UNIVERSITÄT BASEL

Refining abstraction heuristics with mutexes

Bachelor thesis

Natural Science Faculty of the University of Basel
Department of Computer Science
http://ai.cs.unibas.ch/

Examiner: Prof. Dr. Malte Helmert
Supervisor: Dr. Martin Wehrle

Matthias Solèr
matthias.soler@stud.unibas.ch
09-055-179

31-07-2012

Abstract

Planning as heuristic search is a powerful approach to solve domain-independent planning problems. An important class of heuristics is based on abstractions of the original planning task. However it comes with loss in precision. The contribution of this thesis is the concept of constrained abstraction heuristics in general, and the application of this concept to pattern database and merge and shrink abstractions in particular. The idea is to use a subclass of mutexes which represent sets of variable-value-pairs so that only one of these pairs can be true at any given time, to regain some of the precision which is lost in the abstraction without increasing its size. By removing states and operators in the abstraction which conflict with such a mutex, the abstraction is refined and hence, the corresponding abstraction heuristic can get more informed. We have implemented the refinements of these heuristics in the Fast-Downward planner and evaluated the different approaches using standard IPC benchmarks. The results show that the concept of constrained abstraction heuristics can improve the planning as heuristic search in terms of time and coverage.

Table of Contents

1 Introduction

Domain-independent planning is a generalized way of solving search problems where the domain of a given problem is part of the input as well. However, the complexity of the problems which can be solved by a planner is still quite restricted due to limited resources. A powerful approach for solving planning problems is based on heuristic search. A heuristic is a mapping of states onto numerical values and represents an approximation of the actual costs remaining to reach a goal. An important class of heuristics are based on abstractions which describe the problem in a smaller version of the problem. It's easier to calculate the costs in these abstractions and use those costs as heuristic values.

There are two main representatives of abstraction heuristics, namely the *pattern databases heuristics* and the *merge and shrink heuristics*. Pattern database heuristics are a well-established type of abstraction heuristics where a subset of variables of the problem gets chosen as a pattern and therefore a smaller state space with just those variables gets created while disregarding all the other variables which aren't part of the pattern [Culberson and Schaeffer, 1998, Edelkamp, 2001]. The merge and shrink is a newer and more generalized approach which starts with abstractions for each variable, where all the others are disregarded and then merges those abstractions consecutively and shrinks the abstractions whenever they get too big. [Helmert et al., 2007] The problem of these abstraction heuristics however is that they lose quite a bit of precision.

The goal of this thesis is regain some of this precision without increasing the size of the abstraction by using so called *mutexes* as suggested for the pattern databases by Haslum et al. [2005]. A mutex is a set of facts with the property that not all of the facts can simultaneously be true in all reachable states. As computing all possible mutexes is computationally hard, we focus on an efficiently computable subclass thereof. This subclass is characterised by variable-value-pairs where only one of those pairs per mutex group can be true at any time and hence as soon as two or more pairs are true in a state in the abstraction, we know that this state can't exist in the original graph and therefore is invalid and can be removed from the abstraction. There is also a similar rule for operators as well. By removing those invalid states and operators, some precision is gained back, hence increasing the heuristic values while maintaining admissibility. This is desirable because in the general case, this will improve the performance of the A*-search algorithm as shown in the work of Pearl [1984].

The A* algorithm is a standard search algorithm which is based on heuristics for solving problems and the solutions are always optimal as long as the heuristic is admissible.
The contribution of this thesis is the development of an equivalent concept for merge and shrink, the implementation for both PDB and M&S heuristics in the Fast-Downward framework as well as the evaluation based on standard IPC benchmarks. Another notable difference to the original work of Haslum et al. [2005] is the usage of SAS^+ instead of *STRIPS*. *STRIPS* is a formalisation of a problem based on Boolean variables whereas SAS^+ uses multivalued variables.
The remainder of this thesis is organized as follows: The second chapter will provide the theoretical background and the third chapter will introduce the refinement theoretically as well as on the level of the implementation. In the fourth chapter we'll evaluate the implementations of the refinements based on standard IPC benchmarks and in the fifth chapter we will conclude the thesis.

2 Background

2.1 Planning

Planning is, simply spoken, a generalized form of a search problem where the problem definition itself is part of the input. Therefore, all problems (including their domains) which can be formalized in a general problem domain definition language (PDDL [McDermott et al., 1998]) should be theoretically able to be solved by any planning system. Those planning tasks can be formalized in various ways. Two, which are important for this paper, are presented below.

2.1.1 Task Formalisation

A *STRIPS* planning task can be described as a 4-tuple $\langle V, O, s_0, S_* \rangle$, where V is a set of Boolean variables, O is a set of operators which are described by preconditions and effects, s_0 is the starting state defined by the initial values of all variables and S_* is a set of goal states which are defined by a goal value of variables in a specific subset of all variables.
The SAS^+ planning tasks are defined similarly except that the variables aren't Boolean but instead each variable has an arbitrary, finite domain size. In addition, operators also use prevail variables. Prevail variables can be seen as constants which have to have that value throughout the action. A conversion of a *STRIPS* to a SAS^+ task can be achieved by clustering the *STRIPS* variables on a semantic layer where only one of the variables in every cluster can be true at any given time. The prevail variables of the operators can be simply derived by the intersection of the precondition and effect pairs (both kinds look like $\langle variable, value \rangle$). [Bäckström, 1992]
A *transition graph* is a semantic representation of a given planning task. It is a 5-tuple $\langle S, L, A, s_0, S_* \rangle$, where S is a finite set of states, L is a set of labels, A is a set of labelled actions (or transitions) with $A \subseteq S \times L \times S$, s_0 is the initial state and S_* is a set of solution states. It is easy to convert either task into a transition graph where solving the problem is equivalent to finding a path form the initial state to any goal state. An optimal solution corresponds to the cheapest path.

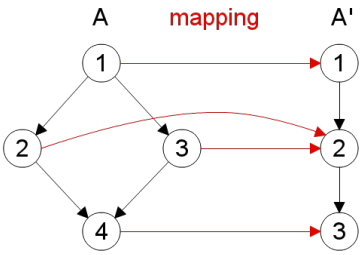

Figure 2.1: This image shows an example of a simple abstraction mapping.

2.1.2 Abstractions and Projections

An abstraction of a transition graph is a graph which ignores some information of the original graph and therefore is typically smaller. An example is shown in Figure 2.1. Mathematically, it is a simple mapping with a function $f : S \to S'$ so that $L' = L, \langle f(s), l, f(s') \rangle \in A'$ for all $\langle s, l, s' \rangle \in A, f(s_0) = s'_0$ and $f(s_*) \in S'_*$ for all $s_* \in S_*$ [Helmert et al., 2007]. There are special forms of abstraction, for example projections. A projection is a mapping where certain variables get completely ignored whereas the information of other variables stays fully intact. Formally it is equivalent to $f(s) = f(s')$ if $s(v) = s'(v)$ for all $v \in V$ where V is a subset of all variables. The special case where the graph gets projected onto a single variable is called atomic abstraction or atomic projection as shown in Figure 2.2.

2.2 Heuristics

To speed up the solving of the planning tasks, we use informed search which is based on heuristics. A heuristic maps states onto numerical values and tries to approximate the costs of the remaining dpath to the closest goal. A* uses those values as follows. For a given state s, the costs up to this state $g(s)$ and the heuristic value for it $h(s)$ it, A* evaluates $f(s) := g(s) + h(s)$. Therefore $f(s)$ is an estimate of the total costs of the path which leads through the state s. A* processes the states in their ascending order respective to their f-values hence higher heuristic values lead to better estimates. Since we want to solve the problems optimally, the heuristic must be admissible, hence never overestimating the actual costs, to reach any given goal or else the A* algorithm might find a suboptimal solution.

2.2.1 Abstraction Heuristics

In this thesis we focus on abstraction heuristics, more precisely pattern databases (PDBs) and merge and shrink heuristics (M&S). Both of these approaches are introduced in more detail below. The general idea is as follows. By using an abstraction with fewer states than the original graph we can use use a simple shortest-path algorithm like Dijkstra to obtain the costs for these abstract states and use those costs of each node as heuristic values of the actual states which are represented in that specific abstract node. You could actually calculate different values for heuristic purposes, but this is the basic idea, which is used in both PDB and M&S.

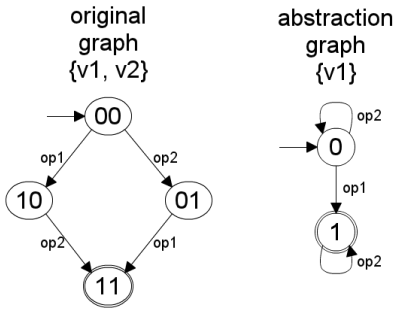

Figure 2.2: This image shows an example abstraction with the pattern {v1} which is the same as the atomic projection of the variable v1.

```
create_pdb(pattern) {
    build_abstract_problem(pattern)
    initialize_Dijkstra()
    calculate_distances_using_reversed_Dijkstra()
}
```

Figure 2.3: Pseudo algorithm of a standard PDB implementation. Note that the abstract states get implicitly built in the Dijkstra loop whereas the abstract operators get created as part of the abstract problem.

2.2.2 Pattern Databases

A pattern database heuristic (also referenced as PDB heuristic) consists of a pattern which contains a subset of variable names and a lookup table with heuristic values. The basic idea is that each pattern describes a projection of the problem. All the variables which aren't in the pattern are ignored and therefore collapsed together as shown in Figure 2.2. The remaining graph gets solved with a breath-first-search in reverse direction and the resulting costs are saved in a lookup table to be reused as heuristic values. Hence all the states which get projected onto the same abstract state will have the heuristic value which corresponds to that particular abstract state which again corresponds to the costs of the path from that state to the nearest goal. [Culberson and Schaeffer, 1998, Edelkamp, 2001]
Figure 2.3 shows the main part of the pattern database algorithm, given the pattern, described in pseudo code. It uses the Dijkstra which is an algorithm which finds the cheapest path in a graph. The reversed Dijkstra is actually the same except that it starts at the goals and looks for the start. This has several advantages over the other direction when it comes to the evaluation of the heuristic values.

2.2.3 iPDB

There's one huge problem with pattern databases: Depending on the chosen pattern, the results can vary drastically. Another important fact is that heuristic values of several pattern databases can be combined ($\max(h^1(s), h^2(s)) =: h^{new}(s)$ and in some cases even $h^1(s) + h^2(s) =: h^{new}(s)$ is admissible). The iPDB tries to account for these facts. It searches in the space of patterns and combines a lot of rather small PDBs to one big heuristic based on the rules just mentioned. There are other optimisations included as well, but these are the core ideas of this approach. [Sievers et al., 2012]

```
compute_abstraction(problem, N) {
    abs := {atomic_projection(v) | v is a variable of the problem}
    while(size(abs) ≤ 1) {
        Select A1, A2 ∈ abs.
        Shrink A1 and/or A2 until size(A1)*size(A2) ≤ N.
        abs := (abs \{A1, A2}) ∪ {A1 ⊗ A2}
    }
    return the last element in abs
}
```

Figure 2.4: General algorithm of the merge and shrink heuristic taken out of the work of Helmert et al. [2007]. N is the maximal size of the abstractions.

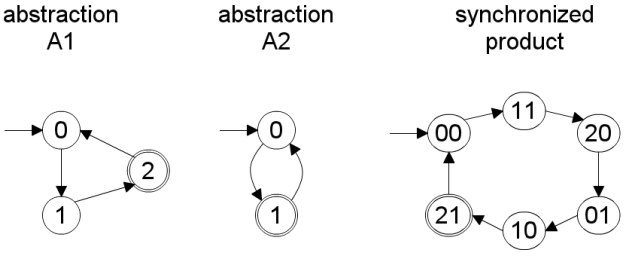

Figure 2.5: This image shows the result of a simple synchronized product.

2.2.4 Merge and Shrink

Merge and Shrink is another abstraction heuristic which consists out of two main steps after the initialisation. In the simple setup, which I used in my work, the initialisation consists out of atomic projections onto each variable and the decision in which order those abstractions will be combined. The first abstraction in the order gets declared as the working abstraction. Now, the basic concept is quite simple. Merge the current working abstraction together with the next atomic abstraction by using a *synchronized product* and repeat this until all abstractions are combined. The synchronized product is an operator which takes two abstractions and results in a new abstraction which contains the information of both inputs by pairing the states in all possible combinations as illustrated in Figure 2.5. The mathematical sign for the synchronized product is the $\otimes$. The size of an abstraction which is the number of states in the abstraction, of the resulting abstraction is the product of the sizes of the input abstractions. Whenever the resulting abstraction would become too big (compared to some threshold) you have to shrink the working abstraction. (Theoretically you can also shrink the atomic abstractions.) Shrinking creates an abstraction where several states of the working abstraction get mapped together to one, preferably not total random states but instead as similar as possible ones. As soon as all abstractions are merged, the distances in this abstraction will be used as heuristic values, similar to the PDB. Actually, it is possible to describe every PDB with a specific merge and shrink abstraction but not the other way round. A more general and formal pseudo algorithm is shown in Figure 2.4. [Helmert et al., 2007]

3

Refining Abstraction Heuristics with Mutexes

The goal of this thesis it to refine the abstraction heuristics by gaining back some of the lost precision in the abstractions without increasing their size. It is based on the usage of *mutexes* which describe some invariants and all states and operators who violate these invariants should be ignored when finding a path through the abstraction. Since the so called constrained abstractions are smaller than the standard abstractions and don't contain any extra paths the costs in the constrained graphs is equal or higher compared to those in the standard abstraction graphs and therefore the resulting heuristic values change as well. Pearl [1984] has shown, that the A*-algorithm finds solutions faster and therefore with less memory needed in the general case when it uses more precise admissible heuristics. We hope that the constrained abstractions will result in better coverages because in most cases the bottleneck is the memory needed by the search-algorithm. In the section 3.1 we explain the theoretical background of the *mutexes* and how we can use them. In the later sections we describe how we implemented this refinement in the Fast-Downward environment.

3.1 Idea

The basic idea, on which this whole thesis is based on, is to use a specific type of *mutexes*, so called "at-most-one-*atom*" mutexes. An atom is another word for variable-value-pair and comes from the *STRIPS* environment. The reason, why we don't use all mutexes is, because in general computing all mutexes for a planning task is computationally hard, however, this subclass is known to be computable in polynomial time. These "at-most-one-atom" mutexes each describe a set of variable-value-pairs of which only one can be true at any given time in the reachable graph. This means, that there can't exist a state in the original graph where two of those variable-value-pairs are active. Formally, this results in the following. Let M be a set of variable-value-pairs which describes a mutex of the "at-most-one-atom" type. Let s be a state defined as a set of variable-value-pairs too. All reachable states in the original graph fulfil the following inequation: $|M \cap s| \leq 1$. However in an abstraction of the graph, it is possible that a state doesn't fulfil the inequation and thus the states in the original graph which map onto this abstract state aren't reachable. Hence we can ignore this "invalid" state which is equal to removing it from the graph. Operators can contain

additional information about the states they connect and therefore it can happen, that the states are valid but the operators between them aren't. Formally, the operators have some preconditions $pre(o)$ which can, together with the knowledge about the state, "contradict" such a mutex: $|(pre(o) \cup s) \cap M| \leq 1$. [Haslum et al., 2005] This works for *STRIPS* as well as SAS^+. The difference is, that in SAS^+ the variable values are aligned along such mutexes (as suggested in Helmert [2004]), hence just containing *STRIPS* variables which would be in a mutex themselves. Thus the number of available mutexes in a SAS^+ environment is smaller compared to the number of mutexes in a *STRIPS* environment, however those missing mutexes already get used implicitly.
A constrained graph is the same as the normal graph just without any states or operators which violate those "at-most-one-atom" mutexes, hence it contains a subset of the states and operators of the non-constrained graph. Therefore the costs in the constrained graph are higher or equal to the costs of the non-constrained graph. As long as the original graph has a solution, the constrained abstractions do have to have one as well since we only removed states which are unreachable in the original graph while containing all the reachable states and thus the states onto which the reachable states get mapped to still have to be contained in the constrained abstraction. Since the costs are equal or higher, no new detours have been introduced and the heuristic values correspond to the costs in the abstract graph, the inequation 3.1 has to be valid. (Note: $h^*(s)$ is the perfect heuristic where the heuristic values correspond to the actual remaining costs.)

$$h^{abstraction}(s) \leq h^{constrainedabstraction}(s) \leq h^*(s), \forall s \in S \tag{3.1}$$

As shown in the book "Heuristics - intelligent search strategies for computer problem solving" which was written by Pearl [1984], the A*-search algorithm finds optimal solutions in shorter amounts of time in a general case, the better the heuristics are, as long as they stay admissible. Because the search becomes faster, it will also require less memory which is the current bottleneck in most cases.

