Planning with Primary Effects: Experiments and Analysis

Eugene Fink *
Computer Science, Carnegie Mellon University
Pittsburgh, Pennsylvania 15213, USA
eugene@cs.cmu.edu

Qiang Yang *
Computer Science, University of Waterloo
Waterloo, Ontario, Canada N2L3G1
qyang@logos.uwaterloo.ca

Abstract

The use of primary effects in planning is an effective approach to reducing search. The underlying idea of this approach is to select certain "important" effects among the effects of each operator and to use an operator only for achieving its important effects. In the past, there has been little analysis of planning with primary effects and few experimental results. We provide empirical and analytical results on the use of primary effects. First, we experimentally demonstrate that the use of primary effects may lead to an exponential reduction of the planning time. Second, we analytically explain the experimental results and identify the factors that influence the efficiency of planning with primary effects. Third, we describe an application of our analysis to predicting the performance of a planner for a given selection of primary effects.

1 Introduction

Planning with primary effects is an effective approach to reducing search. The underlying idea of this approach is to select *primary* effects among the effects of each operator and to use an operator only when we need to achieve one of its primary effects. A *primary-effect restricted* planner never inserts an operator into a plan in order to achieve any of the side effects of the operator. For example, the primary effect of lighting a fireplace is to heat the house. If we have lamps in the house, we view illumination of the room as a side effect of using the fireplace: we would not use the fireplace just to illuminate the room.

The advantages of using primary effects in planning are well-known. If a planner considers only operators whose primary effects match a current goal and ignores operators with matching side effects, the branching factor of search can be reduced, which results in an exponential reduction of running time. For this reason, primary effects are used by many implemented planning systems, such as SIPE [Wilkins, 1988], PRODIGY [Carbonell *et al.*, 1990], and ABTWEAK [Yang and Tenenberg, 1990].

In our previous work, we formalized the notion of planning with primary effects and developed two algorithms, MARGIE [Fink and Yang, 1992] and Prim-TWEAK [Fink and Yang, 1993], for automatically selecting primary effects of operators.

The purpose of this paper is to present empirical and analytical evaluation of the efficiency of planning with primary effects and identify the important factors that determine the effectiveness of primary effects in reducing planning time. We describe experiments on automatically selecting and using primary effects in planning; the experiments demonstrate that the use of primary effects may lead to an exponential efficiency improvement. We then analytically explain the efficiency improvements observed in the experiments. We also describe an application of our analysis to *predicting* the performance of a planner for a given selection of primary effects. This predictive analysis enables us to decide whether the use of primary effects selected by the user improves the efficiency of planning.

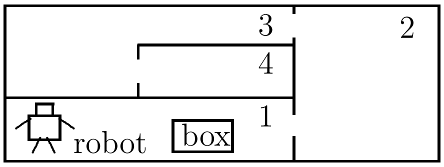
2 Planning with Primary Effects

In this section, we describe the use of primary effects in planning, outline an algorithm for automatically selecting primary effects of operators, and discuss possible problems of planning with primary effects and a method for avoiding these problems.

2.1 Motivating Example

Consider a planning domain with a robot, four rooms, and a box (see Table 1). The robot may go between two rooms connected by a door, and it may carry the box. If the robot has an ax, it can break through the wall between two rooms to create a new door.

*The first author is supported by the Wright Laboratory, Aeronautical Systems Center, Air Force Materiel Command, USAF, and the Advanced Research Projects Agency (ARPA) under grant number F33615-93-1-1330. The second author is supported in part by grants from the Natural Sciences and Engineering Research Council of Canada and ITRC.



operator	**precond's**	**effects**
go(x, y)	robot-in(x) door(x, y)	robot-in(y) ¬robot-in(x)
carry-box(x, y)	robot-in(x) box-in(x) door(x, y)	robot-in(y) box-in(y) ¬robot-in(x) ¬box-in(x)
break(x, y)	robot-in(x) have-ax	robot-in(y) ¬robot-in(x) door(x, y)

Table 1: Robot domain

To describe a current *state* of the domain, we have to specify the location of the robot and the box, list the pairs of rooms connected by doors, and indicate whether the robot has an ax. We may describe states of the domain with four predicates, *robot-in*(x), *box-in*(x), *door*(x, y), and *have-ax*. Literals describing a current state may be obtained from these predicates by substituting specific room numbers for x and y. For example, the literal *robot-in*(1) means that the robot is in Room 1, *box-in*(1) means that the box is in Room 1, and *door*(1,2) means that Rooms 1 and 2 are connected by a doorway.

