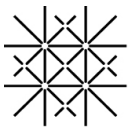


University of Basel

Correlation Complexity and Different Notions of Width

Master's Thesis

Faculty of Science of the University of Basel
Department of Mathematics and Computer Science
Artificial Intelligence
https://ai.dmi.unibas.ch

Examiner: Prof. Dr. Malte Helmert
Supervisors: Dr. Thomas Keller & Augusto B. Corrêa

Simon Dold

simon.dold@unibas.ch

2015-050-461

May 15, 2021

Abstract

In classical planning, there are tasks that are hard and tasks that are easy. We can measure the complexity of a task with the correlation complexity, the improvability width, and the novelty width. In this work, we compare these measures.

We investigate what causes a correlation complexity of at least 2. To do so we translate the state space into a vector space which allows us to make use of linear algebra and convex cones.

Additionally, we introduce the Basel measure, a new measure that is based on potential heuristics and therefore similar to the correlation complexity but also comparable to the novelty width. We show that the Basel measure is a lower bound for the correlation complexity and that the novelty width +1 is an upper bound for the Basel measure.

Furthermore, we compute the Basel measure for some tasks of the International Planning Competitions and show that the translation of a task can increase the Basel measure by removing seemingly irrelevant state variables.

Contents

1 Introduction

Planning is an area of artificial intelligence that is useful for many applications for example logistics. In classical planning, we try to find a plan, a sequence of actions to transition from an initial state to a goal state. We assume everything to be deterministic and fully observable (e.g. no dice rolls and no hidden cards). So the solution of this task is not affected by the execution skill of the agent or randomness but only by the cognitive ability to think ahead. We consider satisficing planning which looks for a plan that leads us to a goal state. In contrast to optimal planning which looks for the plan with the lowest cost. Both satisficing planning and optimal planning are PSPACE-complete in general (Bylander, 1994) but for some domains, it is drastically easier to find a satisficing plan than to find an optimal one. The Traveling Salesperson Problem on a fully connected weighted graph is a good example of this. A satisficing plan would be visiting all cities in any arbitrary order (we do not even have to look at the weights) but for the optimal plan, it is a common example of an NP-complete problem (Papadimitriou, 1977).

Some planning tasks are easier to solve than others even though they have the same amount of variables and operators. This difference is hidden somewhere in the structure of the task. We want to discuss in this work different measures for the difficulty of a planning task. The measures assign a number to the task that reflects how hard it is to solve the task with the corresponding algorithm. Harder tasks have greater measures. In Chapter 2 we provide the formal background of planning. Afterwards we describe in Chapter 3 the measures and corresponding algorithms created by Hoffmann (2001, 2002), Chen and Giménez (2007), Lipovetzky and Geffner (2012) as well as Seipp et al. (2016).

Each of these measures looks at a different kind of structure of the task. The maximal exit distance by Hoffmann (2001, 2002) checks how many transitions are needed at most to improve the heuristic value of h^+. The improvability width[1] by Chen and Giménez (2007) looks at how many state variables have to be changed to get to a state with more goal facts being true (without turning those false that were already true). The novelty width by Lipovetzky and Geffner (2012) checks how many facts have to be considered at once to detect that a state in the closed list of a `Breadth First Search` is a duplicate. The correlation complexity by Seipp et al. (2016) looks at the minimal dimension of a potential heuristic that is descending and dead-end avoiding.

Each of these measures provides a bound which implies that the search (with the fitting algorithm) is exponential only in the measure. However, the search is linear in the length of the plan, which can be exponential in the number of state variables. We formulate these algorithms in a common framework. This makes their similarities and differences more apparent.

In Chapter 4 we look at the sufficient conditions for a correlation complexity that is at least 2. There we add to the work of Seipp et al. (2016) by formulating

[1] Chen and Giménez (2007) as well as Lipovetzky and Geffner (2012) defined *width* differently, in this work we will call them improvability width and novelty width respectively to reduce confusion.

a new criterion that is more general and detects a larger correlation complexity in more cases. We show that the new criterion is more general by examples and, with a counterexample, we show that it is not a necessary condition. Additionally, we provide another new criterion to detect a correlation complexity of at least 3.

In Chapter 5 we translate state transitions into vectors and 1-dimensional potential heuristics into linear mappings. This way we can use the language of linear algebra and linear programming to formulate a new criterion that can be used to detect that the correlation complexity of a given task is larger than 1.