3.2 Mutexes in the Fast-Downward Planner

To test this concept of using mutexes to increase the precision the Fast-Downward planner was used [Helmert, 2006] with it's implementations of the pattern database as well as the M&S-algorithm. The planner already calculates the "at-most-one-atom" mutexes in a preprocessing step since it uses them for the *STRIPS* to SAS^+ conversion thus those were already given. However, it is important to know that the formalisation is different in the solver than the mutexes defined in the previous chapter. The first snippet (Figure 3.1) shows a mutex as a set of variable-value-pairs as described before which is part of the output of the preprocessing part. This output file is used as an input for the search-program part of the planning system which converts it into an internal form. The only possibility to access is from the heuristics is the function are_mutex which takes two pairs as an input. (One pair represents the <variable, value> data.) Therefore the sets of variable-value-pairs degenerate into multiple sets of the size two. The access to the function as well as a few examples are depicted in Figure 3.2. This is equivalent because as soon as one of these mutex-pairs fires, the corresponding bigger set would fire too and the other way round as well. However, there is a loss in precision because it is impossible in the general case to trace back which pair

```
begin_mutex_group
6 //the following mutex group has 6 members
2 0 //var 2 with value 0
7 0 //var 7 with value 0
8 1 //...
0 1
9 1
10 1
end_mutex_group
```

Figure 3.1: This shows a snippet of the output of the pre-compiler for the probBLOCKS-5-0 task.

```
are_mutex(pair<int, int>, pair<int, int>): bool

are_mutex(<2, 0>, <7, 0>) → true
are_mutex(<2, 0>, <8, 1>) → true
are_mutex(<7, 0>, <8, 1>) → true
```

Figure 3.2: Internal access and return values

belongs to which set. This has an impact on the possible uses of those mutexes but the needed functionality is still available.

3.3 Constrained Pattern Databases

Since the goal is to remove nodes and operators in abstract graphs to improve the heuristic values, the main part of the implementation will be at the part of code, where the abstract graphs either get constructed or traversed. In case of pattern database algorithm, this is basically only true for the last line of the pseudo-algorithm (as shown in Figure 2.3), the reversed Dijkstra loop.
In the Fast-Downward environment the abstract operators are *reversed* operators because they're needed for the reversed Dijkstra search. A reversed operator is an operator which points into the opposite direction. This is done by exchanging preconditions and effects. However, there are some variables which have a defined effect and no precondition. Those operators get split into several unique operators which define the effect value for this variable as well. With this in mind it is quite easy now to check the variable-value-pairs for mutexes. Whenever we reach a state we can check if the state itself together with the preconditions of the actual operator which leads to the state we just came from contradict any mutexes. The needed information is depicted in Figure 3.3. If that's true, they get removed, if not, the search goes on. The reason why we want to remove the states as early as possible is simply because by removing a state we can actually remove up to a whole sub-graph, however there still might be another operator or state which leads into this sub-graph and therefore the extra time we save might be rather small, but the precision gain remains.
There are a few possibilities to optimize this check, however they haven't been implemented yet. For example, the abstract operators can contain additional information about the mutexes. One can pre-calculate which mutexes conflict with the precondition of the operator and only check if any of these mutexes is contained in the abstract state. Since the abstract

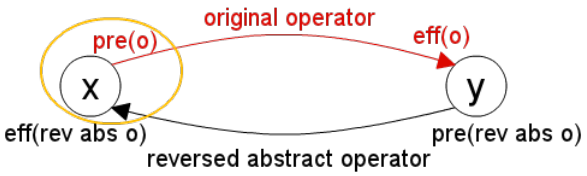

Figure 3.3: This image shows the relation between the original and the reversed abstract operator as well as the data which we need for the validity check, which is shown with an orange circle.

operators always have a precondition variable and a corresponding effect variable, it is even possible to pre-calculate the mutexes which will be broken, depending on the variables of the target state, which is where we come from in the reversed Dijkstra. Some abstract operators might already conflict a mutex themselves even without the additional information of the state. This can happen, when the original operator gets split into several abstract operators and therefore specified. A rather huge performance boost would be possible, if the mutexes are represented as sorted sets internally. Because calculating the intersection between two sorted sets, one being the information about the state and the preconditions of the operator, the second being a mutex, is rather efficient compared to checking the validity of each possible pair.

3.4 Constrained Merge and Shrink

The main focus of this thesis was how to find out how it is possible to use the mutexes in the M&S environment. The core idea, removing invalid states and operators which violate mutexes, stays the same yet there are some differences in the theory as well as the in the implementation. For one, the M&S cycles through merge and shrink steps repeatedly, therefore the question arises. How often and where should the validity be checked. There can be unforeseeable side-effects: By changing one temporary abstraction, the following ones which result after another few merge and shrink steps can be completely different and therefore can wield even lower heuristic values theoretically. Implementation-wise, it is for example rather difficult to backtrack which variable-value-pairs are present in a specific abstract state since the indexes of the states are chosen arbitrary except during the initialisation part of the heuristic where the atomic abstractions get created and the information we have is easily attainable with the data we need but not the other way round (key-value-pair problematic). During the whole project we followed two possible implementations and compared them. Both rely on the principle of *information tagging* and just work with the information which is packed into those tags to avoid the problematic which arises since we don't know the variable-value-pairs of the states. Information tagging is nothing else but adding information to states from the outside. We tagged *labels* onto the states which in this case means that we attached variable-value-pairs to the states.
In the following chapters we'll often use the name *combined* abstraction. This is simply the current working abstraction of the merge and shrink heuristic as described in chapter 2.2.4

3.4.1 Mutex based Labelling

The first idea was to label all abstract states of the M&S abstraction with variable-value-pairs which are in contradict any mutex and remove operators who themselves have preconditions so that the intersection with those tags isn't the empty set. For this approach to work, each pair which together with the current abstract state contradict any mutex gets tagged at the current state. This however is only possible during the initialisation of the atomic abstractions since it is too complex to backtrack all the variables of a state later on in the combined abstraction. This is due to the fact that the data structures are designed for the other way round. Therefore our goal is to remove the operators since we can easily access the preconditions they have by using the tags and this is only possible if we find a way to conserve the information in the tags for as long as possible. Thus we need rules which describe how to propagate those tags to merged or shrinked states.
Our approach was to use simple set operations. During a merge step, we combined the tags of both states. We can unify both informations without introducing new detours as proven in Figure 3.4. During a shrink step however, this disjoint property isn't given any more and therefore we can't just combine the tags that easily. We decided to use the intersection because of two simple facts: First, it is obvious to see that there aren't any new detours introduced because we only conserve the labels which were restraining both original states and second, we haven't found any additional information which is as easy to calculate which isn't introducing errors in any possible case. So this second part is a compromise between precision and efficiency. It would be possible to track the single states inside a combined abstract state however this would defy the whole idea of the abstraction because you nearly have to save the same amount of information as in the actual not-abstracted graph. (In the end you'll have the same amount of states however probably a few less operators.) An example can be seen in Figure 3.5.
With these rules it is possible to track those tags throughout the whole heuristic process and therefore it is possible to check the operators whenever they get created for a new combined abstraction by using a simple intersection of their preconditions and the tags at state where they are attached to. Whenever the intersection isn't empty, which implies, that some precondition is the same as one of the variable-value-pairs in the tags, they operator is invalid and therefore can be removed. The downside of this approach is obviously that the additional memory usage can become quite huge, depending on the number of available mutexes, as well as the fact that it isn't possible to remove states from the abstraction except in the absence of some operators which let the state become unreachable or irrelevant.
A minor but still useful optimisation, which is also implemented, is based on the fact, that in the SAS^+ environment, the variable-value-pairs automatically are mutex to other variable-value-pairs of the same variable. This however doesn't have to be checked with the labels since the operators initially only are applicable where they actually are applicable and in the process of merging they'll never get attached to new ones. In the shrinking step this however might happen and that's why the tags get intersected and not unified. If a label remains attached after the shrinking phase, it implies that all the states which got shrinked into it had this label already and therefore all the operators which are attached to it are consistent to it. Therefore the labels which contain a variable-value-pair of the same variable as the state of the atomic abstraction corresponds to, can be ignored.
One rather big downside of this type of labels is that the label reduction of the operators,

Facts:
Let V be the set of all variables and M be the set of all mutexes,
$v_1 \subset V, v_2 \in V, v_2 \notin v_1$,
s_1 is a state of the combined abstraction with the current varset v_1,
s_2 is a state of the next atomic abstraction which is the atomic abstraction of v_2,
T_i are the tags attached to the state $s_i, \forall i \in \{1, 2\}$,
$\forall t \in T_i$ applies $\exists v \in v_i$ where the pair $\langle t, s_i(v)\rangle$ is in conflict with a mutex $m \in M$,
$v_{12} := v_1 \dot{\cup} v_2$,
$s_{12} := (s_1, s_2)$.
To be shown:
$\forall v \in v_{12}, a := s_{12}(v), \forall t \in T_1 \cup T_2 \Rightarrow \langle a, t\rangle$ is in conflict with a mutex $m \in M$.
Proof:
$\forall t \in T_1 \cup T_2$ applies: $t \in T_1$ or $t \in T_2$
$\Rightarrow \forall t \in T_i, \exists v \in v_i$ where the pair $\langle t, s_i(v)\rangle$ is in conflict with a mutex $m \in M$
$\forall i \in \{1, 2\}$
Because of $v_i \subset v_{12}$ and $s_{12} := (s_1, s_2)$ follows:
$\Rightarrow \exists v \in v_{12}$ where the pair $\langle t, s_{12}(v)\rangle$ is in conflict with a mutex $m \in M\square$

Figure 3.4: Proof, that the union of the labels isn't introducing any unwanted detours.

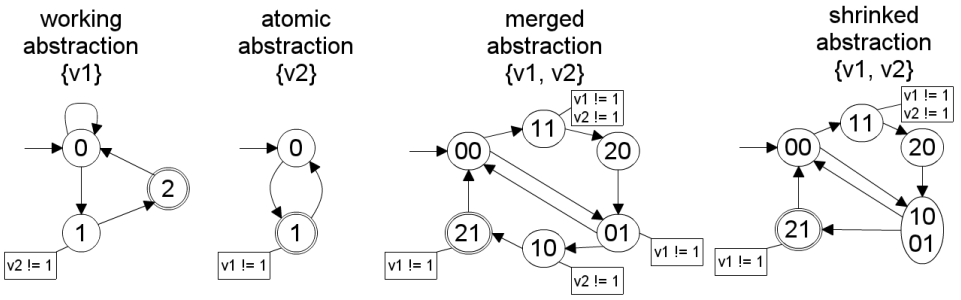

Figure 3.5: This image shows a merge and a shrink step with mutex labeled abstractions.

which is similar to grouping operators, has either to be turned off or the intersections of the preconditions of the grouped operators has to be saved as well and you'd have to put up with implicitly lowering the chance of the algorithm to fire since the size of the intersection of the preconditions can't be greater. We chose the first option and concentrated us on rather comparing different approaches on a higher level and maintaining the bigger "hitting sets" than saving this memory.

3.4.2 "Possible variables" based Labelling

The second approach was basically the exact opposite: Tag all the abstract states with variable-value-pairs which won't conflict any mutexes and based on this information removing all abstract states where one variable essentially gets "impossible" and therefore there isn't any label left of that variable. Again, the only moment where this is possible to initialize is during the creation of the atomic abstractions. It is quite simple to find all possible variable-value pairs and tag those to the given states. It is necessary to tag the variable-value-pair of the abstract state itself or else the check will fail for all states. This is because of the following: are_mutex(<v, x>, <v, y>) $\rightarrow$ true $\forall x!{=}y$. Therefore, if we don't attach the variable-value-pair the state corresponds to, there won't be any variable-value-pair of

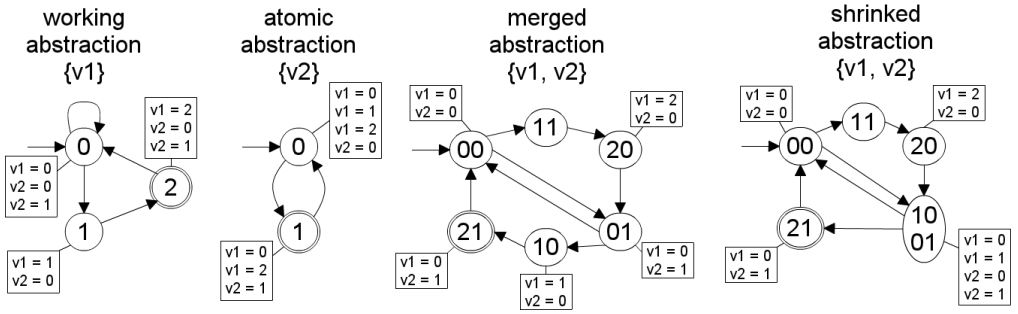

Figure 3.6: This image shows a merge and a shrink step with "possible variables" labeled abstractions.

that specific variable left and thus our check will fire. The remaining question is, how to propagate the tags through the whole process of the M&S heuristic.

We used a similar approach as with the mutex labels and because we also tagged the variable-value-pair of the stated itself onto it, it is possible to use just as simple rules. During the merge step the additional information about the mutexes can be simply added, as shown in Figure 3.4. This however translates to a set intersection in this "possible variables" tagging approach. The same parallel is applicable during the shrink step. For mutex based labelling it was a set intersection, for this kind of labelling, we need a union. If we don't chose a set which has at least all the elements of the union we can introduce new detours to at least one of the states which get shrunk together. However we want it to be as small as possible since the smaller the set the bigger chance of finding a variable-set which is empty and therefore the bigger chance of finding an illegal state. Again, with this approach it is not the case that all invalid states get removed from the abstract graph, but it is a viable trade-off. The rules are shown in an example in Figure 3.6.d

The tags will use a fair amount of space in the beginning but it won't increase too much during the whole heuristic calculation because at some point the number of states in the abstract graph won't increase any more (or at least not by a lot) whereas the number of possible variables per state will decrease. At the end the tags will actually describe which variable-value-pairs are inside a state, but this still contains less information than tracking the actual states in the abstract states.

In the implementation, the way the states get removed is rather simple. As soon as the states get created they already get checked for their validity. If they're invalid all the operators coming from or leading to this state won't get created. Since the creation of the new operators in the combined abstraction comes after the creation of the states, all invalid states won't have any operator attached to them. After this the M&S algorithm performs a pruning step anyway and in this step, those invalid and thus now unreachable states, get removed. This proofed to be way more efficient than the implementation of the mutex labelling but this might really just be the implementation as well as the fact that in the current implementation the tags or ordered by variable and therefore it's only needed to check the size of the variable-sets instead of calculating intersections for every operator.

4 Evaluation

4.1 Setup

The evaluation is based on the benchmark sets included in the Fast-Downward repository. There are older problem domains like airport, blocks or gripper as well as the newer IPC 11 bundle. Out of all domains, only the ones for optimal solving were chosen. Under these candidates there were a few domains (e.g. openstacks-strips) which used all the mutexes during the conversion from *STRIPS* to SAS^+ problems and therefore were not really useful either. The remaining domains were used and tested extensively. All the heuristics were tested with standard settings with the exception of the label reduction which got removed for the mutex label based M&S implementation. The tested heuristics are the standard pdb implementation (pdb), the constrained pdb (cpdb), the newer ipdb implementation (ipdb), ipdb which uses constrained pdbs internally (cipdb) as well as the standard M&S (ms), M&S with mutex labels (ms-m), M&S with 'possible variables' labels (ms-p) and M&S with both kinds of labels (ms-mp).
All tasks were given a maximum of 15 minutes and 2 GB memory. The problems were run on a CPU with 3.2 GHZ (dual core but the planner is not multi-threaded) and the PC has 4GB RAM in total thus as long as only one problem is running at a time it basically has full control over one kernel as well as the maximum 2GB RAM at all times and shouldn't be disturbed too much by other programs or the OS itself.

