PbFCEGAR: Pre-refined by Facts Counterexample-Guided Abstraction Refinement

Martín Pozo, [1] **Álvaro Torralba,** [2] **Carlos Linares López** [1]

Universidad Carlos III de Madrid, Madrid, Spain
Aalborg University, Aalborg, Denmark

Abstract

Counterexample-Guided Abstraction Refinement (CEGAR) is a prominent technique to generate Cartesian abstractions for guiding search in cost-optimal planning. The core idea is to iteratively refine an abstraction, by finding a flaw of the current optimal abstract plan when it is replicated in the concrete state space. It has shown a significant improvement in performance when used to generate additive abstractions combined by saturated cost partitioning.

But other ways of refining the abstraction are possible, and one of the problems of additive heuristics is the lack of diversity, alleviated by computing a different abstraction for subproblems that use each landmark as the only goal of the problem. At first, refining the abstraction by facts chosen by other domain-independent heuristics could be a good idea to speed up the abstraction creation and to provide diversity, by applying a different heuristic for each abstraction, getting one heuristic to dominate them all, the so long-awaited legendary omni-heuristic! We observed, however, that it does not work well because the quality of the refinements (and even the order in which they are applied) is paramount, and limiting the size of the abstraction is essential too.

Introduction

Counterexample-guided abstraction refinement (CEGAR) is a method that originated in model-checking (Clarke et al. 2000), where it is widely used to prove that error states are unreachable. The key idea is to iteratively refine an abstraction by finding an optimal abstract plan, understanding why it is not an actual plan by using interpolation to find a logic formula that explains the difference, and changing the abstraction to reflect the difference. The idea was brought to planning by Seipp and Helmert (2013b; 2018), using CEGAR to generate Cartesian abstractions that result in informative admissible heuristics for A* search (Hart, Nilsson, and Raphael 1968). Since then, CEGAR has become one of the dominant approaches for generating all kinds of abstraction heuristics (Rovner, Sievers, and Helmert 2019; Kreft et al. 2023). This is best exemplified by the state-of-the-art Scorpion planner (Seipp 2018), which uses this method to compute a diverse set of Cartesian abstraction heuristics and combine them additively (Seipp and Helmert 2013a, 2014) using saturated cost-partitioning (Katz and Domshlak 2010; Seipp, Keller, and Helmert 2017, 2020).

But the question of how to guide the refinement process in CEGAR to generate informative abstractions has comparatively received much less attention. One possibility is to perform batch refinement (Speck and Seipp 2022), where the abstraction is refined at each iteration based on multiple plans instead of only one. Regression flaws is another successful approach that has been proposed recently (Pozo, Torralba, and Linares López 2024), consisting in searching flaws backwards from the goals partial state. In this paper, we pre-refine the abstraction before the loop by the refinement of facts chosen by other families of domain-independent heuristics, with the hope of speeding up the abstraction creation as much as incorporating the knowledge of other heuristics into it. Furthermore, each heuristic can be used to refine a different abstraction, getting another source of diversity for additive abstractions.

Background

We consider tasks in SAS$^+$ representation (Bäckström and Nebel 1995), where states are described in terms of a set of variables V, and each $v \in V$ has a finite domain, $\mathcal{D}_v$. A *partial state* p is a partial variable assignment over some variables $\mathrm{vars}(p) \subseteq V$. A (concrete) state s is a full assignment, i.e., $\mathrm{vars}(s) = V$. We write $p[v] = d$ for the value d assigned to the variable $v \in \mathrm{vars}(p)$ in the partial state p, and we call a *fact* to this assignment. Two partial states p and c are consistent if $p[v] = c[v]$ for all $v \in \mathrm{vars}(p) \cap \mathrm{vars}(c)$. We denote by $S(p) \subseteq S$ the set of states consistent with p.

