Additive Counterexample-guided Cartesian Abstraction Refinement

Jendrik Seipp and **Malte Helmert**
Universität Basel
Basel, Switzerland
{jendrik.seipp,malte.helmert}@unibas.ch

Abstract

We recently showed how counterexample-guided abstraction refinement can be used to derive informative heuristics for optimal classical planning. In this work we introduce an algorithm for building *additive* abstractions and demonstrate that they outperform other state-of-the-art abstraction heuristics on many benchmark domains.

Introduction

Recently, we presented a new algorithm for deriving admissible heuristics for classical planning (Seipp and Helmert 2013) based on the counterexample-guided abstraction refinement (CEGAR) methodology (Clarke et al. 2000).

Starting from a coarse abstraction of a planning task, we iteratively compute an optimal abstract solution, check if and why it fails for the concrete planning task and refine it so that the same failure cannot occur in future iterations. After a given time or memory limit is hit, we use the abstraction as an admissible search heuristic.

As the number of CEGAR iterations grows, one can observe diminishing returns: it takes more and more iterations to obtain further improvements in heuristic value. Therefore, we propose building multiple smaller *additive* abstractions instead of a single big one and show that this increases the number of solved benchmark tasks.

Background

We consider optimal planning in the classical setting, using a SAS$^+$-like (Bäckström and Nebel 1995) representation.

Definition 1. ***Planning tasks.***
*We define a **planning task** as a 5-tuple* $\Pi = \langle \mathcal{V}, \mathcal{O}, c, s_0, s_\star \rangle$.

- $\mathcal{V}$ *is a finite set of **state variables*** v_i*, each with an associated finite domain* $\mathcal{D}(v_i)$.
  *A **fact** is a pair* $\langle v, d \rangle$ *with* $v \in \mathcal{V}$ *and* $d \in \mathcal{D}(v)$.
  *A **partial state** is a function* s *defined on a subset of* $\mathcal{V}$*. This subset is denoted by* $\mathcal{V}_s$*. For all* $v \in \mathcal{V}_s$*, we must have* $s(v) \in \mathcal{D}(v)$*. Partial states defined on all variables are called **states**, and* $\mathcal{S}(\Pi)$ *is the set of all states of* Π.
  *The **update** of partial state* s *with partial state* t*,* $s \oplus t$*, is the partial state defined on* $\mathcal{V}_s \cup \mathcal{V}_t$ *which agrees with* t *on all* $v \in \mathcal{V}_t$ *and with* s *on all* $v \in \mathcal{V}_s \setminus \mathcal{V}_t$.
- $\mathcal{O}$ *is a finite set of **operators**. Each operator* o *has a **precondition*** $pre(o)$ *and **effect*** $eff(o)$*, which are partial states. The **cost function*** c *assigns a cost* $c(o) \in \mathbb{N}_0$ *to each operator.*
- s_0 *is the **initial state** and* $s_\star$ *is a partial state, the **goal**.*

A planning task $\Pi = \langle \mathcal{V}, \mathcal{O}, c, s_0, s_\star \rangle$ induces a transition system with states $\mathcal{S}(\Pi)$, labels $\mathcal{O}$, initial state s_0, goal states $\{s \in \mathcal{S}(\Pi) \mid s_\star \subseteq s\}$ and transitions $\{\langle s, o, s \oplus eff(o) \rangle \mid s \in \mathcal{S}(\Pi), o \in \mathcal{O}, pre(o) \subseteq s\}$. Optimal planning is the problem of finding a shortest path from the initial to a goal state in the transition system induced by a planning task, or proving that no such path exists. For a formal definition of transition systems we refer to Seipp and Helmert (2013).

Abstractions

Abstracting a planning task means losing some distinctions between states to obtain a more "coarse-grained", and hence smaller, transition system.

Definition 2. ***Abstraction.***
*An **abstraction** of a transition system* $\mathcal{T} = \langle S, L, T, s_0, S_\star \rangle$ *is a pair* $\mathcal{A} = \langle \mathcal{T}', \alpha \rangle$ *where* $\mathcal{T}' = \langle S', L', T', s'_0, S'_\star \rangle$ *is a transition system called the **abstract transition system** and* $\alpha : S \to S'$ *is a function called the **abstraction mapping**, such that* $L' = L$*,* $\langle \alpha(s), l, \alpha(s') \rangle \in T'$ *for all* $\langle s, l, s' \rangle \in T$*,* $\alpha(s_0) = s'_0$*, and* $\alpha(s_\star) \in S'_\star$ *for all* $s_\star \in S_\star$.