4.2 Results

Since the implementations of the constrained heuristics aren't fully optimized yet but rather held in a state which fulfils the purpose of the 'prove of concept' the heuristic time values might change for the better in future versions but nevertheless it is an important measurement of the efficiency. The same applies to the used memory values. The most important nominal values however are the initial heuristic value as well as the general size of the search time since both indicate how much actually was gained compared to the standard implementations. The total time used as well as the coverage are also shown. All the numbers can be found in the appendix (5).

coverage	**pdb**	**cpdb**	**ipdb**	**cipdb**	**ms**	**ms-m**	**ms-p**	**ms-mp**	**out of**
airport	**21**	**21**	20	20	18	12	12	12	50
blocks	21	23	**26**	**26**	21	25	24	24	26
depot	**6**	**6**	**6**	5	5	4	5	3	10
driverlog	10	10	**13**	**13**	**13**	**13**	**13**	**13**	15
freecell	16	17	**19**	10	9	8	13	7	25
gripper	7	7	7	7	7	7	7	7	8
pipesworld-tankage	11	**12**	**12**	7	7	8	9	8	14
trucks-strips	6	7	7	**10**	6	6	6	6	10
total	98	103	**110**	98	86	83	89	80	158
barman	**4**	**4**	**4**	**4**	**4**	0	**4**	0	8
floortile	2	3	2	**8**	2	5	2	5	10
scanalyzer	**6**	**6**	**6**	**6**	**6**	5	**6**	**6**	10
sokoban	**10**	**10**	**10**	**10**	0	1	1	1	10
tidybot	**10**	0	**10**	**10**	0	0	0	0	10
woodworking	5	5	0	0	5	**7**	6	**7**	10
total IPC 11	37	28	32	**38**	17	18	19	19	58

Table 4.1: This table shows the coverage of the different implementations. The upper half of the table contains the results of older benchmarks whereas the lower half contains the results of the IPC 11 benchmarks.

Even though the best coverage (shown in Figure 4.1) wasn't necessarily the goal since the set resource boundaries favour the optimized algorithms, it is nice to see that in most cases the number of solved problems of the improved versions either were the same or even higher. What's also notable is that there are huge differences between the domains, even if you only compare the three label based improvements of the M&S. But even if the coverage might be rather low sometimes, it doesn't mean that the wielded results regarding the heuristic values are worse, but rather that the memory ran out too early. Quite notable for example are the increase of the coverage in the floortile domain as well as in the blocks domain and the decrease of the coverage in domains like airport and freecell.
In Figure 4.2, the results of the latter half of the tests in the blocks domain are shown. As we can see, in this domain the values get improved in most cases which is a very promising result. This is probably due to a good ratio of mutexes to variables as well as the simplicity of the domain which results in rather few operators as well as the fact that there aren't too many optimal solutions compared to other domains like gripper, where the heuristic values have improved a lot as well but due to the special kind of parallelism in the domain, the search can't really gain a huge boost through higher heuristics.
As seen in the first half of the results in the airport domain (Table 4.3), there aren't really any improvements at all. The coverage of the label-based M&S approaches are even quite a bit lower compared to the standard M&S. That's probably due to the fact, that the airport domain has quite a lot of variables and therefore the additional memory which is needed to save all the tags is too big as well as the time needed for the set unions and intersection accumulate quite soon. The same goes for the freecell domain which did show rather weak results compared to other domains. To improve the coverage, it'd be necessary to improve the additional memory usage as well as the time needed for the checks. There aren't any improvements of the initial heuristic values either except for one. This, however, doesn't mean that the heuristic values are the same for the whole graph and the whole search process still were sped up. To improve this, the only option for M&S would be to find better rules for propagating the labels during the whole heuristic process or use a different approach.

initial_h_value	pdb	cpdb	ipdb	cipdb	ms	ms-m	ms-p	ms-mp
8-0	12	14	12	12	16	**18**	**18**	**18**
8-1	8	12	10	10	15	**20**	**20**	**20**
8-2	12	14	14	14	**16**	**16**	**16**	**16**
9-0	10	14	14	14	16	**24**	**24**	**24**
9-1	8	14	14	14	16	24	24	**26**
9-2	10	12	16	16	16	**24**	22	**24**
10-0	None	14	18	18	None	**24**	**24**	**24**
10-1	10	14	18	18	16	**22**	**22**	**22**
10-2	None	16	18	18	None	**24**	**24**	**24**
11-0	None	None	16	16	None	**24**	**24**	**24**
11-1	None	None	20	20	None	**22**	None	None
11-2	None	None	**16**	**16**	None	None	None	None

Table 4.2: This table shows the initial heuristic values which were returned of the different implementations using the blocks benchmarks.

initial_h_value	pdb	cpdb	ipdb	cipdb	ms	ms-m	ms-p	ms-mp
1	8	8	8	8	8	8	8	8
2	9	9	9	9	9	9	9	9
3	17	17	17	17	17	17	17	17
4	20	20	20	20	20	20	20	20
5	21	21	21	21	21	21	21	21
6	41	41	41	41	41	41	41	41
7	41	41	41	41	41	41	41	41
8	43	43	**62**	**62**	43	45	None	None
9	44	44	**71**	**71**	37	None	None	None
10	18	18	18	18	18	18	18	18

Table 4.3: The initial heuristic values for the first half of the airport benchmarks.

4.2.1 PDB compared to cPDB

Overall, the results are encouraging. The heuristic values improved in several domains but not in all which isn't unusual in the AI. Compared to the standard pattern database implementation, the constrained one actually improved quite a bit, but, as for now, it's implemented rather inefficiently. However if we optimized it further, it could become rather competitive. For example the structure of the mutexes internally could be changed so that the groups could be interpreted as a mutex group again. An intersection of two sorted lists has a runtime of $O(m + n)$. The check with every possible pair however is $O(n^2)$ which is quite a bit slower. The reason why this would probably have a rather huge impact is, because in the pattern database, the part where the constraints have to get checked is called rather often and therefore slows down the code quite a bit as of now. Of course the current implementation of mutexes has it's advantages as well. Another possibility would be to pre-calculate more information about the mutexes regarding the operators and store it into the *match-tree*, a data structure which is used to efficiently find all applicable operators for a given state. There are also different approaches where the checks of the operators could be ignored in example. This might yield a huge performance boost in the current implementation but the question which remains is how much precision would we lose as well as if the performance boost in an optimized version would justify it.

4.2.2 iPDB compared to ciPDB

The iPDB and ciPDB implementations had nearly always similar results. Only in a few domains mainly the iPDB was a bit better since the PDBs are more efficient to calculate than the cPDBs. Thus we'd suggest to stick with the standard version for now, but this might change in the future. We assume this is due to the patterns themselves which are rather small and therefore the hitting sets for the mutexes is rather diminutive as well which results in only small improvements opposed to a rather huge investment.
If we look at both iPDBs compared to PDBs and cPDBs however, we see huge advantages, as long as the standart settings won't let the iPDB search for a set of patterns nearly endlessly.

4.2.3 The M&S Approaches

Comparing the standard M&S with the extended versions, it is always a lot faster in the calculation of the heuristic and the improvements might not be as visible as it is for the constrained pattern database, but they've wield better results in several domains. One huge exception are the problems which need a lot of space for the additional data so that it exceeded the memory limit before the heuristic even started. Regarding the current speed, the ms-p is way faster than the ms-m while both wield similar results regarding the heuristic with a similar usage of memory. However the difference in time might be due to the degree of optimisation in the code. The main parts are just as efficient, however the checks are at different locations and thus it might yield quite a performance boost for one or another, if we replaced them. While both single label implementations have their pros and cons (depending on the domain) the combined version wasn't as successful, mainly due to the high resources which are needed for the labelling. Also the performance isn't great because it has to do double the work at every step. Therefore we think that both ms-m and ms-p might have some success whereas ms-mp probably is too slow compared to the gained precision.

5 Conclusions

The main goal was to see if the usage of mutexes actually improves the abstraction heuristics without producing too much overhead. There were most definitely improvements in several domains, but the overhead might become a problem unless the implementations get further optimized. There are already certain ideas which start at simpler tasks like repositioning the newly introduced checks. Another problem which has been highlighted is the limited use of the whole idea in general due to the fact that in an SAS^+ environment, not every problem will contain any additional 'at-most-one-atom' mutexes since they're already used during the conversion itself. This fact however is not a problem since the preprocessor knows how many mutexes are left over and whenever no mutexes are useful any more it can change to a different heuristic before starting a constrained approach.
The constrained pattern database which is based on the idea of Haslum et al. [2005] outperformed the original implementation in most of the viable domains regarding the heuristic values but as already mentioned at a currently rather high price in runtime. The idea is based on 'at-most-one-atom' mutexes which can be calculated rather effectively. As soon as a state itself contradicts a mutex, it gets removed from the abstraction and as soon as an operators precondition combined with the knowledge of the source state contradicts a mutex, the operator can be removed. We ported this idea over to M&S and derived rules and algorithms which led to a similar result for this respective heuristic.
Now, there is still the question, if these basic ideas stay useful regarding the efficiency versus the information gain. We have showed that they definitely can improve the results to some degree but as of right now the costs are a bit too high. So, the central point lies in the question, how far it can be optimized. Maybe there are still more efficient ways of implementing a mutex based concept into M&S which we haven't found yet. We tried out two different approaches which were quite similar regarding the results besides the fact that both approaches were nearly the opposite of each other except that both are based on state-labelling.
What's most definitely interesting is the development of the iPDB and how it changes it's effectiveness as soon as constrained PDBs get used. We already have a few numbers from test runs in this thesis as well but we only scratched the surface because it is rather difficult to foresee how the iPDB reacts to usage of cPDB. For now the results were, as expected,

rather mediocre, but depending on the settings of the iPDB and the problem size, this might change.
All in all, this general idea which is based on mutexes shows potential to a certain degree which isn't overwhelming but nevertheless satisfactory, at least in some domains, and therefore shouldn't be completely disregarded. The next step will involve to find out, why the results are that fluctuating and what the reason is behind it. If a simple reason can be found, maybe the effectiveness can be increased quite a bit and therefore this concept can become quite valuable.

Bibliography

[Bäckström, 1992] Christer Bäckström. Equivalence and tractability results for SAS^+ planning. In *KR*, pages 126–137. Morgan Kaufmann, 1992.

[Culberson and Schaeffer, 1998] Joseph C. Culberson and Jonathan Schaeffer. Pattern databases. *Computational Intelligence*, 14(3):318–334, 1998.

[Edelkamp, 2001] Stefan Edelkamp. Planning with pattern databases. In *Proceedings of the 6th European Converence on Planning*, pages 13–24, 2001.

[Fikes and Nilsson, 1971] Richard Fikes and Nils J. Nilsson. Strips: A new approach to the application of theorem proving to problem solving. *Artif. Intell.*, 2(3/4):189–208, 1971.

[Haslum et al., 2005] Patrik Haslum, Blai Bonet, and Hector Geffner. New admissible heuristics for domain-independent planning. In *AAAI*, pages 1163–1168. AAAI Press / The MIT Press, 2005.

[Helmert, 2004] Malte Helmert. A planning heuristic based on causal graph analysis. In *ICAPS*, pages 161–170. AAAI, 2004.

[Helmert, 2006] Malte Helmert. The fast downward planning system. *J. Artif. Intell. Res. (JAIR)*, 26:191–246, 2006.

[Helmert et al., 2007] Malte Helmert, Patrik Haslum, and Jörg Hoffmann. Flexible abstraction heuristics for optimal sequential planning. In *ICAPS*, pages 176–183. AAAI, 2007.

[McDermott et al., 1998] Drew McDermott, Malik Ghallab, Adele Howe, Craig Knoblock, Ashwin Ram, Manuela Veloso, Daniel Weld, and David Wilkings. Pddl - the planning domain definition language. In *Technical Report CVC TR98003/DCS TR1165*, 1998.

[Pearl, 1984] Judea Pearl. *Heuristics - intelligent search strategies for computer problem solving.* Addison-Wesley series in artificial intelligence. Addison-Wesley, 1984.

[Sievers et al., 2012] Silvan Sievers, Manuela Ortlieb, and Malte Helmert. Efficient implementation of pattern database heuristics for classical planning. In *Proceedings of the Fifth Annual Symposium on Combinatorial Search*, pages 105–111, 2012.

Appendix A: Testresults

It is important to note that the search time of all pdb-implementations can't be below 10 ms due to the measurement.

labels

pdb	standart pattern database
cpdb	constrained pattern database
ipdb	iterative pattern database using pdbs internally
cipdb	iterative pattern database using cpdb internally
ms	standart merge and shrink
ms-m	merge and shrink with mutex labels enabled
ms-p	merge and shrink with 'possible variables' labels enabled
ms-mp	merge and shrink with both kinds of labels enabled

measurements

1. initial_h_value
2. memory
3. heuristic time
4. search time
5. total time
6. coverage

initial_h_value

airport standart

initial_h_value	pdb	cpdb	ipdb	cipdb	ms	ms-m	ms-p	ms-mp
1	8	8	8	8	8	8	8	8
2	9	9	9	9	9	9	9	9
3	17	17	17	17	17	17	17	17
4	20	20	20	20	20	20	20	20
5	21	21	21	21	21	21	21	21
6	41	41	41	41	41	41	41	41
7	41	41	41	41	41	41	41	41
8	43	43	**62**	**62**	43	45	None	None
9	44	44	**71**	**71**	37	None	None	None
10	18	18	18	18	18	18	18	18
11	21	21	21	21	21	21	21	21
12	39	39	39	39	39	39	39	39
13	37	37	37	37	37	37	37	37
14	43	43	**60**	**60**	43	None	None	None
15	41	41	**58**	**58**	41	None	None	None
16	59	59	**79**	**79**	38	None	None	None
17	**59**	**59**	None	None	None	None	None	None
18	None	None	None	None	None	None	None	None
19	58	58	**90**	**90**	None	None	None	None
20	None	None	None	None	None	None	None	None

airport half MUC

initial_h_value	pdb	cpdb	ipdb	cipdb	ms	ms-m	ms-p	ms-mp
1	54	54	**101**	**101**	54	None	None	None
2	54	54	**148**	**148**	None	None	None	None
3	None	None	None	None	None	None	None	None
4	None	None	None	None	None	None	None	None
5	None	None	None	None	None	None	None	None
6	None	None	None	None	None	None	None	None
7	None	None	None	None	None	None	None	None
8	None	None	None	None	None	None	None	None
9	None	None	None	None	None	None	None	None
10	None	None	None	None	None	None	None	None
11	None	None	None	None	None	None	None	None
12	None	None	None	None	None	None	None	None
13	None	None	None	None	None	None	None	None
14	None	None	None	None	None	None	None	None
15	None	None	None	None	None	None	None	None

airport MUC

initial_h_value	pdb	cpdb	ipdb	cipdb	ms	ms-m	ms-p	ms-mp
1	60	60	**109**	**109**	51	None	None	None
2	None	None	None	None	None	None	None	None
3	None	None	None	None	None	None	None	None
4	None	None	None	None	None	None	None	None
5	None	None	None	None	None	None	None	None
6	None	None	None	None	None	None	None	None
7	None	None	None	None	None	None	None	None
8	None	None	None	None	None	None	None	None
9	None	None	None	None	None	None	None	None
10	None	None	None	None	None	None	None	None
11	None	None	None	None	None	None	None	None
12	None	None	None	None	None	None	None	None
13	None	None	None	None	None	None	None	None
14	None	None	None	None	None	None	None	None
15	None	None	None	None	None	None	None	None

barman ipc11

initial_h_value	pdb	cpdb	ipdb	cipdb	ms	ms-m	ms-p	ms-mp
1	6	6	**19**	**19**	17	None	17	None
2	6	6	**20**	**20**	17	None	17	None
3	6	6	**20**	**20**	17	None	17	None
4	6	6	**20**	**20**	17	None	17	None
5	None	None	None	None	None	None	None	None
6	None	None	None	None	None	None	None	None
7	None	None	None	None	None	None	None	None
8	None	None	None	None	None	None	None	None

blocks

initial_h_value	pdb	cpdb	ipdb	cipdb	ms	ms-m	ms-p	ms-mp
1-0	1	1	1	1	1	1	1	1
2-0	2	2	2	2	2	2	2	2
4-0	6	6	6	6	6	6	6	6
4-1	**10**	**10**	4	4	**10**	**10**	**10**	**10**
4-2	6	6	6	6	6	6	6	6
5-0	**12**	**12**	6	6	**12**	**12**	**12**	**12**
5-1	**10**	**10**	6	6	**10**	**10**	**10**	**10**
5-2	**16**	**16**	8	8	**16**	**16**	**16**	**16**
6-0	10	10	10	10	**12**	**12**	**12**	**12**
6-1	10	10	10	10	10	10	10	10
6-2	13	**20**	10	10	18	**20**	**20**	**20**
7-0	12	18	12	12	17	**20**	**20**	**20**
7-1	10	16	10	10	16	**22**	**22**	**22**
7-2	10	14	10	10	16	**20**	**20**	**20**
8-0	12	14	12	12	16	**18**	**18**	**18**
8-1	8	12	10	10	15	**20**	**20**	**20**
8-2	12	14	14	14	**16**	**16**	**16**	**16**
9-0	10	14	14	14	16	**24**	**24**	**24**
9-1	8	14	14	14	16	24	24	**26**
9-2	10	12	16	16	16	**24**	22	**24**
10-0	None	14	18	18	None	**24**	**24**	**24**
10-1	10	14	18	18	16	**22**	**22**	**22**
10-2	None	16	18	18	None	**24**	**24**	**24**
11-0	None	None	16	16	None	**24**	**24**	**24**
11-1	None	None	20	20	None	**22**	None	None
11-2	None	None	**16**	**16**	None	None	None	None