A SAS$^+$ task Π is a tuple $\langle V, O, s_0, G\rangle$ where s_0 is the initial state, G is a partial state that describes the goals, and O is a set of operators. An operator $o \in O$ has preconditions $pre(o)$ and effects $\mathit{eff}(o)$, both of which are partial states, and a non-negative cost $\mathsf{cost}(o) \in \mathbb{R}_0^+$. An operator o is applicable in progression in a state s if s is consistent with $pre(o)$. The result of applying o to s is another state $s[\![o]\!]$ where $s[\![o]\!][v] = \mathit{eff}(o)[v]$ if $v \in \mathrm{vars}(\mathit{eff}(o))$ and $s[\![o]\!][v] = s[v]$ otherwise. $s \xrightarrow{o} s'$ is a shorthand for the application of o on s when $s' = s[\![o]\!]$. The *state space* of a task Π is a transition system, $\Theta = \langle S, O, T, s_0, S_G\rangle$, where S is the set of all states, $S_G = \{s \in S \mid s \text{ is consistent with } G\}$ is the set of goal states, and $T = \{(s, o, s') \mid s \xrightarrow{o} s'\}$ is

the set of transitions. A *plan* π for s is a sequence of operators $\langle o_1, o_2, \ldots, o_n \rangle$, s.t. the trace $s \xrightarrow{o_1} s_1 \xrightarrow{o_2} \ldots \xrightarrow{o_n} s_n$ reaches a goal state $s_n \in S_G$. The cost of π is the summed up cost of its operators. The goal distance from s to the goal, $h^*(s)$, is the minimum cost of any plan for s, or ∞ if no plan exists. A plan for Π is a plan for the initial state, s_0.

A common approach to find optimal plans is to use A* search with an admissible heuristic. A *heuristic* is a function $h : S \mapsto \mathbb{R}_0^+ \cup \{\infty\}$. The heuristic is admissible if $h(s) \leq h^*(s)$ for all $s \in S$.

An abstraction is a function $\alpha : S \mapsto S^\alpha$, where S^α is a finite set of abstract states. The abstract state space $\Theta^\alpha = \langle S^\alpha, O, T^\alpha, s_0^\alpha, S_G^\alpha \rangle$ is a homomorphism of the state space, i.e., $T^\alpha = \{(\alpha(s) \xrightarrow{o} \alpha(t) \mid s \xrightarrow{o} t \in T)\}$, $s_0^\alpha = \alpha(s_0)$, $S_G^\alpha = \{\alpha(s) \mid s \in S_G\}$. Each abstraction induces a heuristic function where $h^\alpha(s)$ is the distance from $\alpha(s)$ to the goal in Θ^α. Each abstract state $s^\alpha \in S^\alpha$ is identified with the set of states mapped to it, $S(s^\alpha) = \{s \mid s \in S, \alpha(s) = s^\alpha\}$.

Cartesian abstractions are a type of abstractions where the set of states $S(s^\alpha)$ is Cartesian for all $s^\alpha \in S^\alpha$ (Seipp and Helmert 2018). A set of states is Cartesian if it is of the form $A_1 \times A_2 \times \cdots \times A_n$, where $A_i \subseteq \mathcal{D}_{v_i}$ for all $v_i \in V$. Given a Cartesian set a, we denote by $a[v_i]$ the set of values that v_i can take in a, i.e., $a[v_i] = A_i \subseteq \mathcal{D}_{v_i}$. The intersection of two Cartesian sets $a' = a_1 \cap a_2$ is also a Cartesian set, where $a'[v] = a_1[v] \cap a_2[v]$ for all $v \in V$.

Note that Cartesian sets generalize partial states. For any partial state p, we can build a Cartesian set $C(p)$ such that $C(p) = S(p)$, by making $C(p)[v] = \{p[v]\}$ if $v \in \text{vars}(p)$ and $C(p)[v] = \mathcal{D}_v$ otherwise. Slightly abusing notation, we will keep this conversion implicit and define the intersection of a Cartesian set a and a partial state p as $a \cap p = a \cap C(p)$.

