psi(u)$ must be satisfied.
- $\tau(o_p, \vec{t}, o_c, \vec{u})$: Encodes that there exists some q such that the $\langle o_p, q(\vec{t}), o_c, q(\vec{u}) \rangle$ causal link remains threat-free within the final POP.

To compute the MRR of a partial-order plan $\pi_p = \langle \mathcal{O}, \Psi, \prec \rangle$, the following encoding together with formulae (3) through (6) is used. Formulae (11) and (12) specify that the equality relation over variables and constants is both symmetric and transitive, whereas the mapping of each variable to a single object is guaranteed by Formula (13). Formulae (14) and (15) formalize the conditions required for the validity of the target POP.

$$\varepsilon(u, v) \leftrightarrow \varepsilon(v, u) \quad (11)$$

$$\varepsilon(u, v) \wedge \varepsilon(v, t) \rightarrow \varepsilon(u, t) \quad (12)$$

$$\bigwedge_{x \in \text{vars}_{\mathcal{O}}} \left(\bigvee_{c \in \text{consts}_{\mathcal{O}}} \varepsilon(x, c) \right) \wedge \bigwedge_{\substack{c', c'' \in \text{consts}_{\mathcal{O}}: \\ c' \neq c''}} \neg \varepsilon(x, c') \vee \neg \varepsilon(x, c'') \quad (13)$$

$$\bigwedge_{q(\vec{u}) \in \text{cons}_{o_c}} \bigvee_{q(\vec{t}) \in \text{prod}_{o_p}} \tau(o_p, \vec{t}, o_c, \vec{u}) \wedge \kappa(o_p, o_c) \quad (14)$$

$$\tau(o_p, \vec{t}, o_c, \vec{u}) \rightarrow \bigwedge_{1 \leq i \leq |\vec{t}|} \varepsilon(\vec{t}[i], \vec{u}[i]) \wedge \bigwedge_{\substack{q(\vec{v}) \in \text{del}_{o_d}: \\ \vec{t} = \vec{v}, o_d \neq o_c}} (\kappa(o_d, o_p) \vee \kappa(o_c, o_d)) \quad (15)$$

If a POP has fewer than k ordering constraints, determining if it admits a minimal reinstantiated reordering is *NP-complete*. It is impossible to approximate the problem of finding such a reordering within a constant factor. A key limitation of MRR is that substitutions are restricted to operators sharing the same name, but it does not permit replacing operator sets with different operator names or sizes.

5 Explanation-based Order Generalization

Another well-known approach, Explanation-based Order Generalization (EOG), employs validation structures as proofs of correctness and modifies plans to resolve inconsistencies (Veloso et al. 2002; Kambhampati 1994; Kambhampati and Hendler 1992). EOG has been further refined to support conditional effects (Noor and Siddiqui 2022). The EOG deorders plans by creating a validation structure, creating a causal link for each operator’s preconditions, and using *promotions* or *demotions* to mitigate threats to those causal links (Kambhampati and Kedar 1994; Veloso et al. 2002). Given a total-order plan π of a planning task $\Pi = \langle \mathcal{X}, \mathcal{O}, s_i, s_g \rangle$, EOG emulates the initial and goal conditions of Π by augmenting π with a couple of additional operators, o_I and o_G . Specifically, $\text{pre}_{o_I} = \emptyset$, $\text{eff}_{o_I} = s_i$, $\text{pre}_{o_G} = s_g$, and $\text{eff}_{o_G} = \emptyset$, with $o_I \prec o_G$ and $o_I \prec o \prec o_G$ for all $o \in (\mathcal{O} \setminus \{o_I, o_G\})$. The validation structure is subsequently constructed, and threats are resolved by introducing promotion and

demotion orderings. To minimize unnecessary transitive orderings, it systematically binds causal links to the earliest possible producers.

5.1 Plan Deordering with Conditional Effects

Conditional effects (Haslum et al. 2019) enable operators to present complex scenarios more effectively by allowing a single operator to produce different outcomes depending on the specific conditions. Noor and Siddiqui (2022) introduce a method for annotating operator orderings in plans with conditional effects, allowing such plans to be transformed into a Partial-Order Plan (POP).

Definition 10. Let $\Pi = \langle \mathcal{X}, \mathcal{O}_c, s_i, s_g \rangle$ be a planning task, where $\mathcal{O}_c$ is a set of a finite number of operators with conditional effects. An effect of an operator $o \in \mathcal{O}_c$ consists of triples $\langle \text{cond}, x, d \rangle$, forming a partial state where cond denotes the effect condition, which may be empty. When o is applicable in the state s , applying it yields the state $s' = \text{apply}(o, s)$, where $x \in \mathcal{X}$ takes the value $d \in D_x$ if and only if $\text{cond} \subseteq s$.

Definition 11. Let $\pi_{\text{pop}} = \langle \mathcal{O}, \prec \rangle$ be a POP and two operators $o_i, o_j \in \mathcal{O}$ with conditional effects.

- o_i is considered as a **candidate producer** of $\langle x, d \rangle$ for operator o_j iff:
 1. $o_j \not\prec o_i$,
 2. $\langle \text{cond}, x, d \rangle \in \text{eff}_{o_i}$ and there exists other candidate producer $o_p \in \mathcal{O}$ for every $\langle x', d' \rangle \in \text{cond}$ where $o_p \neq o_j$, and
 3. there exists no operator $o_k \in \mathcal{O}$ with $\langle \text{cond}_1, x, d_1 \rangle \in \text{eff}_{o_k}$ such that $o_j \not\prec o_k \not\prec o_i$, $d_1 \in (D_x \setminus \{d\})$, and there exists other candidate producer $o_p \in \mathcal{O}$ for each $\langle x'', d'' \rangle \in \text{cond}_1$.
- o_i is considered as an **earliest candidate producer** of $\langle x, d \rangle$ for operator o_j iff o_i is a candidate producer of $\langle x, d \rangle$ for o_j and there does not exist any candidate producer o_k of $\langle x, d \rangle$ for o_j with $o_k \prec o_i$.

Definition 12. Let $\pi_{\text{pop}} = \langle \mathcal{O}, \prec \rangle$ be a POP and $o \in \mathcal{O}$ be an operator with conditional effects. The operator o produces, consumes, and deletes facts, which are represented by prod_o , cons_o , and del_o , respectively. The fact—:

- $\langle x, d \rangle \in \text{cons}_o$ if $\langle x, d \rangle \in \text{pre}_o$ or $\langle x, d \rangle \in \text{cond}$ such that $\langle \text{cond}, x', d' \rangle \in \text{eff}_o$ and there exists an operator $o_k \in \mathcal{O}$ with $o \xrightarrow{\langle x', d' \rangle} o_k$.
- $\langle x, d \rangle \in \text{prod}_o$ if $\langle \text{cond}, x, d \rangle \in \text{eff}_o$ and there exists other candidate producer $o_p \in \mathcal{O}$ for each $\langle x', d' \rangle \in \text{cond}$ such that $o_p \prec o$.
- $\langle x, d \rangle \in \text{del}_o$ if $x \notin \text{vars}(\text{cons}_o)$ or $\text{cons}_o(x) = d$, $\langle \text{cond}, x, d' \rangle \in \text{eff}_o$, where $d' \in (D_x \setminus \{d\})$, and there exists a candidate producer $o_p \in \mathcal{O}$ for each $\langle x', d'' \rangle \in \text{cond}$.