depot

initial_h_value	pdb	cpdb	ipdb	cipdb	ms	ms-m	ms-p	ms-mp
1	8	8	8	8	**10**	**10**	**10**	**10**
2	6	6	14	14	**15**	**15**	**15**	**15**
3	10	10	16	None	15	**18**	17	**18**
4	10	10	14	**17**	None	None	15	None
5	None	None	None	None	None	None	None	None
6	None	None	None	None	None	None	None	None
7	8	8	14	**17**	12	14	13	None
8	None	None	None	None	None	None	None	None
9	None	None	None	None	None	None	None	None
10	8	8	14	**18**	12	None	None	None

driverlog

initial_h_value	pdb	cpdb	ipdb	cipdb	ms	ms-m	ms-p	ms-mp
1	**7**	**7**	3	3	**7**	**7**	**7**	**7**
2	19	19	19	19	19	19	19	19
3	**12**	**12**	11	11	**12**	**12**	**12**	**12**
4	13	13	15	15	**16**	**16**	**16**	**16**
5	15	17	14	14	**18**	**18**	**18**	**18**
6	6	6	8	**11**	**11**	**11**	**11**	**11**
7	8	8	11	11	**13**	**13**	**13**	**13**
8	None	None	**21**	19	20	**21**	20	**21**
9	6	6	18	18	**22**	**22**	**22**	**22**
10	10	10	14	14	**17**	**17**	**17**	**17**
11	8	8	15	15	15	**16**	15	**16**
12	None	None	None	None	None	None	None	None
13	None	None	**20**	**20**	19	19	19	19
14	None	None	18	20	21	**23**	21	**23**
15	None	None	None	None	None	None	None	None

floortile ipc11

initial_h_value	pdb	cpdb	ipdb	cipdb	ms	ms-m	ms-p	ms-mp
1	20	25	20	26	21	**38**	25	**38**
2	23	25	23	25	24	**33**	24	**33**
3	None	None	None	26	None	**44**	None	**44**
4	None	24	None	24	None	41	None	**42**
5	None	None	None	33	None	**43**	None	**43**
6	None	None	None	**30**	None	None	None	None
7	None	None	None	**33**	None	None	None	None
8	None	None	None	**36**	None	None	None	None
9	None	None	None	None	None	None	None	None
10	None	None	None	None	None	None	None	None

freecell v1

initial_h_value	pdb	cpdb	ipdb	cipdb	ms	ms-m	ms-p	ms-mp
1	6	6	**8**	**8**	**8**	**8**	**8**	**8**
2	6	6	**10**	**10**	**10**	None	**10**	None
3	9	9	**12**	9	11	None	None	None
4	8	8	**11**	None	None	None	None	None
5	7	7	**12**	None	None	None	11	None
6	None	None	None	None	None	None	None	None
7	None	None	None	None	None	None	None	None
8	None	None	None	None	None	None	None	None
9	None	None	None	None	None	None	None	None
10	None	None	None	None	None	None	None	None

freecell v2

initial_h_value	pdb	cpdb	ipdb	cipdb	ms	ms-m	ms-p	ms-mp
2-1	4	4	**9**	**9**	None	**9**	**9**	**9**
2-2	6	6	**8**	**8**	**8**	**8**	**8**	**8**
2-3	5	5	7	7	**8**	**8**	**8**	**8**
2-4	5	5	**8**	**8**	**8**	**8**	**8**	**8**
2-5	5	5	**9**	**9**	None	**9**	**9**	**9**
3-1	4	4	**9**	None	**9**	None	**9**	None
3-2	6	6	**12**	None	None	None	9	None
3-3	4	5	4	4	None	None	**8**	**8**
3-4	6	6	**10**	None	**10**	**10**	None	None
3-5	6	7	6	6	9	**10**	**10**	None
4-1	None	None	**13**	None	None	None	None	None
4-2	6	6	**12**	None	None	None	10	None
4-3	None	**6**	None	None	None	None	None	None
4-4	None	None	**12**	None	None	None	None	None
4-5	None	None	**10**	None	None	None	None	None

gripper

initial_h_value	pdb	cpdb	ipdb	cipdb	ms	ms-m	ms-p	ms-mp
1	**11**	**11**	5	9	**11**	**11**	**11**	**11**
2	**17**	**17**	7	13	**17**	**17**	**17**	**17**
3	16	16	9	17	18	19	**21**	19
4	10	20	11	21	14	**23**	14	**23**
5	12	24	13	**25**	15	**25**	15	**25**
6	12	24	15	**29**	17	**29**	17	**29**
7	12	24	17	**33**	18	**33**	18	**33**
8	None	None	None	None	None	None	None	None

pipesworld-tankage

initial_h_value	pdb	cpdb	ipdb	cipdb	ms	ms-m	ms-p	ms-mp
1	5	5	5	5	5	5	5	5
2	12	12	12	12	12	12	12	12
3	7	7	7	7	**8**	**8**	**8**	**8**
4	9	9	9	9	**11**	**11**	**11**	**11**
5	7	7	7	7	**8**	**8**	**8**	**8**
6	9	9	9	9	9	**10**	**10**	**10**
7	**8**	**8**	**8**	**8**	None	**8**	**8**	**8**
8	7	**10**	9	None	None	None	9	None
9	None	None	None	None	None	None	None	None
10	None	**17**	16	None	None	None	None	None
11	6	6	**10**	None	8	8	8	None
12	8	8	**12**	None	None	None	None	None
13	10	10	**12**	None	None	None	None	None
14	None	None	None	None	None	None	None	None

scanalyzer ipc11

initial_h_value	pdb	cpdb	ipdb	cipdb	ms	ms-m	ms-p	ms-mp
1	**13**	**13**	12	12	**13**	**13**	**13**	**13**
2	**22**	**22**	18	18	**22**	**22**	**22**	**22**
3	**26**	**26**	18	18	24	**26**	**26**	**26**
4	24	24	24	24	24	24	24	24
5	**30**	**30**	**30**	**30**	28	None	29	29
6	**30**	**30**	24	24	25	26	26	26
7	None	None	None	None	None	None	None	None
8	None	None	None	None	None	None	None	None
9	None	None	None	None	None	None	None	None
10	None	None	None	None	None	None	None	None

sokoban ipc11

initial_h_value	pdb	cpdb	ipdb	cipdb	ms	ms-m	ms-p	ms-mp
1	3	5	1	1	None	**7**	**7**	**7**
2	4	12	5	**13**	None	None	None	None
3	8	**10**	**10**	**10**	None	None	None	None
4	17	18	**24**	**24**	None	None	None	None
5	12	**23**	15	15	None	None	None	None
6	13	13	**15**	**15**	None	None	None	None
7	6	6	**8**	**8**	None	None	None	None
8	2	**8**	3	3	None	None	None	None
9	7	**9**	7	7	None	None	None	None
10	**8**	**8**	**8**	**8**	None	None	None	None

tidybot ipc11

initial_h_value	pdb	cpdb	ipdb	cipdb	ms	ms-m	ms-p	ms-mp
1	**4**	None	**4**	**4**	None	None	None	None
2	7	None	**9**	**9**	None	None	None	None
3	4	None	**7**	**7**	None	None	None	None
4	4	None	**8**	**8**	None	None	None	None
5	6	None	**9**	**9**	None	None	None	None
6	6	None	**10**	**10**	None	None	None	None
7	6	None	**7**	**7**	None	None	None	None
8	6	None	**9**	**9**	None	None	None	None
9	6	None	**8**	**8**	None	None	None	None
10	6	None	**10**	**10**	None	None	None	None

trucks-strips

initial_h_value	pdb	cpdb	ipdb	cipdb	ms	ms-m	ms-p	ms-mp
1	**13**	**13**	11	11	**13**	**13**	**13**	**13**
2	13	13	14	14	**17**	**17**	**17**	**17**
3	10	15	11	**17**	12	13	14	14
4	10	14	13	**20**	11	13	12	13
5	11	15	15	**23**	11	14	13	14
6	None	None	None	**26**	None	None	None	None
7	12	12	**20**	**20**	12	12	12	12
8	None	13	15	**22**	None	None	None	None
9	None	None	None	**26**	None	None	None	None
10	None	None	None	**29**	None	None	None	None

woodworking ipc11

initial_h_value	pdb	cpdb	ipdb	cipdb	ms	ms-m	ms-p	ms-mp
1	130	130	None	None	**195**	**195**	**195**	**195**
2	155	155	None	None	**205**	**205**	**205**	**205**
3	115	115	None	None	185	185	**190**	185
4	195	195	None	None	**275**	**275**	**275**	**275**
5	120	120	None	None	220	220	**230**	**230**
6	None	None	None	None	None	250	None	**255**
7	None	None	None	None	None	**180**	None	**180**
8	None	None	None	None	None	None	None	None
9	None	None	None	None	None	None	None	None
10	None	None	None	None	**270**	None	**270**	None

memory

airport standart

memory	pdb	cpdb	ipdb	cipdb	ms	ms-m	ms-p	ms-mp
1	7524	7524	**3088**	**3088**	**3088**	**3088**	3224	3224
2	7480	7480	3220	3216	**3088**	**3088**	3356	3356
3	6188	6188	**3476**	4276	3604	4012	8892	9032
4	7612	7616	4012	4008	**3616**	3748	5196	5204
5	6940	6944	4912	4864	**3748**	3880	5864	5864
6	8360	8280	7676	23560	**6896**	10044	62896	63528
7	8360	8280	7668	23564	**6808**	10024	62900	62876
8	9556	**9548**	11888	11836	157012	448656	None	None
9	36408	36400	19564	**18092**	206080	None	None	None
10	6580	6580	4400	4404	**3876**	3880	5660	5732
11	6508	6512	5612	5692	**3880**	4012	7048	7052
12	7356	**7256**	9812	31400	7804	11148	80136	80836
13	**7252**	7256	11120	565408	7628	10648	86784	86832
14	**12100**	12112	15488	15492	175192	None	None	None
15	**11860**	11872	18744	18812	169892	None	None	None
16	107484	107236	**31264**	31288	416428	None	None	None
17	1305144	**1304888**	None	None	2080780	None	2097152	2092584
18	2084736	2082760	None	None	**1917376**	None	2097152	2097152
19	1362000	1361732	30808	**27932**	2083324	None	2085780	2084892
20	**2083176**	2084104	None	None	2096404	None	2097152	2094776

airport half MUC

memory	pdb	cpdb	ipdb	cipdb	ms	ms-m	ms-p	ms-mp
1	**44420**	44436	599336	1074132	632620	None	2087724	2086076
2	**740352**	740384	874604	1510824	1833936	None	2088420	2084504
3	2082992	2085084	None	None	**1831516**	None	2092320	2086964
4	**2083640**	2085200	None	None	2084356	None	2084188	2085616
5	2084948	2085252	None	None	**1887548**	1925732	2091044	2090956
6	2083220	2083920	None	None	**1804996**	2067552	2088544	2088048
7	2085700	**2083924**	None	None	2097152	2094288	2088024	2087140
8	**2083828**	2085412	None	None	2096320	2097020	2090244	2095128
9	**2083756**	2084684	None	None	2097144	2097144	2087120	2091500
10	2087456	2087060	None	None	2097108	**1723016**	2088328	2086224
11	2084688	2084028	None	None	2096356	**1828804**	2086144	2089192
12	2084164	2086804	None	None	2053596	**1972852**	2097064	2097020
13	2084432	2082980	2083604	None	2084316	**1997128**	2097128	2097100
14	2085088	2086828	2086812	2085648	**2010908**	2097140	2097152	2097020
15	2083556	2084088	2083084	**2082772**	2093136	2085044	2097152	2091748

airport MUC

memory	pdb	cpdb	ipdb	cipdb	ms	ms-m	ms-p	ms-mp
1	83468	**83448**	1126440	1126460	795656	None	2089876	2086908
2	2085352	2087256	None	None	2096588	None	2089092	**2082900**
3	2090928	**2084656**	None	None	None	None	2089424	2087960
4	**2083156**	2085904	None	None	2088956	2097120	2093712	2097128
5	2086760	2083632	None	None	**2050428**	None	2097092	2097148
6	**2083860**	2084916	None	None	None	None	2097084	2097140
7	2087928	2085552	None	None	**1931108**	2097116	2097040	2085676
8	2085796	2087908	None	None	**1865860**	2031048	2097124	2097068
9	2085180	2086916	None	None	**1784288**	None	2097092	2097124
10	2085388	2087444	None	None	**2010612**	2048988	2090132	2087272
11	2087780	2088572	None	None	2093072	2094004	2087864	**2085176**
12	**2085708**	2088476	None	None	None	2091528	2097152	2097152
13	2086852	2091828	2090648	2083316	**2030944**	2053540	2097152	2097152
14	2083980	2085032	2089136	**2083164**	None	None	2097152	2097008
15	2095604	2088872	2086532	**2083720**	2097084	2096800	2097112	2097016

barman ipc11

memory	pdb	cpdb	ipdb	cipdb	ms	ms-m	ms-p	ms-mp
1	605572	605572	**484280**	484292	1899632	None	2034856	None
2	601872	601872	**464720**	464736	647416	None	748088	None
3	601876	601876	**503080**	503084	647336	None	741692	None
4	601876	601876	464720	**464640**	647400	None	740640	None
5	2083172	2086012	None	None	2085716	2089728	2097028	**1867884**
6	2084248	2084776	None	None	2072080	2097000	**1994872**	2086656
7	2082776	2084356	None	None	2095684	**1985732**	2097124	2084664
8	2084264	2083212	2083188	2082756	2097068	2068268	2097044	**1845364**

blocks

memory	pdb	cpdb	ipdb	cipdb	ms	ms-m	ms-p	ms-mp
1-0	2956	2956	2956	2956	2956	2956	2956	2956
2-0	2956	2956	2956	2956	2956	2956	2956	2956
4-0	3088	3088	3416	**3084**	4248	3484	3480	3876
4-1	3088	3088	**2956**	**2956**	4224	3484	3476	3880
4-2	3088	3088	3416	**3084**	4228	3484	3480	3880
5-0	5440	4904	**2956**	**2956**	15292	9684	8136	14528
5-1	5440	4904	**2956**	**2956**	14896	9684	8136	14528
5-2	5440	4904	**2956**	**2956**	15048	9684	8136	14528
6-0	7912	5188	**3084**	**3084**	32484	82656	53200	100400
6-1	7908	5188	**3084**	**3084**	30292	83788	53276	100764
6-2	7912	5188	**3084**	**3084**	39032	81464	53276	100936
7-0	8624	5268	**3084**	3088	72372	146024	83164	173224
7-1	8624	5268	**3340**	3344	42656	144580	82888	172280
7-2	8624	5268	**3084**	3088	54488	140656	82308	170156
8-0	8668	5472	**3216**	**3216**	73164	190168	135764	214864
8-1	8668	5484	**3568**	3576	78296	156064	109088	204816
8-2	8668	5472	**3216**	**3216**	74660	180416	123488	213664
9-0	138572	73900	**18352**	18360	127424	199468	193584	248432
9-1	20900	6948	**3216**	**3216**	139652	171840	165148	296128
9-2	20768	9396	**3636**	**3636**	137420	203432	216100	312004
10-0	2084032	1381176	**203624**	**203624**	2082820	644444	822344	413940
10-1	2030380	1078080	**65392**	**65392**	1613412	713956	417880	790356
10-2	2084352	1385660	**104988**	**104988**	2083924	571568	542432	631672
11-0	2083220	2085788	**58524**	**58524**	2083380	1242060	851644	535844
11-1	2085576	2089232	98432	**98428**	2085744	2047740	2084788	2083392
11-2	2083328	1956672	27148	**27144**	2083228	2082848	2083432	None