We introduce a new measure, in Chapter 6, which we call the Basel measure, and use mixed integer programming to create a method to detect whether or not the Basel measure of a given task is 1. Additionally, we look at the two notions of width and the correlation complexity and compare them with each other. We see that the Basel measure is a lower bound for the correlation complexity and the novelty width +1 is an upper bound for the Basel measure. With two examples we prove that the novelty width and improvability width, in general, do not introduce any bounds on each other.

We measure multiple domains from the International Planning Competitions[2] (IPC) in Chapter 7 and figure out that individual tasks of these domains have a Basel measure of 1 and see that the translation of a task can increase the Basel measure by removing seemingly irrelevant state variables.

[2]http://ipc.icaps-conference.org

2 Background

In this chapter, we introduce important definitions and concepts used throughout this thesis.

2.1 SAS$^+$ Planning Formalism

To argue rigorously about classical planning tasks we express them in a formal way. In this work, we use the SAS$^+$ formalism (Bäckström and Nebel, 1995). A SAS$^+$ planning task Π is defined as a tuple $\Pi = \langle V, I, O, \gamma \rangle$ where V is the set of **state variables** in finite domain representation with the **domain** $dom(v)$ for each $v \in V$. A tuple $\langle v, d \rangle$ with $v \in V$ and $d \in dom(v)$ is called a **fact** and we use the notation $v \mapsto d$. A set of facts is called a **partial assignment** (unless stated otherwise we assume each assignment in this work to be valid, meaning each variable $v \in V$ appears at most in one fact of the partial assignment). We define $vars(p)$ as the set of all state variables that appear in the partial assignment p. For a partial assignment p we define $p(v) = d$ if the fact $v \mapsto d$ is an element of p and undefined otherwise. We say two partial assignments p, q **agree** with each other if $p(v) = q(v)$ for each $v \in vars(p) \cap vars(q)$. We call a partial assignment s with $vars(s) = V$ a **state**. Let $M \subseteq V$ and p a partial assignment, we call p^M the **projection** of p on M and we define it as the partial assignment p^M where $vars(p^M) = M \cap vars(p)$ and p^M agrees with p.

The set O describes the **operators**. Every operator $o \in O$ consists of a **precondition** $pre(o)$ and an **effect** $\mathit{eff}(o)$, both are partial assignments; and a **cost** $cost(o) \in \mathbb{R}_{\geq 0}$. An operator o is **applicable** in state s if $pre(o)$ agrees with s. A task is in **normal form** if $vars(\mathit{eff}(o)) \subseteq vars(pre(o))$ for all $o \in O$. **Applying** an operator o in state s results in the **successor** state $s[\![o]\!]$ with $s[\![o]\!](v) := \mathit{eff}(o)(v)$ for all $v \in \mathit{eff}(o)$ and $s[\![o]\!](v) := s(v)$ for all $v \in V \setminus vars(\mathit{eff}(o))$. The successors of a state s are defined as $succ(s) := \{s[\![o]\!] \mid o \in O, o \text{ apllicable in } s\}$.

A sequence $\pi = \langle o_1, o_2, \ldots, o_n \rangle$ of operators $o_i \in O$ for each $i \in \{1, 2, \ldots, n\}$ applied on a state s is defined as $s[\![\langle . \rangle]\!] = s$ for the empty sequence; and $s[\![\langle o_1, \ldots, o_n \rangle]\!] = s[\![o_1]\!][\![\langle o_2, \ldots, o_n \rangle]\!]$ if o_1 is applicable in s, i.e. applying a sequence on a state applies at first the first operator of the sequence on the state and afterwards the rest of the sequence. We call the sequence $\langle o_1, o_2, \ldots, o_n \rangle$ of operators applicable in state s if o_k is applicable in state $s[\![\langle o_1, \ldots, o_{k-1} \rangle]\!]$ for each $k \in \{1, \ldots, n\}$ and the state $s[\![\langle o_1, \ldots, o_k \rangle]\!]$ is **visited** by π for each $k \in \{0, \ldots, n\}$. We call a sequence π of operators that is applicable in s a **path** from s to $s[\![\pi]\!]$. If a path from s to s' exists we call s' reachable from s. The operators o and o' are **inverse** of each other if there exists a state s with $s[\![o]\!][\![o']\!] = s$ (independent of applicability). We define the cost of a sequence π as the sum of the cost of the individual operators $cost(\pi) = \sum_{i=1}^{n} cost(o_i)$. The notation $d(s, s')$ is used for the **distance** from s to s' which is the cost of the path from s to s' with minimal cost ($d(s, s') = \infty$ if no such path exists), such a path is called **shortest** path from s to s'. Note that the distance is not symmetric.

