Counterexample-Guided Abstraction Refinement for Pattern Selection in Optimal Classical Planning: Additional Material

Alexander Rovner and **Silvan Sievers** and **Malte Helmert**
University of Basel
Basel, Switzerland
alex.rovner@stud.unibas.ch, {silvan.sievers,malte.helmert}@unibas.ch

Technical Report CS-2019-002

Common Parameters	
Π	planning task
MAXTIME	maximum runtime
PDBMAXSIZE	maximum size of individual PDBs
COLLECTIONMAXSIZE	maximum size of all PDBs together
WILDCARD	true iff wildcard plans should be computed
Disjoint Pattern Collections (Single CEGAR Algorithm)	
Blacklist	a subset of state variables of Π
Multiple CEGAR Algorithm	
MAXSTAG	maximum runtime that may pass without finding a new pattern before triggering stagnation
BLTRIGGERTIME	maximum runtime that may pass without finding a new pattern before blacklisting is used
BLONSTAG	if true, turn on blacklisting when MAXSTAG is hit the first time; terminate the second time

Table 1: Parameters of both CEGAR algorithms (top), additional parameters of the single CEGAR algorithm (middle), and additional parameters of the multiple CEGAR algorithm (bottom).

This supplementary material contains some additional details referenced from Rovner, Sievers, and Helmert (2019). In particular, we provide pseudo-code of some functions not shown in the main paper. Furthermore, we report per-domain coverage of all experiments and include domain comparisons where they were missing in the main paper.

Pseudo-Code

This section provides pseudo-code for some functions that are only described in words in the main paper. Table 1 shows global parameters of both CEGAR algorithms which are not shown as parameters of the individual functions.

Single CEGAR Runs

Algorithm 1 shows pseudo-code for computing a plan of a given pattern P. The algorithm first computes the PDB for P (line 2). Note that this needs to be done only once for each pattern since we store the plan we compute for each pattern. The remainder of the computation is steepest-ascent enforced hill climbing with h^P as the perfect heuristic. In each

Algorithm 1 COMPUTEPLAN of CEGAR.

```
1:  function ComputePlan($\Pi$, $P$)
2:      $h^P \leftarrow$ PDB for $P$ // perfect heuristic for $\Pi^P$
3:      $\pi = \langle\rangle$ // sequence of states
4:      $s \leftarrow s_0^P$ // $s_0^P$ initial state in $\Pi^P$
5:      $f^* \leftarrow h^P(s)$
6:      if $f^* = \infty$ then
7:          exit with UNSOLVABLE
8:      while $s \notin S_*^P$ do
9:          $\pi', s \leftarrow$ BFS$(s, f^*, h^P)$
10:         $\pi \leftarrow \pi \circ \pi'$
11:     return $\pi$
12: function BFS($s_{start}$, $f^*$, $h^P$)
13:     queue $\leftarrow [s_{start}]$
14:     closed $\leftarrow \{s_{start}\}$
15:     while TRUE do
16:         $s \leftarrow$ queue.POP()
17:         $h_{best} \leftarrow h(s_{start})$
18:         $s_{best} \leftarrow$ NONE
19:         for $t \in$ RandomOrder(SUCCESSORS$(s)$) do
20:             if $t \notin$ closed then
21:                 if $f(t) = f^*$ then
22:                     if $h(t) < h_{best}$ or $t \in S_\star^P$ then
23:                         $h_{best} \leftarrow h(t)$
24:                         $s_{best} \leftarrow t$
25:                     else
26:                         open.PUSH$(t)$
27:                         closed $\leftarrow$ closed $\cup \{t\}$
28:         if $s_{best}$ then
29:             return ExtractPlan$(s_{best})$, $s_{best}$
```

outer iteration (lines 8–10), the algorithm uses a breadth-first search (BFS) from the current state s for a state with the lowest h-value, which initially is chosen to be the initial state (line 4). The result of BFS is a sequence of states which extends the maintained state sequence π (line 3). The last state of BFS serves for updating s for the next iteration. The loop terminates when reaching a goal state. BFS itself is shown in the second procedure of Algorithm 1. The key elements are that we use early duplicate detection and that we can prune all successor states that do not have an f-value equal to the known perfect f-value. The function EXTRACTPLAN

Algorithm 2 FINDFLAWS of CEGAR.

```
1: function FindFlaws($\Pi$, $\pi$, Blacklist)
2:     // $s_0$ initial state of $\Pi$, $s_\star$ goal of $\Pi$
3:     $V \leftarrow \emptyset$
4:     $s \leftarrow s_0$
5:     for $O$ in $\pi$ do
6:         failed ← TRUE
7:         for $o \in O$ do
8:             flaw ← FALSE
9:             for $\langle v, d \rangle \in pre(o)$ do
10:                if $v \notin$ Blacklist and $s[v] \neq d$ then
11:                    flaw ← TRUE
12:                    $V \leftarrow V \cup \{v\}$
13:            if not flaw then
14:                failed ← FALSE
15:                $V \leftarrow \emptyset$
16:                $s \leftarrow$ Apply$(s, o)$
17:                break
18:        if failed then
19:            break
20:    if $V = \emptyset$ then
21:        if $s[v] = s_\star[v]$ for all $v \in vars(s_\star)$ then
22:            if Blacklist $= \emptyset$ then
23:                exit with plan $\pi$
24:        else
25:            $V \leftarrow \{v \mid v \in vars(s_\star) \wedge s[v] \neq s_\star[v]\}$
26:    return $V$
```

extracts a plan starting from a given search node by following the predecessor node pointers. Depending on whether regular or wildcard plans should be computed, the function either chooses a random cheapest operator of those that lead from a predecessor node to a current node or all of them (in uniformly random order).