depot

memory	pdb	cpdb	ipdb	cipdb	ms	ms-m	ms-p	ms-mp
1	7332	7132	**3088**	**3088**	**3088**	3884	3880	4264
2	7708	**7020**	7200	7156	14352	132640	87244	171748
3	43476	39080	**11264**	None	269492	337956	568896	662572
4	545964	423956	111596	**41668**	2089872	None	1180708	None
5	2083604	2083448	**2083376**	None	2083616	None	2093956	None
6	**2082844**	2083744	2083004	None	2095312	None	2096788	None
7	59360	48204	**10876**	39232	357868	883120	409504	None
8	2083680	2084720	None	None	1907540	None	**1865736**	None
9	2083880	**2083540**	None	None	2087748	None	None	None
10	491932	459692	143624	**133180**	1709092	None	1970836	None

driverlog

memory	pdb	cpdb	ipdb	cipdb	ms	ms-m	ms-p	ms-mp
1	3268	3268	**2960**	**2960**	4536	7108	12308	13960
2	5044	4580	3100	**3088**	11756	23792	53088	61380
3	4896	4716	**3088**	**3088**	11672	23624	52984	61264
4	9408	8932	4164	**3576**	24348	36440	66888	82784
5	12340	11736	**4016**	**4016**	32388	43256	101540	105176
6	7412	7400	**3356**	8784	55492	55788	138920	152508
7	12556	12556	**3356**	4792	53228	68448	142588	173272
8	2082904	2085212	**12156**	14424	61488	116220	157952	220040
9	1766024	1766024	**5884**	**5884**	54424	131304	172784	241180
10	60924	60924	36820	**29344**	60020	107148	179088	227820
11	158416	158416	30092	**30084**	59988	161556	170192	249948
12	2084100	2084364	**2083036**	2083300	2083080	2084240	2085156	2083636
13	2083368	2084160	**42720**	**42720**	231272	397824	341616	544976
14	2083220	2083880	952828	**463764**	1860692	978640	1792384	1401212
15	2083496	2083492	2082940	2085116	2083248	2083276	**2082764**	2084936

floortile ipc11

memory	pdb	cpdb	ipdb	cipdb	ms	ms-m	ms-p	ms-mp
1	49100	**6160**	51256	37728	71984	87880	125376	150784
2	25388	**5332**	42896	37720	76652	86836	125400	154984
3	2083172	2085300	2083088	**102792**	2084356	108184	2083948	218532
4	2083392	1793796	2083168	**16412**	2082872	184896	2084348	282696
5	2085328	2085268	2082940	**580744**	2083088	1641980	2083340	1640608
6	2084476	2083432	2082984	**137908**	2083900	2083612	2083148	2085056
7	2084716	2083168	None	**1573068**	2084000	2084104	2083008	2086724
8	2083992	2085044	None	**334264**	2085764	2082816	2083856	2084492
9	**2082968**	2083240	None	None	2083264	2084672	2083404	2083888
10	2087484	**2083436**	None	None	2083904	2084580	2084600	2083716

freecell v1

memory	pdb	cpdb	ipdb	cipdb	ms	ms-m	ms-p	ms-mp
1	8808	**6752**	62340	61172	100376	383156	279208	498724
2	**7704**	7740	70592	68896	145940	None	1435312	None
3	17436	17436	100220	**17056**	1077372	None	2097140	None
4	**71724**	**71724**	132268	None	2084428	None	2096616	2096688
5	515624	515536	**161788**	None	2086332	2093976	2053584	None
6	2082964	2082832	2082964	None	**2061868**	2096928	2097152	None
7	2085084	2083432	2084992	None	**2022520**	2045180	2095844	2096256
8	2083172	2082772	2082860	None	2093552	2096180	2096916	**1863696**
9	2082736	2082868	2085436	2083256	**2065280**	2083276	2096504	2096888
10	2083056	2083324	2083868	2083064	2097140	**2070804**	2096696	2097100

freecell v2

memory	pdb	cpdb	ipdb	cipdb	ms	ms-m	ms-p	ms-mp
2-1	**7048**	7204	48076	46936	1890460	346708	185400	834328
2-2	10440	**9724**	46056	45524	62156	353136	658860	772368
2-3	7056	**6796**	12836	14888	1065300	242180	208688	1938520
2-4	12120	**10076**	47864	46836	85612	586392	468792	2012556
2-5	10472	**9728**	101108	99884	1738048	1231668	200112	2031192
3-1	42400	**36900**	100132	None	234648	None	358348	2004008
3-2	31352	**28316**	100196	None	1721136	None	552476	None
3-3	22788	**20676**	21948	21368	2097076	1860164	863700	2073128
3-4	29392	**25832**	86872	None	611428	1603044	1945164	2091656
3-5	33784	28500	32736	**28004**	194996	1011596	345172	None
4-1	2083296	2082744	**357152**	None	2054348	1842964	2083104	None
4-2	1725240	1472132	**119400**	None	2088360	None	1726352	None
4-3	2083376	**1920244**	2082872	2082740	2097108	2090996	2083216	1984672
4-4	2085096	2082740	**237996**	None	2082928	None	2073228	None
4-5	2082936	2085092	**386816**	None	2085076	2097116	2084860	1755032

gripper

memory	pdb	cpdb	ipdb	cipdb	ms	ms-m	ms-p	ms-mp
1	**2960**	**2960**	**2960**	**2960**	3088	3352	3352	3752
2	3240	3240	**3088**	**3088**	6440	10096	11884	18016
3	5640	5700	**3600**	**3600**	32560	51268	40760	66104
4	9664	9828	**7176**	7184	55176	61476	91728	121264
5	27380	28104	**25536**	26064	71696	77476	115548	151384
6	126304	129172	**126052**	128568	132628	134388	140448	177296
7	**608356**	619756	613772	626948	623192	631852	622508	632340
8	2083748	2083572	2083232	2083740	2085288	2083052	**2082744**	2083040

pipesworld-tankage

memory	pdb	cpdb	ipdb	cipdb	ms	ms-m	ms-p	ms-mp
1	6948	5952	20376	21256	**4612**	7576	15372	17292
2	6156	**5540**	62888	45688	6284	10408	26216	32540
3	5864	5328	6068	**4876**	79128	98952	199840	246040
4	7008	**6516**	76096	70952	378892	120340	260228	277848
5	8912	**7800**	24660	8092	279860	256268	230464	398944
6	8508	**7656**	101180	19544	486776	197556	331776	329264
7	10572	**9840**	76264	17440	2088100	445584	776128	554412
8	143236	**11304**	204640	None	2096940	1998304	1029796	2061420
9	2083524	2088476	2083104	None	**1996520**	2083060	2083172	2083748
10	2083052	**214456**	620772	None	2096780	None	2084700	2078660
11	108656	105748	**42252**	None	755076	1992648	733080	1874588
12	1641064	1606636	**396392**	None	2058224	2087296	2083260	2044984
13	29880	**25792**	163192	None	1706872	1980000	1960976	1983340
14	2084084	2083080	2084504	None	2086952	2082532	2082976	**2014860**

scanalyzer ipc11

memory	pdb	cpdb	ipdb	cipdb	ms	ms-m	ms-p	ms-mp
1	3368	3368	3468	3480	**3352**	3356	3616	3616
2	8308	**5324**	6204	6676	38932	99476	64440	127040
3	7432	**5300**	6404	5672	37488	98708	63792	126248
4	5092	**4864**	12624	12248	262664	224556	114448	295980
5	**234220**	**234220**	721976	721976	1972868	None	612868	1882152
6	4716	**4540**	601412	601408	510656	391980	382996	391632
7	2083720	None	**2082772**	None	2097060	None	2097048	2095980
8	**2083748**	None	2084048	2084444	2085584	None	2096088	None
9	2085536	None	2086452	None	1895468	None	**1646376**	None
10	2085980	None	2085356	None	**1966336**	None	2095952	None

sokoban ipc11

memory	pdb	cpdb	ipdb	cipdb	ms	ms-m	ms-p	ms-mp
1	4908	4912	**3616**	**3616**	1954396	651076	844020	669708
2	9268	8128	**6248**	140856	2082956	None	2097068	None
3	6016	5356	**4136**	4140	2096996	None	2097144	None
4	9384	8004	**5032**	**5032**	1938156	None	2097120	None
5	40596	32432	**26164**	26172	2084856	None	1982368	None
6	34424	34424	25276	**25268**	2075924	None	2096704	None
7	40812	40700	7740	**7728**	2097024	None	2091004	None
8	13376	**11028**	18728	18728	2083180	None	2097152	None
9	7404	6408	**4260**	4264	2097132	None	2097012	None
10	7652	**7564**	7900	7848	2093884	None	2097076	None

tidybot ipc11

memory	pdb	cpdb	ipdb	cipdb	ms	ms-m	ms-p	ms-mp
1	15380	None	**10088**	**10088**	1943840	2068656	2078464	1989292
2	31292	None	**24400**	24620	2036996	2039772	2056192	1977616
3	**20544**	None	23288	23568	2089716	2088000	2084828	2081836
4	46988	None	**23076**	23240	2096548	2088000	2084828	2081836
5	483968	None	**152192**	**152192**	2097016	2093104	2087532	2096428
6	55332	None	**41868**	42012	2097032	2085356	2097144	2097120
7	**30468**	None	36656	36800	2092260	2097012	2097144	2097120
8	335268	None	**110396**	**110396**	2097016	2097012	2097144	2095736
9	81080	None	**64224**	64448	2097064	2092828	2097108	2082800
10	214576	None	**67760**	67984	2094436	2097120	2097064	2097080

trucks-strips

memory	pdb	cpdb	ipdb	cipdb	ms	ms-m	ms-p	ms-mp
1	4136	4140	**3088**	**3088**	3804	5992	7064	9748
2	5848	5360	**3220**	**3220**	6484	18608	24676	38128
3	12416	6948	5488	**3628**	65260	87724	85376	129560
4	68076	34388	26012	**7976**	172588	185292	172340	209312
5	385824	222048	113088	**12260**	1276488	1113196	1216300	1045456
6	2083320	2082820	None	**151168**	2083176	2082976	2082760	2083100
7	168120	168192	**10952**	10960	669520	660352	632968	627300
8	2083008	1489188	358780	**20648**	2085384	2082788	2083104	2083368
9	2084868	2084920	None	**228028**	2083232	None	2082852	2083004
10	2083052	2082860	2082988	**1423396**	2083136	None	2083136	2083076

woodworking ipc11

memory	pdb	cpdb	ipdb	cipdb	ms	ms-m	ms-p	ms-mp
1	9688	**9172**	None	None	74412	124708	233684	286676
2	**13116**	13168	None	None	83124	216180	266876	348724
3	362232	**134432**	None	None	174592	365888	369212	567448
4	**135756**	**135756**	None	None	140044	354956	297936	526936
5	1460088	1391096	None	None	**258532**	847320	389808	1078240
6	2084480	2083396	None	None	2083800	**1287952**	2083728	1605248
7	2084312	2085232	None	None	2083136	**498340**	2083408	829272
8	2086108	**2084792**	None	None	2097096	2091952	2097112	2096184
9	2084104	2083860	None	None	2083496	None	**2083464**	None
10	2085304	2084308	None	None	**744092**	None	757168	None

heuristic_time

airport standart

heuristic_time	pdb	cpdb	ipdb	cipdb	ms	ms-m	ms-p	ms-mp
1	0.06	0.10	**0.00**	**0.00**	**0.00**	**0.00**	0.02	0.02
2	**0.00**	0.08	**0.00**	0.02	**0.00**	**0.00**	0.02	0.02
3	**0.00**	0.08	0.42	5.28	0.04	0.18	0.66	0.82
4	**0.00**	**0.00**	0.06	0.14	0.02	0.04	0.18	0.18
5	**0.00**	0.10	0.54	0.92	0.02	0.04	0.20	0.22
6	**0.00**	0.14	10.46	31.66	0.16	2.92	10.30	19.88
7	**0.00**	0.14	9.90	22.36	0.16	2.88	9.82	20.16
8	**0.02**	0.04	34.98	33.70	29.44	704.66	None	None
9	**0.04**	0.12	188.42	179.92	55.00	None	None	None
10	**0.00**	**0.00**	0.08	0.08	0.02	0.02	0.20	0.22
11	0.04	0.24	0.62	0.66	**0.02**	0.04	0.26	0.30
12	**0.00**	0.02	11.88	25.14	0.22	2.70	13.80	27.78
13	**0.00**	**0.00**	7.54	43.60	0.20	2.20	17.10	27.40
14	**0.04**	0.06	38.88	41.96	45.40	None	None	None
15	**0.04**	0.08	64.62	76.98	39.64	None	None	None
16	**0.10**	0.62	587.10	580.00	110.36	None	None	None
17	**0.12**	0.70	None	None	None	None	None	None
18	None	None	None	None	None	None	None	None
19	**0.12**	0.72	563.04	804.50	None	None	None	None
20	None	None	None	None	None	None	None	None

airport half MUC

heuristic_time	pdb	cpdb	ipdb	cipdb	ms	ms-m	ms-p	ms-mp
1	**0.20**	**0.20**	49.88	90.66	200.68	None	None	None
2	0.38	**0.36**	196.48	323.34	None	None	None	None
3	None	None	None	None	None	None	None	None
4	None	None	None	None	None	None	None	None
5	None	None	None	None	None	None	None	None
6	None	None	None	None	None	None	None	None
7	None	None	None	None	None	None	None	None
8	None	None	None	None	None	None	None	None
9	None	None	None	None	None	None	None	None
10	None	None	None	None	None	None	None	None
11	None	None	None	None	None	None	None	None
12	None	None	None	None	None	None	None	None
13	None	None	None	None	None	None	None	None
14	None	None	None	None	None	None	None	None
15	None	None	None	None	None	None	None	None

airport MUC

heuristic_time	pdb	cpdb	ipdb	cipdb	ms	ms-m	ms-p	ms-mp
1	**0.34**	**0.34**	67.92	116.20	233.50	None	None	None
2	None	None	None	None	None	None	None	None
3	None	None	None	None	None	None	None	None
4	None	None	None	None	None	None	None	None
5	None	None	None	None	None	None	None	None
6	None	None	None	None	None	None	None	None
7	None	None	None	None	None	None	None	None
8	None	None	None	None	None	None	None	None
9	None	None	None	None	None	None	None	None
10	None	None	None	None	None	None	None	None
11	None	None	None	None	None	None	None	None
12	None	None	None	None	None	None	None	None
13	None	None	None	None	None	None	None	None
14	None	None	None	None	None	None	None	None
15	None	None	None	None	None	None	None	None

barman ipc11

heuristic_time	pdb	cpdb	ipdb	cipdb	ms	ms-m	ms-p	ms-mp
1	**1.88**	151.96	327.04	775.94	103.54	None	262.16	None
2	**1.86**	142.66	389.90	653.20	77.86	None	221.74	None
3	**1.84**	145.16	139.00	219.94	75.76	None	213.50	None
4	**1.84**	178.36	355.86	651.00	75.04	None	230.18	None
5	None	None	None	None	None	None	None	None
6	None	None	None	None	None	None	None	None
7	None	None	None	None	None	None	None	None
8	None	None	None	None	None	None	None	None

blocks

heuristic_time	pdb	cpdb	ipdb	cipdb	ms	ms-m	ms-p	ms-mp
1-0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
2-0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
4-0	**0.00**	**0.00**	0.22	0.02	**0.00**	**0.00**	**0.00**	0.02
4-1	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	0.02
4-2	**0.00**	**0.00**	0.22	0.02	**0.00**	**0.00**	**0.00**	0.02
5-0	0.02	**0.00**	**0.00**	**0.00**	0.26	0.12	0.16	0.32
5-1	0.02	**0.00**	**0.00**	**0.00**	0.24	0.14	0.18	0.34
5-2	0.02	**0.00**	**0.00**	**0.00**	0.26	0.14	0.16	0.32
6-0	0.42	**0.00**	**0.00**	**0.00**	1.02	2.36	2.04	4.46
6-1	0.42	**0.00**	**0.00**	**0.00**	0.96	2.34	2.02	4.56
6-2	0.44	**0.00**	**0.00**	**0.00**	1.10	2.38	2.04	3.36
7-0	0.14	0.10	**0.00**	**0.00**	2.10	16.46	8.20	20.60
7-1	0.14	0.12	**0.00**	**0.00**	1.82	15.64	9.18	21.42
7-2	0.14	0.10	**0.00**	**0.00**	2.10	18.10	9.14	21.64
8-0	0.20	0.18	**0.00**	**0.00**	3.18	46.94	16.84	37.92
8-1	0.26	0.18	**0.00**	**0.00**	3.32	36.52	13.14	38.86
8-2	0.20	0.16	**0.00**	**0.00**	3.44	39.18	13.00	42.28
9-0	**0.04**	0.10	0.08	0.08	5.26	89.86	25.44	69.78
9-1	0.04	0.10	**0.00**	**0.00**	4.84	87.20	24.56	97.12
9-2	0.04	0.10	**0.00**	0.02	5.54	82.40	19.96	124.24
10-0	None	0.20	**0.12**	**0.12**	None	201.10	65.24	279.68
10-1	**0.08**	0.20	0.12	0.14	8.76	220.12	37.86	713.52
10-2	None	0.20	**0.12**	**0.12**	None	245.00	29.60	201.92
11-0	None	None	**0.14**	0.16	None	371.18	60.96	203.98
11-1	None	None	**0.18**	0.20	None	201.08	None	None
11-2	None	None	**0.14**	0.16	None	None	None	None

