Heuristics for Cost-Optimal Classical Planning Based on Linear Programming[†]

Florian Pommerening and **Gabriele Röger** and **Malte Helmert**
Universität Basel
Basel, Switzerland
{florian.pommerening,gabriele.roeger,malte.helmert}@unibas.ch

Blai Bonet
Universidad Simón Bolívar
Caracas, Venezuela
bonet@ldc.usb.ve

Abstract

Many heuristics for cost-optimal planning are based on linear programming. We cover several interesting heuristics of this type by a common framework that fixes the objective function of the linear program. Within the framework, constraints from different heuristics can be combined in one heuristic estimate which dominates the maximum of the component heuristics. Different heuristics of the framework can be compared on the basis of their constraints. We present theoretical results on the relation between existing heuristics and experimental results that demonstrate the potential of the proposed framework.

1 Introduction

The central problem when designing an autonomous agent is the problem of selecting the next action for the agent to execute. In a truly autonomous system, this selection is based only on the sequence of the actions performed and observations gathered, called *execution*, up to the time a new action is needed. This problem is known as the control problem.

There are three main approaches for dealing with the control problem. In the first approach, the programming approach, the controller that selects the actions is hardwired in the form of a program or circuit that is *specific for the task*. This approach is used in simple problems or for agents that implement simple control strategies, but it often falls short of selecting appropriate actions when the agent encounters executions that were not anticipated by the 'programmer'. In the second approach, the learning approach, the agent is embedded in the system and *learns the controller* as it tries the different actions and observes their outcomes. In practice, however, the agent needs to explore too many executions to learn a useful controller making it unfeasible except for small tasks. In the third approach, the model-based approach, the system is described using a model that specifies the possible states of the system, the effects and observations obtained after executing the actions, and the goals that the agent tries to achieve. This model is used to *automatically synthetise* a controller that maps executions to the next action to perform. The problem is thus cast as a synthesis problem from a given specification. Two obstacles for this approach are that a suitable model for the task is needed, and that the synthesis problem is intractable in general. But, this intractability does not preclude the approach from being effective in meaningful cases. Planning is the model-based approach to autonomous behaviour. For a general introduction to planning models and methods along these lines, see Geffner and Bonet [2012].

Classical planning is the simplest type of model in which the initial state is fully known, the actions behave deterministically, and agent tries to achieve one of possibly many goal states. The determinism implies that the agent is capable of predicting the outcomes of the actions and, together with the knowledge of the initial state, a controller is just a sequence of actions to execute from the initial state and that results in a goal state. On the other hand, the synthesis problem for classical planning is the problem of finding a path in an implicit graph of exponential size whose vertices stand for system states and edges stand for action applications. In this graph, we can look for *satisficing* or *(cost-)optimal* plans that correspond to arbitrary or optimal paths in the graph.

In this paper we focus on the problem of computing optimal plans. A common way to find optimal paths in an implicit graph is the standard best-first search with an *admissible heuristic* (i.e. A*). Such a heuristic must map states to cost estimates that are less than or equal to the cost of the best path from the state to a goal [Russell and Norvig, 2003].

Several recent admissible heuristics [van den Briel *et al.*, 2007; Karpas and Domshlak, 2009; Bonet, 2013; Pommerening *et al.*, 2013] for cost-optimal planning show that it is feasible and beneficial to obtain estimates by solving a linear program for every state encountered during the search. The approaches differ in their formulation of the objective and constraints, which depend on the type of information and problem structure that is exploited for the computation of the heuristics. Karpas and Domshlak use landmarks, Pommerening et al. exploit information from abstraction heuristics, and van den Briel *et al.* and Bonet base their linear programs on network flows for the state variables.

We will show that all these approaches are covered by a single framework that fixes the optimization function of the linear program. The framework provides:

[†] This paper was invited for submission to the Best Papers From Sister Conferences Track, based on a paper that appeared in the International Conference on Automated Planning and Scheduling (ICAPS) 2014.

- a clear and crisp formulation that is able to capture many of the existing state-of-the-art heuristics for optimal planning,
- a simple way to combine different heuristics (and thus different types of information) into new heuristics that are more powerful than the isolated component heuristics,
- new methods of analysis that allow us to better understand the limitations of the current heuristics and to reveal deep connections between heuristics, and
- a practical way to define heuristics that currently set the state of the art in heuristics for optimal planning.

Helmert and Domshlak [2009] provide the following classification for most of the heuristics for optimal planning, including all the heuristics that are state of the art:

- the delete relaxation [Bonet and Geffner, 2001],
- critical paths: the h^m family [Haslum and Geffner, 2000],
- abstractions like pattern databases [Edelkamp, 2001], merge-and-shrink abstractions [Helmert *et al.*, 2014] and structural patterns [Katz and Domshlak, 2009], and
- landmarks, like the admissible landmark heuristics of Karpas and Domshlak [2009].

As we will see, many of these heuristics can be captured efficiently in our framework, while for others we still do not know. Characterizing known heuristics within the proposed framework is not only of theoretical interest. The information provided by one heuristic can be easily combined with the information provided by other heuristics to define novel heuristics that often exhibit a synergistic effect: the new estimates are often strictly more informed than the estimates provided by each component alone.

Recently, the LP-based framework was used to express even better heuristics either by extracting more useful information from the problem or by observing novel ways to optimally combine the estimates provided by different heuristics [Bonet and van den Briel, 2014; Pommerening *et al.*, 2015].