$Re(o_a \prec o_b)$ denotes the ordering reasons between two operators o_a and o_b with conditional effects, which can fall into the categories PC, CT, TP, and OC. $OC(\langle x, d \rangle)$

denotes an *obstructor-consumer* relation for a fact $\langle x, d \rangle$, which occurs when o_a prevents o_b from receiving the fact $\langle x, d \rangle \in cond$, such that $\langle cond, x', d' \rangle \in eff_{o_b}$. If there is an operator o_k that yields $\langle x, d \rangle$ and $o_b \prec o_k \prec o_a$, then $OC(\langle x, d \rangle) \in Re(o_a \prec o_b)$ is considered *threatened*. This identified threat may be mitigated in two ways: through *Promotion*, by including $pOC(\langle x, d \rangle)$ in $Re(o_k \prec o_a)$, or through *Demotion*, by including $OCp(\langle x, d \rangle)$ in $Re(o_b \prec o_k)$.

The Deordering Plan with Conditional Effects (DConE) algorithm constructs causal links to build a validation framework for a POP. Similar to the EOG, DConE identifies causal links using the earliest candidate producers for each fact $\langle x, d \rangle$ required by an operator. While determining a candidate producer of $\langle x, d \rangle$ for an operator o_i , additional orderings may be needed due to *OC* (obstructor-consumer) reasons. This occurs when an operator o_d prevents another operator o_p from providing $\langle x, d \rangle$ to o_i through a conditional effect $\langle cond, x, d' \rangle \in eff_{o_d}$, where $d' \in (\mathcal{D}_x \setminus \{d\})$. To address this, the algorithm introduces an ordering reason $OC(\langle x, d \rangle)$ in $Re(o_q \prec o_d)$ for some operator o_q that obstructs the effect $\langle cond, x, d' \rangle$ of o_d from being triggered. The ordering $o_q \prec o_d$ prevents *cond* from holding, ensuring that o_d no longer deletes $\langle x, d \rangle$. This process repeats iteratively until a valid candidate producer is found. After establishing a causal link between two operators, DConE adds conditional causal links for each fact in the producer's effect condition and resolves all threats to *PC* and *OC* reasons through promotion or demotion of the threatening operators.

6 Block Deordering-based Approach

Block deordering (Siddiqui and Haslum 2012) identifies coherent sets of operators, known as blocks, to remove additional orderings from the plan. Block deordering groups coherent operators into blocks to reduce the POP's ordering constraints, resulting in a block decomposed partial-order (BDPO) plan (Siddiqui and Haslum 2015). Each block contains a set of operators; operators from two disjoint blocks cannot interleave, allowing the blocks to be executed in any sequence. Additionally, blocks can be nested, which allows a block to have one or more inner blocks. Overlapping between blocks is not permitted. A block fully enclosed within another is called an inner block, while a block not contained within any other is referred to as an outer block. Block deordering has also been leveraged to improve overall plan quality (Siddiqui and Haslum 2015), generate *macro-actions* (Chrupa and Siddiqui 2015), and improve flexibility via block-substitution (Noor and Siddiqui 2024).

Definition 13. A *BDPO plan* is denoted as $\pi_{bdpop} = \langle \mathcal{O}, \prec, \mathcal{B} \rangle$, where $\mathcal{O}$ be a collection of operators, $\prec$ is a collection of orderings within $\mathcal{O}$, and $\mathcal{B}$ is a set of blocks. For a block $b \in \mathcal{B}$, if $o_p, o_q \in b$ and $o_p \prec o_q$, then there does not exist an operator $o_r \notin b$ such that $o_p \prec o_r \prec o_q$. For any two blocks $b_p, b_q \in \mathcal{B}$, exactly one of the following holds: $b_p \subset b_q$, $b_q \subset b_p$, or $b_p \cap b_q = \emptyset$.

A block b can be characterized by its preconditions and effects, similar to an operator. If some operator $o \in b$ consumes $\langle x, d \rangle$ and no operator $o' \in b$ provides $\langle x, d \rangle$ to operator o , then a fact $\langle x, d \rangle$ is one of the preconditions of b . Conversely, if

an operator $o \in b$ generates $\langle x, d \rangle$ and no subsequent operator within b modifies it, then $\langle x, d \rangle$ is included in the effects of b . Unlike an individual operator, a block can have multiple effects on the same or different variables. For example, if $o_i, o_j \in b$ with $o_i \not\prec o_j$, $o_j \not\prec o_i$, $\langle x, d \rangle \in \text{eff}_{o_i}$, and $\langle x, d' \rangle \in \text{eff}_{o_j}$, then a block b has both $\langle x, d \rangle$ and $\langle x, d' \rangle$ as its effects (where $d \neq d'$). The labels *PC* (producer-consumer), *CT* (consumer-threat), and *TP* (threat-producer) can also be used to annotate ordering relations between blocks.

Definition 14. Let a BDPO plan $\pi_{bdpop} = \langle \mathcal{O}, \prec, \mathcal{B} \rangle$ with $b \in \mathcal{B}$ be a block, then:

- If $o_i \in b$ be an operator with $\langle x, d \rangle \in \text{pre}_{o_i}$ and there exists no operator $o_j \in b$ with $i \neq j$ that delivers $\langle x, d \rangle$ to o_i via a causal link $o_j \xrightarrow{\langle x, d \rangle} o_i$, then the fact $\langle x, d \rangle \in \text{pre}_b$.
- If $o_i \in b$ be an operator with $\langle x, d \rangle \in \text{eff}_{o_i}$ and no subsequent operator $o_j \in b$ (with $o_i \prec o_j$) produces a different value $\langle x, d' \rangle$ for the same variable x with $d' \in (\mathcal{D}_x \setminus \{d\})$, then the fact $\langle x, d \rangle \in \text{eff}_b$.

Definition 15. For a block b , the sets of facts that it consumes, produces, and deletes are denoted as **cons_b**, **prod_b**, and **del_b**, respectively. Where:

- A fact $\langle x, d \rangle \in \text{cons}_b$ if and only if $\langle x, d \rangle \in \text{pre}_b$.
- A fact $\langle x, d \rangle \in \text{prod}_b$ if and only if $\langle x, d \rangle \notin \text{cons}_b$, $\langle x, d \rangle \in \text{eff}_b$, and no other effect $\langle x, d' \rangle \in \text{eff}_b$ exists with $d' \in (\mathcal{D}_x \setminus \{d\})$.
- A fact $\langle x, d \rangle \in \text{del}_b$ if and only if either $x \notin \text{vars}(\text{cons}_b)$ or $\text{cons}_b(x) = d$, and there exists an effect $\langle x, d' \rangle \in \text{eff}_b$ with $d' \in (\mathcal{D}_x \setminus \{d\})$.