The partial assignment γ describes the **goal**. A path π from a state s where $s[\![\pi]\!]$ agrees with γ is called a **plan** from s. If a plan from s exists we call s **solvable** otherwise we call s a **dead end**. I is the **initial state** of the planning task and if a path from I to a state s does exist we call s **reachable**. A state is called **alive** if it is reachable and solvable. If I is solvable (i.e. a path from the initial state to a state that agrees with the goal does exist) we call the planning task solvable. A plan is **optimal** if it is a plan with minimal cost.

An operator o is **critical** (Seipp et al., 2016) in state s if each plan from s contains o. An operator o is critical in the task Π if there exists an alive state s where o is critical in s. (o being critical is equivalent to o being an action landmark.) An operator o is **dangerous** (Seipp et al., 2016) in state s if s is solvable, o is applicable in s, and $s[\![o]\!]$ is unsolvable. An operator o is dangerous in the task Π if there exists an alive state s where o is dangerous in s.

2.2 Heuristics

Finding a plan to solve the planning task is hard, especially for planning tasks with many state variables or many applicable operators in individual states. To reduce the search time we use guidance that estimates how promising a given state is. We represent this guidance as a **heuristic** function h for the planning task Π that maps each state from Π to a value in $\mathbb{R} \cup \{\infty\}$ where smaller values represent more promising states. Note that we allow the heuristic to map onto negative values in this work which is often not the case.

There are many heuristics, classes of heuristics, and properties of heuristics that are well established in the literature. The heuristic that maps each state s to the cost of the optimal plan from s if one exists and ∞ otherwise is called the **perfect heuristic** h^*. Computing the perfect heuristic for a state s is as hard as finding the plan from s. For this reason, h^* is not used in practice but only for comparison. The **goal count heuristic** $h^{\#g}$ is defined as the number of facts from γ that are not in the state s, i.e. $h^{\#g}(s) := |\gamma \setminus s|$. This heuristic is very easy to evaluate for a given state but is very uninformed as it ignores the operators completely.

Additionally, we want to talk about the **delete relaxed heuristic**[3] h^+. We call a sequence $\pi^+ = \langle o_1, \ldots, o_n \rangle$ of operators a **delete relaxed plan** from s if $\mathit{pre}(o_k) \subseteq s \cup \bigcup_{i=1}^{k-1} \mathit{eff}(o_i)$ for each $k \in \{1, \ldots, n\}$ and $\gamma \subseteq s \cup \bigcup_{i=1}^{n} \mathit{eff}(o_i)$ (note that these assignments are generally not valid). The intuition for delete relaxation is that facts are not overwritten by the operator effects but added. The delete relaxed heuristic value $h^+(s)$ is defined as the minimal cost of all delete relaxed plans from s if one exists and ∞ otherwise. Evaluating the h^+

[3]The name *delete* relaxed might be confusing in the context of a finite domain representation because we never talked about deletions in the first place. The name comes originally from STRIPS (Fikes and Nilsson, 1971) planning tasks where each operator is defined with an add-effect and a delete-effect and the delete relaxation ignores these delete-effects (Bonet and Geffner, 2001; Hoffmann and Nebel, 2001). However, the concept can be generalized to SAS$^+$ tasks (Domshlak et al., 2015) and the transformation from SAS$^+$ to STRIPS or vice versa produces with the given definition the same values for the delete relaxed heuristic on equivalent states so we stay with this name.

value of a given state is NP-hard (Bylander, 1997) so it is not feasible to use in practice despite being well informed.