The most successful technique to obtain informative Cartesian abstractions is CEGAR (Counterexample-Guided Abstraction Refinement) (Seipp and Helmert 2013b, 2018). CEGAR starts with the trivial abstraction with a single abstract state a s.t. $a[v] = \mathcal{D}_v \ \forall v \in \text{vars}(v)$. Then, it is iteratively refined until reaching a termination condition or finding a concrete plan. The refinement loop finds an optimal abstract plan trace $\tau^\alpha = a_0 \xrightarrow{o_1} a_1 \xrightarrow{o_2} \ldots \xrightarrow{o_n} a_n$ and executes it in the concrete space, resulting in a trace $s_0 \xrightarrow{o_1} s_1 \xrightarrow{o_2} \ldots \xrightarrow{o_n} s_n$. If this execution succeeds and $s_n \in S_G$, then it is an optimal plan for the task. If the plan execution fails at some step, a flaw is reported and the abstraction is refined by splitting an abstract state along the plan into two in such a way that the same flaw cannot happen again. A *flaw* is a tuple $\langle s_i, c \rangle$ consisting of a state $s_i \in S$ and a Cartesian set c. There are three types of flaws, which correspond to different reasons that can cause the execution of τ^α to fail at step i: (1) s_i is the first state in which o_{i+1} is not applicable and c is the set of states in a_i in which o_{i+1} is applicable, i.e. $c = a_i \cap pre(o_{i+1})$. (2) s_i is the first state where o_{i+1} is applicable but its successor is not mapped to a_{i+1}, so c is the set of states in a_i from which a_{i+1} is reached when applying o_i. (3) The sequence can be executed but s_n is not a goal state, producing the flaw $\langle s_n, G \rangle$.

A flaw $\langle s, c \rangle$ is repaired by splitting $\alpha(s)$ into two abstract states d and e with $s \in S(d)$ and $S(c) \subseteq S(e)$. Usually, multiple possible splits exist in different variables to fix the flaw. A split selection strategy is a criterion to choose a split among the ones that fix the flaw (Seipp and Helmert 2018).

The process refines the abstraction until solving the problem either by finding an optimal plan or proving the task unsolvable. The process can be stopped by some termination condition (typically a time or memory limit), resulting in a Cartesian abstraction that induces a heuristic.

Domain-Independent Heuristics Guided Fact Refinement

The CEGAR algorithm is a well-known technique to obtain Cartesian abstractions in planning. However, it starts with the trivial abstraction, so it needs thousands of refinements until getting a good heuristic, and each iteration consists of several steps: getting an optimal abstract plan, finding a flaw by replicating it in the concrete state space, computing the possible splits, choosing one of them and splitting the abstract state into two states. This process can be computationally expensive, especially the computation of the optimal abstract plan, getting the possible splits and choosing one of them.

In addition, CEGAR abstractions have shown a much better performance when used to create additive abstractions in a saturated cost-partitioning scheme (Seipp and Helmert 2018), but to be successful the used abstractions must be diverse, since otherwise the costs are saturated by the first abstractions so that the last abstractions are useless. To achieve it, a sub-problem is created using each landmark as the only goal, but this solution does not work well in problems where capturing the interactions among different goals is necessary to get good goal-distance estimates.

Therefore, starting the refinement loop from a pre-refined, non-trivial, abstraction seems a promising idea to speed up the creation of the abstractions. Starting the process from different initial abstractions can be an alternative method to diversify additive abstractions without completely ignoring the interactions among goals. In this work we propose to define an initial Cartesian abstraction in which a set of facts of the planning task have been fully refined, similar to Pattern Database heuristics (Edelkamp 2001).

Definition 1 (Fact Refinement)**.** *A fact $(v = d)$ is refined in an abstraction α if and only if the value d it is the only value of the variable v in all abstract states where $(v = d)$ is contained. Formally, $a[v] = \{d\} \vee d \notin a[v] \ \forall a \in S^\alpha$.*

We propose to select facts based on domain-independent heuristics that indicate the relevance of facts, e.g., by assigning a value to each fact. The heuristics we have used are:

- Fact landmarks (Hoffmann, Porteous, and Sebastia 2004) are facts that must be achieved at some point along any plan that solves the task. While computing a complete set of all fact landmarks is NP-hard, there are different methods to find a set of fact landmarks (Richter, Helmert, and Westphal 2008; Keyder, Richter, and Helmert 2010; Bonet and Castillo 2011), often based on the relaxed planning graph (Blum and Furst 1997). As these facts are essential to solve the problem, we expect the initial abstraction to be already informative.