Algorithm 2 shows pseudo-code for computing flaws given an abstract plan. It returns a set of variables V that it identified as flaws. The outer loop (lines 5–19) executes exactly one step of the plan, which may consist of multiple "parallel" operators if using wildcard plans, or of a single operator otherwise. The inner loop (lines 7–17) considers each operator o of the plan step and attempts to apply it in the current state s. To do so, it checks for all non-blacklisted variables v of the precondition of o if they comply with s (lines 9–12). If not, v is added to V. If no flaws of the step were found, the operator is applied (line 16) and the algorithm continues with the next plan step in the outer-most loop. If, when leaving the outer-most loop, there are no flaws (line 20), then we need to check if the concrete goal of the task is satisfied (line 21). If it is and additionally the blacklist is empty, then we know that we found a concrete solution with which we can exit (line 23). Otherwise, we collect all violated goal variables and return them (line 25).

Multiple CEGAR Runs

Algorithm 3 shows pseudo-code of the multiple CEGAR algorithm that repeatedly calls the single CEGAR algorithm to compute a single pattern in each iteration of the main

Algorithm 3 Multiple CEGAR algorithm.

```
Input: Planning task $\Pi$
Output: Pattern collection $C$
1: function MultipleCEGAR($\Pi$)
2:     // $\mathcal{V}$ variables of $\Pi$, $s_\star$ goal of $\Pi$
3:     $C \leftarrow \emptyset$
4:     UseBl ← FALSE
5:     LastChange ← Time()
6:     goals ← RandomOrder(vars($s_\star$))
7:     while TRUE do
8:         for $g \in$ goals do
9:             if Time() > MaxTime then
10:                return $C$
11:            if Time() > LastChange + MaxStag then
12:                if BlOnStag and not UseBl then
13:                    UseBl ← TRUE
14:                    LastChange ← Time()
15:                else
16:                    return $C$
17:            if Time() > BlTriggerTime then
18:                UseBl ← TRUE
19:            Blacklist ← $\emptyset$
20:            if UseBl then
21:                BlSize ← RandomNumber($|\mathcal{V}|$)
22:                nongoals ← $\{v \mid v \notin vars(s_\star)\}$
23:                Blacklist ← SelectUniformly(
                       nongoals, min(BlSize, |nongoals|))
24:            $P \leftarrow$ CEGAR($\Pi[s_\star|_g]$, Blacklist)
25:            if $P \notin C$ then
26:                LastChange ← Time()
27:                $C \leftarrow C \cup \{P\}$
```

loop (lines 7–27). To ensure diversification, it computes a random order on goal variables once (line 6) and repeatedly iterates over them (line 8). For the case where stagnation or blacklisting is used (MAXSTAG < MAXTIME or BLTRIGGERTIME < MAXTIME), the algorithm keeps track of the last time point in which progress was made, i.e., a new pattern was added to the collection (line 5 and line 26). Blacklisting is initially set to FALSE (line 4).

At the start of each iteration, the algorithm first checks if the stagnation limit has been reached (line 11). If so, it additionally checks if also blacklisting should be used on stagnation (BLONSTAG) and it is not using blacklisting yet for diversification (line 12). If that is the case, blacklisting is enabled and the progress timer reset. Otherwise, the collection is returned and the algorithm terminates. Similarly, the next check to be done is to enable blacklisting (line 18) if no progress has been made for long enough time (line 17).

Then, depending on if blacklisting should be used, the algorithm computes a random size of at most $|\mathcal{V}|$ for the blacklist and chooses as many non-goal variables if available (lines 19–23). The resulting blacklist (empty per default) is passed to the single CEGAR algorithm along with the task Π with a goal condition restricted to the goal variable of the iteration (line 24). If the resulting pattern is new, it is added to the collection (line 27) and the progress timer is reset. If

	regP	wcP	SYS	HC1	CPC1	HC9	CPC9	sRCG	G	tot
regP	–	**20**	**34**	9	10	9	8	**40**	**24**	854.7
wcP	17	–	**34**	8	8	8	6	**38**	21	853.8
SYS	14	13	–	0	5	0	3	**28**	17	769
HC1	**38**	**38**	**38**	–	22	1	15	**44**	**30**	935.5
CPC1	**44**	**45**	**49**	**26**	–	24	4	**49**	**35**	951.7
HC9	**40**	**40**	**39**	**10**	**26**	–	21	**47**	**32**	953.4
CPC9	**46**	**44**	**48**	**33**	**34**	**27**	–	**51**	**39**	**975.7**
sRCG	6	5	22	3	3	3	3	–	5	758.8
G	23	**25**	**31**	11	10	9	9	**34**	–	839

	regP	wcP	SYS	HC1	CPC1	HC9	CPC9	tot
regP	–	19	18	19	16	17	23	943.7
wcP	**24**	–	19	22	20	18	25	946.6
SYS	**26**	**26**	–	**17**	20	15	**28**	981
HC1	**27**	**26**	12	–	13	2	23	946.4
CPC1	**35**	**33**	**23**	**28**	–	**28**	**28**	**1033.5**
HC9	**29**	**30**	**17**	**11**	17	–	**25**	965.4
CPC9	**30**	**28**	23	**27**	18	**25**	–	1021.1

Table 2: Domain comparison in terms of coverage for single CEGAR runs (2 variants), SYS, HC, and CPC with the CDPB (top) and the SCP heuristic (bottom), including the single PDB approaches sRCG and G for completeness (top). An entry in row x and column y denotes the number of domains in which x solves more tasks than y. It is bold if $(x, y) \geq (y, x)$. The two rightmost columns show total coverage (tot) and pattern collection construction time (c. t.) of the algorithm of the row.

no time is left, the algorithm returns the collection and terminates (line 10).