Definition 16. Let a BDPO plan $\pi_{bdpop} = \langle \mathcal{O}, \prec, \mathcal{B} \rangle$ with a pair of blocks $b, b' \in \mathcal{B}$ and $\langle x, d \rangle \in \text{cons}_{b'}$, then:

- If $\langle x, d \rangle \in \text{eff}_b$, $b \prec b'$, and no other block $b'' \in \mathcal{B}$ exists such that $\langle x, d \rangle \in \text{del}_{b''}$ and $b' \not\prec b'' \not\prec b$, then block b is a **candidate producer** of the fact $\langle x, d \rangle$ for b' .
- If there is no other candidate producer b'' of $\langle x, d \rangle$ for b' with $b'' \prec b$, then block b is considered as an **earliest candidate producer** of $\langle x, d \rangle$ for b' .

Block deordering builds a valid BDPO plan by taking a total-order plan as input. The total-order plan is first converted into a POP $\pi = \langle \mathcal{O}, \prec \rangle$ using EOG. Next, a block $b = \{o\}$ is created and added to $\mathcal{B}$ for every operator $o \in \mathcal{O}$ to construct an initial BDPO plan $\pi_{bdpop} = \langle \mathcal{O}, \prec, \mathcal{B} \rangle$. Ordering constraint $b \prec b'$ is introduced for every ordering $o \prec o'$ present in $\prec$, where $o \in b$, $o' \in b'$, and $b, b' \in \mathcal{B}$. The blocks with a single operator are denoted as *primitive block* and the blocks with multiple operators are denoted as *compound block*. The term *block* is used in a general sense to cover both. Block deordering then applies a set of rules to remove additional orderings in π_{bdpop} .

Rule 1. A valid BDPO plan $\pi_{bdpop} = \langle \mathcal{O}, \prec, \mathcal{B} \rangle$ with an ordering $b_p \prec b_q$ and let b be a block.

- i. If $b_p \in b$, $b_q \notin b$, and $b_p \not\prec b'$ for all $b' \in (b \setminus b_p)$, therefore, $PC(\langle x, d \rangle)$ can be eliminated from $Re(b_p \prec b_q)$ if $\langle x, d \rangle \in pre_b$ and there exists $b_p \notin b$ which can create causal links $b_p \xrightarrow{\langle x, d \rangle} b_q$ and $b_p \xrightarrow{\langle x, d \rangle} b$.
- ii. If $b_p \in b$, $b_q \notin b$, and $b \cap b_q = \emptyset$, therefore, $CT(\langle x, d \rangle)$ can be eliminated from $Re(b_p \prec b_q)$ if $\langle x, d \rangle \notin cons_b$.
- iii. If $b_p \notin b$, $b_q \in b$, and $b_p \cap b = \emptyset$, therefore, $CT(\langle x, d \rangle)$ can be eliminated from $Re(b_p \prec b_q)$ if $\langle x, d \rangle \notin del_b$.
- iv. If $b_q \in b$ and $b_p \notin b$, therefore, $TP(\langle x, d \rangle)$ can be eliminated from $Re(b_p \prec b_q)$ if b contains all blocks b' with $b_q \xrightarrow{\langle x, d \rangle} b'$.

Block deordering begins by examining each ordering in the initial BDPO plan from top to bottom, attempting to remove them greedily. If Rule 1 can eliminate all of the ordering reasons for an ordering $b_p \prec b_q$, then that ordering is eliminated. The ordering $b_p \prec b_q$ is kept, and the algorithm proceeds to the subsequent ordering if some reasons cannot be eliminated. Whenever an ordering is successfully removed, the modified BDPO plan is returned by the algorithm. The deordering process is then restarted by the algorithm from the top of this most recent plan. Until no more orderings can be eliminated from the most recent BDPO plan, this iterative process keeps going.

6.1 Flexibility Improvement via Block-Substitution

Block-substitution (Noor and Siddiqui 2024) permits changing a block inside a valid BDPO plan without compromising the plan's validity. The block that is being replaced is referred to as the *original block*, and the block that is replacing it is referred to as the *substituting block*. Block substitution permits the replacement block to originate either from within the plan or from an external source. When the replacement block is drawn from within the same plan, the process is referred to as an internal block substitution. Establishing causal links for the replacing block's preconditions is required for the substitution process, as well as restoring all causal links that the original block had previously supported. Furthermore, any threats introduced by the substitution must be identified and resolved to maintain the validity of the plan.

Definition 17. Let $\pi_{bdpop} = \langle \mathcal{O}, \prec, \mathcal{B} \rangle$ be a valid BDPO plan, $b \in \mathcal{B}$, and a subplan $\tilde{b} = \langle \tilde{\mathcal{O}}, \tilde{\prec} \rangle$ with $\tilde{\mathcal{O}} \subset \mathcal{O}$. A new BDPO plan $\pi'_{bdpop} = \langle \mathcal{O}', \prec', \mathcal{B}' \rangle$ is produced if b is substituted with $\tilde{b}$ where $b \notin \mathcal{B}'$ and $\tilde{b} \in \mathcal{B}'$. If π'_{bdpop} is a valid BDPO plan, then the block-substitution is considered valid.

Definition 18. Let $\pi_{bdpop} = \langle \mathcal{O}, \prec, \mathcal{B} \rangle$ be a valid BDPO plan and some causal link $b_p \xrightarrow{\langle x, d \rangle} b_c$ is threatened by $b_t \in \mathcal{B}$, where $\langle x, d \rangle \in del_{b_t}$ and $b_p, b_c \in \mathcal{B}$. A threat is

considered resolved if a **threat-resolution strategy** from the following can be used without creating a cycle in π_{bdpop} :

- i. *Promotion*: introduce the ordering $b_t \prec b_p$ into $\prec$.
- ii. *Demotion*: introduce the ordering $b_c \prec b_t$ into $\prec$.
- iii. *Internal substitution*: replace b_t with b_c , or b_c with b_t .

The procedure replaces a block b with a substituting block $\tilde{b}$. The process starts by making sure that every precondition of the $\tilde{b}$ is supported by a causal link, but $\tilde{b}$'s preconditions are already satisfied if it is internal. For an external block, $\tilde{b}$ is first included in the plan, and causal links are established using the *earliest candidate producers* for each precondition; if any precondition cannot be supported, the substitution fails. Then, using $\tilde{b}$ as the new producer, every causal link that block b initially supported is restored. If $\tilde{b}$ does not produce a required fact for any dependent block, the substitution is unsuccessful. Once these causal links are in place, b is removed from the plan. Finally, any threats introduced by the substitution are identified and resolved using threat-resolution strategies, including promotion, demotion, or internal block-substitution. In some situations, replacing a conflicting block with $\tilde{b}$ resolves threats while maintaining plan validity. The block-substitution process has an overall worst-case complexity of $O(n^2p^2)$, where p is the highest possible number of facts in any operator's effect or precondition and n is the number of operators in the plan.