- Potential heuristics (Seipp, Pommerening, and Helmert 2015) assign a potential value to each fact by solving a linear program. This value can be interpreted as a measure of the importance of the fact. We used the 10 facts with the smallest potential values in our experiments, but other criteria based on potentials are also possible.

Experiments

We implemented the sequence refinement within the Fast Downward planner (Helmert 2006). Our experiments run on the Autoscale 21.11 benchmark set (Torralba, Seipp, and Sievers 2021), which contains the 42 domains of the International Planning Competitions (IPC) up to 2018 with 30 tasks scaled with the number of objects each, so for optimal planning runtime typically scales exponentially. All experiments are limited to 30 minutes and 8 GB of RAM and run in an Ubuntu Linux 20.04 server with an Intel Xeon X3470 processor at 2.93 GHz, 16 GB of RAM and a 1 TB HDD.

Table 1 shows the coverage at each domain of each configuration. All configurations are executed with the default parameters. All landmarks are used to refine facts when refining by facts, and the 10 facts with the smallest potential when refining by potentials in all configurations. Additive abstractions by subtasks use the default strategy of splitting the problem using each landmark as the only goal, and then creating an abstraction for that problem, while additive abstractions by heuristic generate an abstraction over the whole problem refining by landmarks ($C^{\text{add}}_{\text{land}}$), by the lowest potential facts ($C^{\text{add}}_{\text{pot}}$) or one abstraction for each one ($C^{\text{add}}_{\text{land+pot}}$), and then they run CEGAR on each one and over another more abstraction not pre-refined. So they use 2 abstractions for $C^{\text{add}}_{\text{land}}$ and $C^{\text{add}}_{\text{pot}}$ and 3 abstractions for $C^{\text{add}}_{\text{land+pot}}$ combined by saturated cost partitioning.

Additive abstractions by subtasks get the highest coverage, solving 66 more problems than CEGAR over a single abstraction without pre-refinement, the second-best configuration. The total coverage of all heuristic-guided refinements is always lower than using the default CEGAR algorithm, especially for landmarks-based refinements. The main problem is exceeding memory, what happens in more than 1000 problems for configurations over a single abstraction refining landmarks and in more than 800 problems for the other configurations over a single abstraction. For additive heuristics over subtasks, heuristic-guided refinement is always detrimental except in 1 problem of $data\text{-}network$ and 2 problems of $parking$. For a single abstraction, heuristic-guided refinement is always detrimental except when using potentials in ged. Potentials-guided refinements always work better than landmarks-guided refinements, in most cases due to the limited number of refined facts.

We can summarize the problems of heuristic-guided refinements as follows:

- Choosing heuristics and limiting the number of refined facts is not a trivial task, so the pre-refinement can easily create too big abstractions, even exhausting memory.
- The quality of the refinements is paramount, but the heuristic approach favors the speed of getting refinements instead.
- The diversity created by the heuristics does not actually improve the quality of additive abstractions because many portions of the abstraction are not relevant for the optimal abstract plans, so they are irrelevant for the refinements and for the saturated cost reductions.

Conclusions

The quality of the refinements is essential to achieve a good abstraction heuristic for Automated Planning. Also, the order in which the refinements are applied is also crucial because each refinement modifies the abstraction, and only one refinement is needed in the trivial abstraction, while up to 2^n refinements may be needed after applying n refinements.

In this paper, we have explored the use of some criteria based on other domain-independent heuristics to pre-refine the abstraction before searching flaws, with the possibility to use a different combination of heuristics for each abstraction to get diverse additive abstractions. However, the quality of these refinements is lower than the search of flaws in the concrete space, so the resulting abstractions are larger and less effective. Shrinking strategies could be used to undo the lowest quality refinements, but this have limited effects and it only mitigates the problem slightly. Furthermore, the impact of the diversity created this way is low in the saturated cost partitioning because many of the refined states are not involved in the optimal abstract plans.

In addition, the usage of incremental search (Seipp, von Allmen, and Helmert 2020) to get optimal abstract plans largely alleviates the cost of the refinement loop iterations, which is also smaller in the first iterations of the loop, so the impact in the abstractions build time is irrelevant.