Potential heuristics are a class of heuristics introduced by Pommerening et al. (2015). A **potential heuristic** is a heuristic that is computed with a weighted count of the partial assignments that agree with the given state.

$$h^{pot}(s) = \sum_{p \in \mathcal{P}} (w(p) \cdot [p \subseteq s])$$

where $\mathcal{P}$ is the set of all possible partial assignments for the task, $[p \subseteq s]$ is in the Iverson bracket notation, and $w(p)$ is the weight for the partial assignment p. In practice most of the weights are 0. The **dimension** of a potential heuristic is $max_{p \in \mathcal{P}, w(p) \neq 0} |p|$. For planning tasks with n state variables the evaluation of a potential heuristic of dimension d can be performed in time $\mathcal{O}(n^d)$ (Seipp et al., 2016). So potential heuristics of small dimension are quick to evaluate but finding good weights that makes the potential heuristic informed is still complex.

2.3 State Space

An intuitive way to visualize planning tasks is a directed graph with nodes that represent the states and an outgoing arc for each operator that is applicable in this state that points to the node that represents the corresponding successor state. We call this graph the **state space** of the planning task. We formally define the state space $\mathcal{S}$ induced by the planning task $\Pi = \langle V, I, O, \gamma \rangle$ as the directed, arc-labeled, arc-weighted graph $\langle S, E, G, s_0 \rangle$ where S, the set of states of the planning task is the set of nodes of the graph and $E = \{\langle s, o, c, s' \rangle \in S \times O \times \mathbb{R} \times S \mid o \text{ applicable in } s, s[\![o]\!] = s', c = cost(o)\}$ is the set of labeled, weighted arcs we call the **state transitions**. The set $G \subseteq S$ contains all states that agree with γ. We call G the **goal states** and $s_0 \in S$ is the initial state. The pair of a state space and a heuristic $\langle \mathcal{S}, h \rangle$ is called a **state space topology**. The heuristic in the state space topology can be interpreted as a weighting of the graph that is the state space so we get an arc-labeled, arc-weighted, node-weighted directed graph. Note that with this definition of the state space topology as a graph the terms successor of s and path from s to s' as well as its cost and distance from graph theory are equivalent to the definition we gave in the SAS$^+$ context.

A heuristic h is **descending** if each alive state s has a successor s' with $h(s) > h(s')$. A heuristic h is **dead-end avoiding** if it holds for each alive state s that each successor s' of s with $h(s') < h(s)$ are solvable (Seipp et al., 2016).

2.4 Search Algorithms

One of the most basic algorithms used to find a path to a goal state is `Breadth First Search` (Russell and Norvig, 2010). In the algorithm, we first test if the

Algorithm 1: `Breadth First Search`

```
Data: planning task Π
Result: plan π
if γ ⊆ I then
    return empty plan
end
open := [I]
closed := {I}
while open is not empty do
    s := pop first element of open
    foreach s' ∈ succ(s) do
        if γ ⊆ s' then
            return extracted path to s'
        end
        if s' ∉ closed then
            insert s' in closed
            append s' to open
        end
    end
end
return fail
```

initial state is itself a goal state. If this is not the case we initialize the open list and the closed set both containing the initial state. While the open set is not empty we remove the first element from it and iterate through each of its successors. If the successor is not a goal state nor in the closed set we insert it into the closed set and append it to the open list. See Algorithm 1.

The `Breadth First Search` is uninformed, it does not use any heuristic guidance.

3 Previous Work

This chapter summarizes the previous work by Hoffmann (2001, 2002), by Chen and Giménez (2007), by Lipovetzky and Geffner (2012), and by Seipp et al. (2016) that introduced different complexity measures for planning instances. Each implies an upper bound to the search time that is exponential only in the measure.

Before we begin with the summaries we first describe a running example. Afterward, we introduce a general framework for the search algorithms. We describe the search algorithms that are used in these four different works with this framework. This helps us to make the differences and similarities between them more apparent.

3.1 Running Example

We will illustrate the following methods on the well-known *crossing the river* task. A person wants to put a fox, a rabbit, and a carrot from the west shore to the east shore of a river, but his boat has only enough space for himself and one of the objects and the person cannot leave the fox with the rabbit nor the rabbit with the carrot on the same shore. Expressed as a planning task:

$$\begin{aligned}\Pi =& \langle V, O, I, \gamma \rangle \\ V =& \{v_{fox}, v_{rabbit}, v_{carrot}, v_{boat}\} \\ dom(v) =& \{\textit{West, East}\} \text{ for all } v \in V\end{aligned}$$