Flexibility Improvement via Block-Substitution (FIBS) (Noor and Siddiqui 2024) algorithm generates a valid BDPO plan given a valid total-order plan π as input. First, using EOG, a partial-order plan $\pi_{pop} = (\mathcal{O}, \prec)$ is created from π . A BDPO plan $\pi_{bdpop} = \langle \mathcal{O}, \prec, \mathcal{B} \rangle$ is then created from this partial-order plan by constructing a block $b = \{o\}$ for each $o \in \mathcal{O}$. In the next phase, ordering constraints are removed from π_{bdpop} by substituting blocks, where initially only primitive blocks are considered, since no compound blocks have yet been formed. Subsequently, block deordering introduces compound blocks to eliminate additional orderings, further increasing plan flexibility. Finally, the procedure of substituting blocks is employed once more to replace both compound and primitive blocks, minimizing orderings in the plan. The procedure of substituting blocks examines each basic ordering $b_i \prec b_j$ in the BDPO plan and attempts to remove it by substituting b_j or, if necessary, b_i , while ensuring causal links and plan validity are maintained. After a successful removal, the procedure is restarted from the current BDPO plan's beginning, iterating until no further orderings can be eliminated. This two-stage application, before and after block deordering, allows the algorithm to separately evaluate the impact of primitive and compound block substitutions on overall plan flexibility.

FIBS significantly enhances plan flexibility while reducing computational and plan costs through efficient block-substitution within BDPO plans, outperforming existing approaches like EOG and MaxSAT reorderings. However, FIBS restricts candidate subplans for substitution to blocks within a BDPO plan, leaving other potentially useful subplans unexplored, and it has not yet been evaluated across diverse planning paradigms or with planners beyond LAMA.

6.2 Concurrency Improvement via Block- Substitution

Partial-order plans enable parallel execution by indicating which operators cannot run concurrently. A parallel plan is a POP that permits concurrent operator execution. Noor and Siddiqui (2025) transform sequential plans into parallel ones and define non-concurrency constraints in FDR. They improved the flexibility of a plan’s execution by expanding block deordering and block substitution approaches, thereby exploiting potential parallelism to minimize overall execution time.

Definition 19. A **parallel plan** is denoted as $\pi_{\mathcal{P}} = \langle \mathcal{O}, \prec, \# \rangle$, which basically extends a POP $\langle \mathcal{O}, \prec \rangle$ with an irreflexive and symmetric relation $\#$, which represents non-concurrency constraints over $\mathcal{O}$. Two operators, o_a and o_b , have a non-concurrency constraint, expressed as $o_a \# o_b$, if $\text{cons}_{o_a}(x) \neq \text{cons}_{o_b}(x)$, $\text{prod}_{o_a}(x) \neq \text{prod}_{o_b}(x)$, or $\text{cons}_{o_a}(x) \neq \text{prod}_{o_b}(x)$ for any variable $x \in \mathcal{X}$, which means that they cannot be executed simultaneously.

Definition 20. Let $\pi_{\mathcal{P}} = \langle \mathcal{O}, \prec, \# \rangle$ be a parallel plan. The ratio of the operator pairs that can be executed simultaneously to the maximum possible operator pairs is the **concurrent flexibility** value of the parallel plan $\pi_{\mathcal{P}}$, as specified in (16), expressed as $\text{cflex}(\pi_{\mathcal{P}})$.

$$\text{cflex}(\pi_{\mathcal{P}}) = 1 - \frac{\sum_{1 \leq b < a \leq |\mathcal{O}|} \begin{cases} 1 & \text{if } o_a \prec o_b, o_b \prec o_a, \text{ or } o_a \# o_b \\ 0 & \text{otherwise} \end{cases}}{\sum_{n=1}^{|\mathcal{O}|-1} n} \quad (16)$$

Two unordered operators of a partial-order plan can be executed in any sequence without invalidating the plan, but they can’t always run at the same time because they might use the same resources. To safely run operators in parallel, we must make sure they don’t interfere with each other. Knoblock (1994) grouped parallel plans based on how operators depend on each other, describing the simplest kind as those where operators are independent with respect to the goal, meaning the final result is the same whether they run one after another or together. Noor and Siddiqui (2025) focus on using such independent operators to speed up plan execution, not because parallelism is required, but because it can make plans more efficient. Methods like Knoblock (1994) and Boutilier and Brafman (2001) handle this by explicitly defining which resources each operator used, but none established how to identify parallelism without such information. Noor and Siddiqui (2025) extended the idea of the BDPO plan by adding the non-concurrency relation to allow blocks to be executed in parallel.

Definition 21. A **parallel block decomposed partial-order (PBDPO) plan** is denoted as $\pi_{\text{pbdp}} = \langle \mathcal{O}, \prec, \mathcal{B}, \# \rangle$, which basically extends a BDPO plan $\langle \mathcal{O}, \prec, \mathcal{B} \rangle$ with an symmetric, irreflexive relation $\#$ over $\mathcal{B}$, depicting non-concurrency constraints between blocks. Blocks b_p and b_q are considered non-concurrent, expressed as $b_p \# b_q$, if two primitive operators $o_p \in b_p$ and $o_q \in b_q$ are non-concurrent ($o_p \# o_q$). Variables causing this non-concurrency are denoted by the set $\text{vars}(b_p \# b_q)$.

Definition 22. Let $\pi_{pbdp} = \langle \mathcal{O}, \prec, \mathcal{B}, \# \rangle$ be a valid parallel BDPO plan with $b_p \# b_q$, where blocks $b_p, b_q \in \mathcal{B}$ are two disjoint blocks. The $b_p \# b_q$ is considered **necessary non-concurrency constraint** within π_{pbdp} if–

- i. $b_p \not\prec b_q \not\prec b_p$, and
- ii. if b_p is in the block $b_x \in B$, then b_q is also in the block b_x .

The Concurrent Flexibility Improvement via Block-Substitution (CIBS) algorithm generates a valid PBDPO plan from a valid total-order plan π for a planning task Π . The CIBS algorithm extends blocks before substitution. It selects two blocks b_p and b_q with $b_p \# b_q$ in a PBDPO plan π_{pbdp} with respect to a planning task Π . Then, the procedure includes the necessary blocks to the block b_p to achieve a new block b'_p so that another block $\hat{b}_p$ can be used in place of b'_p where $vars(\hat{b}_p \# b_q) = \emptyset$.

The CIBS algorithm enhances the concurrent flexibility through block-substitution in three steps. In the first step, EOG is applied to convert π into a parallel plan $\pi_{eog} = \langle \mathcal{O}, \prec, \# \rangle$ first, and then non-concurrency constraints between every pair of operators are identified. In the next step, cohesive operators are encapsulated into blocks using block deordering, which removes ordering constraints from π_{eog} and yields a PBDPO plan $\pi_{pbdp} = \langle \mathcal{O}, \prec, \mathcal{B}, \# \rangle$. In the last stage, blocks are replaced to improve the concurrent flexibility of π_{pbdp} .