We start by introducing SAS$^+$ planning and our new framework, which is based on *operator-counting constraints*. Afterwards, we present a wide range of such constraints and explain how they are used to express existing and novel heuristics. We then present some experimental results and end with a discussion. The reader is referred to Pommerening *et al.* [2014] for a more complete exposition and results.

2 The SAS$^+$ Model and Heuristics

A SAS$^+$ *task* is a tuple $\Pi = \langle \mathcal{V}, \mathcal{O}, s_I, s_G, \mathit{cost} \rangle$ where $\mathcal{V}$ is a finite set of *variables*. Each variable $V \in \mathcal{V}$ has a finite domain D_V. A (partial) *state* s is a (partial) variable assignment over $\mathcal{V}$. We write $\mathit{vars}(s)$ for the domain of definition of s and $s[V]$ for the value of V in s. A partial state s is *consistent* with a partial state s' if $s[V] = s'[V]$ for all $V \in \mathit{vars}(s')$. An *atom* $V = v$ is true in state s iff $s[V] = v$.

Each operator o in the finite set of *operators* $\mathcal{O}$ is associated with a precondition $\mathit{pre}(o)$ and an effect $\mathit{eff}(o)$, which are both partial variable assignments over $\mathcal{V}$. We require that $V = v$ cannot be both a precondition and an effect of o. This is not a real restriction because such effects would be redundant. The (complete) state s_I is the *initial state* of the task and the partial state s_G describes its *goal*. The *cost function* $\mathit{cost} : \mathcal{O} \to \mathbb{R}_0^+$ assigns a non-negative cost to each operator.

An operator o is *applicable* in a state s if s is consistent with $\mathit{pre}(o)$. The *resulting state* of applying an applicable operator o in state s is the state $\mathit{res}(o, s)$ with

$$\mathit{res}(o,s)[V] = \begin{cases} \mathit{eff}(o)[V] & \text{if } V \in \mathit{vars}(\mathit{eff}(o)) \\ s[V] & \text{otherwise.} \end{cases}$$

A sequence of operators $\pi = \langle o_1, \ldots, o_n \rangle$ is applicable in state s_0 if there are states $s_1, \ldots, s_n$ with $s_i = \mathit{res}(o_i, s_{i-1})$ for $1 \leq i \leq n$. The resulting state of this application is $\mathit{res}(\pi, s_0) = s_n$. The cost of π is the sum of its operator costs $\mathit{cost}(\pi) = \sum_{i=1}^{n} \mathit{cost}(o_i)$.

For state s, an *s-plan* is an operator sequence π applicable in s such that $\mathit{res}(\pi, s)$ is consistent with s_G. An s_I-plan is just called a *plan*. A plan with minimal cost is called *optimal*.

A function h that maps states to non-negative numbers (or ∞) is called a *heuristic*. A heuristic h is called *admissible* if $h(s) \leq h^*(s)$ for all states s, where $h^*(s)$ is the cost of an optimal s-plan (or ∞ if no s-plan exists).

3 Operator-counting Constraints

Recently proposed heuristics [van den Briel *et al.*, 2007; Bonet, 2013; Pommerening *et al.*, 2013] are based on linear programs of similar form. They formalize constraints that must be satisfied by every plan π and use a variable Y_o for each operator o such that setting Y_o to the number of occurrences of o in π satisfies the constraints.

We will show that these heuristics (and some other ones) can be covered by a common framework based on the notion of *operator-counting constraints*:

Definition 1 (Operator-counting constraints) *Let Π be a planning task with operator set $\mathcal{O}$, and let s be one of its states. Let $\mathcal{Y}$ be a set of non-negative real-valued and integer variables, including an integer variable Y_o for each operator $o \in \mathcal{O}$ along with any number of additional variables. The variables Y_o are called* operator-counting variables.

If π is an s-plan, we denote the number of occurrences of operator $o \in \mathcal{O}$ in π with Y_o^π. A set of linear inequalities over $\mathcal{Y}$ is called an operator-counting constraint *for s if for every s-plan π, there exists a feasible solution with $\mathsf{Y}_o = Y_o^\pi$ for all $o \in \mathcal{O}$.*

A constraint set *for s is a set of operator-counting constraints for s where the only common variables between constraints are the operator-counting variables.*

As an example, $C_1 = \{\mathsf{Y}_{o_1} - 2\mathsf{Y}_{o_2} \geq 0\}$ is an operator-counting constraint expressing that in every plan, o_1 must occur at least twice as often as o_2. The set of inequalities $C_2 = \{\mathsf{Y}_{o_1} - \mathsf{Z} \geq 2, \mathsf{Y}_{o_2} + \mathsf{Z} \geq 1\}$ with auxiliary integer variable Z is another constraint. The set $\{C_1, C_2\}$ is a constraint set because C_1 and C_2 do not share auxiliary variables.

Definition 2 (Operator-counting programs) *The* operator-counting integer program IP_C *for constraint set C is:*

$$\textit{Minimize } \textstyle\sum_{o \in \mathcal{O}} \mathit{cost}(o)\mathsf{Y}_o \textit{ subject to } C\,.$$

The operator-counting linear program LP_C *is the LP-relaxation of* IP_C.