Results

This section provides full per-domain coverage results of all algorithms.

Single CEGAR Runs

Table 2 shows a domain comparison of Table 1 in the main paper.

Table 3 shows per-domain coverage of Table 1 in the main paper.

Multiple CEGAR Runs

Table 4 shows per-domain coverage of Table 2 in the main paper.

State of the Art

Table 5 shows a domain comparison of Table 3 in the main paper.

Table 6 shows per-domain coverage of Table 3 in the main paper.

References

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).* AAAI Press.

	CPDB							SCP								
	regP	wcP	SYS	HC1	CPC1	HC9	CPC9	regP	wcP	SYS	HC1	CPC1	HC9	CPC9	sRCG	G
agricola-opt18-strips (20)	0.9	**1.8**	0	0.0	0.0	0.0	0.0	0.9	**1.8**	0	0.0	0.0	0.0	0.1	0.0	0
airport (50)	32.6	32.3	2	31.0	29.9	**38.0**	30.2	36.0	35.6	24	31.0	31.0	**38.0**	31.8	23.1	23
barman-opt11-strips (20)	4.0	4.0	4	4.0	**8.0**	4.0	**8.0**	4.0	4.0	4	4.0	**8.0**	4.0	**8.0**	4.0	4
barman-opt14-strips (14)	0.0	0.0	0	0.0	2.0	0.0	2.4	0.0	0.0	0	0.0	2.6	0.0	**3.0**	0.0	0
blocks (35)	21.0	21.0	**28**	**28.0**	25.9	**28.0**	26.0	**28.0**	**28.0**	**28**	**28.0**	**28.0**	**28.0**	**28.0**	21.0	22
childsnack-opt14-strips (20)	**0.0**	**0.0**	**0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0**
data-network-opt18-strips (20)	11.9	11.9	11	12.0	12.7	12.0	**13.0**	**13.0**	**13.0**	11	12.0	12.7	12.0	**13.0**	9.4	9
depot (22)	7.0	7.0	7	11.0	7.0	11.0	7.0	10.0	9.8	**12**	11.0	8.6	11.0	9.1	6.0	7
driverlog (20)	13.0	13.1	13	13.5	14.0	13.5	14.1	13.0	13.1	**15**	13.5	13.6	13.5	13.1	10.3	11
elevators-opt08-strips (30)	19.2	19.7	21	23.0	24.7	23.0	**25.1**	20.0	20.5	2	23.0	23.6	23.0	23.0	14.9	23
elevators-opt11-strips (20)	16.2	16.2	17	18.0	**19.0**	18.0	**19.0**	16.7	16.9	16	18.0	18.7	18.0	18.3	12.9	**19**
floortile-opt11-strips (20)	2.0	2.4	2	2.5	6.0	2.5	**6.7**	4.1	4.1	4	4.0	6.2	4.0	5.9	2.4	3
floortile-opt14-strips (20)	0.3	0.2	0	0.0	4.2	0.0	**6.6**	2.0	2.0	2	2.0	5.1	2.0	4.7	0.2	1
freecell (80)	20.1	20.0	2	21.0	20.5	21.0	20.9	22.0	22.3	21	21.0	23.5	21.0	**24.2**	20.0	21
ged-opt14-strips (20)	19.0	19.0	19	19.0	19.3	19.0	**19.6**	19.0	19.0	19	19.0	19.0	19.0	17.2	18.6	15
grid (5)	2.1	2.6	2	**3.0**	**3.0**	**3.0**	**3.0**	2.2	2.6	**3**	**3.0**	**3.0**	**3.0**	**3.0**	1.6	**3**
gripper (20)	**8.0**	**8.0**	**8**	**8.0**	**8.0**	**8.0**	**8.0**	**8.0**	**8.0**	**8**	**8.0**	6.2	**8.0**	5.1	**8.0**	**8**
hiking-opt14-strips (20)	18.4	**18.9**	13	13.0	18.1	13.0	**18.9**	18.4	**18.9**	13	13.0	18.5	13.0	18.4	12.1	12
logistics00 (28)	20.5	20.6	19	22.0	22.0	23.4	22.4	20.5	20.6	**26**	22.0	21.9	23.2	21.9	14.5	15