The CIBS effectively enhances concurrent flexibility across a wide range of planning domains, outperforming both EOG and block deordering in most cases. Its consistent improvements in *cflex*, particularly in domains rich with resource-based interactions, show the benefit of incorporating substitution alongside block deordering. The observed correlations suggest that CIBS scales reasonably well with plan complexity, though execution time grows notably with larger plan sizes and more variables. However, the performance drop in extensive plans and the limited impact of block deordering on *cflex* highlight scalability and efficiency challenges. Additionally, the algorithm's reliance on available resources for substitution restricts its applicability in domains lacking such elements, indicating room for improvement in generalizing CIBS to more diverse planning tasks.

7 Analysis

The optimization capabilities of the six algorithms are summarized in Table 1 across six categories: ordering, action handling, parameter handling, plan structure, concurrency, and complexity and optimality.

All six algorithms remove unnecessary orderings from a plan, but they differ in scope. EOG operates at the operator level and produces a standard POP without block structure. Block Deordering (BD) groups coherent operators into blocks and eliminates more orderings than EOG by treating each block as a unit. MR and MRR allow arbitrary reordering and find minimum reorderings via MaxSAT encodings, though neither produces block-structured plans. Flexibility Improvement via Block-Substitution (FIBS) and Concurrency Improvement via Block-Substitution (CIBS)

Table 1: Optimization capability comparison among algorithms

Criteria	EOG	BD	MR	MRR	FIBS	CIBS
Ordering						
Remove Orderings	Yes	Yes	Yes	Yes	Yes	Yes
Arbitrary Reordering	No	No	Yes	Yes	No	No
Block-level Deordering	No	Yes	No	No	Yes	Yes
Minimizes Orderings	No	No	Yes	Yes	No	No
Action Handling						
Remove Redundant Actions	No	No	Yes	Yes	No	No
Introduce New Actions	No	No	No	No	Yes	Yes
Replace Action Sets (Subplans)	No	No	No	No	Yes	Yes
Replace Actions of Different Name	No	No	No	No	Yes	Yes
Parameter Handling						
Rebind Parameters	No	No	No	Yes	No	No
Same-name Operator Rebinding Only	-	-	-	Yes	-	-
Plan structure						
Produces Partial-order Plan	Yes	Yes	Yes	Yes	Yes	Yes
Produces BDPO Plan	No	Yes	No	No	Yes	Yes
Produces Parallel Plan	No	No	No	No	No	Yes
Produces Parallel BDPO Plan	No	No	No	No	No	Yes
External Planner for Subplans	No	No	No	No	Yes	Yes
Concurrency						
Non-concurrency Constraints (#)	No	No	No	No	No	Yes
Minimizes #	No	No	No	No	No	Yes
Improves Parallel Execution	No	Partial	No	No	Partial	Yes
Uses FDR for # Conditions	No	No	No	No	No	Yes
Block Extension via DTG Analysis	No	No	No	No	No	Yes
Complexity and Optimality						
Polynomial-time Algorithm	Yes	Yes	No	No	Partial	Partial
MaxSAT-based Encoding	No	No	Yes	Yes	No	No
Anytime / Iterative Improvement	No	No	No	No	Yes	Yes
Composable on top of MR/MRR	No	No	-	-	Yes	No

use block-level deordering but, unlike MR and MRR, do not seek a globally minimum reordering; instead, they improve the plan iteratively through substitution.

On action handling, EOG and BD leave the operator set unchanged. MR removes redundant actions as part of its encoding. MRR does the same and additionally reorders operator parameters, though only within operators of the same name. FIBS and CIBS replace entire subplans with new ones drawn from the planning task, including operators with different names and different numbers of actions. Block-substitution is therefore the only mechanism in the table that can introduce actions absent from the original plan.

Parameter rebinding is exclusive to MRR, and even there it is constrained to same-name operators. All other algorithms treat operator parameters as fixed.

All six algorithms produce a valid POP. BD, FIBS, and CIBS additionally produce BDPO plans by organizing operators into blocks. Only CIBS produces a parallel plan by incorporating a non-concurrency constraint relation ($\#$), which is a parallel block decomposed (PBD) plan. FIBS and CIBS both use an external planner to generate candidate subplans for substitution; the remaining algorithms are self-contained.

The concurrency category distinguishes CIBS from everything else in the table. EOG, BD, MR, MRR, and FIBS do not model non-concurrency constraints ($\#$). CIBS formalizes necessary and sufficient conditions for non-concurrency using FDR variables, uses a Domain Transition Graph (DTG) to determine when a block must be extended before substitution, and directly minimizes $\#$ constraints to maximize parallel execution flexibility.

EOG and BD run in polynomial time. MR and MRR are NP-complete due to their MaxSAT formulations. FIBS and CIBS are partially polynomial: the block-substitution procedure runs in $O(n^2p^2)$ time, but both use an external planner, which is not guaranteed to be polynomial. Both are anytime algorithms that produce a valid result at any point during execution. Finally, FIBS can be applied on top of MR or MRR-generated POPs for further flexibility gains; CIBS takes a sequential plan directly as input and is not designed to compose with MaxSAT reorderings.

8 Evaluation

We evaluate seven plan execution flexibility-improving methods on 3,345 plans from 46 IPC domains. These methods are EOG, BD, MR, MRR, FIBS, FIBS applied on MR output (FIBS+MR), and FIBS applied on MRR output (FIBS+MRR). EOG and BD achieve full coverage by construction, while MR, MRR, FIBS+MR, and FIBS+MRR solve 81.4%, 47.8%, 48.2%, and 21.0% of plans, respectively. When an algorithm fails to find a solution, we assume its flex value with the EOG flex for that plan, which is the lowest flex any plan achieves before any reordering or substitution.

We measure flexibility using the flex metric defined in (2). For a multi-dimensional comparison, we compute five normalized scores per algorithm: Flex (mean flex value), Coverage (proportion of plans solved), Speed (inverse mean execution time), Consistency (inverse coefficient of variation of flex), and Unique Best (proportion of plans on which the algorithm achieves the strictly highest flex). All five are min-max normalized, so a value of 1.0 denotes the best algorithm on that dimension.

8.1 Flex-Time Trade-off

Figure 1 plots mean flex against mean time per plan for all seven methods, with the Pareto frontier connecting those not dominated on both dimensions at once. EOG is the fastest at 1.06 seconds with a mean flex of 0.247. MR and FIBS+MR reach mean flex values of 0.247 and 0.249 at 29.3 and 30.4 seconds, respectively, both on the frontier. BD and FIBS improve mean flex to 0.339 and 0.341 at 44.0 and 47.4 seconds, also on the frontier. MRR and FIBS+MRR take 58.8 and 59.9 seconds but reach only 0.249 mean flex each, placing them off the frontier. The Pareto front therefore has two natural operating points: fast with modest gains (EOG, MR) and moderate cost with

substantially better flex (BD, FIBS). MRR and FIBS+MRR roughly double MR’s computation time without improving its position on the frontier.