For simplicity we substitute in all facts $v_{fox} \mapsto \textit{East}$ with F and $v_{fox} \mapsto \textit{West}$ with f. We use the initial letter in uppercase or lowercase of the other state variables analogously. For example, state $\{f, R, c, B\}$ is the state where the rabbit and the boat are on the east shore and the fox and the carrot are on the west shore. The state space is visualized in Figure 1.

$$\begin{aligned}O =& \{\langle\{f,r,c,b\},\{R,B\}\rangle, \langle\{f,R,c,B\},\{r,b\}\rangle, \\ & \langle\{f,R,c,B\},\{b\}\rangle, \langle\{f,R,c,b\},\{B\}\rangle, \\ & \langle\{f,R,c,b\},\{F,B\}\rangle, \langle\{F,R,c,B\},\{f,b\}\rangle, \\ & \langle\{f,R,c,b\},\{C,B\}\rangle, \langle\{f,R,C,B\},\{c,b\}\rangle, \\ & \langle\{F,R,c,B\},\{r,b\}\rangle, \langle\{F,r,c,b\},\{R,B\}\rangle, \\ & \langle\{f,R,C,B\},\{r,b\}\rangle, \langle\{f,r,C,b\},\{R,B\}\rangle, \\ & \langle\{F,r,c,b\},\{C,B\}\rangle, \langle\{F,r,C,B\},\{c,b\}\rangle, \\ & \langle\{f,r,C,b\},\{F,B\}\rangle, \langle\{F,r,C,B\},\{f,b\}\rangle, \\ & \langle\{F,r,C,B\},\{b\}\rangle, \langle\{F,r,C,b\},\{B\}\rangle, \\ & \langle\{F,r,C,b\},\{R,B\}\rangle, \\ & \langle\{F,R,C,B\},\{r,b\}\rangle\} \\ I =& \{f,r,c,b\} \\ \gamma =& \{F,R,C\}\end{aligned}$$

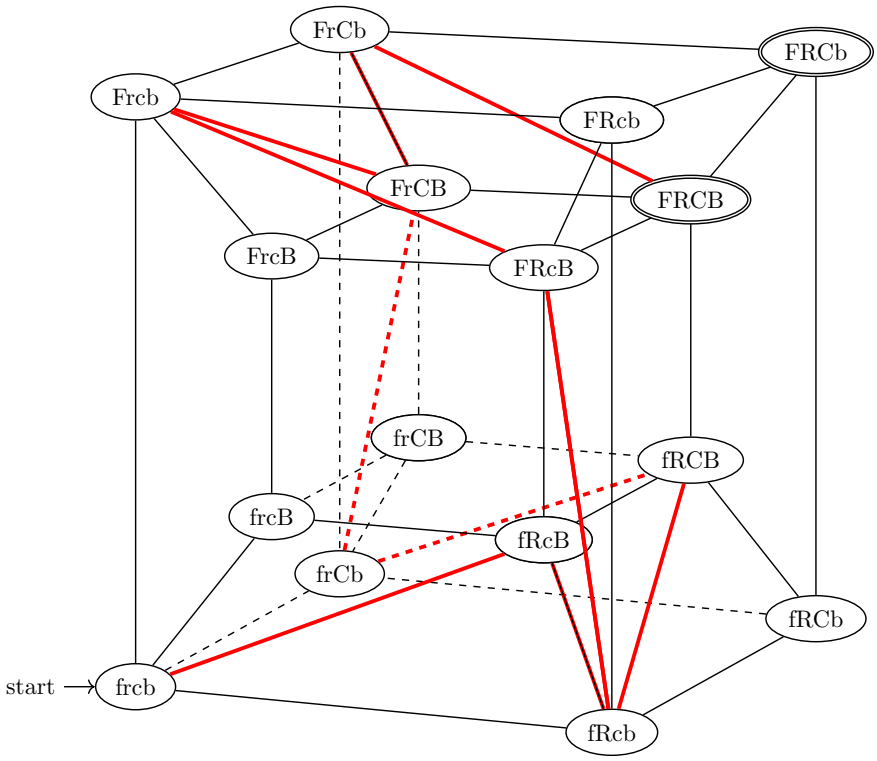


Figure 1: State space for the *crossing the river* task. Note that the black edges are not the state transitions but a visualization of the 4-dimensional hypercube. The red lines indicate the state transitions.

3.2 Search Framework

We introduce a framework to have the upcoming algorithms in a similar form. We call it the `Search Framework`. This allows us to specify a search algorithm by only the 3 subroutines `progressCheck`, `expandCheck` and `updateClosed`.

