Proceedings of the Twenty-Seventh International Conference on Automated Planning and Scheduling (ICAPS 2017)

Abstraction Heuristics, Cost Partitioning and Network Flows

Florian Pommerening, Malte Helmert
University of Basel,
Basel, Switzerland
{florian.pommerening, malte.helmert}@unibas.ch

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

Abstract

Cost partitioning is a well-known technique to make admissible heuristics for classical planning additive. The optimal cost partitioning of explicit-state abstraction heuristics can be computed in polynomial time with a linear program, but the size of the model is often prohibitive. We study this model from a dual perspective and develop several simplification rules to reduce its size. We use these rules to answer open questions about extensions of the state equation heuristic and their relation to cost partitioning.

Introduction

Abstraction is one of the most common techniques to come up with admissible heuristics for classical planning. Often, several smaller abstractions of a task are used instead a large one. Even if all abstraction heuristics are admissible, their sum might not be, so combining heuristic values in an admissible way is an important problem. Maximization is always admissible, but usually a better combination can be found by *cost partitioning* (Katz and Domshlak 2007; Yang et al. 2008; Katz and Domshlak 2010), where operator costs are distributed among the abstractions to make the sum of heuristic values admissible.

Katz and Domshlak (2010) show how to compute the optimal cost partitioning for a set of explicit-state abstractions in polynomial time as the objective value of a linear program. However, this technique is often not used in practice, as the LP model can get prohibitively large. Pommerening et al. (2015) recently showed that there is a dual view on cost partitioning as operator counting heuristics. For example, the state equation heuristic (van den Briel et al. 2007; Bonet 2013) can be described in this framework, and has a close relationship to cost partitioning over projections to individual variables (atomic projections). In contrast to the direct implementation of cost partitioning over these abstractions, the model of the state equation heuristic is small enough that it can be solved efficiently in practice, making it a state-of-the-art heuristic.

We investigate cost partitioning over abstractions from the dual perspective as synchronized network flows. Taking lessons from the state equation heuristic, we derive general rules that can be used to simplify the LP models involved. Applied to atomic projections, our rules end up with the constraints for the state equation heuristic, explaining its good performance compared to an equivalent heuristic implemented as cost partitioning over abstractions. The rules therefore provide a way to apply the lessons learned from the state equation heuristic to the cost partitioning of general abstractions.

Bonet and van den Briel (2014) extend the state equation heuristic with (partial) variable merges and mutexes. So far, the relationship of these extensions to cost-partitioned abstractions was unclear and the definition of partial merges was restricted to the binary case. Pommerening et al. (2015) conjectured a relation to cost partitioning over constrained projections. The constraints for the state equation heuristic turn out to be weaker. We show that the model for these extensions can be strengthened without increasing its size to bridge this gap. The connection to general abstractions also gives a clear definition of merges beyond the binary case.

Our rules might increase the heuristic quality of the state equation heuristic and its extensions, and more importantly, they help us to better understand the heuristics involved.

Background

We consider SAS$^+$ planning tasks $\Pi = \langle \mathcal{V}, \mathcal{O}, s_\mathrm{I}, s_\star, \mathit{cost} \rangle$ in transition normal form (TNF) (Bäckström and Nebel 1995; Pommerening and Helmert 2015) with variables $\mathcal{V}$, operators $\mathcal{O}$, initial state s_I, and goal state $s_\star$. We refer to the literature (e.g., Pommerening and Helmert 2015) for details on the semantics of planning tasks and their plans, and only emphasize that in TNF there is a unique goal state, and an operator mentions the same variables in its precondition $\mathit{pre}(o)$ and its effect $\mathit{eff}(o)$. Considering only TNF tasks does not limit generality as every SAS$^+$ task can be transformed into TNF in linear time with small overhead. (Extending our results to unlimited SAS$^+$ is possible but would severely reduce the readability of the results for little or no practical benefit, as conversion to TNF is an established implementation technique for the planning approaches considered in this paper.)

A TNF task Π induces a labeled, weighted transition system $\mathrm{TS}_\Pi = \langle \mathcal{S}, \mathcal{T}, s_\mathrm{I}, \mathrm{S}_G \rangle$, with states $\mathcal{S}$, transitions $\mathcal{T}$, initial state s_I, and goal states $\mathrm{S}_G = \{s_\star\}$. A transition

$s \xrightarrow{o} s' \in \mathcal{T}$ is labeled with o and weighted with $cost(o)$. A shortest path in TS_Π according to the transition weights corresponds to an optimal plan for Π. The cost of an optimal plan is h^*. An *admissible heuristic estimate* is a value $h \leq h^*$. We consider two types of admissible heuristic estimates: *abstractions* and the *state equation heuristic*.