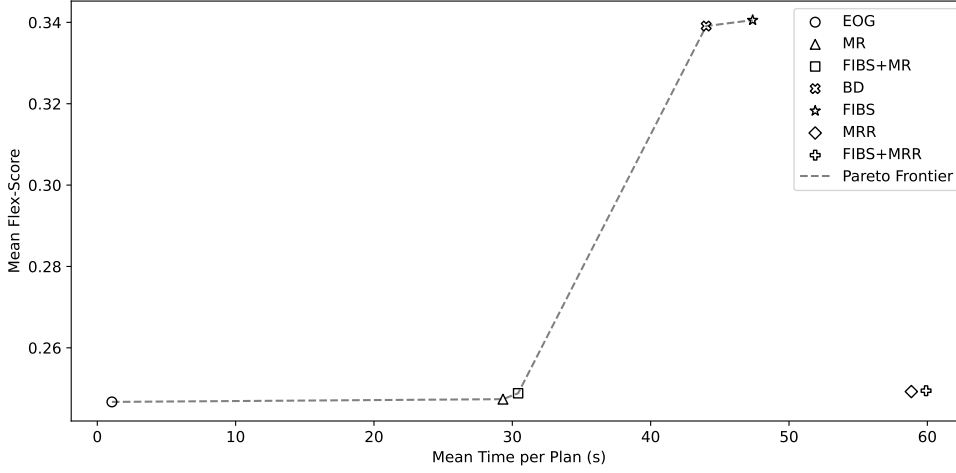


Fig. 1: Mean flex score vs. mean planning time with Pareto Frontier.

8.2 Flex Efficiency

Figure 2 shows flex efficiency per algorithm and domain, measured as flex gain over EOG per additional second spent ($\Delta\text{flex}/\Delta\text{time}$). BD has the highest median efficiency across domains: block deordering removes a large number of ordering constraints at relatively low per-plan overhead. FIBS reaches comparable efficiency because its substitution phases build directly on BD. MR and FIBS+MR cluster at intermediate values. MRR and FIBS+MRR fall to the lowest efficiency, as MRR’s encoding cost is large relative to the flex improvement it achieves.

Bubble size in Figure 2 encodes absolute mean flex gain per domain; the largest bubbles belong to BD and FIBS, confirming that these two algorithms also produce the greatest absolute improvements.

8.3 Statistical Effect Size

Figure 3 presents a pairwise effect-size heatmap using the rank-biserial correlation r from a two-sided Mann–Whitney U test on the flex distributions. A positive r in row A and column B means algorithm A produces higher flex on more plans than algorithm B ; significance levels are marked as $*$ ($p < 0.05$), $**$ ($p < 0.01$), and $***$ ($p < 0.001$).

BD and FIBS show strong positive r values against all other algorithms, with statistically significant differences in the majority of pairwise comparisons. The effect size between BD and FIBS itself is small and non-significant, so the two algorithms perform equivalently at the population level despite FIBS having a marginally higher

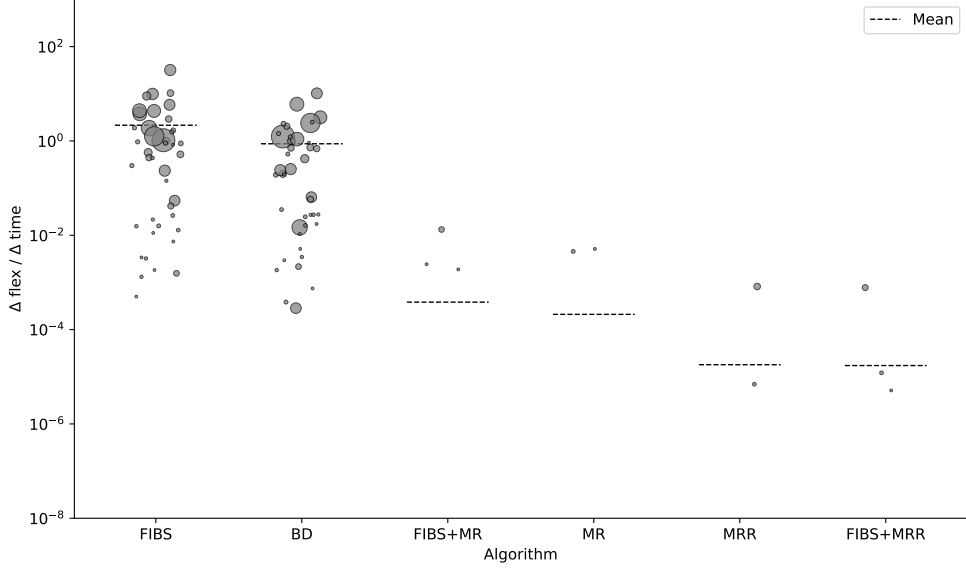


Fig. 2: Flex efficiency

mean. EOG, MR, MRR, FIBS+MR, and FIBS+MRR cluster together, with near-zero or weakly negative r values against the block-based methods. Within this cluster, pairwise effect sizes are consistently small and non-significant, indicating that the MaxSAT-based methods do not meaningfully separate from one another in terms of flex on the full plan distribution.

8.4 Multi-Criteria Comparison

Figure 4 places all seven methods on a radar chart with one axis per normalized dimension; the outer rim corresponds to the best score on each axis. BD and FIBS occupy the largest area, with normalized Flex scores of 0.985 and 1.0, perfect Coverage, and perfect Consistency. FIBS has a Unique Best score of 0.119, meaning it produces the strictly highest flex on roughly 12% of individual plans. BD’s normalized Unique Best score is 1.0, as it achieves the best flex on the largest number of plans overall. EOG scores perfectly on Speed and Coverage, and its normalized Unique Best of 0.625 is notably high: since EOG is used as the fallback for plans where MaxSAT methods time out, it retains the best score wherever all alternatives fail to terminate.

MR achieves moderate Speed (0.520) and Coverage (0.765) but a near-zero normalized Flex score relative to BD and FIBS. MRR scores near zero on Speed, Coverage, and Unique Best. FIBS+MR and FIBS+MRR also score near zero on Unique Best, since when the MaxSAT component does find a solution, FIBS adds only a small additional flex increment, and the combined pipeline rarely produces the globally best result for any individual plan.