The pseudocode for the `Search Framework` is shown in Algorithm 2. (So far we did not explain what the width $k \in \mathbb{N}$ is. We get to it in Section 3.4.)

In the first 7 lines is the initialization. There the initial state I of the planning task Π is testes for agreement with the goal γ. If it does agree the search is already over. If it does not agree, then the *open* list is initialized as containing only the initial state. The *closed* set is first initialized as an empty set but immediately updated by the subroutine `updateClosed` depending on I and the width k. The reference state *reference* is set to the initial state.

After the initialization, the main while loop runs as long as the *open* list contains at least one state. Inside the while loop, we pop the first element of the *open* list and make it to the current state *current*. Then we iterate (in an

Algorithm 2: `Search Framework`

```
   Data: planning task Π = ⟨V, I, O, γ⟩, width k ∈ ℕ, heuristic h,
         subroutines updateClosed, progressCheck, expandCheck
   Result: plan π
 1 if γ ⊆ I then
 2 |  return empty plan
 3 end
 4 open := [I]
 5 closed := ∅
 6 updateClosed(I, closed, k)
 7 reference := I
 8 while open is not empty do
 9 |  current := pop first element of open
10 |  foreach candidate ∈ succ(current) do
11 |  |  if γ ⊆ candidate then
12 |  |  |  return extracted path to candidate
13 |  |  else if progressCheck(candidate, reference, h) then
14 |  |  |  open := [candidate]
15 |  |  |  closed := ∅
16 |  |  |  updateClosed(candidate, closed, k)
17 |  |  |  reference := candidate
18 |  |  |  break
19 |  |  else if expandCheck(candidate, closed, k) then
20 |  |  |  updateClosed(candidate, closed, k)
21 |  |  |  append candidate to open
22 |  |  end
23 |  end
24 end
25 return fail
```

arbitrary order) over all states *candidate* that are a successor of *current*.

Each *candidate* is tested for agreement with the goal γ. If it does the search is over and the path to *candidate* is returned. If it does not agree, then *candidate* is tested for providing progress compared to *reference* for this we call the subroutine `progressCheck`. If progress was made the *open* list, the *closed* set and the *reference* state are re-initialized like in lines 4-7 but with the *candidate* state instead of the initial state I and the foreach loop breaks.

If *candidate* does not agree with the goal nor does it provide progress, then *candidate* is tested if it should be expanded further. We call the subroutine `expandCheck` for this. If *candidate* should be expanded further we call the subroutine `updateClosed` on *candidate* to add elements into the *closed* set and we append the *candidate* to the *open* list. If the while loop ends the search terminates with a fail return.

There is also a width $k \in \mathbb{N}$ and a heuristic h in the input data of the `Search Framework`. These are only used in some of the subroutines. As an example we can reproduce the `Breadth First Search` with the `Search Framework` framework by using the subroutines Algorithm 3, 4, and 5.

Algorithm 3: `progressCheck` (Breadth First Search)

Data: states *candidate*, *reference*, heuristic h
1 **return** *false*

Algorithm 4: `expandCheck` (Breadth First Search)

Data: state *candidate*, set *closed*, $k \in \mathbb{N}$
1 **return** $candidate \notin closed$

Algorithm 5: `updateClosed` (Breadth First Search)

Data: state s, set *closed*, $k \in \mathbb{N}$
1 insert s into *closed*
2 **return**

Plugging these subroutines into the the `Search Framework` results in an algorithm equivalent to the `Breadth First Search` from Algorithm 1. Since the `progressCheck` in line 13 is never fulfilled we can remove the lines 13-18 without changing the behavior of the algorithm. Replacing lines 5 and 6 with the line $closed = \{I\}$ does not change anything of the behavior as well. With these two modifications, we directly see that the `Search Framework` with the mentioned subroutines is, in fact, equivalent to the `Breadth First Search` from Algorithm 1.

In the subroutines of `Breadth First Search`, we see that the heuristic h is not used. This is because `Breadth First Search` does not use any heuristic guidance. We also see that the width k is not used.

Each measure uses a specific algorithm. We will represent its corresponding algorithm in a table like Table 1. This representation of the measure shows only the name and the subroutines for the `Search Framework` that specify the corresponding algorithm. This makes the similarities and differences of the algorithms more apparent.