From Definition 1, if π is a plan, then there exists a solution to both $\text{IP}_\mathcal{C}$ and $\text{LP}_\mathcal{C}$ where $\mathsf{Y}_o = Y_o^\pi$ for all $o \in \mathcal{O}$. The cost of the plan is $cost(\pi) = \sum_{o\in\mathcal{O}} cost(o) \cdot Y_o^\pi$, and hence the optimal plan cost is an upper bound for the objective value of the IP/LP. This allows us to define admissible heuristics:

Definition 3 *Let Π be a planning task, and let $\mathcal{C}$ be a function that maps states s of Π to constraint sets for s.*

The IP heuristic $h^{\text{IP}}_\mathcal{C}(s)$ *is the objective value of the integer program* $\text{IP}_{\mathcal{C}(s)}$. *The* LP heuristic $h^{\text{LP}}_\mathcal{C}(s)$ *is the objective value of the linear program* $\text{LP}_{\mathcal{C}(s)}$. *Infeasible IPs/LPs are treated as having an objective value of ∞.*

Note that the requirement that an operator-counting constraint must have a feasible solution with $\mathsf{Y}_o = Y_o^\pi$ for *every* plan π is stricter than necessary for admissibility. It is sufficient that whenever a solution exists, there is one optimal plan π^* such that *all* operator-counting constraints have a feasible solution with $\mathsf{Y}_o = Y_o^{\pi^*}$.

If all operator costs of a planning task are integer, we can obviously improve the LP heuristic estimate without losing admissibility by rounding up to the nearest integer.

4 Types of Operator-counting Constraints

In this section we describe four types of operator-counting constraints that capture different state-of-the-art heuristics.

Landmark Constraints

A *(disjunctive) action landmark* [Zhu and Givan, 2003; Helmert and Domshlak, 2009] for a state s is a set of operators of which at least one must be part of any s-plan.

Using linear programming to derive heuristic estimates from landmarks was introduced by Karpas and Domshlak [2009] for computing optimal cost partitionings for landmarks. The LP formulation was improved by Keyder *et al.* [2010]. Bonet and Helmert [2010] introduced an alternative formulation that directly fits into our framework and showed that it is the dual of the representation by Keyder et al.

Strengthening heuristics with landmarks is not a new idea: Domshlak *et al.* [2012] propose it for abstraction heuristics and Bonet [2013] for the LP-based state-equation heuristic. He uses the same constraints as Bonet and Helmert [2010]:

Definition 4 *Let $L \subseteq \mathcal{O}$ be an action landmark for state s of task Π. The* landmark constraint $c^{\text{lm}}_{s,L}$ *for L is $\sum_{o\in L} \mathsf{Y}_o \geq 1$.*

Since at least one action of the landmark occurs in every s-plan, landmark constraints are operator-counting constraints.

Post-Hoc Optimization Constraints

Post-hoc optimization heuristics [Pommerening *et al.*, 2013] exploit knowledge on certain operators not being able to contribute to a heuristic estimate:

Definition 5 *Let Π be a planning task with operator set $\mathcal{O}$, let h be an admissible heuristic for Π, and let $N \subseteq \mathcal{O}$ be a set of operators that are* noncontributing *in the sense that the h-values are still admissible in a modified planning task where the cost of all operators in N is set to 0. The* post-hoc optimization constraint $c^{\text{PH}}_{s,h,N}$ *for h, N and state s of Π consists of the single inequality $\sum_{o\in\mathcal{O}\setminus N} cost(o)\mathsf{Y}_o \geq h(s)$.*

An important special case applies to pattern database (PDB) heuristics for planning [Edelkamp, 2001], which are based on projections of a planning task Π to a subset (called a *pattern*) of the state variables. Operators that do not modify any variable in the pattern are noncontributing. We denote with $c^{\text{pdb}}_{s,P}$ the post-hoc optimization constraint for state s and the PDB heuristic h^P for pattern P.

Net Change Constraints

Bonet [2013] introduces the state-equation heuristic h^{SEQ} by relating planning tasks to Petri nets and deriving constraints based on the net change of the number of tokens in the Petri net locations caused by the firing of transitions. Here, we present the general ideas behind h^{SEQ} but working on the planning task directly, leading us to a deeper understanding.

We say that an operator o applied in state s *produces* an atom $V = v$ if $s[V] \neq v$ and $res(o,s)[V] = v$ and that it *consumes* the atom if $s[V] = v$ and $res(o,s)[V] \neq v$. The net change of the atom from state s to the successor state $res(o,s)$ is 1 if o produces the atom, -1 if it consumes it and 0 otherwise. This idea can be lifted to valid sequence of operators (plans). If π is an operator sequence that is applicable at state s, the net change that π induces over an atom is either 1, -1 or 0 whether π produces, consumes or leaves the atom unchanged. A necessary condition for π to be an s-plan is that π should produce every goal atom that does not hold at s and leave unchanged every goal atom that holds at s.

We want to use this necessary condition to derive operator-counting constraints. Informally, the accumulated net change for an atom $V = v$ of an operator sequence π adds up all operator applications that produce the atom and subtracts those that consume the atom. Without knowing the exact sequence π, one cannot predict the states are visited by the path π from s to the goal, and thus decide when and what atoms are produced or consumed. However, we can still obtain useful bounds on the induced net change for *candidate plans* on a given atom $V = v$. For this, let us consider four disjoint classes of operators for the atom $V = v$: the class $AP_{V=v}$ of operators that *always produce* the atom, the class $SP_{V=v}$ of operators that *sometimes produce* the atom, the class $AC_{V=v}$ of operators that *always consume* the atom, and the class $SC_{V=v}$ of operators that *sometimes consume* the atom. Operators that do not fall into one of these classes never change the truth value of the atom. For example, if we are told that the plan π for state s contains exactly two operators in $AP_{V=v}$, one operator in $AC_{V=v}$, and no operator in the other classes for the atom $V = v$, then we can be sure that the atom $V = v$ will hold after executing π at s.