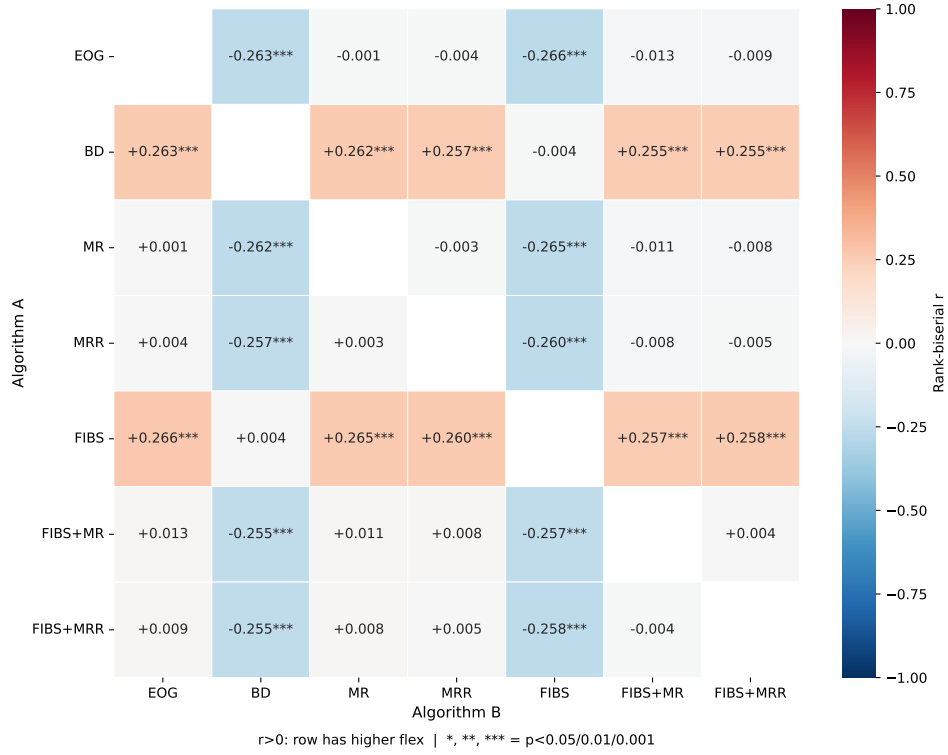


Fig. 3: Effect Size - Rank-biserial r

9 Conclusion

The methods reviewed in this study approach plan flexibility from three distinct angles: causal-link reasoning (POCL, EOG), operator coherence and block structure (BD, FIBS, CIBS), and constraint optimization (MR, MRR). Each angle captures something real about what makes a plan flexible, but the empirical results show that these angles are not equally productive. Restructuring causal dependencies at the block level, without any guarantee of minimality, yields substantially more flexibility than minimizing ordering constraints within a fixed causal structure. This is the central finding of the study, and is somewhat counterintuitive given how much theoretical work has gone into minimum reordering.

The theoretical promise of finding minimum reorderings with the MaxSAT-based methods is legitimate. Yet minimum reordering within a fixed action set is a ceiling set by the original plan’s causal structure. Block deordering raises that ceiling by reorganizing how causal dependencies are grouped, which exposes ordering constraints that would otherwise appear necessary. Block-substitution raises it further by replacing subplans with alternatives that carry different causal entailments entirely. BD’s greedy, polynomial-time heuristic therefore outperforms MR’s NP-complete search on a broad collection of IPC benchmarks.

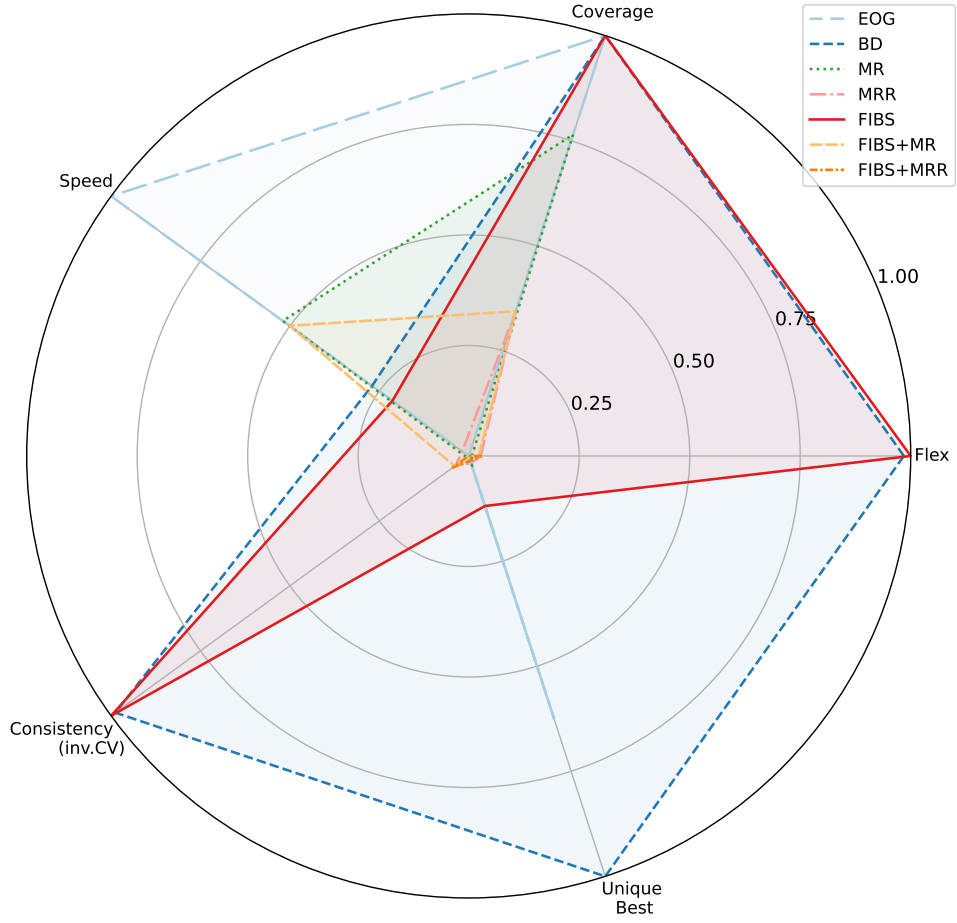


Fig. 4: Radar Chart

Minimum reordering has been studied extensively as an optimization target, but the quality of the causal structure itself has received less attention. The success of block-substitution suggests that reordering and action-set modification interact in non-trivial ways. MRR takes a step in this direction by allowing parameter rebinding, but only within same-name operators, and the evaluation shows this restriction is too tight to close the gap with BD. A method that jointly optimizes ordering and action selection without the coverage failures of MaxSAT encodings is still missing.

CIBS introduces a separate question that none of the other methods address. Two unordered operators may still be concurrent if they do not share a state variable. CIBS introduces *cflex* to account for this, instead of relying on *flex*, and the gap between the two metrics is non-trivial in resource-rich domains. However, whether a domain-independent characterization of practical concurrency is achievable, and how

to extend concurrent flexibility improvement to domains without explicit resource contention, remain open.

Statements and Declarations

Competing Interests

The authors have no relevant financial or non-financial interests to disclose.