logistics98 (35)	5.0	5.0	4	6.0	5.2	6.0	6.0	5.0	5.0	**8**	6.0	7.0	6.3	7.1	3.0	3
miconic (150)	62.9	62.5	58	68.9	68.1	68.9	71.2	62.9	62.5	7	71.3	110.9	71.3	**113.1**	55.0	63
movie (30)	**30.0**	**30.0**	**3**	**30.0**	**30.0**	**30.0**	**30.0**	**30.0**	**30.0**	**3**	**30.0**	**30.0**	**30.0**	**30.0**	**30.0**	**3**
mprime (35)	24.1	24.1	23	24.0	23.2	24.0	23.1	24.1	24.2	**3**	24.0	23.2	24.0	23.1	22.5	25
mystery (30)	15.9	16.0	16	17.0	16.1	17.0	16.0	16.3	16.5	**18**	17.0	16.0	17.0	16.0	16.4	15
nomystery-opt11-strips (20)	18.5	19.3	**2**	**20.0**	**20.0**	**20.0**	**20.0**	18.5	19.3	**2**	**20.0**	**20.0**	**20.0**	19.1	9.0	13
openstacks-opt08-strips (30)	**22.0**	**22.0**	14	**22.0**	21.9	**22.0**	21.8	**22.0**	**22.0**	**22**	**22.0**	21.9	**22.0**	21.8	**22.0**	**22**
openstacks-opt11-strips (20)	**17.0**	**17.0**	9	**17.0**	16.9	**17.0**	16.9	**17.0**	**17.0**	**17**	**17.0**	16.9	**17.0**	**17.0**	**17.0**	**17**
openstacks-opt14-strips (20)	**3.0**	**3.0**	0	**3.0**	**3.0**	**3.0**	**3.0**	**3.0**	**3.0**	**3**	**3.0**	**3.0**	**3.0**	**3.0**	**3.0**	**3**
openstacks-strips (30)	**7.0**	**7.0**	**7**	**7.0**	**7.0**	**7.0**	**7.0**	**7.0**	**7.0**	**7**	**7.0**	**7.0**	**7.0**	**7.0**	**7.0**	**7**
organic-synthesis-opt18-strips (20)	**7.0**	**7.0**	**7**	**7.0**	**7.0**	**7.0**	**7.0**	**7.0**	**7.0**	**7**	**7.0**	**7.0**	**7.0**	**7.0**	**7.0**	**7**
organic-synthesis-split-opt18-strips (20)	9.0	9.0	5	9.0	**10.0**	8.0	**10.0**	**10.0**	**10.0**	**1**	**10.0**	**10.0**	**10.0**	**10.0**	**10.0**	**1**
parcprinter-08-strips (30)	14.8	14.9	9	18.1	17.9	**22.0**	19.1	17.8	18.1	21	19.0	17.1	**22.0**	16.9	11.1	14
parcprinter-opt11-strips (20)	10.8	10.9	5	14.1	13.3	**17.0**	15.1	13.7	14.1	**17**	15.0	12.7	**17.0**	12.6	7.1	1
parking-opt11-strips (20)	1.0	1.0	0	**7.0**	1.0	**7.0**	1.0	**7.0**	6.9	**7**	**7.0**	**7.0**	**7.0**	5.1	0.0	0
parking-opt14-strips (20)	0.0	0.0	0	**6.0**	0.1	**6.0**	0.1	**6.0**	**6.0**	**6**	**6.0**	**6.0**	**6.0**	4.0	0.0	0
pathways-noneg (30)	4.0	4.0	4	4.0	4.0	4.0	4.0	4.2	**4.3**	4	4.0	4.0	4.0	4.0	4.0	4
pegsol-08-strips (30)	28.7	28.3	21	29.0	29.0	29.3	29.1	28.7	28.3	29	29.0	29.0	**30.0**	28.6	29.0	29
pegsol-opt11-strips (20)	18.7	18.3	11	19.0	19.0	19.3	19.1	18.7	18.3	19	19.0	19.0	**20.0**	17.6	19.0	19
petri-net-alignment-opt18-strips (20)	0.0	0.0	0	0.0	4.0	0.0	4.0	4.0	4.0	**7**	0.0	4.0	0.0	4.0	4.0	4
pipesworld-notankage (50)	21.6	22.5	17	21.0	23.0	21.0	24.5	24.1	24.3	2	21.0	24.1	21.0	24.4	17.4	**25**
pipesworld-tankage (50)	15.0	15.7	15	**18.0**	17.0	**18.0**	17.0	16.5	16.3	17	**18.0**	17.0	**18.0**	16.6	13.7	**18**