Abstraction Heuristics

An abstraction of a planning task Π (e.g., Edelkamp 2001; Haslum, Bonet, and Geffner 2005; Helmert et al. 2014) maps the transition system of a planning task to an *abstract transition system* $\mathrm{TS}^\alpha = \langle \mathcal{S}^\alpha, \mathcal{T}^\alpha, s_\mathrm{I}^\alpha, \mathrm{S}_G^\alpha \rangle$. The *abstraction function* α maps states in $\mathcal{S}$ to abstract states in $\mathcal{S}^\alpha$ such that every path in the original transition system of Π corresponds to a path in TS^α with the same weights. The cost h^α of a shortest path in TS^α is a lower bound for the optimal plan cost of Π. (If there is no path from s_I^α to $s_\star^\alpha$ in TS^α, h^α is defined to be ∞.) An important special case of abstractions are *projections* to a subset of variables $P \subseteq \mathcal{V}$ where $\alpha(s)$ is the restriction of s to the variables in P.

State Equation Heuristic

The *state equation heuristic* h^{SEQ} can be defined over flows in domain transition graphs (DTGs) (van den Briel et al. 2007; Bonet and van den Briel 2014). A DTG for a variable V in a TNF task is a directed graph with one node for every value of V and an edge $v \xrightarrow{o} v'$ for every operator o with $pre(o)[V] = v$ and $\mathit{eff}(o)[V] = v'$. The *SEQ constraint* for variable-value pair $\langle V, v \rangle$, with $V \in \mathcal{V}$ and $v \in dom(V)$, is

$$\sum_{v' \xrightarrow{o} v \in in_{\mathrm{DTG}}(v)} \mathsf{Count}_o - \sum_{v \xrightarrow{o} v' \in out_{\mathrm{DTG}}(v)} \mathsf{Count}_o = \Delta(V, v)$$

where $\Delta(V, v) = [s_\star[V] = v] - [s_\mathrm{I}[V] = v]$. The original definition of the SEQ constraint uses inequalities, but with *safety-based improvement* all of them turn into equalities in TNF tasks. The model of the state equation heuristic is defined as minimizing $\sum_{o \in \mathcal{O}} cost(o)\mathsf{Count}_o$ subject to $\mathsf{Count}_o \geq 0$ for all $o \in \mathcal{O}$ and the SEQ constraints for all variable-value pairs. Extensions to h^{SEQ} consider additional constraints, which we will describe as they become relevant.

Cost Partitioning

Multiple heuristics can be combined with cost partitioning by distributing the cost of each operator among the heuristics. For a collection $\mathcal{H}$ of functions mapping costs to admissible heuristic estimates, the sum $\sum_{h \in \mathcal{H}} h(cost_h)$ is admissible if $\sum_{h \in \mathcal{H}} cost_h(o) \leq cost(o)$ for each operator o. An optimal cost partitioning for collection $\mathcal{H}$ is a partitioning that maximizes the sum. Allowing negative operator costs in the partitioning can be beneficial (Pommerening et al. 2015). We denote the value of $\sum_{h \in \mathcal{H}} h(cost_h)$ for an optimal cost partitioning by $h^{\mathrm{OCP}}_{\mathcal{H}}$. Observe that any cost partitioning of abstraction heuristics is ∞ if there is some abstraction α for which there is no path from s_I^α to $s_\star^\alpha$ in TS_α.

Network Flows

Consider a transition system $\mathrm{TS} = \langle \mathcal{S}, \mathcal{T}, s_\mathrm{I}, \{s_\star\} \rangle$ with a single goal state and transition labels $\mathcal{O}$. For a state $s \in \mathcal{S}$ we write $in_{\mathrm{TS}}(s)$ and $out_{\mathrm{TS}}(s)$ for the set of transitions that end and start in s, and write $trans_{\mathrm{TS}}(o)$ to denote the set of transitions labeled with o. A *flow* in TS maps each transition t to a non-negative real number called the *flow along* t, such that the total flow along incoming transitions matches the total flow along outgoing transitions in each node, except at the initial and goal nodes. The cost of a flow is the summed cost of each transition multiplied by its flow. As we want to use the cost of a flow in the context of general cost partitioning, we allow the cost function to take real values.