The operations performed by the robot, such as moving between rooms and carrying the box, are called *operators*. We describe an operator by a *set of preconditions* and a *set of effects* [Fikes and Nilsson, 1971] (see Table 1). The *preconditions* of an operator are the conditions that must hold before the execution of the operator and the *effects* are the results of the execution. If a literal l is an effect of some operator, we say that this operator *achieves* l.

A *plan* is a sequence of operators that achieves some desired goal. For example, suppose that the initial state is as shown in Table 1, and we wish to bring the robot to Room 3 and the box to Room 2; this goal may be achieved by the plan (*carry-box*(1,2), *go*(2,3)).

In this paper, we measure the quality of a plan by the number of operators in the plan: the fewer operators, the better the plan. The number of operators is called the *size* of the plan. An *optimal* plan is a plan of the smallest size that achieves the goal. We may readily extend all our results to more robust definitions of a plan quality [Fink and Yang, 1994]; for example, we may assign a real-valued cost to each operator in a domain and measure the quality of a plan by the total cost of its operators.

If an operator has several effects, we may choose certain "important" effects among them and insert the operator into a plan only for the sake of these effects. The chosen important effects are called *primary* and the other effects are called *side* effects. A planner that adds new operators to a plan only for achieving their primary effects is called a *primary-effect restricted* planner.

For example, consider the following selection of primary effects:

Operators	**Prim. Effects**
go(x, y)	robot-in(y)
carry-box(x, y)	box-in(y)
break(x, y)	door(x, y)

Suppose that the initial state is as shown in Table 1 and the robot must move into Room 3. The robot may achieve this goal by going to Room 3 (through Room 2) or by carrying the box to Room 3; if the robot has an ax, it may also achieve the goal by breaking through the wall between Rooms 1 and 3. Since *robot-in* is *not* a primary effect of the operators *carry-box* and *break*, the planner will not consider the use of either of these operators, and it will find the plan (*go*(1,2), *go*(2,3)).

2.2 Selecting Primary Effects

The utility of planning with primary effects depends crucially on a "good" selection of primary effects. An improper selection may result in the loss of completeness of planning, which happens when a primary-effect restricted planner cannot find a plan for a solvable planning problem. For example, if we use the primary effects chosen in the end of Section 2.1, a planner cannot find a plan for removing the box from Room 4, since ¬*box-in* is not a primary effect of any operator.

In [Fink and Yang, 1992] we showed that enforcing the following two restrictions usually (although not always) results in preserving completeness of planning:

1. every achievable literal must be a primary effect of some operator,

2. and every operator must have a primary effect.

Restriction 1 guarantees that, if a literal can be achieved at all, then it can be achieved as a primary effect of some operator, and Restriction 2 ensures that the planner can use all operators of the domain. We use the following simple algorithm for selecting primary effects according to the two restrictions:

1. For every literal l in the domain,
   if there are operators that achieve l,
   then make l a primary effect of one of these operators.
2. For every operator that does not have primary effects,
   make one of its effects primary.

Note that there may be several different ways of selecting "one of the operators" at Step 1 of the algorithm and "one of the effects" at Step 2; we presented heuristics for making these choices in [Fink and Yang, 1993] and [Fink and Yang, 1994]. The algorithm would select the following primary effects in the robot domain:

Operators	Prim. Effects
go(x, y)	robot-in(y), ¬robot-in(x)
carry-box(x, y)	box-in(y), ¬box-in(x)
break(x, y)	door(x, y)

2.3 Learning Additional Primary Effects

While the use of primary effects chosen by the above algorithm may improve the efficiency of planning, it can also cause two serious problems. First, the resulting selection of primary effects is not immune to the loss of completeness. Second, planning with the selected primary effects may result in finding a nonoptimal plan. For example, if the robot has an ax, the goal of moving the robot from Room 1 to Room 4 may be achieved by the operator *break*(1,4); however, a primary-effect restricted planner will not consider this possibility, since the new position of the robot is not a primary effect of *break*. Instead, the planner will find the plan (*go*(1,2), *go*(2,3), *go*(3,4)), with a size of 3. In this example, the shortest primary-effect restricted plan is 3 times longer than the optimal plan. We say that 3 is the *plan-size increase* of the planning problem.