Abstraction preserves paths in the transition system and can therefore be used to define admissible and consistent heuristics for planning. Specifically, $h^{\mathcal{A}}$ is defined as the cost of an optimal plan starting from s'_0 in the abstract transition system.

We call a set of abstractions *additive* if the sum of their heuristic estimates is admissible.

Additive Abstractions Algorithm

Our algorithm for building additive abstractions is based on the following observation:

Theorem 3. *Consider transition systems* $\mathcal{T}$ *and* $\mathcal{T}'$ *that only differ in the weight of a single transition* $a \to b$*, which is* w *in* $\mathcal{T}$ *and* w' *in* $\mathcal{T}'$*. Let* h *and* h' *denote the goal distance functions in* $\mathcal{T}$ *and* $\mathcal{T}'$.
If $h(a) - h(b) \le w' \le w$*, then* $h = h'$.

Copyright © 2013, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

Proof. $\mathcal{T}$ and $\mathcal{T}'$ only differ in the weight of $a \to b$, so it suffices to show $h'(a) = h(a)$. We have $h'(a) \leq h(a)$ because $w' \leq w$. It remains to show $h'(a) \geq h(a)$.

Clearly $h'(b) = h(b)$: we can assume that shortest paths from b are acyclic, hence do not use the transition $a \to b$, and all other transitions have the same cost in $\mathcal{T}'$ and $\mathcal{T}$.

If a shortest path from a in $\mathcal{T}'$ does not use $a \to b$, then clearly $h'(a) = h(a)$. If it does, then its cost is $h'(a) = w' + h'(b) = w' + h(b) \geq h(a) - h(b) + h(b) = h(a)$. $\square$

Assume we are given an abstraction heuristic h with abstract transition system $\mathcal{T}$. By iteratively applying the theorem, we can reduce the cost of each transition $a \to b$ to $\max(0, h(a) - h(b))$ without affecting h. For each operator o, let $c'(o)$ be the maximal transition cost after this modification over all transitions induced by o. Then we can interpret h as an admissible heuristic under a *cost partitioning* (Katz and Domshlak 2010) with cost function c', and the remaining operator cost $c(o) - c'(o)$ can be used to define further heuristics that can be admissibly added to h.

In our additive abstraction algorithm, we exploit this idea by iteratively computing a sequence of CEGAR heuristics $h_1, \dots, h_n$ (Seipp and Helmert 2013) where h_{i+1} uses the remaining operator costs after computing h_i. The key for an informative overall estimate is to *diversify* the heuristics so that the operator costs are not consumed early on in the process. Towards this end, we follow a simple strategy: we set n to the number of goal facts of the planning task and only consider the i-th goal fact when constructing h_i. If many operators are only relevant for a small number of goals, this may lead to the desired diversity.

Note that the algorithm depends on the order in which the goal facts are considered. However, we observed only minor variations in coverage for different orderings (including random ones).

As a final remark, we point out that the overall idea of the algorithm can be used with any abstraction method. One advantage of CEGAR abstractions is that they are comparatively easy to diversify. Also, they use an explicit representation of the transition system allowing for an easy computation of the c' values. This is not true, for example, for the most efficient implementations of merge-and-shrink heuristics (Nissim, Hoffmann, and Helmert 2011), whose use of *label reduction* removes information about which operator induces which transition.

Experiments

We implemented additive CEGAR abstractions in the Fast Downward system and compared them to state-of-the-art abstraction heuristics already present in the planner: h^{iPDB} (Sievers, Ortlieb, and Helmert 2012) and the two merge-and-shrink heuristics that competed as components of planner portfolios in the IPC 2011 sequential optimization track, $h_1^{\text{m\&s}}$ and $h_2^{\text{m\&s}}$ (Nissim, Hoffmann, and Helmert 2011). We applied a time limit of 30 minutes and memory limit of 2 GB and let h^{CEGAR} refine for at most 15 minutes. For $h_{\text{add}}^{\text{CEGAR}}$ we distributed the refinement time equally among the abstractions.