Future Work

Other heuristics can be used as a criterion to choose the facts to refine. Specifically, PDBs (Edelkamp 2001) and the h^2 (Helmert and Domshlak 2009) heuristics can be used to increase the diversity of additive abstractions.

One way to improve our results is limiting the number of refined fact landmarks. Likewise, the creation of a higher number of additive abstractions using a lower number of refined facts in each one, and even combining facts chosen by different heuristics in the same abstraction.

Also, when using potential heuristics as a criterion to choose facts, the maximum between the sum of potentials of the state and the abstraction heuristics could be used almost for free as the heuristic value.

Shrinking the abstractions could reduce the size of the abstractions by merging states, but the only criterion we have found to do it is merging the abstracts states with the same distance to goal which are strictly equal but in one variable, whose values can be merged in a single state. We think this criterion merges very few states, so it was not implemented.

Finally, we think that the pre-refinement of the abstraction before CEGAR is not a totally wrong idea, and other types of pre-refinements could be done. One promising approach is to generate a small domain abstraction by using CEGAR (Kreft et al. 2023) and then mapping it into a Cartesian abstraction to continue the CEGAR refinements.

Coverage	Single abstraction				Additive abstractions by subtasks				Add. abstr. by heuristic		
	C	C_{land}	C_{pot}	$C_{\text{land+pot}}$	$C^{\text{add\_subt}}$	$C^{\text{add\_subt}}_{\text{land}}$	$C^{\text{add\_subt}}_{\text{pot}}$	$C^{\text{add\_subt}}_{\text{land+pot}}$	$C^{\text{add}}_{\text{land}}$	$C^{\text{add}}_{\text{pot}}$	$C^{\text{add}}_{\text{land+pot}}$
agricola (30)	**5**	0	**5**	0	4	0	1	0	0	**5**	0
airport (30)	**5**	0	**5**	0	**5**	3	1	0	0	**5**	0
barman (30)	**12**	6	10	1	**12**	10	9	9	6	10	6
blocksworld (30)	**12**	9	**12**	7	**12**	**12**	**12**	**12**	9	**12**	9
childsnack (30)	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**
data-network (30)	10	8	8	8	13	**14**	11	**14**	9	8	8
depots (30)	11	3	11	1	**12**	10	**12**	11	3	10	3
driverlog (30)	**5**	3	3	3	**5**	3	2	2	3	3	3
elevators (30)	11	11	11	11	**12**	**12**	11	11	11	11	11
floortile (30)	5	4	5	4	**6**	5	4	5	4	5	4
freecell (30)	6	0	6	0	**17**	0	0	0	0	6	0
ged (30)	17	16	**22**	12	20	20	16	16	16	**22**	16
grid (30)	13	10	11	10	**14**	13	8	8	11	10	11
gripper (30)	**12**	**12**	**12**	9	**12**	**12**	**12**	11	**12**	**12**	**12**
hiking (30)	10	9	9	9	**11**	9	7	7	10	9	10
logistics (30)	11	11	11	11	**20**	13	11	12	11	11	11
miconic (30)	4	2	4	0	**5**	3	3	3	2	4	2
mprime (30)	6	5	5	5	**7**	6	5	6	5	5	5
nomystery (30)	6	4	6	4	**7**	**7**	4	5	4	5	4
openstacks (30)	**6**	0	**6**	0	**6**	2	**6**	2	0	**6**	0
organic-synth (30)	**10**	0	**10**	0	9	4	8	4	0	**10**	0
parcprinter (30)	10	3	10	3	**19**	9	9	9	3	9	3
parking (30)	8	3	8	3	8	10	10	**11**	3	8	3
pathways (30)	9	9	9	6	**10**	**10**	9	**10**	9	9	9
pegsol (30)	**28**	1	**28**	0	**28**	27	27	27	1	**28**	1
pipesworld-not (30)	17	12	16	4	**19**	**19**	18	16	12	16	12
pipesworld-tank (30)	**14**	12	**14**	10	**14**	13	10	10	12	13	12
rovers (30)	**4**	2	**4**	1	**4**	**4**	2	3	2	**4**	2
satellite (30)	5	4	5	1	**9**	**9**	5	**9**	4	5	4
scanalyzer (30)	**9**	**9**	**9**	**9**	**9**	**9**	**9**	**9**	**9**	**9**	**9**
snake (30)	**15**	2	**15**	1	**15**	**15**	14	14	2	**15**	2
sokoban (30)	10	0	9	0	**13**	6	6	5	0	9	0
storage (30)	5	5	5	5	**14**	11	5	11	5	5	5
termes (30)	**13**	**13**	**13**	**13**	**13**	**13**	1	0	**13**	**13**	**13**
tetris (30)	**17**	5	**17**	1	16	15	15	15	5	**17**	5
thoughtful (30)	**5**	0	**5**	0	**5**	0	0	0	0	**5**	0
tidybot (30)	9	1	8	0	**10**	8	8	8	1	8	1
tpp (30)	2	2	2	2	**3**	2	2	2	2	2	2
transport (30)	12	11	11	11	**13**	**13**	12	**13**	11	11	11
visitall (30)	**8**	2	**8**	1	5	4	2	3	2	7	2
woodworking (30)	**5**	2	**5**	1	**5**	**5**	**5**	**5**	2	**5**	2
zenotravel (30)	7	6	6	5	**14**	12	8	12	5	7	5
Sum (1260)	394	222	384	177	**460**	377	325	335	224	379	223