depot

heuristic_time	pdb	cpdb	ipdb	cipdb	ms	ms-m	ms-p	ms-mp
1	**0.00**	**0.00**	0.52	0.54	**0.00**	0.02	0.04	0.04
2	0.28	**0.02**	38.24	26.22	0.60	13.08	5.30	13.38
3	**0.20**	0.76	23.20	None	11.34	312.84	64.44	353.54
4	**0.44**	3.10	15.28	49.78	None	None	122.28	None
5	None	None	None	None	None	None	None	None
6	None	None	None	None	None	None	None	None
7	**0.04**	0.22	9.28	137.30	17.80	679.02	80.22	None
8	None	None	None	None	None	None	None	None
9	None	None	None	None	None	None	None	None
10	**0.52**	4.24	6.48	323.32	66.80	None	None	None

driverlog

heuristic_time	pdb	cpdb	ipdb	cipdb	ms	ms-m	ms-p	ms-mp
1	**0.00**	**0.00**	**0.00**	**0.00**	0.02	0.04	0.10	0.18
2	**0.04**	0.54	0.28	0.36	0.08	0.40	0.70	1.38
3	**0.02**	0.52	0.38	0.42	0.10	0.36	0.66	1.20
4	0.56	6.42	1.28	1.14	**0.54**	2.32	2.60	6.64
5	1.00	13.44	1.06	1.20	**0.80**	3.30	4.14	10.58
6	**0.32**	6.60	7.38	27.92	1.80	8.06	7.58	21.88
7	**0.26**	5.28	48.20	117.48	2.42	14.34	10.16	28.22
8	None	None	64.92	179.42	**3.38**	19.96	12.66	36.66
9	**0.18**	2.74	4.14	6.18	2.82	12.86	17.50	30.32
10	**0.36**	6.04	15.40	41.00	3.18	12.58	20.12	33.60
11	**0.04**	0.88	4.46	20.70	1.98	11.84	14.40	26.10
12	None	None	None	None	None	None	None	None
13	None	None	**1.80**	5.76	3.14	34.38	34.42	61.68
14	None	None	15.90	55.10	**4.62**	66.16	39.76	103.62
15	None	None	None	None	None	None	None	None

floortile ipc11

heuristic_time	pdb	cpdb	ipdb	cipdb	ms	ms-m	ms-p	ms-mp
1	0.26	**0.06**	22.64	9.50	2.96	10.08	9.02	16.02
2	0.26	**0.04**	17.10	11.50	3.10	12.52	11.30	19.14
3	None	None	None	**42.40**	None	100.88	None	100.06
4	None	**0.60**	None	22.94	None	102.34	None	107.00
5	None	None	None	**52.26**	None	163.02	None	204.22
6	None	None	None	**77.60**	None	None	None	None
7	None	None	None	**141.90**	None	None	None	None
8	None	None	None	**150.68**	None	None	None	None
9	None	None	None	None	None	None	None	None
10	None	None	None	None	None	None	None	None

freecell v1

heuristic_time	pdb	cpdb	ipdb	cipdb	ms	ms-m	ms-p	ms-mp
1	**0.62**	1.72	9.66	75.90	5.38	48.68	26.04	99.52
2	**0.44**	4.42	27.98	749.40	15.04	None	92.14	None
3	**0.48**	13.00	36.78	10.42	51.86	None	None	None
4	**2.26**	68.86	37.12	None	None	None	None	None
5	**1.52**	43.98	34.80	None	None	None	315.20	None
6	None	None	None	None	None	None	None	None
7	None	None	None	None	None	None	None	None
8	None	None	None	None	None	None	None	None
9	None	None	None	None	None	None	None	None
10	None	None	None	None	None	None	None	None

freecell v2

heuristic_time	pdb	cpdb	ipdb	cipdb	ms	ms-m	ms-p	ms-mp
2-1	**0.26**	3.12	14.30	286.42	None	89.72	26.68	140.22
2-2	**0.68**	5.86	14.40	531.78	5.22	98.58	34.74	158.84
2-3	**0.24**	2.74	2.84	137.14	25.12	70.76	30.26	166.92
2-4	**0.80**	4.62	17.58	535.44	5.74	80.90	38.64	136.40
2-5	**0.66**	6.04	28.44	698.86	None	111.32	27.24	256.04
3-1	**0.70**	9.90	38.04	None	16.86	None	69.32	None
3-2	**0.80**	11.10	70.80	None	None	None	69.42	None
3-3	**0.68**	11.12	1.02	5.84	None	None	80.60	762.08
3-4	**0.90**	12.66	42.10	None	42.76	525.10	None	None
3-5	**0.68**	10.12	**0.68**	21.20	19.42	450.44	73.64	None
4-1	None	None	**83.84**	None	None	None	None	None
4-2	**1.14**	29.92	61.96	None	None	None	133.70	None
4-3	None	**58.54**	None	None	None	None	None	None
4-4	None	None	**58.68**	None	None	None	None	None
4-5	None	None	**83.92**	None	None	None	None	None

gripper

heuristic_time	pdb	cpdb	ipdb	cipdb	ms	ms-m	ms-p	ms-mp
1	**0.00**	**0.00**	**0.00**	0.08	**0.00**	**0.00**	**0.00**	**0.00**
2	**0.00**	**0.00**	**0.00**	0.20	0.04	0.22	0.18	0.36
3	**0.00**	**0.00**	0.10	0.48	0.42	4.66	1.52	5.02
4	0.62	6.02	**0.22**	0.56	1.06	20.84	4.38	16.70
5	0.54	27.62	**0.28**	0.74	1.90	31.94	8.32	40.78
6	0.56	27.20	**0.40**	1.84	2.94	54.24	13.02	55.98
7	0.54	26.64	**0.50**	1.74	4.22	79.46	16.66	85.38
8	None	None	None	None	None	None	None	None

pipesworld-tankage

heuristic_time	pdb	cpdb	ipdb	cipdb	ms	ms-m	ms-p	ms-mp
1	0.14	0.14	2.88	14.56	**0.06**	0.48	0.96	1.86
2	**0.00**	**0.00**	4.98	10.60	0.10	0.76	1.14	2.28
3	**0.10**	1.26	0.66	2.24	7.94	65.00	30.50	152.78
4	**0.14**	1.34	7.92	154.96	30.76	76.54	41.98	120.72
5	**0.36**	3.88	3.60	46.86	15.84	78.24	53.36	121.74
6	**0.20**	2.60	10.92	223.64	20.64	122.40	67.26	156.64
7	**0.36**	8.24	11.32	374.66	None	525.18	132.68	565.38
8	**0.94**	47.22	27.72	None	None	None	117.68	None
9	None	None	None	None	None	None	None	None
10	None	341.84	**21.76**	None	None	None	None	None
11	**0.50**	25.10	23.38	None	53.22	243.22	170.62	None
12	**1.84**	140.00	266.40	None	None	None	None	None
13	**2.36**	246.98	132.60	None	None	None	None	None
14	None	None	None	None	None	None	None	None

scanalyzer ipc11

heuristic_time	pdb	cpdb	ipdb	cipdb	ms	ms-m	ms-p	ms-mp
1	**0.00**	**0.00**	0.82	0.92	**0.00**	**0.00**	0.02	**0.00**
2	**0.48**	0.82	9.16	9.92	1.28	5.00	1.16	5.76
3	**0.44**	0.68	2.88	2.50	1.10	4.30	1.06	5.60
4	**0.26**	18.42	7.10	14.34	7.28	156.70	8.16	103.80
5	**0.60**	70.70	44.32	230.66	55.10	None	34.20	649.68
6	**0.12**	7.66	7.18	12.34	4.20	110.98	7.82	81.14
7	None	None	None	None	None	None	None	None
8	None	None	None	None	None	None	None	None
9	None	None	None	None	None	None	None	None
10	None	None	None	None	None	None	None	None

sokoban ipc11

heuristic_time	pdb	cpdb	ipdb	cipdb	ms	ms-m	ms-p	ms-mp
1	**0.00**	0.20	**0.00**	**0.00**	None	315.90	41.46	208.28
2	**0.12**	0.58	0.48	58.26	None	None	None	None
3	**0.00**	0.14	0.46	0.50	None	None	None	None
4	**0.02**	0.14	0.56	0.76	None	None	None	None
5	**0.08**	0.54	0.74	0.82	None	None	None	None
6	**0.04**	0.24	1.92	2.20	None	None	None	None
7	**0.04**	0.24	0.52	0.70	None	None	None	None
8	0.22	1.62	**0.14**	0.20	None	None	None	None
9	**0.02**	0.34	0.50	0.56	None	None	None	None
10	**0.12**	4.76	89.04	108.66	None	None	None	None

tidybot ipc11

heuristic_time	pdb	cpdb	ipdb	cipdb	ms	ms-m	ms-p	ms-mp
1	3.88	None	**0.42**	0.44	None	None	None	None
2	**4.96**	None	19.38	20.18	None	None	None	None
3	**4.96**	None	10.36	11.50	None	None	None	None
4	**4.94**	None	12.42	13.46	None	None	None	None
5	**7.48**	None	32.76	32.70	None	None	None	None
6	**7.94**	None	51.38	51.64	None	None	None	None
7	**8.04**	None	15.46	17.16	None	None	None	None
8	**7.96**	None	31.78	33.06	None	None	None	None
9	**10.94**	None	36.76	56.46	None	None	None	None
10	**13.46**	None	77.36	116.44	None	None	None	None

trucks-strips

heuristic_time	pdb	cpdb	ipdb	cipdb	ms	ms-m	ms-p	ms-mp
1	**0.00**	0.08	0.42	0.32	**0.00**	0.12	0.10	0.20
2	**0.00**	0.14	0.48	0.28	0.06	0.74	0.58	1.16
3	**0.82**	4.06	5.12	2.56	1.34	34.24	5.48	23.26
4	**0.40**	3.28	9.54	8.96	2.20	94.02	7.04	57.56
5	**0.62**	7.40	11.78	6.76	3.16	206.32	11.78	137.72
6	None	None	None	**35.14**	None	None	None	None
7	**0.12**	1.42	4.76	4.84	1.84	104.42	9.70	90.82
8	None	**4.68**	40.74	33.06	None	None	None	None
9	None	None	None	**17.56**	None	None	None	None
10	None	None	None	**23.08**	None	None	None	None

woodworking ipc11

heuristic_time	pdb	cpdb	ipdb	cipdb	ms	ms-m	ms-p	ms-mp
1	**0.02**	0.14	None	None	4.36	21.54	31.70	76.02
2	**0.08**	0.58	None	None	9.10	61.10	41.78	95.72
3	**0.04**	0.58	None	None	23.24	164.00	108.36	267.54
4	**0.06**	0.64	None	None	14.42	58.46	60.08	124.02
5	**0.04**	0.18	None	None	21.30	112.48	103.52	258.08
6	None	None	None	None	None	**383.04**	None	505.58
7	None	None	None	None	None	**517.18**	None	720.00
8	None	None	None	None	None	None	None	None
9	None	None	None	None	None	None	None	None
10	None	None	None	None	**86.10**	None	259.72	None

search_time

airport standart

search_time	pdb	cpdb	ipdb	cipdb	ms	ms-m	ms-p	ms-mp
1	0.10	0.10	0.10	0.10	0.10	0.10	**0.08**	**0.08**
2	0.10	0.10	0.10	0.10	0.10	0.10	**0.08**	**0.08**
3	0.10	0.10	0.10	0.10	0.06	**0.00**	**0.00**	**0.00**
4	0.10	0.10	0.10	0.10	0.08	0.06	**0.00**	**0.00**
5	0.10	0.10	0.10	0.10	0.08	0.06	**0.00**	**0.00**
6	0.10	0.10	0.10	0.10	**0.02**	**0.02**	**0.02**	**0.02**
7	0.10	0.10	0.10	0.10	0.02	0.02	**0.00**	0.02
8	**0.10**	**0.10**	**0.10**	**0.10**	0.30	0.24	None	None
9	0.98	0.98	**0.10**	**0.10**	4.50	None	None	None
10	0.10	0.10	0.10	0.10	0.08	0.08	**0.00**	**0.00**
11	0.10	0.10	0.10	0.10	0.08	0.06	**0.00**	**0.00**
12	0.10	0.10	0.10	0.10	0.02	0.02	0.02	**0.00**
13	0.10	0.10	0.10	0.10	**0.02**	**0.02**	**0.02**	**0.02**
14	**0.10**	0.12	**0.10**	**0.10**	0.34	None	None	None
15	**0.10**	**0.10**	**0.10**	**0.10**	0.42	None	None	None
16	3.90	3.90	0.12	**0.10**	27.28	None	None	None
17	59.98	**59.94**	None	None	None	None	None	None
18	None	None	None	None	None	None	None	None
19	60.96	60.76	**0.10**	0.30	None	None	None	None
20	None	None	None	None	None	None	None	None

airport half MUC

search_time	pdb	cpdb	ipdb	cipdb	ms	ms-m	ms-p	ms-mp
1	**0.10**	**0.10**	**0.10**	**0.10**	0.74	None	None	None
2	20.08	20.02	**0.10**	**0.10**	None	None	None	None
3	None	None	None	None	None	None	None	None
4	None	None	None	None	None	None	None	None
5	None	None	None	None	None	None	None	None
6	None	None	None	None	None	None	None	None
7	None	None	None	None	None	None	None	None
8	None	None	None	None	None	None	None	None
9	None	None	None	None	None	None	None	None
10	None	None	None	None	None	None	None	None
11	None	None	None	None	None	None	None	None
12	None	None	None	None	None	None	None	None
13	None	None	None	None	None	None	None	None
14	None	None	None	None	None	None	None	None
15	None	None	None	None	None	None	None	None

airport MUC

search_time	pdb	cpdb	ipdb	cipdb	ms	ms-m	ms-p	ms-mp
1	0.22	0.22	**0.10**	**0.10**	2.20	None	None	None
2	None	None	None	None	None	None	None	None
3	None	None	None	None	None	None	None	None
4	None	None	None	None	None	None	None	None
5	None	None	None	None	None	None	None	None
6	None	None	None	None	None	None	None	None
7	None	None	None	None	None	None	None	None
8	None	None	None	None	None	None	None	None
9	None	None	None	None	None	None	None	None
10	None	None	None	None	None	None	None	None
11	None	None	None	None	None	None	None	None
12	None	None	None	None	None	None	None	None
13	None	None	None	None	None	None	None	None
14	None	None	None	None	None	None	None	None
15	None	None	None	None	None	None	None	None

barman ipc11

search_time	pdb	cpdb	ipdb	cipdb	ms	ms-m	ms-p	ms-mp
1	**41.02**	57.18	87.58	117.08	55.64	None	75.44	None
2	**40.24**	47.50	117.82	107.86	73.00	None	77.70	None
3	**40.14**	40.66	113.46	118.20	55.34	None	72.58	None
4	**40.42**	40.50	98.22	102.26	54.40	None	76.20	None
5	None	None	None	None	None	None	None	None
6	None	None	None	None	None	None	None	None
7	None	None	None	None	None	None	None	None
8	None	None	None	None	None	None	None	None