The net change on an atom $V = v$ that a sequence π induces between a state s and the resulting state $res(\pi, s)$ can only take a limited number of values if π is an s-plan:

$$pnc^{s\to\star}_{V=v} = \begin{cases} \{0,1\} & \text{if } s_G[V] \text{ is undefined and } s[V] \neq v \\ \{-1,0\} & \text{if } s_G[V] \text{ is undefined and } s[V] = v \\ \{1\} & \text{if } s_G[V] = v \text{ and } s[V] \neq v \\ \{-1\} & \text{if } s_G[V] = v' \text{ and } s[V] = v \neq v' \\ \{0\} & \text{otherwise.} \end{cases}$$

These values and operator classes are combined as follows:

Definition 6 *Let Π be a planning task with operator set $\mathcal{O}$. The* lower-bound net change constraint $c^{\text{ncl}}_{s,V=v}$ *for atom $V = v$ and state s is the constraint*

$$\sum_{o \in AP_{V=v}} \mathsf{Y}_o \;+\; \sum_{o \in SP_{V=v}} \mathsf{Y}_o \;-\; \sum_{o \in AC_{V=v}} \mathsf{Y}_o \;\geq\; \min pnc^{s \to \star}_{V=v}\,;$$

the upper-bound net change constraint $c^{\text{ncu}}_{s,V=v}$ *is*

$$\max pnc^{s \to \star}_{V=v} \;\geq\; \sum_{o \in AP_{V=v}} \mathsf{Y}_o \;-\; \sum_{o \in AC_{V=v}} \mathsf{Y}_o \;-\; \sum_{o \in SC_{V=v}} \mathsf{Y}_o\,.$$

As before, it is not very difficult to show that the lower- and upper-bound net change constraints are operator-counting constraints. That is, if π is an s-plan for state s, the assignment $\mathsf{Y}_o = Y^\pi_o$ for $o \in \mathcal{O}$ satisfies all such constraints.

Cost Partitioning Constraints for Abstractions

An *abstraction heuristic* maps each state s of a planning task Π through a homomorphic mapping α to an *abstract state* $\alpha(s)$. The heuristic estimate for s is the cost of the cheapest path from $\alpha(s)$ to an abstract goal state in the transition system $\mathcal{T}^\alpha$ induced by α on the planning task Π.

In general, the estimates of provided by a set $\mathcal{A}$ of abstraction heuristics can only be combined admissibly by taking their maximum. *Action cost partitionings* permit the estimates to be added up while retaining admissibility by distributing the costs of the actions across all the abstractions in $\mathcal{A}$; i.e., if $cost_\alpha(o) \geq 0$ is the cost assigned to operator o in the abstraction α, an action cost partitioning sets the action costs such that $\sum_{\alpha \in \mathcal{A}} cost_\alpha(o) \leq cost(o)$ for each $o \in \mathcal{O}$.

An *optimal cost partitioning* [Katz and Domshlak, 2010] computes the best estimate obtainable by any action cost partitioning for a given set $\mathcal{A}$ of abstractions. Unfortunately, the best partitioning is state-dependent and its computation at each state is often too expensive [Pommerening *et al.*, 2013].

For a set $\mathcal{A}$ of abstraction mappings for task Π, let $\mathcal{T}^\alpha = \langle S^\alpha, s^\alpha_I, G^\alpha, T^\alpha \rangle$ denote the transition system induced by $\alpha \in \mathcal{A}$, where S^α is the set of abstract states, s^α_I is the abstract initial state and G^α is the set of abstract goal states. Each transition $\langle s, o, s' \rangle \in T^\alpha$ from state s to state s' is labeled with the operator o that induces it. The subset $SCT^\alpha \subseteq T^\alpha$ contains all *state-changing* transitions $\langle s, o, s' \rangle$ with $s \neq s'$.

The estimate of the optimal cost partitioning heuristic $h^{\text{OCP}}_{\mathcal{A}}$ for $\mathcal{A}$ in state $s \in S$ is the objective value of the following LP or ∞ if the LP is not bounded feasible:

Maximize $\sum_{\alpha \in \mathcal{A}} \mathsf{H}^\alpha$ subject to

$$\begin{aligned}
\mathsf{D}^\alpha_{s'} &= 0 && \text{for all } \alpha \in \mathcal{A} \text{ and } s' = \alpha(s) \\
\mathsf{D}^\alpha_{s''} &\leq \mathsf{D}^\alpha_{s'} + \mathsf{C}^\alpha_o && \text{for all } \alpha \in \mathcal{A} \text{ and } \langle s', o, s'' \rangle \in SCT^\alpha \\
\mathsf{H}^\alpha &\leq \mathsf{D}^\alpha_{s'} && \text{for all } \alpha \in \mathcal{A} \text{ and } s' \in G^\alpha \\
\textstyle\sum_{\alpha \in \mathcal{A}} \mathsf{C}^\alpha_o &\leq cost(o) && \text{for all } o \in \mathcal{O}
\end{aligned}$$

with all variables restricted to be non-negative.

Intuitively, variable C^α_o describes the cost of operator o in $\mathcal{T}^\alpha$. Given this cost partitioning, variable $\mathsf{D}^\alpha_{s'}$ measures the cheapest cost to reach s' from $\alpha(s)$ in the abstract system and H^α is the (additive) heuristic estimate for abstraction α.