The following LP is the standard formulation of a minimum-cost flow problem that "moves" one unit of flow from the source node s_I to the sink node $s_\star$ in TS:

$$\text{Minimize } \sum_{o \in \mathcal{O}} \sum_{t \in trans_{\mathrm{TS}}(o)} cost(o)\mathsf{Count}_t \text{ subject to}$$

$$\sum_{t \in in_{\mathrm{TS}}(s)} \mathsf{Count}_t - \sum_{t \in out_{\mathrm{TS}}(s)} \mathsf{Count}_t = \Delta(s) \qquad \forall s \in \mathcal{S} \quad (1)$$

$$\mathsf{Count}_t \geq 0 \qquad \forall t \in \mathcal{T} \quad (2)$$

where $\Delta(s) = [s = s_\star] - [s = s_\mathrm{I}]$ denotes the amount of flow "consumed" at state s; i.e., one unit produced at s_I, one unit consumed at $s_\star$, and zero units at other states. In this LP, the flow along a transition t is denoted by Count_t.

We call this LP the *flow model* for TS and the constraint (1) the *flow constraint* for s. It is very similar to the model of h^{SEQ}, but there are two important differences: firstly, flow constraints consider abstract states and transitions, whereas SEQ constraints consider variable-value pairs and DTG transitions. Secondly, and more fundamentally, the flow model has one LP variable for each abstract transition, while SEQ constraints only use one LP variable for each operator. We will analyze the differences between the two models in more detail later on.

It can be shown that the flow model is feasible iff there is a path from s_I to $s_\star$ in TS (Korte and Vygen 2001). If costs are non-negative, feasible solutions always have bounded cost, but if the LP is feasible and there is a cycle of transitions in TS with negative total cost (not necessarily connected to initial or goal state), the LP is unbounded and its value is $-\infty$ because an unbounded amount of flow may circulate through the negative cost cycle.

The *min-flow abstraction heuristic* for a task Π and abstraction α is the objective value of the flow model for the transition system TS^α of α, or ∞ if it is infeasible. We denote the value of the min-flow abstraction heuristic for α by f^α. Our first result explains the relationship of f^α to the abstraction heuristic h^α, which we will later use to relate h^{SEQ} to certain abstraction heuristics. To state this result, we define an abstract state $s \in \mathcal{S}^\alpha$ as *alive* in TS^α if there is a path from s_I^α to s and from s to a state in S_G^α. Otherwise, we call the state *dead*.

Proposition 1. *Let α be an abstraction of a planning task Π and let $cost_\alpha : \mathcal{O} \to \mathbb{R}$ be a cost function. Then, $f^\alpha \leq h^\alpha$ with equality if there are no dead states in TS^α.*

Proof: As mentioned above, $h^\alpha = \infty$ iff there is no path in TS^α from s_I^α to $s_\star^\alpha$ iff $f^\alpha = \infty$.

Assume now that $h^\alpha < \infty$. For a path in TS^α from s_I^α to $s_\star^\alpha$ with cost c, there is a flow function with the same cost.

Therefore, $f^\alpha \leq h^\alpha$ (this includes the case where a negative cost cycle occurs on an alive state and $h^\alpha = f^\alpha = -\infty$).

If h^α is finite and there are no dead states in TS^α, no negative cost cycles exist in TS^α and thus f^α is also finite. The flow along all transitions may be assumed to be integral (Korte and Vygen 2001). This flow moves one unit from s^α_I to $s^\alpha_\star$ along a single minimum cost path. Therefore, $h^\alpha = f^\alpha$. □

Operator-Counting Constraints

Pommerening et al. (2014) define the operator-counting framework for obtaining admissible heuristic estimates. Heuristics are defined as the objective value of minimizing $\sum_{o\in\mathcal{O}} \mathit{cost}(o)\mathsf{Count}_o$ subject to $\mathsf{Count}_o \geq 0$ for operators o and a collection of constraints $\mathcal{C}$. Each constraint in $\mathcal{C}$ must encode a necessary property of plans and each two constraints may only share *operator-counting variables* Count_o. The operator-counting heuristic for $\mathcal{C}$ is denoted by $h^\mathrm{LP}_\mathcal{C}$.