psr-small (50)	**50.0**	**50.0**	48	**50.0**	**50.0**	**50.0**	**50.0**	**50.0**	**50.0**	**5**	**50.0**	**50.0**	**50.0**	49.0	**50.0**	**5**
rovers (40)	7.9	7.2	7	8.0	8.0	8.0	8.5	8.6	8.3	7	8.0	11.4	8.0	**11.9**	6.0	7
satellite (36)	6.0	6.0	4	6.0	**7.0**	6.0	**7.0**	6.0	6.1	6	6.0	**7.0**	6.0	**7.0**	6.0	6
scanalyzer-08-strips (30)	12.0	12.0	13	13.0	12.2	13.0	13.0	13.9	15.9	**18**	14.0	14.3	13.0	11.4	12.0	12
scanalyzer-opt11-strips (20)	9.0	9.0	1	10.0	9.2	10.0	10.0	10.8	12.9	**15**	11.0	11.3	10.0	9.1	9.0	9
snake-opt18-strips (20)	11.3	11.7	13	13.0	13.3	**14.0**	13.8	12.5	12.6	13	13.0	13.0	13.5	12.4	11.2	**14**
sokoban-opt08-strips (30)	28.1	27.8	**3**	**30.0**	27.9	**30.0**	28.8	29.8	29.7	**3**	**30.0**	**30.0**	**30.0**	**30.0**	23.3	28
sokoban-opt11-strips (20)	**20.0**	**20.0**	**2**	**20.0**	**20.0**	**20.0**	**20.0**	**20.0**	**20.0**	**2**	**20.0**	**20.0**	**20.0**	**20.0**	19.5	**2**
spider-opt18-strips (20)	12.0	11.6	0	14.0	12.4	**15.0**	13.0	14.1	14.5	14	14.0	13.6	**15.0**	14.2	11.0	11
storage (30)	15.2	15.1	**16**	**16.0**	15.0	**16.0**	15.0	15.8	15.7	**16**	**16.0**	15.6	**16.0**	**16.0**	14.7	15
termes-opt18-strips (20)	12.0	12.0	12	**13.0**	12.7	**13.0**	12.4	12.0	12.0	12	**13.0**	12.8	**13.0**	11.0	12.0	**13**
tetris-opt14-strips (17)	9.0	9.0	4	9.0	10.0	10.0	10.1	9.5	9.4	**11**	10.0	10.6	10.0	10.6	9.0	1
tidybot-opt11-strips (20)	15.1	14.0	14	14.0	14.0	14.0	14.0	**16.9**	16.0	14	14.0	14.0	14.0	13.9	13.2	15
tidybot-opt14-strips (20)	11.1	9.9	9	9.0	9.5	9.0	9.9	**12.9**	12.0	9	9.0	9.2	9.0	9.8	6.3	9
tpp (30)	8.3	7.9	6	6.0	**12.0**	6.0	**12.0**	8.3	7.9	6	6.0	**12.0**	6.0	**12.0**	6.0	6
transport-opt08-strips (30)	11.5	11.4	11	13.8	**14.0**	13.8	**14.0**	11.5	11.6	11	13.8	**14.0**	13.8	**14.0**	11.1	**14**
transport-opt11-strips (20)	6.7	7.0	6	12.7	13.0	12.7	**14.0**	6.8	7.2	6	11.9	12.4	11.9	11.9	6.2	11
transport-opt14-strips (20)	7.1	7.0	7	9.0	9.1	9.0	**10.0**	7.1	7.0	7	9.0	9.0	9.0	9.6	7.0	7
trucks-strips (30)	7.9	7.5	9	9.0	10.0	9.0	10.0	8.9	8.8	9	9.0	10.0	9.0	**10.2**	6.3	8
visitall-opt11-strips (20)	10.0	9.9	16	16.0	17.6	16.0	**18.0**	16.8	17.0	17	16.0	17.5	16.0	17.3	9.0	11
visitall-opt14-strips (20)	4.6	4.4	12	12.0	14.7	12.0	**15.0**	13.1	13.1	13	12.0	13.8	12.0	13.0	3.0	5
woodworking-opt08-strips (30)	10.8	10.5	8	13.9	15.8	14.0	16.8	15.7	14.2	**29**	14.0	20.6	16.0	22.5	8.9	1
woodworking-opt11-strips (20)	5.8	5.5	3	9.0	10.7	9.0	11.5	10.3	9.2	**2**	9.0	15.5	11.0	16.0	3.9	5
zenotravel (20)	11.1	11.2	12	**13.0**	12.6	**13.0**	**13.0**	11.1	11.2	**13**	12.9	12.9	12.9	12.5	8.0	9
Sum (1827)	854.7	853.8	769	935.5	951.7	953.4	975.7	943.7	946.6	981	946.4	**1033.5**	965.4	1021.1	758.8	839