At first glance, the cost partitioning LP seems unrelated to the operator-counting constraint framework. However, it turns out that the *dual* of the LP (which must have the same objective value) is a direct fit for our framework, offering a new perspective on optimal cost partitioning. This dual LP is to Minimize $\sum_{o \in \mathcal{O}} cost(o)\mathsf{Y}_o$ subject to the constraints $c^{\text{OCP}}_{s,\alpha}$ (defined next) for all abstraction mappings $\alpha \in \mathcal{A}$.

Definition 7 *Let α be an abstraction mapping for task Π and let $\mathcal{T}^\alpha = \langle S^\alpha, s^\alpha_I, G^\alpha, T^\alpha \rangle$ be the induced abstract transition system with $SCT^\alpha \subseteq T^\alpha$ being the set of state-changing transitions. For a state s of Π, the* optimal cost partitioning constraint $c^{\text{OCP}}_{s,\alpha}$ *is defined in terms of the following variables: operator-counting variables Y_o for $o \in \mathcal{O}$, goal-achievement indicator variables G^α_s for $s \in G^\alpha$, and transition-counting variables T^α_t for $t \in SCT^\alpha$. The constraint $c^{\text{OCP}}_{s,\alpha}$ consists of:*

1. *a* transition count inequality *for each operator $o \in \mathcal{O}$:*

$$\mathsf{Y}_o \;\geq\; \sum_{\substack{t \in SCT^\alpha \\ t \text{ labeled with } o}} \mathsf{T}^\alpha_t\,,$$

2. *a* goal inequality $\sum_{s' \in G^\alpha} \mathsf{G}^\alpha_{s'} \geq 1$*, and*
3. *a* flow inequality *for all abstract states $s' \neq \alpha(s)$ in S^α:*

$$\sum_{\substack{t \in SCT^\alpha \\ t \text{ ends in } s'}} \mathsf{T}^\alpha_t \;-\; \sum_{\substack{t \in SCT^\alpha \\ t \text{ starts in } s'}} \mathsf{T}^\alpha_t \;\geq\; \begin{cases} \mathsf{G}^\alpha_{s'} & \text{if } s' \in G^\alpha \\ 0 & \text{if } s' \notin G^\alpha \end{cases}.$$

Given an s-plan π for state s, we can find an assignment to the variables in $c^{\text{OCP}}_{s,\alpha}$ that satisfies the constraint. Namely, we set Y_o to Y^π_o, $\mathsf{G}^\alpha_{s'}$ to 1 or 0 depending on whether executing π in $\mathcal{T}^\alpha$ ends in s' or not, and T^α_t to the number of times that the transition t is used when executing π in $\mathcal{T}^\alpha$. Thus, the $c^{\text{OCP}}_{s,\alpha}$ constraints are operator-counting constraints.

A closer look at the proof of this claim shows that there is a solution of the LP that uses only integer values. This means that the constraints remain sound when interpreting $c^{\text{OCP}}_{s,\alpha}$ as an *integer program*. This results in a heuristic that *dominates* Katz and Domshlak's optimal cost partitioning, yet the new heuristic is not known to be efficiently computable.

Delete Relaxation Constraints

The delete-relaxation heuristic h^+ is best outlined in planning tasks that are expressed with a propositional language like STRIPS [Fikes and Nilsson, 1971]. The estimate $h^+(s)$ for state s in a such task Π refers to the cost of an optimal s-plan in the task Π^+ that results from removing the "delete effects" [Bonet and Geffner, 2001].

Imai and Fukunaga [2014] present an IP that exactly computes the estimate $h^+(s)$ for a given state s using a formulation that associates feasible solutions with s-plans in Π^+, and every such plan with a feasible solution that encodes it.

Imai and Fukunaga use an encoding that uses indicator variables $\mathsf{U}_o \in \{0, 1\}$ for each $o \in \mathcal{O}$, and with an objective that minimizes $\sum_{o \in \mathcal{O}} cost(o)\mathsf{U}_o$. We can make the set of constraints into an operator-counting constraint by adding the operator-counting variables Y_o with constraints $\mathsf{Y}_o \geq \mathsf{U}_o$, for every $o \in \mathcal{O}$, and by changing the objective function to minimize $\sum_{o \in \mathcal{O}} cost(o)\mathsf{Y}_o$ [Röger and Pommerening, 2015]. The resulting constraint for state s is denoted by c^{dr}_s.

5 Relationship to Existing Heuristics

We now sketch how several existing heuristics can be expressed within the operator-counting constraint framework.

Landmark Heuristic with Optimal Cost Partitioning
Optimal cost partitioning for landmarks [Karpas and Domshlak, 2009] can be expressed in our framework.

Theorem 1 *For state s, let $\mathcal{C}(s)$ be a set of landmark constraints for s. Then $h^{\mathrm{LP}}_{\mathcal{C}}(s) = h^{\mathrm{LM}}_{\mathrm{opt}}(s)$, where $h^{\mathrm{LM}}_{\mathrm{opt}}$ is the landmark heuristic with optimal cost partitioning using the same landmarks as in $\mathcal{C}$.*

The LM-cut heuristic [Helmert and Domshlak, 2009] computes a cost partitioning for the set of action landmarks that it finds. Bonet [2013] proposed to use these landmarks for specifying constraints in the sense of this paper. Thus, any LP heuristic using these constraints dominates LM-cut.

Post-Hoc Optimization Heuristic
The definition of the post-hoc optimization heuristic $h^{\mathrm{PhO}}_{\mathcal{H},\mathcal{N}}$ involves a variable merging step which speeds up its computation but has no influence on the heuristic estimate. If we omit this step, it is defined by an LP which minimizes the objective function $\sum_{o\in\mathcal{O}} \mathsf{C}_o$, where C_o represents the total cost incurred by operator o in a plan. Replacing each occurrence of C_o by $cost(o)\mathsf{Y}_o$ brings the LP into the required form.