Table 1: Coverage of all experiments in all domains.

Acknowledgments

This work was partially funded by grant PID2021-127647NB-C21 from MCIN/AEI/10.13039/501100011033, by the ERDF "A way of making Europe", and by the Madrid Government under the Multiannual Agreement with UC3M in the line of Excellence of University Professors (EPUC3M17) in the context of the V PRICIT (Regional Programme of Research and Technological Innovation). This work has been supported by the Otto Mønsted foundation.

References

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

Blum, A.; and Furst, M. L. 1997. Fast Planning Through Planning Graph Analysis. *Artificial Intelligence*, 90(1–2): 281–300.

Bonet, B.; and Castillo, J. 2011. A Complete Algorithm for Generating Landmarks. In Bacchus, F.; Domshlak, C.; Edelkamp, S.; and Helmert, M., eds., *Proceedings of the Twenty-First International Conference on Automated Planning and Scheduling (ICAPS 2011)*, 315–318. AAAI Press.

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.

Edelkamp, S. 2001. Planning with Pattern Databases. In Cesta, A.; and Borrajo, D., eds., *Proceedings of the Sixth European Conference on Planning (ECP 2001)*, 84–90. AAAI Press.

Hart, P. E.; Nilsson, N. J.; and Raphael, B. 1968. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2): 100–107.

Helmert, M. 2006. The Fast Downward Planning System. *Journal of Artificial Intelligence Research*, 26: 191–246.

Helmert, M.; and Domshlak, C. 2009. Landmarks, Critical Paths and Abstractions: What's the Difference Anyway? In Gerevini, A.; Howe, A.; Cesta, A.; and Refanidis, I., eds., *Proceedings of the Nineteenth International Conference on Automated Planning and Scheduling (ICAPS 2009)*, 162–169. AAAI Press.

Hoffmann, J.; Porteous, J.; and Sebastia, L. 2004. Ordered Landmarks in Planning. *Journal of Artificial Intelligence Research*, 22: 215–278.

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

Keyder, E.; Richter, S.; and Helmert, M. 2010. Sound and Complete Landmarks for And/Or Graphs. In Coelho, H.; Studer, R.; and Wooldridge, M., eds., *Proceedings of the 19th European Conference on Artificial Intelligence (ECAI 2010)*, 335–340. IOS Press.

Kreft, R.; Büchner, C.; Sievers, S.; and Helmert, M. 2023. Computing Domain Abstractions for Optimal Classical Planning with Counterexample-Guided Abstraction Refinement. In Koenig, S.; Stern, R.; and Vallati, M., eds., *Proceedings of the Thirty-Third International Conference on Automated Planning and Scheduling (ICAPS 2023)*. AAAI Press.