The min-flow abstraction heuristic can be seen as an *operator-counting heuristic*. Indeed, it is enough to modify the flow model by adding the operator-counting variables Count_o for each operator o, replacing the objective function by $\sum_{o\in\mathcal{O}} \mathit{cost}(o)\mathsf{Count}_o$, and adding constraints $\sum_{t\in \mathit{trans}_\mathrm{TS}(o)} \mathsf{Count}_t = \mathsf{Count}_o$ for each operator o. We call each such constraint the *strong linking constraint* for o and the modified LP the *OC model*.

Proposition 2. *The OC model and the flow model are equivalent: both have the same objective value and their solutions are in 1:1 correspondence.*

Pommerening et al. (2014) show that $h^\mathrm{LP}_\mathcal{C}$ is equal to the optimal cost partitioning of the collection $\{h^\mathrm{LP}_{\{C\}} : C \in \mathcal{C}\}$ of LP heuristics, one heuristic for each individual set C of constraints in $\mathcal{C}$. Proposition 1 can then be straightforwardly used to show the following proposition.

Proposition 3. *Let $\mathcal{A}$ be a set of abstractions of a planning task and let C^α be the constraints in the OC model for* TS^α *and $\alpha \in \mathcal{A}$. Then, $h^\mathrm{LP}_{\{C^\alpha:\alpha\in\mathcal{A}\}} = h^\mathrm{OCP}_{\{f^\alpha:\alpha\in\mathcal{A}\}} \leq h^\mathrm{OCP}_{\{h^\alpha:\alpha\in\mathcal{A}\}}$ with equality if, for all $\alpha \in \mathcal{A}$,* TS^α *contains no dead state.*

Together, Propositions 2 and 3 show how several min-flow abstraction heuristics can be combined in an LP that computes their optimal cost partitioning and that this can be weaker than the optimal cost partitioning of the abstraction heuristics for the same abstractions. In the following section, we will show how this gap can be closed.

To see that $h^\mathrm{OCP}_{\{f^\alpha:\alpha\in\mathcal{A}\}}$ may indeed lead to a lower value in the presence of dead states, consider a task with a single operator o and two binary variables V_1, V_2. Operator o changes V_1 from 0 to 1 and requires V_2 to be 1 without changing it. Both variables are initially 0 and the goal is to set V_1 to 1 and V_2 to 0. Let α_1 and α_2 be the projections to V_1 and V_2. The task is unsolvable but the abstract goal is reachable in both projections. In TS^{α_1}, o induces a transition from the initial state to the goal; in TS^{α_2}, the initial and goal state are the same and o induces a self-loop on an unreachable state. For an arbitrarily large M, let $\mathit{cost}^{\alpha_1}(o) = M + 1$ and $\mathit{cost}^{\alpha_2}(o) = -M$. The shortest paths under these cost functions have cost $M + 1$ and 0. Because there is a sequence of cost partitionings with arbitarily large heuristic values, $h^\mathrm{OCP}_{\{h^\alpha:\alpha\in\mathcal{A}\}} = \infty$. The minimal flows have cost $M + 1$ and $-\infty$, which is not optimal. The best cost partitioning for the flows uses costs of 1 in α_1 and 0 in α_2 for a total value of $h^\mathrm{OCP}_{\{f^\alpha:\alpha\in\mathcal{A}\}} = 1$.

Simplifying the OC Model

We now show how the OC model can be simplified and strengthened by introducing a set of transformation rules. Using them, we show how the state equation heuristic relates to an optimal cost partitioning over certain abstractions.

Dead States

Dead states cannot be part of a shortest path in any abstraction. Removing such states is an obvious step.

Rule 1. *Remove the flow constraints for all dead states and all transition-counting variables for transitions adjacent to a dead state. This may strengthen the OC model.*

In the context of operator-counting, OC models strengthened in this way behave like shortest path models. To show this, we have to consider them under general cost functions.