Theorem 2 *Let $h^{\mathrm{PhO}}_{\mathcal{H},\mathcal{N}}$ be the post-hoc optimization heuristic for a set of heuristics $\mathcal{H}$ where $\mathcal{N}(h)$ denotes the noncontributing operators of $h \in \mathcal{H}$. For state s, let $\mathcal{C}_{\mathcal{H},\mathcal{N}}(s) = \{c^{\mathrm{PH}}_{s,h,\mathcal{N}(h)} \mid h \in \mathcal{H}\}$. Then $h^{\mathrm{PhO}}_{\mathcal{H},\mathcal{N}} = h^{\mathrm{LP}}_{\mathcal{C}_{\mathcal{H},\mathcal{N}}}$.*

State-Equation Heuristic
The state-equation heuristic fits directly out framework.

Theorem 3 *For state s, let $\mathcal{C}(s)$ denote the set of lower-bound net change constraints for s and all atoms. Then the state-equation heuristic h^{SEQ} equals the LP heuristic $h^{\mathrm{LP}}_{\mathcal{C}}$.*

Optimal Cost Partitioning for Abstractions
We derived the optimal cost partitioning constraints from the dual of the LP which defines the optimal cost partitioning heuristic. As a bounded feasible LP and its dual have the same optimal value, the heuristic fits our framework.

Theorem 4 *Let $\mathcal{A}$ be a set of abstractions. For state s, let $\mathcal{C}_{\mathcal{A}}(s)$ be the constraints $\{c^{\mathrm{OCP}}_{s,\alpha} \mid \alpha \in \mathcal{A}\}$. Then $h^{\mathrm{OCP}}_{\mathcal{A}} = h^{\mathrm{LP}}_{\mathcal{C}_{\mathcal{A}}}$.*

Delete-Relaxation Heuristic
As shown by Imai and Fukunaga [2014] (cf. Röger and Pommerening [2015]) the objective of the integer program $\mathrm{IP}_{\mathcal{C}^{\mathrm{dr}}_s}$ equals the delete-relaxation estimate $h^+(s)$ for state s. The LP relaxation $\mathrm{LP}_{\mathcal{C}^{\mathrm{dr}}_s}$ provides a lower bound to h^+ that is accurate in several cases [Imai and Fukunaga, 2014].

Critical Path Heuristics
Among the four classes of heuristics identified by Helmert and Domshlak [2009], the class of critical path heuristics is the only that we do not know how to capture. This class only contains the heuristics h^m [Haslum and Geffner, 2000] that are in some sense different from the others and that exploit a type of information that the other heuristics do not consider. It is an open and important question whether such heuristics can be efficiently captured with operator-counting constraints.

	SEQ	PhO-Sys^1	PhO-Sys^2	LMC	OPT-Sys^1	LMC+ PhO-Sys^2	LMC+ SEQ	PhO-Sys^2+ SEQ	LMC+ PhO-Sys^2+ SEQ	$h^{\text{LM-cut}}$
barman (20)	4	4	4	4	4	4	4	4	4	4
elevators (20)	7	9	16	16	4	17	16	15	16	**18**
floortile (20)	4	2	2	6	2	6	6	4	6	**7**
nomystery (20)	10	11	**16**	14	8	**16**	12	14	14	14
openstacks (20)	11	**14**	**14**	**14**	5	**14**	11	11	11	**14**
parcprinter (20)	**20**	11	13	13	7	14	**20**	**20**	**20**	13
parking (20)	3	**5**	1	2	1	1	2	1	1	3
pegsol (20)	**18**	17	17	17	10	17	**18**	17	16	17
scanalyzer (20)	11	9	4	11	7	10	10	10	8	**12**
sokoban (20)	16	19	**20**	**20**	13	**20**	**20**	**20**	19	**20**
tidybot (20)	7	13	**14**	**14**	4	**14**	10	8	10	**14**
transport (20)	**6**	**6**	**6**	**6**	4	**6**	**6**	5	**6**	**6**
visitall (20)	17	16	16	10	15	17	**19**	17	18	11
woodworking (20)	9	5	10	11	2	13	**16**	10	**16**	12
Sum IPC 2011 (280)	143	141	153	158	86	169	**170**	156	165	165
IPC 1998–2008 (1116)	487	446	478	586	357	589	**618**	516	598	598
Sum (1396)	630	587	631	744	443	758	**788**	672	763	763

Table 1: Coverage on the IPC benchmark suite. Best results are highlighted in bold.

6 Experimental Evaluation

We implemented a general framework for LP heuristics in the Fast Downward planning system [Helmert, 2006], supporting net change, landmark, PDB and optimal cost partitioning constraints. The underlying LP solver is CPLEX v12.5.1. For our evaluation, we use all tasks for optimal planning from the IPC benchmark suite. All experiments were conducted on Intel Xeon E5-2660 processors (2.2 GHz) with a time limit of 30 minutes and a memory limit of 2 GB for each planner run.

For the different types of operator-counting constraints, we consider the following five groups of constraints:

SEQ All lower-bound net change constraints, corresponding to the state-equation heuristic h^{SEQ}.

PhO-Sys^1 All post-hoc optimization constraints for projections on goal variables.

PhO-Sys^2 All post-hoc optimization constraints for systematically generated projections on up to two variables.