blocks

search_time	pdb	cpdb	ipdb	cipdb	ms	ms-m	ms-p	ms-mp
1-0	0.10	0.10	0.10	0.10	0.10	0.10	0.10	0.10
2-0	0.10	0.10	0.10	0.10	0.10	0.10	0.10	0.10
4-0	0.10	0.10	0.10	0.10	0.10	0.10	0.10	**0.08**
4-1	0.10	0.10	0.10	0.10	0.10	0.10	0.10	**0.08**
4-2	0.10	0.10	0.10	0.10	0.10	0.10	0.10	**0.08**
5-0	0.10	0.10	0.10	0.10	**0.00**	**0.00**	**0.00**	**0.00**
5-1	0.10	0.10	0.10	0.10	**0.00**	**0.00**	**0.00**	**0.00**
5-2	0.10	0.10	0.10	0.10	**0.00**	**0.00**	**0.00**	**0.00**
6-0	0.10	0.10	0.10	0.10	**0.00**	**0.00**	**0.00**	**0.00**
6-1	0.10	0.10	0.10	0.10	**0.00**	**0.00**	**0.00**	**0.00**
6-2	0.10	0.10	0.10	0.10	**0.00**	**0.00**	**0.00**	**0.00**
7-0	0.10	0.10	0.10	0.10	**0.00**	**0.00**	**0.00**	0.02
7-1	0.10	0.10	0.10	0.10	0.02	**0.00**	**0.00**	**0.00**
7-2	0.10	0.10	0.10	0.10	**0.00**	**0.00**	**0.00**	0.02
8-0	0.10	0.10	0.10	0.10	**0.00**	**0.00**	**0.00**	0.02
8-1	0.10	0.10	0.10	0.10	0.02	**0.00**	**0.00**	0.02
8-2	0.10	0.10	0.10	0.10	**0.00**	**0.00**	**0.00**	**0.00**
9-0	8.48	3.90	0.70	0.70	7.46	1.38	**0.34**	1.90
9-1	0.80	0.12	0.10	0.10	0.48	**0.00**	**0.00**	0.04
9-2	0.78	0.24	0.10	0.10	1.30	**0.02**	**0.02**	**0.02**
10-0	None	85.96	10.46	**10.26**	None	51.78	46.24	35.50
10-1	131.48	62.80	2.74	**2.70**	105.00	53.58	25.70	55.72
10-2	None	83.50	4.94	**4.88**	None	36.42	35.68	39.30
11-0	None	None	2.62	**2.60**	None	93.28	68.20	30.84
11-1	None	None	4.42	**4.36**	None	142.18	None	None
11-2	None	None	**1.04**	**1.04**	None	None	None	None

depot

search_time	pdb	cpdb	ipdb	cipdb	ms	ms-m	ms-p	ms-mp
1	0.10	0.10	0.10	0.10	0.10	0.08	**0.06**	**0.06**
2	0.10	0.10	0.10	0.10	**0.00**	**0.00**	**0.00**	**0.00**
3	2.66	3.22	**1.48**	None	2.38	2.44	1.96	2.10
4	38.86	42.40	12.92	**5.68**	None	None	52.14	None
5	None	None	None	None	None	None	None	None
6	None	None	None	None	None	None	None	None
7	3.70	2.84	0.72	**0.40**	2.20	2.00	2.46	None
8	None	None	None	None	None	None	None	None
9	None	None	None	None	None	None	None	None
10	45.56	42.98	16.34	**3.58**	47.76	None	None	None

driverlog

search_time	pdb	cpdb	ipdb	cipdb	ms	ms-m	ms-p	ms-mp
1	0.10	0.10	0.10	0.10	0.08	0.06	**0.00**	**0.00**
2	0.10	0.10	0.10	0.10	0.02	**0.00**	**0.00**	0.02
3	0.10	0.10	0.10	0.10	**0.00**	**0.00**	**0.00**	0.02
4	0.10	0.10	0.10	0.10	**0.00**	**0.00**	**0.00**	**0.00**
5	0.10	0.10	0.10	0.10	**0.00**	**0.00**	**0.00**	**0.00**
6	0.10	0.10	0.10	0.10	**0.00**	**0.00**	**0.00**	**0.00**
7	0.26	0.26	0.10	0.10	**0.00**	**0.00**	**0.00**	**0.00**
8	None	None	0.98	15.60	0.66	**0.02**	0.24	0.04
9	108.86	95.10	0.22	0.32	0.02	**0.00**	**0.00**	**0.00**
10	2.12	2.04	0.10	0.10	**0.00**	**0.00**	**0.00**	0.02
11	6.32	6.44	**0.12**	**0.12**	0.26	0.24	0.26	0.24
12	None	None	None	None	None	None	None	None
13	None	None	2.46	**2.42**	9.16	11.64	11.80	9.16
14	None	None	135.24	34.44	90.22	**14.90**	95.30	15.70
15	None	None	None	None	None	None	None	None

floortile ipc11

search_time	pdb	cpdb	ipdb	cipdb	ms	ms-m	ms-p	ms-mp
1	1.72	0.10	2.62	0.10	3.10	**0.00**	1.86	0.02
2	0.68	0.10	0.98	0.10	1.24	**0.00**	1.22	0.02
3	None	None	None	8.62	None	**4.14**	None	4.36
4	None	106.00	None	**0.96**	None	2.34	None	1.74
5	None	None	None	**77.30**	None	86.38	None	88.42
6	None	None	None	**19.12**	None	None	None	None
7	None	None	None	**342.86**	None	None	None	None
8	None	None	None	**74.22**	None	None	None	None
9	None	None	None	None	None	None	None	None
10	None	None	None	None	None	None	None	None

freecell v1

search_time	pdb	cpdb	ipdb	cipdb	ms	ms-m	ms-p	ms-mp
1	0.10	0.10	0.10	0.10	**0.02**	**0.02**	**0.02**	0.04
2	0.16	0.16	**0.10**	**0.10**	0.24	None	0.20	None
3	0.86	1.18	**0.10**	1.04	1.38	None	None	None
4	5.90	5.70	**2.14**	None	None	None	None	None
5	55.94	65.64	**15.80**	None	None	None	79.92	None
6	None	None	None	None	None	None	None	None
7	None	None	None	None	None	None	None	None
8	None	None	None	None	None	None	None	None
9	None	None	None	None	None	None	None	None
10	None	None	None	None	None	None	None	None

freecell v2

search_time	pdb	cpdb	ipdb	cipdb	ms	ms-m	ms-p	ms-mp
2-1	**0.10**	**0.10**	**0.10**	**0.10**	None	0.12	0.12	0.12
2-2	**0.10**	**0.10**	**0.10**	**0.10**	0.14	0.12	0.12	0.16
2-3	**0.10**	**0.10**	**0.10**	**0.10**	0.16	0.16	0.14	0.12
2-4	**0.10**	**0.10**	**0.10**	**0.10**	0.16	0.16	0.16	0.18
2-5	**0.10**	**0.10**	**0.10**	**0.10**	None	0.12	0.14	0.18
3-1	4.10	3.20	**0.36**	None	6.88	None	6.52	None
3-2	2.28	1.84	**0.10**	None	None	None	2.40	None
3-3	1.28	**1.04**	1.56	1.96	None	None	1.48	1.32
3-4	1.78	1.40	**0.10**	None	2.42	0.94	None	None
3-5	2.68	**1.86**	2.72	2.72	3.40	2.46	3.02	None
4-1	None	None	**31.48**	None	None	None	None	None
4-2	245.98	213.54	**4.10**	None	None	None	285.56	None
4-3	None	**280.96**	None	None	None	None	None	None
4-4	None	None	**21.60**	None	None	None	None	None
4-5	None	None	**33.66**	None	None	None	None	None

gripper

search_time	pdb	cpdb	ipdb	cipdb	ms	ms-m	ms-p	ms-mp
1	0.10	0.10	0.10	0.10	0.10	0.10	0.10	0.10
2	0.10	0.10	0.10	0.10	0.06	**0.00**	**0.00**	**0.00**
3	0.10	0.10	0.10	0.10	**0.02**	0.06	0.04	0.06
4	**0.24**	0.26	0.30	0.34	0.34	0.46	0.44	0.30
5	**1.52**	1.80	2.00	2.48	2.26	3.02	2.74	2.88
6	**9.58**	10.30	12.36	18.50	13.00	12.76	16.10	12.36
7	**52.66**	54.30	68.44	80.60	68.38	68.62	77.34	69.60
8	None	None	None	None	None	None	None	None

pipesworld-tankage

search_time	pdb	cpdb	ipdb	cipdb	ms	ms-m	ms-p	ms-mp
1	0.10	0.10	0.10	0.10	0.04	**0.00**	**0.00**	**0.00**
2	0.10	0.10	0.10	0.10	**0.00**	**0.00**	**0.00**	**0.00**
3	0.10	0.10	0.10	0.10	**0.04**	0.06	**0.04**	0.12
4	0.10	0.10	0.10	0.10	0.06	**0.04**	**0.04**	0.22
5	0.10	0.10	0.10	0.10	0.08	**0.04**	0.08	0.06
6	0.10	0.10	0.10	0.10	0.08	0.08	0.08	**0.06**
7	**0.10**	**0.10**	**0.10**	**0.10**	None	0.28	0.30	0.30
8	4.82	**0.10**	0.18	None	None	None	2.10	None
9	None	None	None	None	None	None	None	None
10	None	**14.98**	42.66	None	None	None	None	None
11	8.70	5.58	**2.10**	None	11.24	8.06	9.38	None
12	132.58	99.52	**23.90**	None	None	None	None	None
13	0.66	0.56	**0.24**	None	None	None	None	None
14	None	None	None	None	None	None	None	None

scanalyzer ipc11

search_time	pdb	cpdb	ipdb	cipdb	ms	ms-m	ms-p	ms-mp
1	0.10	0.10	0.10	0.10	0.10	0.10	**0.08**	0.10
2	0.10	0.10	0.20	0.18	**0.00**	**0.00**	**0.00**	0.02
3	0.10	0.10	0.36	0.38	**0.00**	**0.00**	**0.00**	**0.00**
4	0.10	0.10	0.30	0.32	0.06	**0.02**	0.04	**0.02**
5	**8.34**	8.72	43.18	43.90	129.42	None	39.16	33.88
6	**0.10**	**0.10**	111.20	120.92	63.56	40.26	38.14	45.22
7	None	None	None	None	None	None	None	None
8	None	None	None	None	None	None	None	None
9	None	None	None	None	None	None	None	None
10	None	None	None	None	None	None	None	None

sokoban ipc11

search_time	pdb	cpdb	ipdb	cipdb	ms	ms-m	ms-p	ms-mp
1	0.10	0.10	0.10	0.10	None	0.02	**0.00**	0.04
2	0.12	**0.10**	0.12	**0.10**	None	None	None	None
3	**0.10**	**0.10**	**0.10**	**0.10**	None	None	None	None
4	0.18	0.14	**0.10**	**0.10**	None	None	None	None
5	1.28	**0.96**	1.02	1.04	None	None	None	None
6	**0.92**	**0.92**	0.94	0.98	None	None	None	None
7	1.46	1.48	**0.20**	**0.20**	None	None	None	None
8	0.20	**0.12**	0.54	0.68	None	None	None	None
9	**0.10**	**0.10**	**0.10**	**0.10**	None	None	None	None
10	**0.10**	**0.10**	**0.10**	**0.10**	None	None	None	None

tidybot ipc11

search_time	pdb	cpdb	ipdb	cipdb	ms	ms-m	ms-p	ms-mp
1	**0.10**	None	**0.10**	**0.10**	None	None	None	None
2	3.30	None	1.58	**1.56**	None	None	None	None
3	0.24	None	**0.10**	**0.10**	None	None	None	None
4	6.56	None	**1.40**	1.42	None	None	None	None
5	217.70	None	59.52	**59.28**	None	None	None	None
6	12.40	None	**2.24**	2.26	None	None	None	None
7	0.26	None	**0.14**	**0.14**	None	None	None	None
8	134.30	None	**40.22**	40.24	None	None	None	None
9	29.52	None	**11.08**	14.28	None	None	None	None
10	164.46	None	**18.58**	33.26	None	None	None	None

trucks-strips

search_time	pdb	cpdb	ipdb	cipdb	ms	ms-m	ms-p	ms-mp
1	0.10	0.10	0.10	0.10	0.10	**0.00**	**0.00**	**0.00**
2	0.10	0.10	0.10	0.10	0.04	**0.00**	**0.00**	**0.00**
3	0.50	**0.10**	0.38	**0.10**	2.40	2.44	2.00	1.84
4	5.84	2.18	9.08	**1.32**	17.88	16.32	19.48	16.66
5	61.96	28.70	50.90	**2.66**	236.34	169.04	230.68	180.18
6	None	None	None	**129.38**	None	None	None	None
7	10.96	12.72	**1.12**	1.14	64.42	65.74	61.26	67.88
8	None	181.24	320.26	**11.48**	None	None	None	None
9	None	None	None	**126.40**	None	None	None	None
10	None	None	None	**409.06**	None	None	None	None

woodworking ipc11

search_time	pdb	cpdb	ipdb	cipdb	ms	ms-m	ms-p	ms-mp
1	0.10	0.10	None	None	**0.00**	**0.00**	**0.00**	0.02
2	0.16	0.16	None	None	0.16	**0.02**	**0.02**	0.06
3	20.00	6.72	None	None	6.90	**0.64**	6.64	1.04
4	5.02	4.88	None	None	**0.00**	**0.00**	**0.00**	0.02
5	84.22	80.00	None	None	15.68	**1.06**	9.90	1.20
6	None	None	None	None	None	**9.30**	None	10.50
7	None	None	None	None	None	8.54	None	**6.64**
8	None	None	None	None	None	None	None	None
9	None	None	None	None	None	None	None	None
10	None	None	None	None	**32.78**	None	37.04	None

total_time

airport standart

total_time	pdb	cpdb	ipdb	cipdb	ms	ms-m	ms-p	ms-mp
1	0.16	0.20	**0.10**	**0.10**	**0.10**	**0.10**	**0.10**	**0.10**
2	**0.10**	0.18	**0.10**	0.12	**0.10**	**0.10**	**0.10**	**0.10**
3	**0.10**	0.18	0.52	5.38	**0.10**	0.18	0.66	0.82
4	**0.10**	**0.10**	0.16	0.24	**0.10**	**0.10**	0.18	0.18
5	**0.10**	0.20	0.64	1.02	**0.10**	**0.10**	0.20	0.22
6	**0.10**	0.24	10.56	31.76	0.18	2.94	10.32	19.90
7	**0.10**	0.24	10.00	22.46	0.18	2.90	9.82	20.18
8	**0.12**	0.14	35.08	33.80	29.74	704.90	None	None
9	**1.02**	1.10	188.52	180.02	59.50	None	None	None
10	**0.10**	**0.10**	0.18	0.18	**0.10**	**0.10**	0.20	0.22
11	0.14	0.34	0.72	0.76	**0.10**	**0.10**	0.26	0.30
12	**0.10**	0.12	11.98	25.24	0.24	2.72	13.82	27.78
13	**0.10**	**0.10**	7.64	43.70	0.22	2.22	17.12	27.42
14	**0.14**	0.18	38.98	42.06	45.74	None	None	None
15	**0.14**	0.18	64.72	77.08	40.06	None	None	None
16	**4.00**	4.52	587.22	580.10	137.64	None	None	None
17	**60.10**	60.64	None	None	None	None	None	None
18	None	None	None	None	None	None	None	None
19	**61.08**	61.48	563.14	804.80	None	None	None	None
20	None	None	None	None	None	None	None	None

airport half MUC

total_time	pdb	cpdb	ipdb	cipdb	ms	ms-m	ms-p	ms-mp
1	**0.30**	**0.30**	49.98	90.76	201.42	None	None	None
2	20.46	**20.38**	196.58	323.44	None	None	None	None
3	None	None	None	None	None	None	None	None
4	None	None	None	None	None	None	None	None
5	None	None	None	None	None	None	None	None
6	None	None	None	None	None	None	None	None
7	None	None	None	None	None	None	None	None
8	None	None	None	None	None	None	None	None
9	None	None	None	None	None	None	None	None
10	None	None	None	None	None	None	None	None
11	None	None	None	None	None	None	None	None
12	None	None	None	None	None	None	None	None
13	None	None	None	None	None	None	None	None
14	None	None	None	None	None	None	None	None
15	None	None	None	None	None	None	None	None

airport MUC

total_time	pdb	cpdb	ipdb	cipdb	ms	ms-m	ms-p	ms-mp
1	**0.56**	**0.56**	68.02	116.30	235.70	None	None	None
2	None	None	None	None	None	None	None	None
3	None	None	None	None	None	None	None	None
4	None	None	None	None	None	None	None	None
5	None	None	None	None	None	None	None	None
6	None	None	None	None	None	None	None	None
7	None	None	None	None	None	None	None	None
8	None	None	None	None	None	None	None	None
9	None	None	None	None	None	None	None	None
10	None	None	None	None	None	None	None	None
11	None	None	None	None	None	None	None	None
12	None	None	None	None	None	None	None	None
13	None	None	None	None	None	None	None	None
14	None	None	None	None	None	None	None	None
15	None	None	None	None	None	None	None	None