Pozo, M.; Torralba, Á.; and Linares López, C. 2024. When CEGAR Meets Regression: A Love Story in Optimal Classical Planning. In *Proceedings of the Thirty-Eighth AAAI Conference on Artificial Intelligence (AAAI 2024)*, 20238–20246. AAAI Press.

Richter, S.; Helmert, M.; and Westphal, M. 2008. Landmarks Revisited. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence (AAAI 2008)*, 975–982. AAAI Press.

Rovner, A.; Sievers, S.; and Helmert, M. 2019. Counterexample-Guided Abstraction Refinement for Pattern Selection in Optimal Classical Planning. In Lipovetzky, N.; Onaindia, E.; and Smith, D. E., eds., *Proceedings of the Twenty-Ninth International Conference on Automated Planning and Scheduling (ICAPS 2019)*, 362–367. AAAI Press.

Seipp, J. 2018. Fast Downward Scorpion. In *Ninth International Planning Competition (IPC-9): Planner Abstracts*, 77–79.

Seipp, J.; and Helmert, M. 2013a. Additive Counterexample-guided Cartesian Abstraction Refinement. In desJardins, M.; and Littman, M. L., eds., *Late-Breaking Developments in the Field of Artificial Intelligence – Papers Presented at the Twenty-Seventh AAAI Conference on Artificial Intelligence (AAAI 2013) – AAAI Technical Report WS-13-17*, 119–121. AAAI Press.

Seipp, J.; and Helmert, M. 2013b. Counterexample-guided Cartesian Abstraction Refinement. In Borrajo, D.; Kambhampati, S.; Oddi, A.; and Fratini, S., eds., *Proceedings of the Twenty-Third International Conference on Automated Planning and Scheduling (ICAPS 2013)*, 347–351. AAAI Press.

Seipp, J.; and Helmert, M. 2014. Diverse and Additive Cartesian Abstraction Heuristics. In Chien, S.; Fern, A.; Ruml, W.; and Do, M., eds., *Proceedings of the Twenty-Fourth International Conference on Automated Planning and Scheduling (ICAPS 2014)*, 289–297. AAAI Press.

Seipp, J.; and Helmert, M. 2018. Counterexample-Guided Cartesian Abstraction Refinement for Classical Planning. *Journal of Artificial Intelligence Research*, 62: 535–577.

Seipp, J.; Keller, T.; and Helmert, M. 2017. Narrowing the Gap Between Saturated and Optimal Cost Partitioning for Classical Planning. In Singh, S.; and Markovitch, S., eds., *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI 2017)*, 3651–3657. AAAI Press.

Seipp, J.; Keller, T.; and Helmert, M. 2020. Saturated Cost Partitioning for Optimal Classical Planning. *Journal of Artificial Intelligence Research*, 67: 129–167.

Seipp, J.; Pommerening, F.; and Helmert, M. 2015. New Optimization Functions for Potential Heuristics. In Brafman, R.; Domshlak, C.; Haslum, P.; and Zilberstein, S., eds., *Proceedings of the Twenty-Fifth International Conference on Automated Planning and Scheduling (ICAPS 2015)*, 193–201. AAAI Press.

Seipp, J.; von Allmen, S.; and Helmert, M. 2020. Incremental Search for Counterexample-Guided Cartesian Abstraction Refinement. In Beck, J. C.; Karpas, E.; and Sohrabi, S., eds., *Proceedings of the Thirtieth International Conference on Automated Planning and Scheduling (ICAPS 2020)*, 244–248. AAAI Press.

Speck, D.; and Seipp, J. 2022. New Refinement Strategies for Cartesian Abstractions. In Thiébaux, S.; and Yeoh, W., eds., *Proceedings of the Thirty-Second International Conference on Automated Planning and Scheduling (ICAPS 2022)*, 348–352. AAAI Press.

Torralba, Á.; Seipp, J.; and Sievers, S. 2021. Automatic Instance Generation for Classical Planning. In Goldman, R. P.; Biundo, S.; and Katz, M., eds., *Proceedings of the Thirty-First International Conference on Automated Planning and Scheduling (ICAPS 2021)*, 376–384. AAAI Press.