Table 3: Per-domain coverage of the CPDB (left) and the SCP (right) heuristic over PDBs computed with a single CEGAR run (regP and wcP), SYS, HC, CPC, and also of the single PDB heuristics generated with sRCG and G.

	regP				wcP				mRCG	SYS	HC1	CPC1	HC9	CPC9
	v	s	b	sb	v	s	b	sb						
agricola-opt18-strips (20)	3.6	3.6	3.3	3.3	**3.8**	3.7	3.6	3.6	0.0	0	0.0	0.0	0.0	0.1
airport (50)	37.1	37.2	37.1	36.8	37.9	37.8	36.3	36.4	26.8	24	31.0	31.0	**38.0**	31.8
barman-opt11-strips (20)	4.0	4.0	4.0	4.0	4.0	4.0	4.0	4.0	4.0	4	4.0	**8.0**	4.0	**8.0**
barman-opt14-strips (14)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	0.0	2.6	0.0	**3.0**
blocks (35)	**28.0**	**28.0**	**28.0**	**28.0**	**28.0**	**28.0**	**28.0**	**28.0**	**28.0**	**28**	**28.0**	**28.0**	**28.0**	**28.0**
childsnack-opt14-strips (20)	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0**	**0.0**	**0.0**	**0.0**	**0.0**
data-network-opt18-strips (20)	13.0	13.0	13.0	13.0	**13.1**	**13.1**	**13.1**	**13.1**	11.9	11	12.0	12.7	12.0	13.0
depot (22)	**12.3**	**12.3**	**12.3**	**12.3**	11.7	11.7	11.7	11.7	10.0	12	11.0	8.6	11.0	9.1
driverlog (20)	13.4	13.5	14.0	14.0	13.6	13.7	14.0	14.0	14.0	**15**	13.5	13.6	13.5	13.1
elevators-opt08-strips (30)	22.0	22.0	22.0	22.0	22.2	22.2	22.2	22.2	22.8	2	23.0	**23.6**	23.0	23.0
elevators-opt11-strips (20)	18.0	18.0	18.0	18.0	18.0	18.0	18.0	18.0	18.1	16	18.0	**18.7**	18.0	18.3
floortile-opt11-strips (20)	6.0	6.0	6.0	6.0	6.0	6.0	6.0	6.0	**6.8**	4	4.0	6.2	4.0	5.9
floortile-opt14-strips (20)	5.0	5.0	5.0	5.0	4.0	4.0	5.0	5.0	**6.3**	2	2.0	5.1	2.0	4.7
freecell (80)	23.4	23.4	23.4	23.4	**26.0**	**26.0**	**26.0**	**26.0**	20.8	21	21.0	23.5	21.0	24.2
ged-opt14-strips (20)	19.0	19.0	19.0	19.0	19.0	19.0	19.0	19.0	**19.3**	19	19.0	19.0	19.0	17.2
grid (5)	**3.0**	**3.0**	**3.0**	**3.0**	**3.0**	**3.0**	**3.0**	**3.0**	**3.0**	**3**	**3.0**	**3.0**	**3.0**	**3.0**
gripper (20)	**8.0**	**8.0**	**8.0**	**8.0**	**8.0**	**8.0**	**8.0**	**8.0**	6.6	**8**	**8.0**	6.2	**8.0**	5.1
hiking-opt14-strips (20)	14.6	14.5	14.2	14.3	18.0	18.0	14.3	15.7	14.1	13	13.0	**18.5**	13.0	18.4
logistics00 (28)	21.1	21.0	25.8	25.9	21.0	21.0	25.7	25.6	21.7	**26**	22.0	21.9	23.2	21.9
logistics98 (35)	5.4	5.4	7.4	7.4	5.4	5.4	7.3	7.4	7.0	**8**	6.0	7.0	6.3	7.1
miconic (150)	137.4	137.1	135.6	136.6	136.3	137.1	136.0	136.4	**149.0**	7	71.3	110.9	71.3	113.1
movie (30)	**30.0**	**30.0**	**30.0**	**30.0**	**30.0**	**30.0**	**30.0**	**30.0**	**30.0**	**3**	**30.0**	**30.0**	**30.0**	**30.0**
mprime (35)	28.2	28.2	28.3	28.3	28.9	28.9	29.2	29.4	25.6	**3**	24.0	23.2	24.0	23.1
mystery (30)	18.8	18.8	18.8	18.8	**19.0**	**19.0**	**19.0**	**19.0**	17.4	18	17.0	16.0	17.0	16.0
nomystery-opt11-strips (20)	**20.0**	**20.0**	**20.0**	**20.0**	**20.0**	**20.0**	**20.0**	**20.0**	**20.0**	**2**	**20.0**	**20.0**	**20.0**	19.1
openstacks-opt08-strips (30)	**22.0**	**22.0**	**22.0**	**22.0**	**22.0**	**22.0**	**22.0**	**22.0**	20.0	**22**	**22.0**	21.9	**22.0**	21.8
openstacks-opt11-strips (20)	**17.0**	**17.0**	**17.0**	**17.0**	**17.0**	**17.0**	**17.0**	**17.0**	15.0	**17**	**17.0**	16.9	**17.0**	**17.0**
openstacks-opt14-strips (20)	**3.0**	**3.0**	**3.0**	**3.0**	**3.0**	**3.0**	**3.0**	**3.0**	**3.0**	**3**	**3.0**	**3.0**	**3.0**	**3.0**
openstacks-strips (30)	**7.0**	**7.0**	**7.0**	**7.0**	**7.0**	**7.0**	**7.0**	**7.0**	**7.0**	**7**	**7.0**	**7.0**	**7.0**	**7.0**
organic-synthesis-opt18-strips (20)	**7.0**	**7.0**	**7.0**	**7.0**	**7.0**	**7.0**	**7.0**	**7.0**	5.0	**7**	**7.0**	**7.0**	**7.0**	**7.0**
organic-synthesis-split-opt18-strips (20)	**10.0**	**10.0**	**10.0**	**10.0**	**10.0**	**10.0**	**10.0**	**10.0**	9.4	**1**	**10.0**	**10.0**	**10.0**	**10.0**