To address the problem of completeness and optimality, we present an algorithm that tests whether a given choice of primary effects guarantees the completeness of planning and estimates the maximal plan-size increase.

Planning with primary effects is complete if a primary-effect restricted planner can find a plan for every solvable problem. Completeness is lost when we can achieve a goal as a side effect of some operator, but cannot find a primary-effect restricted plan that achieves this goal. To guarantee completeness, we must ensure that such a situation cannot happen; that is, we have to make sure that, *for every operator, there is always a primary-effect restricted plan that achieves its side effects.*

Now suppose that a somewhat stronger condition holds: for some constant C, every operator Op can always be replaced by a primary-effect restricted plan *whose size is at most* C. Then, the plan-size increase due to the use of primary effects is never larger than C. Thus, this stronger condition not only ensures completeness, but also guarantees a limited plan-size increase for all possible problems in a planning domain. Below, we summarize our observations:

> **Completeness and Limited Plan-Size Increase:** Suppose that, for every operator Op and every initial state S satisfying the preconditions of Op, there exists a primary-effect restricted plan $\mathcal{P}$ with the initial state S such that
> (1) $\mathcal{P}$ achieves the side effects of Op
> (2) and $\mathcal{P}$ contains at most C operators.
> Then, for every solvable problem in the planning domain, there is a primary-effect restricted solution plan at most C times larger than an optimal plan.

A formal proof and discussion of this observation can be found in [Fink and Yang, 1994]. We use this result to design a learning algorithm that adds new primary effects to a selection to make sure that primary-effect restricted planning is complete and that the plan-size increase is within the user-specified bound. For each operator Op in the domain, the learner generates several states that satisfy the preconditions of Op and checks whether the side effects of Op can be achieved in all these states. If not, the algorithm "promotes" some side effects of Op to primary effects.

Let C be the maximal allowed plan-size increase, specified by the user. The learning algorithm can be informally described as follows:

For every operator Op, repeat "several" times:
1. Pick at random a state S
   that satisfies the preconditions of Op.
2. Try to find a primary-effect restricted plan $\mathcal{P}$ such that
   ∘ $\mathcal{P}$ achieves the side effects of Op from the state S,
   ∘ and $\mathcal{P}$ contains at most C operators.
3. If such a plan is not found,
   make one of the side effects of Op be a primary effect.

In [Fink and Yang, 1993] we presented a formal description of this algorithm, described methods for generating random legal states satisfying the preconditions of a given operator, and estimated the number of different states that must be considered for every operator to ensure a high probability of completeness.

As an example, suppose that we apply this learning algorithm to the robot domain, with primary effects shown in the end of Section 2.2 and $C = 2$. The learner is likely to notice that, if the robot has an ax, the effect *robot-in*(4) of the operator *break*(1,4) cannot be efficiently achieved by a primary-effect restricted plan. After noticing it, the algorithm will select an additional primary effect of the *break* operator, producing the following selection:

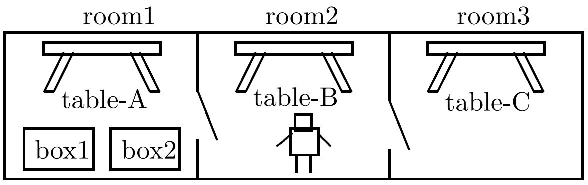
Operators	Prim. Effects
go(x, y)	robot-in(y), ¬robot-in(x)
carry-box(x, y)	box-in(y), ¬box-in(x)
break(x, y)	door(x, y), robot-in(y)

3 Search Reduction: Experiments

We now present a series of experiments on planning with primary effects; the experiments demonstrate that the use of primary effects may drastically improve the efficiency of planning. We used the ABTWEAK planning system [Yang and Tenenberg, 1990] in the experiments; the primary effects were selected automatically by the learning algorithm outlined in Section 2.3.