References

- Anderson JS, Farley AM (1988) Plan abstraction based on operator generalization. In: Proceedings of the Seventh AAAI National Conference on Artificial Intelligence. AAAI Press, AAAI'88, p 100–104
- Bercher P, Höller D, Behnke G, et al (2016) More than a name? on implications of preconditions and effects of compound HTN planning tasks. In: Proceedings of the Twenty-Second European Conference on Artificial Intelligence. IOS Press, NLD, ECAI'16, p 225–233, <https://doi.org/10.3233/978-1-61499-672-9-225>
- Bercher P, Behnke G, Höller D, et al (2017) An admissible HTN planning heuristic. In: Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17, pp 480–488, <https://doi.org/10.24963/ijcai.2017/68>
- Bit-Monnot A, Smith DE, Do M (2016) Delete-free Reachability Analysis for Temporal and Hierarchical Planning (full version). In: ICAPS Workshop on Heuristics and Search for Domain-independent Planning (HSDIP), London, United Kingdom, URL <https://ut3-toulouseinp.hal.science/hal-01319768>
- Bit-Monnot A, Ghallab M, Ingrand F, et al (2020) FAPE: a constraint-based planner for generative and hierarchical temporal planning. arXiv:2010.13121
- Blum AL, Furst ML (1997) Fast planning through planning graph analysis. Artificial Intelligence 90(1):281–300. [https://doi.org/10.1016/S0004-3702\(96\)00047-1](https://doi.org/10.1016/S0004-3702(96)00047-1)
- Boutilier C, Brafman RI (2001) Partial-order planning with concurrent interacting actions. Journal of Artificial Intelligence Research 14:105–136. <https://doi.org/10.1613/jair.740>
- Bäckström C (1998) Computational aspects of reordering plans. Journal of Artificial Intelligence Research 9:99–137
- Chrapa L, Siddiqui FH (2015) Exploiting block deordering for improving planners efficiency. In: Proceedings of the 24th International Conference on Artificial Intelligence. AAAI Press, IJCAI'15, p 1537–1543

- Coles A, Coles A, Fox M, et al (2021) Forward-chaining partial-order planning. Proceedings of the International Conference on Automated Planning and Scheduling 20(1):42–49. <https://doi.org/10.1609/icaps.v20i1.13403>, URL <https://ojs.aaai.org/index.php/ICAPS/article/view/13403>
- Fikes RE, Nilsson NJ (1971) STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence* 2(3):189–208. [https://doi.org/10.1016/0004-3702\(71\)90010-5](https://doi.org/10.1016/0004-3702(71)90010-5)
- Fox M, Long D (2001) Hybrid STAN: Identifying and managing combinatorial optimisation sub-problems in planning. In: Proceedings of the 17th International Joint Conference on Artificial Intelligence - Volume 1. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, IJCAI’01, p 445–450
- Graham JR, Decker KS, Mersic M (2003) DECAF - a flexible multi agent system architecture. *Autonomous Agents and Multi-Agent Systems* 7(1–2):7–27. <https://doi.org/10.1023/A:1024120703127>
- Haslum P, Lipovetzky N, Magazzeni D, et al (2019) An Introduction to the Planning Domain Definition Language. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, Morgan & Claypool
- Helmert M (2011) The fast downward planning system. *Journal of Artificial Intelligence Research* 26. <https://doi.org/10.1613/jair.1705>
- Hickmott S, Rintanen J, Thiébaux S, et al (2007) Planning via petri net unfolding. In: Proceedings of the 20th International Joint Conference on Artificial Intelligence. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, IJCAI’07, p 1904–1911
- Kambhampati S (1994) Multi-contributor causal structures for planning: a formalization and evaluation. *Artificial Intelligence* 69(1–2):235–278. [https://doi.org/10.1016/0004-3702\(94\)90083-3](https://doi.org/10.1016/0004-3702(94)90083-3)
- Kambhampati S, Hendler JA (1992) A validation-structure-based theory of plan modification and reuse. *Artificial Intelligence* 55(2):193–258. [https://doi.org/10.1016/0004-3702\(92\)90056-4](https://doi.org/10.1016/0004-3702(92)90056-4)
- Kambhampati S, Kedar S (1994) A unified framework for explanation-based generalization of partially ordered and partially instantiated plans. *Artificial Intelligence* 67(1):29–70. [https://doi.org/10.1016/0004-3702\(94\)90011-6](https://doi.org/10.1016/0004-3702(94)90011-6)
- Kautz H, Selman B (1998) BLACKBOX: A new approach to the application of theorem proving to problem solving. In: AIPS98 Workshop on Planning as Combinatorial Search
- Knoblock CA (1994) Generating parallel execution plans with a partial-order planner. In: International Conference on Artificial Intelligence Planning Systems, pp 98–103

- Koehler J (1999) Handling of conditional effects and negative goals in IPP
- Muise C, Beck J, McIlraith S (2016) Optimal partial-order plan relaxation via MaxSAT. *Journal of Artificial Intelligence Research* 57:113–149. <https://doi.org/10.1613/jair.5128>
- Nguyen X, Kambhampati S (2001) Reviving partial order planning. pp 459–466
- Noor SB, Siddiqui FH (2022) Plan deordering with conditional effects. In: Arai K (ed) *Intelligent Systems and Applications*. Springer International Publishing, pp 852–870
- Noor SB, Siddiqui FH (2024) Improving plan execution flexibility using block-substitution. *arXiv preprint arXiv:240603091*
- Noor SB, Siddiqui FH (2025) Improving execution concurrency in partial-order plans via block-substitution. *Autonomous Agents and Multi-Agent Systems* 39(1):1–43
- Penberthy JS, Weld DS (1992) UCPOP: A sound, complete, partial order planner for adl. In: *Proceedings of the Third International Conference on Principles of Knowledge Representation and Reasoning*. Morgan Kaufmann Publishers Inc., KR’92, p 103–114
- Regnier P, Fade B (1991) Complete determination of parallel actions and temporal optimization in linear plans of action. In: *European Workshop on Planning*, volume 522 of *Lecture*. Springer-Verlag, pp 100–111
- Siddiqui FH, Haslum P (2012) Block-structured plan deordering. In: Thielscher M, Zhang D (eds) *AI 2012: Advances in Artificial Intelligence*. Springer Berlin Heidelberg, Berlin, Heidelberg, pp 803–814
- Siddiqui FH, Haslum P (2015) Continuing plan quality optimisation. *Journal of Artificial Intelligence Research* 54:369–435
- Simmons R, Younes H (2011) VHPOP: Versatile heuristic partial order planner. *Journal of Artificial Intelligence Research* 20. <https://doi.org/10.1613/jair.1136>
- Veloso M, Perez M, Carbonell J (2002) Nonlinear planning with parallel resource allocation
- Waters M, Nebel B, Padgham L, et al (2018) Plan relaxation via action debinding and deordering. *Proceedings of the International Conference on Automated Planning and Scheduling* 28(1):278–287. <https://doi.org/10.1609/icaps.v28i1.13901>, URL <https://ojs.aaai.org/index.php/ICAPS/article/view/13901>
- Waters M, Padgham L, Sardina S (2020) Optimising partial-order plans via action reinstantiation. In: *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI’20*

Weld DS (1994) An introduction to least commitment planning. *AI Magazine* 15(4):27.
<https://doi.org/10.1609/aimag.v15i4.1109>

Winner E, Veloso MM (2002) Analyzing plans with conditional effects. In: *AIPS*, pp 23–33