Coverage	h^0	h^{iPDB}	$h_1^{\text{m\&s}}$	$h_2^{\text{m\&s}}$	h^{CEGAR}	$h_{\text{add}}^{\text{CEGAR}}$
airport (50)	19	21	22	15	19	**32**
blocks (35)	18	**28**	**28**	20	18	18
depot (22)	4	**7**	**7**	6	4	5
driverlog (20)	7	**13**	12	12	10	12
elevators-08 (30)	11	**20**	1	12	13	12
freecell (80)	14	**20**	16	3	15	15
grid (5)	1	**3**	2	**3**	2	2
gripper (20)	7	7	7	**20**	7	7
logistics-00 (28)	10	**21**	16	20	14	20
logistics-98 (35)	2	4	4	5	3	**6**
miconic (150)	50	55	50	74	55	**77**
mprime (35)	19	22	23	11	**24**	**24**
mystery (19)	15	**16**	**16**	7	**16**	**16**
openstacks-08 (30)	**19**	**19**	9	**19**	**19**	**19**
parcprinter-08 (30)	10	12	15	**17**	11	16
pegsol-08 (30)	27	4	2	**29**	27	28
pipesworld-nt (50)	14	**17**	15	8	15	14
pipesworld-t (50)	10	**17**	16	7	11	11
psr-small (50)	49	49	**50**	49	49	49
rovers (40)	5	7	6	**8**	6	7
satellite (36)	4	6	6	**7**	6	**7**
scanalyzer-08 (30)	12	**13**	6	12	12	12
sokoban-08 (30)	19	**29**	3	23	20	20
tpp (30)	5	6	6	7	6	**10**
trucks (30)	6	8	6	8	7	**13**
woodworking-08 (30)	7	7	**14**	9	9	11
zenotravel (20)	8	11	9	11	9	**12**
Sum (1026)	372	442	367	422	407	**475**

Table 1: Number of solved tasks by domain.

Table 1 shows the number of solved instances for most pre-2011 IPC domains. We omit three domains were all heuristics yield the same coverage. In 14 of the 27 domains $h_{\text{add}}^{\text{CEGAR}}$ solves more problems than h^{CEGAR}, raising the total coverage by 68 problems. While the single h^{CEGAR} heuristic solves fewer tasks than h^{iPDB} and $h_2^{\text{m\&s}}$ in total, the additive version outperforms all competitors. In a direct comparison $h_{\text{add}}^{\text{CEGAR}}$ solves more tasks than h^{iPDB}, $h_1^{\text{m\&s}}$ and $h_2^{\text{m\&s}}$ on 11, 16 and 12 domains.

The heuristic estimate for the initial state made by $h_{\text{add}}^{\text{CEGAR}}$ is 31% higher on average than the one made by h^{CEGAR}.

Conclusion

We presented an algorithm for building additive abstractions of classical planning tasks and showed that the derived heuristics often outperform state-of-the-art heuristics.

Future research could try to refine each abstraction until its heuristic values do not increase fast enough anymore instead of applying fixed timeouts.

Acknowledgments

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

References

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

Clarke, E. M.; Grumberg, O.; Jha, S.; Lu, Y.; and Veith, H. 2000. Counterexample-guided abstraction refinement. In Emerson, E. A., and Sistla, A. P., eds., *Proceedings of the 12th International Conference on Computer Aided Verification (CAV 2000)*, 154–169.

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

Nissim, R.; Hoffmann, J.; and Helmert, M. 2011. Computing perfect heuristics in polynomial time: On bisimulation and merge-and-shrink abstraction in optimal planning. In Walsh, T., ed., *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI 2011)*, 1983–1990.

Seipp, J., and Helmert, M. 2013. Counterexample-guided Cartesian abstraction refinement. In *Proceedings of the Twenty-Third International Conference on Automated Planning and Scheduling (ICAPS 2013).* AAAI Press.

Sievers, S.; Ortlieb, M.; and Helmert, M. 2012. Efficient implementation of pattern database heuristics for classical planning. In Borrajo, D.; Felner, A.; Korf, R.; Likhachev, M.; Linares López, C.; Ruml, W.; and Sturtevant, N., eds., *Proceedings of the Fifth Annual Symposium on Combinatorial Search (SOCS 2012)*, 105–111. AAAI Press.