3.1 Experiments in a Robot Domain

We first describe experiments in a robot domain, where the robot can move between rooms, open and close doors, carry boxes, and climb tables (with or without boxes). We ran our learning algorithm to select primary effects in this domain, with the plan-size increase $C = 1$; the



Goal	Solution Size	CPU time, ms w/ prim	w/o
robot-on(table-B)	1	50	50
status(door12, open)	2	183	217
robot-in(room1)	3	267	350
box-at(box1, door12)	5	800	1933
robot-on(table-C)	5	783	1200
box-on(box1, table-A)	5	800	1250
box-at(box2, table-B)	7	6550	14450

Table 2: Planning time in the Robot Domain

time of learning primary effects was 2980 CPU msec. In Table 2, we show the performance of the ABTWEAK planner without primary effects ("w/o") and with the use of the learned primary effects ("w/ prim") when achieving different goals. The initial state in all cases is as shown in the picture, with both doors being closed. For all problems, the solution found with the use of primary effects was the same as the solution found without primary effects. As can be seen from the table, the use of primary effects considerably improves the efficiency of ABTWEAK.

3.2 Experiments in Artificial Domains

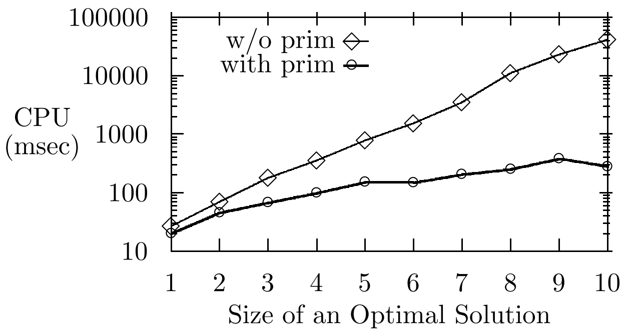
We next describe a series of experiments in artificial domains, similar to the domains used in [Barrett and Weld, 1992]. These domains have a number of features that can be varied independently, which enables us to perform controlled experiments.

We define a problem by m initial-state literals, $init_1, .., init_m$, and m goal literals, $goal_1, .., goal_m$. We use m operators, $Op_1, .., Op_m$. Each operator Op_i has the single precondition $init_i$. Op_i deletes the literal $init_{i-1}$ and establishes the goal literals $goal_{i \bmod m}, goal_{(i+1) \bmod m}, .., goal_{(i+k-1) \bmod m}$. Thus, each operator has $(k+1)$ effects: one negative effect and k positive effects. The artificial domains allow us to vary the following problem features:

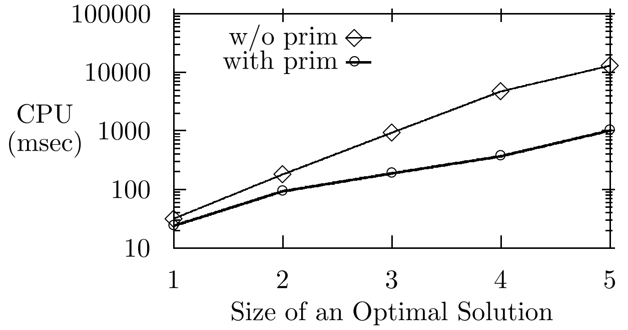
Goal Size The *goal size* is the number of goal literals, m. The size of an optimal solution plan changes in proportion to the number of goal literals.

Effect Overlap The *effect overlap*, k, is the average number of operators establishing the same literal.

We vary these two features in the controlled experiments. Although the domains are artificial, they demonstrate some important characteristics of real-world problems: first, if the goal size increases, the size of the optimal solution plan also increases; second, if the effect overlap increases, then every operator can achieve more goal literals and the size of the solution decreases.



Effect overlap k equals 2



Effect overlap k equals 4

Figure 1: Dependency of the planning time on the size of an optimal solution plan

Varying solution sizes.
First, we demonstrate that the savings in running time grow exponentially with the size of an optimal solution plan. We use domains with effect overlaps $k = 2$ and $k = 4$. The results of the experiments are presented in Figure 1: we show the running times of the ABTWEAK planner without primary effects ("w/o prim") and with the use of the learned primary effects ("with prim") for problems with different optimal-solution sizes; every point on each figure is the average of five different problems. (Note that the time scale is *logarithmic*.) These graphs show that *the savings in planning time increase exponentially with an increase in the optimal-solution size*; a significant improvement is achieved in both cases.