Proposition 4. *Let α be an abstraction of a planning task Π and let $\mathcal{C}$ be the constraints of the OC model for* TS^α *strengthened with Rule 1. Then $h^\mathrm{LP}_\mathcal{C}(\mathit{cost}) = h^\alpha(\mathit{cost})$ for any cost function $\mathit{cost} : \mathcal{O} \to \mathbb{R}$.*

Proof: Let TS' be the transition system TS^α without dead states and their adjacent transitions. Let f' and h' be the cost of a minimal flow and a shortest path in TS' under cost function *cost*. Proposition 2 shows $h^\mathrm{LP}_\mathcal{C}(\mathit{cost}) = f'$. Because TS^α and TS' have the same goal paths, $h' = h^\alpha(\mathit{cost})$. Proposition 1 establishes the missing link $f' = h'$. □

Operators Inducing a Single Transition

A common situation when considering small projections is that an operator only induces a single transition. In this case the linking constraint $\sum_{t\in \mathit{trans}_\mathrm{TS}(o)} \mathsf{Count}_t = \mathsf{Count}_o$ trivializes to $\mathsf{Count}_{t_o} = \mathsf{Count}_o$ for some transition t_o. We can reduce the size of the model by using Count_o directly:

Rule 2. *If an operator o only induces a single transition t_o in an abstraction, replace Count_{t_o} with Count_o in all constraints. Then remove the linking constraint for o and the variable Count_{t_o}. This does not affect the solutions of operator-counting variables in the OC model.*

Self-Loops

Self-looping transitions cancel out in flow constraints because they are incoming and outgoing transitions of the same state. Thus, they only occur in linking constraints. We can use different simplifications depending on how many self-loops and state-changing transitions an operator o induces.

If o induces no state-changing transitions, its transition-counting variables only occur in the linking constraint. But for every value of Count_o the constraint can always be satisfied by setting Count_t to Count_o for some transition t of o and to 0 for all others.

Rule 3. *If an operator o induces at least one self-loop and no state-changing transition, remove the linking constraint for o and all transition-counting variables for transitions labeled with o. This does not affect the solutions of operator-counting variables in the OC model.*

SEQ constraints for a variable V are based on the DTG of V without operators that do not affect V. Except for these self-loops its transition system matches that of a projection on V, so the following proposition is easy to verify.

Proposition 5. *The model of the state equation heuristic has the same constraints as the union of OC models for each atomic projection simplified using Rules 2 and 3.*

This is an alternative proof to show that the state equation heuristic computes an optimal cost partitioning over atomic projections (Pommerening et al. 2014), but this only holds when the atomic projections contain no dead states.

If o induces both self-loops *and* state-changing transitions, then the linking constraint can be simplified. We can think of the linking constraint as two inequalities $\sum_{t \in trans_{\mathrm{TS}}(o)} \mathsf{Count}_t \leq \mathsf{Count}_o$ and $\sum_{t \in trans_{\mathrm{TS}}(o)} \mathsf{Count}_t \geq \mathsf{Count}_o$, where only one of them can be unsatisfied at the same time. The latter can always be satisfied if a transition t_o only occurs in the linking constraint (i.e., if t_o is a self-loop) by setting Count_{t_o} high enough. In the other inequality, there is no need to mention the counting variables for self-loops. Any solution that assigns a positive flow to them still is a solution if their flow is reduced to 0.

Rule 4. *If an operator o induces at least one self-loop and at least one state-changing transition, replace the strong linking constraint for o with the* weak linking constraint *for o:*

$$\sum_{\substack{t \in trans_{\mathrm{TS}}(o) \\ t \text{ is no self-loop}}} \mathsf{Count}_t \leq \mathsf{Count}_o$$

and remove all transition-counting variables for self-loops labeled with o.

Bonet and van den Briel (2014) consider merging two variables X and Y into a new variable Z and introducing flow constraints for values of Z. The DTG of Z is defined as the *parallel composition* (Dräger, Finkbeiner, and Podelski 2006) of the DTGs of X and Y where states violating mutexes are removed. We want to consider mutexes separately, so for now we assume that such states are not removed, i.e., the nodes in the DTG of Z are $dom(X) \times dom(Y)$. Since both X and Y have a goal value and are safe, Z also has a goal value and is safe, so the constraints have the same bounds as in the single-variable case. The only difference is that an operator can induce more than one transition in the DTG of Z. To accurately represent this in the constraints, Bonet and van den Briel introduce an *action copy* for each transition in the DTG and add a constraint to link them to the operator counts. We write the LP variable that counts occurrences of the action copy for transition t as the transition count variable Count_t. The constraint that links them to the operator-counting variables then is the weak linking constraint and the constraints introduced for the values of Z are the flow constraints for the projection on $\{X, Y\}$.