parcprinter-08-strips (30)	18.0	18.0	18.0	18.0	18.0	18.0	18.0	18.0	16.2	21	19.0	17.1	**22.0**	16.9
parcprinter-opt11-strips (20)	14.0	14.0	14.0	14.0	14.0	14.0	14.0	14.0	12.2	**17**	15.0	12.7	**17.0**	12.6
parking-opt11-strips (20)	6.9	6.9	6.9	6.9	**7.0**	**7.0**	**7.0**	**7.0**	**7.0**	**7**	**7.0**	**7.0**	**7.0**	5.1
parking-opt14-strips (20)	5.9	5.9	**6.0**	**6.0**	5.9	5.9	**6.0**	**6.0**	**6.0**	**6**	**6.0**	**6.0**	**6.0**	4.0
pathways-noneg (30)	4.1	4.1	4.1	4.1	**4.9**	**4.9**	**4.9**	**4.9**	4.0	4	4.0	4.0	4.0	4.0
pegsol-08-strips (30)	29.0	29.0	29.0	29.0	29.0	29.0	29.0	29.0	29.0	29	29.0	29.0	**30.0**	28.6
pegsol-opt11-strips (20)	19.0	19.0	19.0	19.0	19.0	19.0	19.0	19.0	19.0	19	19.0	19.0	**20.0**	17.6
petri-net-alignment-opt18-strips (20)	5.5	5.5	5.5	5.5	5.4	5.4	5.4	5.4	**7.6**	7	0.0	4.0	0.0	4.0
pipesworld-notankage (50)	23.0	23.0	23.5	23.5	22.6	22.6	22.6	22.6	22.8	2	21.0	24.1	21.0	**24.4**
pipesworld-tankage (50)	16.6	16.6	16.6	16.6	16.8	16.8	16.7	16.7	16.6	17	**18.0**	17.0	**18.0**	16.6
psr-small (50)	**50.0**	**50.0**	**50.0**	**50.0**	**50.0**	**50.0**	**50.0**	**50.0**	**50.0**	**5**	**50.0**	**50.0**	**50.0**	49.0
rovers (40)	9.0	9.0	9.0	9.0	9.0	9.0	9.0	9.0	8.3	7	8.0	11.4	8.0	**11.9**
satellite (36)	**8.0**	**8.0**	**8.0**	**8.0**	**8.0**	**8.0**	**8.0**	**8.0**	6.7	6	6.0	7.0	6.0	7.0
scanalyzer-08-strips (30)	16.0	16.1	16.0	16.1	17.4	17.4	17.4	17.4	14.1	**18**	14.0	14.3	13.0	11.4
scanalyzer-opt11-strips (20)	13.1	13.1	13.1	13.1	14.4	14.4	14.4	14.4	11.1	**15**	11.0	11.3	10.0	9.1
snake-opt18-strips (20)	12.0	12.0	12.0	12.0	12.0	12.0	12.0	12.0	11.8	13	13.0	13.0	**13.5**	12.4
sokoban-opt08-strips (30)	**30.0**	**30.0**	29.5	29.5	**30.0**	**30.0**	29.2	29.5	**30.0**	**3**	**30.0**	**30.0**	**30.0**	**30.0**
sokoban-opt11-strips (20)	**20.0**	**20.0**	19.4	19.5	**20.0**	**20.0**	19.4	19.3	**20.0**	**2**	**20.0**	**20.0**	**20.0**	**20.0**
spider-opt18-strips (20)	13.0	13.0	13.0	13.0	13.0	13.0	13.2	13.2	13.2	14	14.0	13.6	**15.0**	14.2
storage (30)	**16.0**	**16.0**	**16.0**	**16.0**	**16.0**	**16.0**	**16.0**	**16.0**	**16.0**	**16**	**16.0**	15.6	**16.0**	**16.0**
termes-opt18-strips (20)	12.1	12.1	12.1	12.1	12.0	12.0	12.0	12.0	12.0	12	**13.0**	12.8	**13.0**	11.0
tetris-opt14-strips (17)	9.6	9.6	10.0	10.0	9.5	9.5	10.0	10.0	9.6	**11**	10.0	10.6	10.0	10.6
tidybot-opt11-strips (20)	15.8	15.8	15.8	15.9	15.9	15.9	**16.0**	**16.0**	13.4	14	14.0	14.0	14.0	13.9
tidybot-opt14-strips (20)	11.8	11.9	11.8	11.8	11.9	11.9	**12.0**	**12.0**	4.3	9	9.0	9.2	9.0	9.8
tpp (30)	**12.1**	**12.1**	**12.1**	**12.1**	12.0	12.0	12.0	12.0	8.0	6	6.0	12.0	6.0	12.0
transport-opt08-strips (30)	13.0	13.0	**14.0**	**14.0**	13.0	13.0	**14.0**	**14.0**	**14.0**	11	13.8	**14.0**	13.8	**14.0**
transport-opt11-strips (20)	9.4	9.4	10.4	10.4	10.0	10.0	11.0	11.0	10.0	6	11.9	**12.4**	11.9	11.9
transport-opt14-strips (20)	9.5	9.5	9.5	9.5	9.5	9.5	9.5	9.5	7.0	7	9.0	9.0	9.0	**9.6**
trucks-strips (30)	12.0	12.0	**13.0**	**13.0**	12.0	12.0	**13.0**	**13.0**	10.0	9	9.0	10.0	9.0	10.2
visitall-opt11-strips (20)	10.0	10.0	17.0	17.0	10.0	10.0	17.0	17.0	10.0	17	16.0	**17.5**	16.0	17.3
visitall-opt14-strips (20)	4.0	4.0	13.0	13.0	4.0	4.0	13.0	13.0	4.0	13	12.0	**13.8**	12.0	13.0
woodworking-opt08-strips (30)	22.3	22.2	22.9	22.7	21.9	21.9	21.9	21.8	23.1	**29**	14.0	20.6	16.0	22.5
woodworking-opt11-strips (20)	15.1	15.1	15.1	15.1	14.0	14.0	14.0	14.0	16.1	**2**	9.0	15.5	11.0	16.0
zenotravel (20)	12.0	12.0	12.6	12.6	12.2	12.0	**13.0**	**13.0**	**13.0**	**13**	12.9	12.9	12.9	12.5
Sum (1827)	1055.1	1054.9	1080.1	1081.1	1063.2	1063.7	1085.0	**1087.2**	1018.7	981	946.4	1033.5	965.4	1021.1