Varying effect overlap.
Next, we consider efficiency improvement for different values of the effect overlap. Recall that the effect overlap k is defined as the average number of operators achieving the same literal. We vary the effect overlap k from 2 to 6. (If the effect overlap is 1, then all effects must be selected as primary, and primary-effect restricted planning is equivalent to planning without primary effects.)

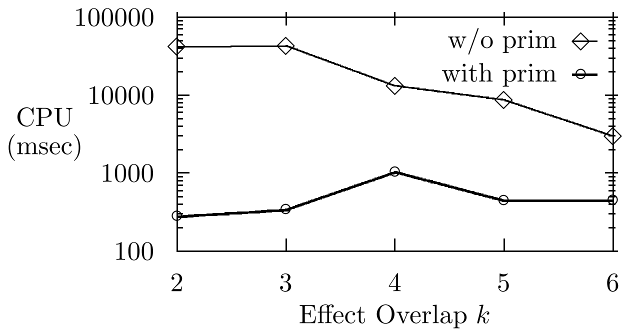


Figure 2: Dependency of the time on the effect overlap

In Figure 2, we present the running times of ABTWEAK without primary effects ("w/o prim") and with the use of learned primary effects ("with prim") for every effect overlap. The graphs show that the use of primary effects improves performance for all effect overlaps, even though the time savings are smaller for large overlaps.

4 Analysis of the Search Reduction

We next present an analytical comparison of planning efficiency with and without the use of primary effects. The purpose of the analysis is to (1) explain the exponential efficiency improvement observed in the experiments and (2) identify the factors that determine the efficiency of planning with primary effects. The analysis is an approximation based on several simplifying assumptions about properties of planning domains.

Assumptions of the analysis.
When searching for a solution to a planning problem, a planning algorithm expands a search space, whose nodes correspond to intermediate (possibly incorrect) plans found in the process of planning. The planning time is proportional to the number of nodes in the expanded search space. Total-order planners, such as ABSTRIPS and PRODIGY, create a new node by inserting a new operator into a plan. Partial-order planners, such as TWEAK and SNLP, create a node by inserting a new operator or by imposing a constraint on the order of executing old operators. For simplicity, we assume that the planner expands nodes breadth first.

When inserting a new operator to achieve some literal l, we may use any operator whose primary effect is l. To estimate the number of the matching operators, we assume that, for all literals l, the number of operators achieving l as a primary effect is the same. We define P as $P = \sum_{Op} \#(\textit{Prim-Eff}(Op))$; that is, we count the number of primary effects, $\#(\textit{Prim-Eff}(Op))$, of each operator Op and define P as the sum of these numbers for all operators. The number of different achievable literals in the domain is denoted by L. Then, the number of operators achieving a particular literal l as a primary effect is estimated as $\frac{P}{L}$.

P	number of primary effects of all operators
E	number of all effects of all operators
L	number of achievable literals in the domain
b	number of different ways to impose ordering constraints after inserting an operator
n	number of operators in the optimal solution plan
C	max. size increase of planning with primary effects
R	ratio of running times w/ and w/o primary effects

Table 3: Summary of the notation

After inserting some operator into a partial-order plan, we may have to impose new constraints on the order of operators in the plan. Let b denote the number of different ways of imposing these ordering constraints. Thus, we can create $b\frac{P}{L}$ different plans by adding an operator that achieves l and imposing necessary ordering constraints. A breadth-first planner considers all these plans and, therefore, the total number of new nodes created after inserting an operator is $b\frac{P}{L}$. This value represents the branching factor of inserting an operator into a primary-effect restricted plan.

Similarly, we assume that the number of all operators achieving l is the same for all literals and define E by $E = \sum_{Op} \#(\textit{Effects}(Op))$, where $\#(\textit{Effects}(Op))$ is the number of all effects of Op. Then, the branching factor of inserting an operator by a planner that does not use primary effects is $b\frac{E}{L}$. (The effect overlap k, defined in the last section, equals $\frac{E}{L}$.)

The notation used in the analysis is summarized in Table 3 (the last three symbols listed in Table 3 are introduced in the next few paragraphs).