We can see from Rules 3 and 4 that self-loops can be ignored or used to weaken the linking constraint in some cases, but the constraints of the state equation heuristic still differ from the OC model using these two rules. Operators inducing only state-changing transitions use a strong linking constraint in the flow model and a weak linking constraint in the model of the state equation heuristic. Obviously, a strong linking constraint implies the weak linking constraint. We have equivalence if we minimize $\sum_{o \in \mathcal{O}} cost(o)\mathsf{Count}_o$, all costs are non-negative, and we consider a single abstraction. However, in the context of general cost partitioning, using the strong linking constraint can make a difference for operators that cannot induce self-loops. The constraints for the merge of X and Y in h^{SEQ} can be strengthened by using strong linking constraints for such operators. (Whether an operator only induces state-changing transitions in a projection can be checked syntactically: an operator o should use a strong linking constraint iff there is an affected variable V in the projection with $pre(o)[V] \neq eff(o)[V]$.)

Proposition 6. *Let $C^{X,Y}$ be the constraints generated by h^{SEQ} when merging variables X and Y, where weak linking constraints are replaced by strong linking constraints for operators that only induce state-changing transitions in the projection to $\{X, Y\}$. Then $C^{X,Y}$ is the OC model for the projection to $\{X, Y\}$ simplified with Rules 3 and 4.*

Mutex Information

Removing nodes that violate mutexes from abstractions is a well-known technique called *constrained abstraction* (Haslum, Bonet, and Geffner 2005). States violating a mutex condition are similar to unreachable states, While they can lie on a path in the abstraction, no concrete plan visits a state that is abstracted to them. Removing them and their adjacent transitions cannot decrease the heuristic value. The example after Proposition 3 can be adapted to show that removing such states can strengthen the OC model.

Rule 5. *Remove the flow constraints for all states that violate mutex conditions and all transition-counting variables for adjacent transitions. This may strengthen the OC model.*

The result from Proposition 6 can be extended to using mutex information. After merging variables X and Y into Z, the DTG of Z with mutexes removed is equal to the constrained projection to $\{X, Y\}$ except for self-loops of operators that do not mention X and Y.

Proposition 7. *Let $C^{X,Y}_{\mathrm{mutex}}$ be defined like $C^{X,Y}$ in Proposition 6 but with the state equation heuristic's extension to mutexes. Then $C^{X,Y}_{\mathrm{mutex}}$ is the OC model for the constrained projection to $\{X, Y\}$ simplified with Rules 3, 4, and 5.*

Ignoring a Single Abstract State

In the OC model of any transition system TS, the constraint for a single state can be removed without affecting the set of solutions. In contrast to previous sections, where states are removed from the transition system including all of their adjacent transitions, here we only consider removing one of the flow constraints. This is a minor modification of the LP,

and the simplification is not likely to result in a performance boost in practice. However, on the theoretical side it allows to ignore certain parts of an abstraction, which is useful for the analysis of the final extension of h^{SEQ}, *partial merges*.

We show that the flow constraint for a state d is redundant in the presence of flow constraints for all other states. Since every transition has to start and end in some state we have $\bigcup_{s\in\mathcal{S}} in_{\text{TS}}(s) = \bigcup_{s\in\mathcal{S}} out_{\text{TS}}(s)$. Thus, the sum over the left-hand side of all flow constraints except the one for d is:

$$\sum_{s\in\mathcal{S}\setminus\{d\}} \sum_{t\in in_{\text{TS}}(s)} \mathsf{Count}_t - \sum_{s\in\mathcal{S}\setminus\{d\}} \sum_{t\in out_{\text{TS}}(s)} \mathsf{Count}_t$$
$$= -\sum_{t\in in_{\text{TS}}(d)} \mathsf{Count}_t + \sum_{t\in out_{\text{TS}}(d)} \mathsf{Count}_t$$

Since all constraints are equations, the result has to be the sum over the right-hand sides of these constraints:

$$\sum_{s\in\mathcal{S}\setminus\{d\}} [s = s_\star] - \sum_{s\in\mathcal{S}\setminus\{d\}} [s = s_{\text{I}}] = -[d = s_\star] + [d = s_{\text{I}}]$$

Multiplying both sides by -1 results in the flow constraint for d.