LMC All landmark constraints for the landmarks found by the LM-cut heuristic.

OPT-Sys^1 All optimal cost partitioning constraints for projections on goal variables.

Individual Constraint Groups
To get an idea of the quality of the different constraint groups, we ran experiments with A*, using one of the above configurations at a time. The resulting coverage is reported in the first block of Table 1.

Optimal cost partitioning on LM-cut landmarks leads to the highest coverage with a clear lead over the state-equation heuristic and the post-hoc optimization heuristic for Sys^2 patterns, which are almost on par. Using only Sys^1 patterns, the post-hoc optimization heuristic solves 44 fewer tasks and the optimal cost partitioning lags far behind with only 443 solved tasks, compared to 744 by the best LP configuration.

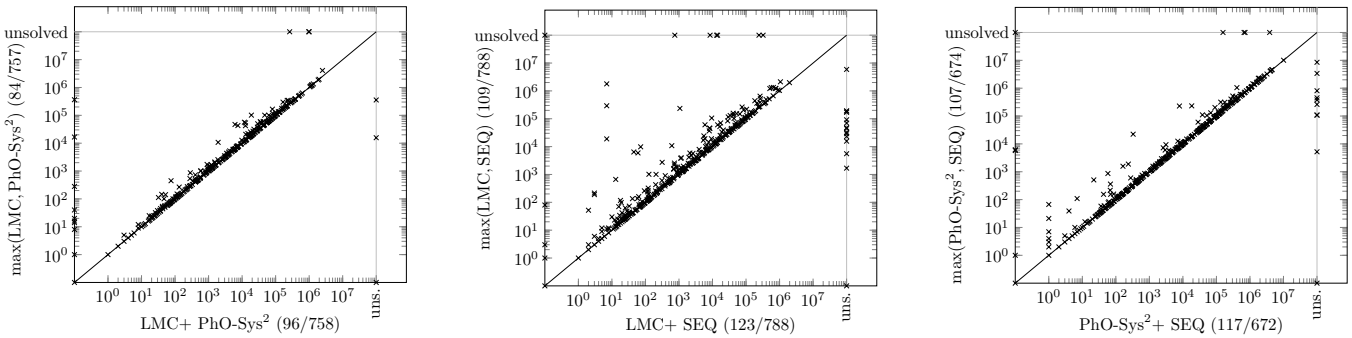

Figure 1: Number of expansions (excluding nodes on the final f-layer). The numbers (x/y) behind the configurations express that among the y solved tasks, x have been solved with perfect heuristic estimates.

Comparing the coverage of the standard LM-cut heuristic and of the LMC configuration reveals that the additional effort of computing the *optimal* cost partitioning for the same landmarks does not pay off in terms of providing sufficiently better guidance. A possible reason for this is that the LM-cut heuristic already approximates h^+ very closely, and the corresponding LP heuristic is also bounded by h^+.

For SEQ, we proved that that the upper bound net change constraints are implied by the lower bound net change constraints, so their addition to SEQ cannot increase the value of the resulting heuristic, and observed that the upper bound net change constraints are strictly weaker than the upper bound net change constraints [Pommerening *et al.*, 2014].

Combinations of Constraint Groups

We showed that SEQ dominates OPT-Sys^1 in terms of heuristic guidance [Pommerening *et al.*, 2014]. Since OPT-Sys^1 also requires much more time to compute the heuristic, we do not consider it in the combinations. Likewise, we also omit PhO-Sys^1 as SEQ and PhO-Sys^2 give better coverage. This leaves us with all combinations of SEQ, PhO-Sys^2, and LMC. The coverage results appear in Table 1.

A combination of SEQ and PhO-Sys^2 looks promising because they have their strengths and weaknesses in different domains. For example, using PhO-Sys^2 solves 14 tasks in the tidybot domain, while only 7 can be solved with SEQ. In parcprinter the picture is reversed: using SEQ, we solve 20 tasks in contrast to only 13 with PhO-Sys^2. Indeed, the combination solves 672 instances, a clear improvement on each individual heuristic solving 630 and 631 tasks, respectively.

The combination of PhO-Sys^2 and LMC also pays off, solving 758 task instead of 631 and 744, respectively.

The best combination is SEQ and LMC: with 788 tasks, it solves 44 more tasks than its best component LMC and 25 more tasks than the standard LM-cut heuristic which is the state of the art for optimal planning.

However, adding more constraints does not always have a positive effect because the extra time needed to compute the resulting LP may be in excess of the resulting extra guidance.

Constraint Interactions

Can we explain the better performance of the combinations with the better guidance of more individual components, or is there an additional positive effect through interactions of the different constraints in the LP? The plots in Figure 1 show the number of expansions using one LP heuristic with two constraint groups against the expansions using the maximum of the two individual LP heuristics.

In all three cases, we see clear synergy effects: combining two sets of constraints in a single LP indeed leads to stronger heuristic estimates than maximizing the estimates from two separate LPs. These synergy effects are more pronounced in the combinations of SEQ and PhO-Sys^2 and of SEQ and LMC than in the combination of PhO-Sys^2 and LMC.

Considering coverage, however, the picture is somewhat more mixed: some tasks can only be solved by the approaches using a single large LP, others only by the maximum over two LP heuristics, and both approaches end up too close to tell apart in terms of overall coverage.

7 Discussion

We introduced a class of IP/LP heuristics based on operator-counting constraints that subsumes many existing heuristics, including the state-equation, post-hoc optimization and admissible landmark heuristic as well as optimal cost partitioning of (explicit-state) abstraction heuristics. Our new LP for optimal cost partitioning of abstraction heuristics is based on a dualization of the originally suggested LP and suggests new ways to combine abstraction heuristics with other sources of knowledge, such as landmarks.