Explaining exponential efficiency improvement.
First, we consider planning without primary effects. Assume that an optimal plan that solves a problem contains n operators. The branching factor of inserting an operator is $b\frac{E}{L}$; thus, the number of nodes with one-operator plans is $b\frac{E}{L}$, the number of nodes with two-operator plans is $(b\frac{E}{L})^2$, etc. The total number of nodes is

$$1 + b\frac{E}{L} + (b\frac{E}{L})^2 + ... + (b\frac{E}{L})^n = \frac{(b\frac{E}{L})^{n+1} - 1}{b\frac{E}{L} - 1}. \quad (1)$$

If the maximal plan-size increase is C, then the number of operators in the solution found by a primary-effect restricted planner is at most $C \cdot n$. Since the branching factor of inserting an operator in primary-effect restricted planning is $b\frac{P}{L}$, the number of nodes in the search space of a primary-effect restricted planner is *at most*

$$\frac{(b\frac{P}{L})^{C \cdot n+1} - 1}{b\frac{P}{L} - 1}. \quad (2)$$

Let R denote the ratio of running times of planning with and without primary effects. Since the running time is proportional to the number of nodes in the search space, R is determined by the ratio of the search-space sizes:

$$R \leq \frac{((b\frac{P}{L})^{C\cdot n+1}-1)/(b\frac{P}{L}-1)}{((b\frac{E}{L})^{n+1}-1)/(b\frac{E}{L}-1)} = O\left(\left(\frac{(b\frac{P}{L})^{C}}{b\frac{E}{L}}\right)^{n}\right) \quad (3)$$

Let us denote the base of the power in Formula 3 by r:

$$r = \frac{(b\frac{P}{L})^{C}}{b\frac{E}{L}} = \frac{L}{E\cdot b}\cdot\left(\frac{P\cdot b}{L}\right)^{C}. \quad (4)$$

Then, Formula 3 may be rewritten as $R = O(r^n)$. This expression explains the experimental results: it demonstrates that *the savings in running time grow exponentially with the size of an optimal plan, n*.

The value of r in the robot domain (Section 3.1) is 0.9 and, thus, the ratio R of running times of planning with and without primary effects is approximately 0.9^n, where n is the size of an optimal solution; the value of r in the artificial domains of Section 3.2 varies from 0.6 to 0.8.

Factors that determine the search reduction. The use of primary effects improves the efficiency of planning only if $r < 1$, which means that $\frac{L}{E\cdot b}\cdot(\frac{P\cdot b}{L})^{C} < 1$. Solving this inequality with respect to the plan-size increase C, we conclude that primary effects improve performance when

$$C < \frac{\log E + \log b - \log L}{\log P + \log b - \log L}. \quad (5)$$

We can draw some other conclusions from the expression for r (Formula 4). The base of the power in this expression, $\frac{P\cdot b}{L}$, is the branching factor of the primary-effect restricted search and, therefore, it is at least 1. Since $r = \frac{L}{E\cdot b}\cdot(\frac{P\cdot b}{L})^{C}$, we conclude that r increases with an increase of C, which implies that *the time savings increase with a decrease of the plan-size increase C*. Also, r increases with an increase of P and, therefore, *the time savings increase with a decrease of the number of primary effects P*.

However, there is a tradeoff between the number of primary effects P and the plan-size increase C: as we select more primary effects, P increases, whereas C decreases. To minimize r, we have to strike the right balance between C and P, which may be done by a modified version of the learner outlined in Section 2.3; we described this modified version in [Fink and Yang, 1994].

5 Predicting Savings in Planning Time

We next describe an application of the analysis to determining whether the use of a given selection of primary effects improves the efficiency of planning. The key component of this predictive analysis is estimating the plan-size increase C for given primary effects.

To describe our method for estimating the plan-size increase, we introduce the notion of a *primary-effect cover* of an operator. We say that operators $Op_1, .., Op_n$ form a primary-effect cover of an operator Op if every side effect of Op is a primary effect of at least one of the operators $Op_1, .., Op_n$.

operator	**precond's**	**effects**	**prim eff**
use-lamps(x)	have-lamps(x)	light(x)	light(x)
use-fireplc(x)	have-fireplc(x) have-wood	warm(x) light(x)	warm(x)

Table 4: Fireplace domain

To estimate the plan-size increase, we generate a primary-effect cover with as few operators as possible for every operator Op in the planning domain. We use the number of operators in the primary-effect cover of Op as an estimate of the size of a primary-effect restricted plan $\mathcal{P}$ that achieves the side effects of Op.