Table 4: Per-domain coverage of the SCP heuristic over PDBs computed with multiple CEGAR runs (8 variants), mRCG, SYS, HC, and CPC.

	best	CPC-S-P	$h^{\text{SCP}}_{\text{hybrid-opt}}$	$h^{\text{SCP}}_{\text{hybrid-opt}}$+best	tot
best	–	23	14	14	1087.2
CPC-S-P	**28**	–	**22**	**26**	1073
$h^{\text{SCP}}_{\text{hybrid-opt}}$	**24**	21	–	**18**	1129.5
$h^{\text{SCP}}_{\text{hybrid-opt}}$+best	**24**	24	15	–	**1138.8**

Table 5: Domain comparison in terms of coverage of the SCP heuristic over PDBs generated by our best method (best) and by CPC-S-P, and the planner $h^{\text{SCP}}_{\text{hybrid-opt}}$ and an extension of $h^{\text{SCP}}_{\text{hybrid-opt}}$ to also compute PDBs by using our best method. See Table 2 for an explanation of the entries.

	best	CPC-S-P	$h^{\text{SCP}}_{\text{hybrid-opt}}$	$h^{\text{SCP}}_{\text{hybrid-opt}}$+best
agricola-opt18-strips (20)	**3.6**	0	0.0	0.0
airport (50)	**36.4**	26	24.3	28.1
barman-opt11-strips (20)	**4.0**	**4**	**4.0**	**4.0**
barman-opt14-strips (14)	0.0	**2**	0.0	0.0
blocks (35)	**28.0**	**28**	**28.0**	**28.0**
childsnack-opt14-strips (20)	0.0	**2**	0.0	0.0
data-network-opt18-strips (20)	13.1	13	**14.0**	13.0
depot (22)	11.7	8	**13.0**	12.9
driverlog (20)	14.0	**15**	**15.0**	**15.0**
elevators-opt08-strips (30)	22.2	**25**	**25.0**	22.0
elevators-opt11-strips (20)	18.0	**19**	**19.0**	18.0
floortile-opt11-strips (20)	6.0	**11**	4.0	6.0
floortile-opt14-strips (20)	5.0	**11**	2.0	5.0
freecell (80)	26.0	25	**68.0**	65.1
ged-opt14-strips (20)	19.0	**2**	19.0	18.9
grid (5)	**3.0**	**3**	**3.0**	**3.0**
gripper (20)	8.0	**11**	8.0	8.0
hiking-opt14-strips (20)	15.7	**19**	14.0	16.7
logistics00 (28)	**25.6**	23	25.0	24.5
logistics98 (35)	7.4	6	**11.6**	9.0
miconic (150)	136.4	105	140.8	**141.6**
movie (30)	**30.0**	**3**	**30.0**	**30.0**
mprime (35)	29.4	24	**30.0**	29.6
mystery (30)	**19.0**	17	**19.0**	**19.0**
nomystery-opt11-strips (20)	**20.0**	**2**	**20.0**	**20.0**
openstacks-opt08-strips (30)	22.0	**29**	22.0	22.0
openstacks-opt11-strips (20)	17.0	**2**	17.0	17.0
openstacks-opt14-strips (20)	3.0	**12**	3.0	3.0
openstacks-strips (30)	7.0	7	**9.0**	**9.0**
organic-synthesis-opt18-strips (20)	**7.0**	**7**	**7.0**	**7.0**
organic-synthesis-split-opt18-strips (20)	**10.0**	**1**	**10.0**	**10.0**
parcprinter-08-strips (30)	**18.0**	**18**	**18.0**	17.6
parcprinter-opt11-strips (20)	**14.0**	13	**14.0**	13.3
parking-opt11-strips (20)	**7.0**	3	**7.0**	**7.0**
parking-opt14-strips (20)	**6.0**	5	**6.0**	**6.0**
pathways-noneg (30)	4.9	**5**	**5.0**	4.8
pegsol-08-strips (30)	29.0	29	29.0	**30.0**
pegsol-opt11-strips (20)	19.0	19	19.0	**20.0**
petri-net-alignment-opt18-strips (20)	**5.4**	4	0.0	0.0
pipesworld-notankage (50)	22.6	**25**	**25.0**	24.1
pipesworld-tankage (50)	16.7	**18**	**18.0**	17.4
psr-small (50)	**50.0**	**5**	**50.0**	**50.0**
rovers (40)	9.0	**13**	8.0	10.6
satellite (36)	8.0	**1**	7.0	7.9
scanalyzer-08-strips (30)	17.4	13	**17.5**	17.0
scanalyzer-opt11-strips (20)	14.4	1	**14.5**	14.0
snake-opt18-strips (20)	12.0	**14**	13.0	13.0
sokoban-opt08-strips (30)	29.5	**3**	**30.0**	**30.0**
sokoban-opt11-strips (20)	19.3	**2**	**20.0**	**20.0**
spider-opt18-strips (20)	13.2	12	**15.0**	**15.0**
storage (30)	**16.0**	15	**16.0**	**16.0**
termes-opt18-strips (20)	12.0	**16**	12.0	12.0
tetris-opt14-strips (17)	10.0	**14**	11.0	11.0
tidybot-opt11-strips (20)	16.0	14	15.0	**16.9**
tidybot-opt14-strips (20)	12.0	1	10.0	**12.9**
tpp (30)	12.0	**15**	8.0	12.0
transport-opt08-strips (30)	**14.0**	**14**	13.8	**14.0**
transport-opt11-strips (20)	11.0	12	12.8	**13.8**
transport-opt14-strips (20)	9.5	**1**	9.0	9.0
trucks-strips (30)	13.0	12	13.2	**15.9**
visitall-opt11-strips (20)	17.0	**18**	17.0	17.0
visitall-opt14-strips (20)	13.0	**15**	13.0	13.0
woodworking-opt08-strips (30)	21.8	19	**26.0**	22.5
woodworking-opt11-strips (20)	14.0	13	**19.0**	16.7
zenotravel (20)	**13.0**	**13**	**13.0**	**13.0**
Sum (1827)	1087.2	1073	1129.5	**1138.8**

Table 6: Per-domain coverage of the SCP heuristic over PDBs generated by our best method and by CPC-S-P, and the planner $h^{\text{SCP}}_{\text{hybrid-opt}}$ and an extension of $h^{\text{SCP}}_{\text{hybrid-opt}}$ to also compute PDBs by using our best method.