Rule 6. *Removing the constraint for a single state from an OC model does not influence the set of solutions.*

We can use this rule to analyze partial merges in h^{SEQ}. In a partial merge of variables X and Y, only the flow constraints for a subset of values $M \subseteq dom(X) \times dom(Y)$ are part of the model.

Proposition 8. *Let $\alpha_M^{X,Y}$ be the abstraction with abstract states $M \cup \{d\}$ that maps every state s to $z = \langle s[X], s[Y]\rangle$ if $z \in M$ and to d otherwise. Let $C_M^{X,Y}$ be defined like $C^{X,Y}$ in Proposition 6 but only considering merged values in M. Then $C_M^{X,Y}$ is the OC model for $\alpha_M^{X,Y}$ simplified with Rules 3 and 4, then using Rule 6 to remove the constraint for d.*

The proposition suggests a clean way for generalizing partial merges beyond two variables: project the task to all involved variables, then abstract the projection further by mapping all unrepresented abstract states to a new state d. The partial merge then is the OC model for the resulting abstraction simplified with Rules 3 and 4, then using Rule 6 to remove the constraint for d. Using Rules 1, 2, and 5 as well can strengthen the model and reduce its size.

Conclusion

We have shown how optimal cost partitioning over min-flow abstraction heuristics can be modeled and how this model can be simplified and strengthened. Removing dead states turns this into a way to compute optimal cost partitionings for abstraction heuristics. This leads to a better understanding of such cost partitionings in general and the state equation heuristic in particular.

It turns out that h^{SEQ} is not a cost partitioning over abstraction heuristics but one over min-flow abstraction heuristics. This is weaker in general, but the gap can be closed without increasing the model size by removing isolated states and using stronger linking constraints.

In general the simplification rules produce a more compact model for computing the cost partitioning of any set of abstraction heuristics. This means that ideas that make h^{SEQ} efficient in practice can be used for other abstractions and hopefully lead to new efficient and high-quality heuristics.

Acknowledgments

This work was supported by the European Research Council as part of the project "State Space Exploration: Principles, Algorithms and Applications".

References

Bäckström, C., and Nebel, B. 1995. Complexity results for SAS$^+$ planning. *Computational Intelligence* 11(4):625–655.

Bonet, B., and van den Briel, M. 2014. Flow-based heuristics for optimal planning: Landmarks and merges. In *Proc. ICAPS 2014*, 47–55.

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

Dräger, K.; Finkbeiner, B.; and Podelski, A. 2006. Directed model checking with distance-preserving abstractions. In *Proc. SPIN 2006*, 19–34.

Edelkamp, S. 2001. Planning with pattern databases. In *Proc. ECP 2001*, 84–90.

Haslum, P.; Bonet, B.; and Geffner, H. 2005. New admissible heuristics for domain-independent planning. In *Proc. AAAI 2005*, 1163–1168.

Helmert, M.; Haslum, P.; Hoffmann, J.; and Nissim, R. 2014. Merge-and-shrink abstraction: A method for generating lower bounds in factored state spaces. *JACM* 61(3):16:1–63.

Katz, M., and Domshlak, C. 2007. Structural patterns heuristics: Basic idea and concrete instance. In *ICAPS 2007 Workshop on Heuristics for Domain-Independent Planning*.

Katz, M., and Domshlak, C. 2010. Optimal admissible composition of abstraction heuristics. *AIJ* 174(12–13):767–798.

Korte, B., and Vygen, J. 2001. *Combinatorial Optimization: Theory and Algorithms*. Springer, 2nd edition.

Pommerening, F., and Helmert, M. 2015. A normal form for classical planning tasks. In *Proc. ICAPS 2015*, 188–192.

Pommerening, F.; Röger, G.; Helmert, M.; and Bonet, B. 2014. LP-based heuristics for cost-optimal planning. In *Proc. ICAPS 2014*, 226–234.

Pommerening, F.; Helmert, M.; Röger, G.; and Seipp, J. 2015. From non-negative to general operator cost partitioning. In *Proc. AAAI 2015*, 3335–3341.

van den Briel, M.; Benton, J.; Kambhampati, S.; and Vossen, T. 2007. An LP-based heuristic for optimal planning. In *Proc. CP 2007*, 651–665.

Yang, F.; Culberson, J.; Holte, R.; Zahavi, U.; and Felner, A. 2008. A general theory of additive state space abstractions. *JAIR* 32:631–662.