The framework is also effective for the theoretical analysis of heuristics. Indeed, using the framework and formulations presented here, we were able to show [Pommerening *et al.*, 2014] that the state-equation heuristic dominates the optimal cost partitioning on single-variable abstractions and that the safety-based extension of the state-equation heuristic cannot improve heuristic accuracy.

More recently, Pommerening *et al.* [2015] show that solving the net change constraints alone is equivalent to computing an optimal *general* cost partitioning for the abstraction in Sys^1 (all the projections of variables). This answers an open question on how the state-equation heuristic fits into the general classification of heuristics.

For the future, we look forward for using the framework to obtain new theoretical insight of existing heuristics, and to define new constraints leading to more powerful heuristics.

Acknowledgments

This work was supported by the Swiss National Science Foundation (SNSF) as part of the project "Abstraction Heuristics for Planning and Combinatorial Search" (AHPACS).

References

[Bonet and Geffner, 2001] Blai Bonet and Héctor Geffner. Planning as heuristic search. *AIJ*, 129(1):5–33, 2001.

[Bonet and Helmert, 2010] Blai Bonet and Malte Helmert. Strengthening landmark heuristics via hitting sets. In *Proc. ECAI 2010*, pages 329–334, 2010.

[Bonet and van den Briel, 2014] Blai Bonet and Menkes van den Briel. Flow-based heuristics for optimal planning: Landmarks and merges. In *Proc. ICAPS 2014*, pages 47–55, 2014.

[Bonet, 2013] Blai Bonet. An admissible heuristic for SAS^+ planning obtained from the state equation. In *Proc. IJCAI 2013*, pages 2268–2274, 2013.

[Domshlak *et al.*, 2012] Carmel Domshlak, Michael Katz, and Sagi Lefler. Landmark-enhanced abstraction heuristics. *AIJ*, 189:48–68, 2012.

[Edelkamp, 2001] Stefan Edelkamp. Planning with pattern databases. In *Proc. ECP 2001*, pages 13–24, 2001.

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

[Geffner and Bonet, 2012] Héctor Geffner and Blai Bonet. *A Concise Introduction to Models and Methods for Automated Planning*. Morgan & Claypool Publishers, 2012.

[Haslum and Geffner, 2000] Patrik Haslum and Héctor Geffner. Admissible heuristics for optimal planning. In *Proc. AIPS 2000*, pages 140–149, 2000.

[Helmert and Domshlak, 2009] Malte Helmert and Carmel Domshlak. Landmarks, critical paths and abstractions: What's the difference anyway? In *Proc. ICAPS 2009*, pages 162–169, 2009.

[Helmert *et al.*, 2014] Malte Helmert, Patrik Haslum, Jörg Hoffmann, and Raz Nissim. Merge-and-shrink abstraction: A method for generating lower bounds in factored state spaces. *JACM*, 61(3):16: 1–63, 2014.

[Helmert, 2006] Malte Helmert. The Fast Downward planning system. *JAIR*, 26:191–246, 2006.

[Imai and Fukunaga, 2014] Tatsuya Imai and Alex Fukunaga. A practical, integer-linear programming model for the delete-relaxation in cost-optimal planning. In *Proc. ECAI 2014*, pages 459–464, 2014.

[Karpas and Domshlak, 2009] Erez Karpas and Carmel Domshlak. Cost-optimal planning with landmarks. In *Proc. IJCAI 2009*, pages 1728–1733, 2009.

[Katz and Domshlak, 2009] Michael Katz and Carmel Domshlak. Structural-pattern databases. In *Proc. ICAPS 2009*, pages 186–193, 2009.

[Katz and Domshlak, 2010] Michael Katz and Carmel Domshlak. Optimal admissible composition of abstraction heuristics. *AIJ*, 174(12–13):767–798, 2010.

[Keyder *et al.*, 2010] Emil Keyder, Silvia Richter, and Malte Helmert. Sound and complete landmarks for and/or graphs. In *Proc. ECAI 2010*, pages 335–340, 2010.

[Pommerening *et al.*, 2013] Florian Pommerening, Gabriele Röger, and Malte Helmert. Getting the most out of pattern databases for classical planning. In *Proc. IJCAI 2013*, pages 2357–2364, 2013.

[Pommerening *et al.*, 2014] Florian Pommerening, Gabriele Röger, Malte Helmert, and Blai Bonet. LP-based heuristics for cost-optimal planning. In *Proc. ICAPS 2014*, pages 226–234, 2014.

[Pommerening *et al.*, 2015] Florian Pommerening, Gabriele Röger, Malte Helmert, and Jendrik Seipp. From non-negative to general operator cost partitioning. In *Proc. AAAI 2015*, pages 3335–3341, 2015.

[Röger and Pommerening, 2015] Gabriele Röger and Florian Pommerening. Linear programming for heuristics in optimal planning. In *Proceedings of AAAI-15 Workshop on Planning, Search, and Optimization (PlanSOpt-15)*, 2015.

[Russell and Norvig, 2003] Stuart Russell and Peter Norvig. *Artificial Intelligence — A Modern Approach*. Prentice Hall, 2003.

[van den Briel *et al.*, 2007] Menkes van den Briel, J. Benton, Subbarao Kambhampati, and Thomas Vossen. An LP-based heuristic for optimal planning. In *Proc. CP 2007*, pages 651–665, 2007.

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