barman ipc11

total_time	pdb	cpdb	ipdb	cipdb	ms	ms-m	ms-p	ms-mp
1	**42.90**	209.14	414.62	893.02	159.18	None	337.60	None
2	**42.10**	190.16	507.72	761.06	150.86	None	299.44	None
3	**41.98**	185.82	252.46	338.14	131.10	None	286.08	None
4	**42.26**	218.86	454.08	753.26	129.44	None	306.38	None
5	None	None	None	None	None	None	None	None
6	None	None	None	None	None	None	None	None
7	None	None	None	None	None	None	None	None
8	None	None	None	None	None	None	None	None

blocks

total_time	pdb	cpdb	ipdb	cipdb	ms	ms-m	ms-p	ms-mp
1-0	0.10	0.10	0.10	0.10	0.10	0.10	0.10	0.10
2-0	0.10	0.10	0.10	0.10	0.10	0.10	0.10	0.10
4-0	**0.10**	**0.10**	0.32	0.12	**0.10**	**0.10**	**0.10**	**0.10**
4-1	0.10	0.10	0.10	0.10	0.10	0.10	0.10	0.10
4-2	**0.10**	**0.10**	0.32	0.12	**0.10**	**0.10**	**0.10**	**0.10**
5-0	0.12	**0.10**	**0.10**	**0.10**	0.26	0.12	0.16	0.32
5-1	0.12	**0.10**	**0.10**	**0.10**	0.24	0.14	0.18	0.34
5-2	0.12	**0.10**	**0.10**	**0.10**	0.26	0.14	0.16	0.32
6-0	0.52	**0.10**	**0.10**	**0.10**	1.02	2.36	2.04	4.46
6-1	0.52	**0.10**	**0.10**	**0.10**	0.96	2.34	2.02	4.56
6-2	0.54	**0.10**	**0.10**	**0.10**	1.10	2.38	2.04	3.36
7-0	0.24	0.20	**0.10**	**0.10**	2.10	16.46	8.20	20.62
7-1	0.24	0.22	**0.10**	**0.10**	1.84	15.64	9.18	21.42
7-2	0.24	0.20	**0.10**	**0.10**	2.10	18.10	9.14	21.66
8-0	0.30	0.28	**0.10**	**0.10**	3.18	46.94	16.84	37.94
8-1	0.36	0.28	**0.10**	**0.10**	3.34	36.52	13.14	38.88
8-2	0.30	0.26	**0.10**	**0.10**	3.44	39.18	13.00	42.28
9-0	8.52	4.00	**0.78**	**0.78**	12.72	91.24	25.78	71.68
9-1	0.84	0.22	**0.10**	**0.10**	5.32	87.20	24.56	97.16
9-2	0.82	0.34	**0.10**	0.12	6.84	82.42	19.98	124.26
10-0	None	86.16	10.58	**10.38**	None	252.88	111.48	315.18
10-1	131.56	63.00	2.86	**2.84**	113.76	273.70	63.56	769.24
10-2	None	83.70	5.06	**5.00**	None	281.42	65.28	241.22
11-0	None	None	**2.76**	**2.76**	None	464.46	129.16	234.82
11-1	None	None	4.60	**4.56**	None	343.26	None	None
11-2	None	None	**1.18**	1.20	None	None	None	None

depot

total_time	pdb	cpdb	ipdb	cipdb	ms	ms-m	ms-p	ms-mp
1	**0.10**	**0.10**	0.62	0.64	**0.10**	**0.10**	**0.10**	**0.10**
2	0.38	**0.12**	38.34	26.32	0.60	13.08	5.30	13.38
3	**2.86**	3.98	24.68	None	13.72	315.28	66.40	355.64
4	39.30	45.50	**28.20**	55.46	None	None	174.42	None
5	None	None	None	None	None	None	None	None
6	None	None	None	None	None	None	None	None
7	3.74	**3.06**	10.00	137.70	20.00	681.02	82.68	None
8	None	None	None	None	None	None	None	None
9	None	None	None	None	None	None	None	None
10	46.08	47.22	**22.82**	326.90	114.56	None	None	None

driverlog

total_time	pdb	cpdb	ipdb	cipdb	ms	ms-m	ms-p	ms-mp
1	**0.10**	**0.10**	**0.10**	**0.10**	**0.10**	**0.10**	**0.10**	0.18
2	0.14	0.64	0.38	0.46	**0.10**	0.40	0.70	1.40
3	0.12	0.62	0.48	0.52	**0.10**	0.36	0.66	1.22
4	0.66	6.52	1.38	1.24	**0.54**	2.32	2.60	6.64
5	1.10	13.54	1.16	1.30	**0.80**	3.30	4.14	10.58
6	**0.42**	6.70	7.48	28.02	1.80	8.06	7.58	21.88
7	**0.52**	5.54	48.30	117.58	2.42	14.34	10.16	28.22
8	None	None	65.90	195.02	**4.04**	19.98	12.90	36.70
9	109.04	97.84	4.36	6.50	**2.84**	12.86	17.50	30.32
10	**2.48**	8.08	15.50	41.10	3.18	12.58	20.12	33.62
11	6.36	7.32	4.58	20.82	**2.24**	12.08	14.66	26.34
12	None	None	None	None	None	None	None	None
13	None	None	**4.26**	8.18	12.30	46.02	46.22	70.84
14	None	None	151.14	89.54	94.84	**81.06**	135.06	119.32
15	None	None	None	None	None	None	None	None

floortile ipc11

total_time	pdb	cpdb	ipdb	cipdb	ms	ms-m	ms-p	ms-mp
1	1.98	**0.16**	25.26	9.60	6.06	10.08	10.88	16.04
2	0.94	**0.14**	18.08	11.60	4.34	12.52	12.52	19.16
3	None	None	None	**51.02**	None	105.02	None	104.42
4	None	106.60	None	**23.90**	None	104.68	None	108.74
5	None	None	None	**129.56**	None	249.40	None	292.64
6	None	None	None	**96.72**	None	None	None	None
7	None	None	None	**484.76**	None	None	None	None
8	None	None	None	**224.90**	None	None	None	None
9	None	None	None	None	None	None	None	None
10	None	None	None	None	None	None	None	None

freecell v1

total_time	pdb	cpdb	ipdb	cipdb	ms	ms-m	ms-p	ms-mp
1	**0.72**	1.82	9.76	76.00	5.40	48.70	26.06	99.56
2	**0.60**	4.58	28.08	749.50	15.28	None	92.34	None
3	**1.34**	14.18	36.88	11.46	53.24	None	None	None
4	**8.16**	74.56	39.26	None	None	None	None	None
5	57.46	109.62	**50.60**	None	None	None	395.12	None
6	None	None	None	None	None	None	None	None
7	None	None	None	None	None	None	None	None
8	None	None	None	None	None	None	None	None
9	None	None	None	None	None	None	None	None
10	None	None	None	None	None	None	None	None

freecell v2

total_time	pdb	cpdb	ipdb	cipdb	ms	ms-m	ms-p	ms-mp
2-1	**0.36**	3.22	14.40	286.52	None	89.84	26.80	140.34
2-2	**0.78**	5.96	14.50	531.88	5.36	98.70	34.86	159.00
2-3	**0.34**	2.84	2.94	137.24	25.28	70.92	30.40	167.04
2-4	**0.90**	4.72	17.68	535.54	5.90	81.06	38.80	136.58
2-5	**0.76**	6.14	28.54	698.96	None	111.44	27.38	256.22
3-1	**4.80**	13.10	38.40	None	23.74	None	75.84	None
3-2	**3.08**	12.94	70.90	None	None	None	71.82	None
3-3	**1.96**	12.16	2.58	7.80	None	None	82.08	763.40
3-4	**2.68**	14.06	42.20	None	45.18	526.04	None	None
3-5	**3.36**	11.98	3.40	23.92	22.82	452.90	76.66	None
4-1	None	None	**115.32**	None	None	None	None	None
4-2	247.12	243.46	**66.06**	None	None	None	419.26	None
4-3	None	**339.50**	None	None	None	None	None	None
4-4	None	None	**80.28**	None	None	None	None	None
4-5	None	None	**117.58**	None	None	None	None	None

gripper

total_time	pdb	cpdb	ipdb	cipdb	ms	ms-m	ms-p	ms-mp
1	**0.10**	**0.10**	**0.10**	0.18	**0.10**	**0.10**	**0.10**	**0.10**
2	**0.10**	**0.10**	**0.10**	0.30	**0.10**	0.22	0.18	0.36
3	**0.10**	**0.10**	0.20	0.58	0.44	4.72	1.56	5.08
4	0.86	6.28	**0.52**	0.90	1.40	21.30	4.82	17.00
5	**2.06**	29.42	2.28	3.22	4.16	34.96	11.06	43.66
6	**10.14**	37.50	12.76	20.34	15.94	67.00	29.12	68.34
7	**53.20**	80.94	68.94	82.34	72.60	148.08	94.00	154.98
8	None	None	None	None	None	None	None	None

pipesworld-tankage

total_time	pdb	cpdb	ipdb	cipdb	ms	ms-m	ms-p	ms-mp
1	0.24	0.24	2.98	14.66	**0.10**	0.48	0.96	1.86
2	**0.10**	**0.10**	5.08	10.70	**0.10**	0.76	1.14	2.28
3	**0.20**	1.36	0.76	2.34	7.98	65.06	30.54	152.90
4	**0.24**	1.44	8.02	155.06	30.82	76.58	42.02	120.94
5	**0.46**	3.98	3.70	46.96	15.92	78.28	53.44	121.80
6	**0.30**	2.70	11.02	223.74	20.72	122.48	67.34	156.70
7	**0.46**	8.34	11.42	374.76	None	525.46	132.98	565.68
8	**5.76**	47.32	27.90	None	None	None	119.78	None
9	None	None	None	None	None	None	None	None
10	None	356.82	**64.42**	None	None	None	None	None
11	**9.20**	30.68	25.48	None	64.46	251.28	180.00	None
12	**134.42**	239.52	290.30	None	None	None	None	None
13	**3.02**	247.54	132.84	None	None	None	None	None
14	None	None	None	None	None	None	None	None

scanalyzer ipc11

total_time	pdb	cpdb	ipdb	cipdb	ms	ms-m	ms-p	ms-mp
1	**0.10**	**0.10**	0.92	1.02	**0.10**	**0.10**	**0.10**	**0.10**
2	**0.58**	0.92	9.36	10.10	1.28	5.00	1.16	5.78
3	**0.54**	0.78	3.24	2.88	1.10	4.30	1.06	5.60
4	**0.36**	18.52	7.40	14.66	7.34	156.72	8.20	103.82
5	**8.94**	79.42	87.50	274.56	184.52	None	73.36	683.56
6	**0.22**	7.76	118.38	133.26	67.76	151.24	45.96	126.36
7	None	None	None	None	None	None	None	None
8	None	None	None	None	None	None	None	None
9	None	None	None	None	None	None	None	None
10	None	None	None	None	None	None	None	None

sokoban ipc11

total_time	pdb	cpdb	ipdb	cipdb	ms	ms-m	ms-p	ms-mp
1	**0.10**	0.30	**0.10**	**0.10**	None	315.92	41.46	208.32
2	**0.24**	0.68	0.60	58.36	None	None	None	None
3	**0.10**	0.24	0.56	0.60	None	None	None	None
4	**0.20**	0.28	0.66	0.86	None	None	None	None
5	**1.36**	1.50	1.76	1.86	None	None	None	None
6	**0.96**	1.16	2.86	3.18	None	None	None	None
7	1.50	1.72	**0.72**	0.90	None	None	None	None
8	**0.42**	1.74	0.68	0.88	None	None	None	None
9	**0.12**	0.44	0.60	0.66	None	None	None	None
10	**0.22**	4.86	89.14	108.76	None	None	None	None

tidybot ipc11

total_time	pdb	cpdb	ipdb	cipdb	ms	ms-m	ms-p	ms-mp
1	3.98	None	**0.52**	0.54	None	None	None	None
2	**8.26**	None	20.96	21.74	None	None	None	None
3	**5.20**	None	10.46	11.60	None	None	None	None
4	**11.50**	None	13.82	14.88	None	None	None	None
5	225.18	None	92.28	**91.98**	None	None	None	None
6	**20.34**	None	53.62	53.90	None	None	None	None
7	**8.30**	None	15.60	17.30	None	None	None	None
8	142.26	None	**72.00**	73.30	None	None	None	None
9	**40.46**	None	47.84	70.74	None	None	None	None
10	177.92	None	**95.94**	149.70	None	None	None	None

trucks-strips

total_time	pdb	cpdb	ipdb	cipdb	ms	ms-m	ms-p	ms-mp
1	**0.10**	0.18	0.52	0.42	**0.10**	0.12	**0.10**	0.20
2	**0.10**	0.24	0.58	0.38	**0.10**	0.74	0.58	1.16
3	**1.32**	4.16	5.50	2.66	3.74	36.68	7.48	25.10
4	6.24	**5.46**	18.62	10.28	20.08	110.34	26.52	74.22
5	62.58	36.10	62.68	**9.42**	239.50	375.36	242.46	317.90
6	None	None	None	**164.52**	None	None	None	None
7	11.08	14.14	**5.88**	5.98	66.26	170.16	70.96	158.70
8	None	185.92	361.00	**44.54**	None	None	None	None
9	None	None	None	**143.96**	None	None	None	None
10	None	None	None	**432.14**	None	None	None	None

woodworking ipc11

total_time	pdb	cpdb	ipdb	cipdb	ms	ms-m	ms-p	ms-mp
1	**0.12**	0.24	None	None	4.36	21.54	31.70	76.04
2	**0.24**	0.74	None	None	9.26	61.12	41.80	95.78
3	20.04	**7.30**	None	None	30.14	164.64	115.00	268.58
4	**5.08**	5.52	None	None	14.42	58.46	60.08	124.04
5	84.26	80.18	None	None	**36.98**	113.54	113.42	259.28
6	None	None	None	None	None	**392.34**	None	516.08
7	None	None	None	None	None	**525.72**	None	726.64
8	None	None	None	None	None	None	None	None
9	None	None	None	None	None	None	None	None
10	None	None	None	None	**118.88**	None	296.76	None

coverage

coverage	**pdb**	**cpdb**	**ipdb**	**cipdb**	**ms**	**ms-m**	**ms-p**	**ms-mp**	**out of**
airport	**21**	**21**	20	20	18	12	12	12	50
blocks	21	23	**26**	**26**	21	25	24	24	26
depot	**6**	**6**	**6**	5	5	4	5	3	10
driverlog	10	10	**13**	**13**	**13**	**13**	**13**	**13**	15
freecell	16	17	**19**	10	9	8	13	7	25
gripper	7	7	7	7	7	7	7	7	8
pipesworld-tankage	11	**12**	**12**	7	7	8	9	8	14
trucks-strips	6	7	7	**10**	6	6	6	6	10
total	98	103	**110**	98	86	83	89	80	158
barman	**4**	**4**	**4**	**4**	**4**	0	**4**	0	8
floortile	2	3	2	**8**	2	5	2	5	10
scanalyzer	**6**	**6**	**6**	**6**	**6**	5	**6**	**6**	10
sokoban	**10**	**10**	**10**	**10**	0	1	1	1	10
tidybot	**10**	0	**10**	**10**	0	0	0	0	10
woodworking	5	5	0	0	5	**7**	6	**7**	10
total-ipc11	37	28	32	**38**	17	18	19	19	58

Declaration of Authorship

Formular drucken

Philosophisch-Naturwissenschaftliche Fakultät
der Universität Basel
Dekanat

Erklärung zur wissenschaftlichen Redlichkeit
(beinhaltet Erklärung zu Plagiat und Betrug)

(bitte ankreuzen)
☒ Bachelorarbeit
☐ Masterarbeit

Titel der Arbeit *(Druckschrift)*:

Refining abstraction heuristics with mutexes

Name, Vorname *(Druckschrift)*: Solèr, Matthias

Matrikelnummer: 09-055-179

Hiermit erkläre ich, dass mir bei der Abfassung dieser Arbeit nur die darin angegebene Hilfe zuteil wurde und dass ich sie nur mit den in der Arbeit angegebenen Hilfsmitteln verfasst habe.

Ich habe sämtliche verwendeten Quellen erwähnt und gemäss anerkannten wissenschaftlichen Regeln zitiert.

Diese Erklärung wird ergänzt durch eine separat abgeschlossene Vereinbarung bezüglich der Veröffentlichung oder öffentlichen Zugänglichkeit dieser Arbeit.

☐ ja ☒ nein

Ort, Datum: Basel, 31.07.2012

Unterschrift: Matthias Solèr

Dieses Blatt ist in die Bachelor-, resp. Masterarbeit einzufügen.