We use a greedy algorithm to generate a small primary-effect cover of every operator in the domain. The algorithm for finding a cover of a given operator, Op, may be informally described as follows:

Set *Uncovered-Eff* = *Side-Eff*(*Op*) and *Cover* = $\emptyset$.
While *Uncovered-Eff* is not empty, repeat:
1. Select an operator whose primary effects cover the maximal number of predicates in *Uncovered-Eff*.
2. Add this operator to *Cover*.
3. Remove newly covered predicates from *Uncovered-Eff*.

After generating a cover of every operator in the planning domain, we count the number of operators, $\#(Cover(Op))$, in the cover of every operator Op; we use the maximum of these numbers as an estimate of C:

$$C \approx \max_{Op} \#(Cover(Op)). \quad (6)$$

We substitute this estimate of C into Formula 5 to determine whether the use of primary effects improves the efficiency of planning. We may also use Formula 3 to estimate the savings in planning time.

For example, suppose that we apply this predictive analysis to the robot domain in Table 1 (Section 2.1), with primary effects shown in the end of Section 2.3. The cover size of every operator in this domain is 0 or 1 and, thus, we estimate the plan-size increase as $C \approx 1$. We next note that the total number of primary effects of operators in this domain is $P = 6$ and the total number of all effects is $E = 9$. Substituting these values into Formula 5, we find out that the inequality of Formula 5 is satisfied and, therefore, the use of primary effects improves the efficiency of planning.

As another example, consider a fireplace domain, mentioned in Section 1. We show a formal description of this domain in Table 4, where x can be substituted with different rooms. In this domain, $P = 2$, $E = 3$, and the cost increase is estimated as $C \approx 1$. Using Formula 5, we again conclude that primary effects improve efficiency.

6 Conclusions

We have presented an empirical and analytical evaluation of planning with primary effects and demonstrated that the use of primary effects considerably reduces the planning time, but an improper selection of primary effects may lead to generating nonoptimal plans and increasing the planning time. The crucial factor is the plan-size increase C, which determines the optimality of plans based on primary effects and the efficiency of planning.

The experimental and analytical results show that good selections of primary effects result in considerable efficiency improvement and that the savings in planning time grow exponentially with the problem complexity.

Acknowledgements

The authors would like to thank Manuela Veloso, Steven Minton, Yury Smirnov, Alicia Pérez, Alissa Kaplounova, Constantine Domashnev, and Robert Monroe for helpful comments and discussions.

References

[Barrett and Weld, 1992] Anthony Barrett and Dan Weld. Partial order planning: Evaluating possible efficiency gains. University of Washington, Computer Science and Engineering, 1992. Tech. Report 92-05-01.

[Carbonell *et al.*, 1990] Jaime G. Carbonell, Craig A. Knoblock, and Steven Minton. PRODIGY: An integrated architecture for planning and learning. In *Architectures for Intelligence*, ed.: Kurt VanLehn, Erlbaum, Hillside, NJ, 1990.

[Fikes and Nilsson, 1971] Richard E. Fikes and Nils J. Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2, pages 189–208, 1971.

[Fink and Yang, 1992] Eugene Fink and Qiang Yang. Automatically abstracting effects of operators. *First International Conference on AI Planning Systems*, pages 243–251, 1992.

[Fink and Yang, 1993] Eugene Fink and Qiang Yang. Characterizing and automatically finding primary effects in planning. *International Joint Conference on Artificial Intelligence*, pages 1374–1379.

[Fink and Yang, 1994] Eugene Fink and Qiang Yang. *Automatically Selecting and Using Primary Effects in Planning: Theory and Experiments.* Carnegie Mellon University, Computer Science, 1994. Tech. Report CMU-CS-94-206.

[Wilkins, 1988] David Wilkins. *Practical Planning: Extending the Classical AI Planning Paradigm*, Morgan Kaufmann, CA, 1988.

[Yang and Tenenberg, 1990] Qiang Yang and Josh Tenenberg. ABTWEAK: abstracting a nonlinear, least commitment planner. *Eighth National Conference on Artificial Intelligence*, pages 923–928, 1990.