3.3 Maximal Exit Distance

The empirical and theoretical analysis of Hoffmann (2001, 2002) on local search topology considers the delete relaxed heuristic h^+. This heuristic is not guaranteed to be dead-end avoiding because some states are easy to solve in the delete relaxation but are dead-ends in the original task. However, Hoffmann shows

Name	Name of the algorithm
`progressCheck`	what does `progressCheck` return
`expandCheck`	what does `expandCheck` return
`updateClosed`	what does `updateClosed` do (besides return)

Table 1: Pattern to represent an algorithm in the `Search Framework` compactly.

that a plan can be found with this heuristic guidance in a time only exponential to the *maximal exit distance* if there is no local minimum.

In the *crossing the river* example, an optimal relaxed plan traverses through the invalid assignments

$$I \cup \bigcup_{i=1}^{0} \mathit{eff}(o_i) = \{f, r, c, b\},$$

$$I \cup \bigcup_{i=1}^{1} \mathit{eff}(o_i) = \{f, r, c, b, R, B\},$$

$$I \cup \bigcup_{i=1}^{2} \mathit{eff}(o_i) = \{f, r, c, b, R, B, F\},$$

$$I \cup \bigcup_{i=1}^{3} \mathit{eff}(o_i) = \{f, r, c, b, R, B, F, C\} \supseteq \gamma$$

and has the initial heuristic value $h^+(I) = 3$. Figure 2 shows the h^+ value for each state of the *crossing the river* task.

We see in Figure 2 that the *crossing the river* task with h^+ is not descending. The initial state has no successor that improves the heuristic value. The delete relaxed heuristic is dead-end avoiding in the *crossing the river* task but not in general.

Hoffmann (2001) partitioned the states of the state space topology into different plateaus. A **plateau** of **level** l is a maximal subgraph P of a given state space topology $\langle \mathcal{S}, h \rangle$ where all states $s \in P$ have the heuristic value $h(s) = l$. A state $e \in P$ is called an **exit** if e has an successor e' that is not in the same plateau and $h(e') < h(e)$. A **local minimum** is a plateau without any exit nor goal state. A path is **flat** if it traverses only through states of the same heuristic value.

Definition 3.1 (Maximal Exit Distance)**.** The **exit distance** for a state $s \in \mathcal{S}$ is the minimal cost of a flat path from s to an exit. The **maximal exit distance** of a task is defined as the maximum exit distance over all states.

In Figure 2 we see on the left a plateau of level 3 with state *fRcB* as exit. All states with the heuristic value of 2 are in the same plateau. The state *fRcb* has an exit distance of 3 because *FrCB* is the only exit in this plateau and there exists a flat path from *fRcb* via *FRcB* and *Frcb* to *FrCB* of cost 3. No state in

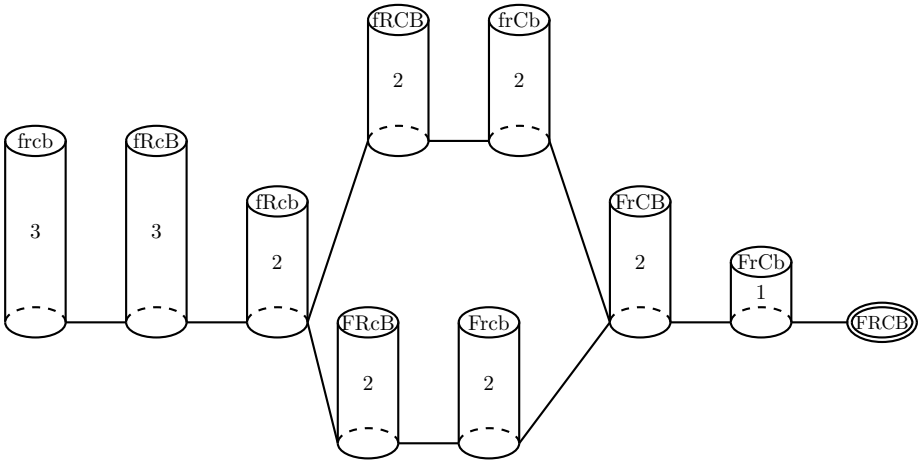


Figure 2: Graphical representation of the h^+ heuristic. The height of the cylinders represents the heuristic value of the state.

the task has a greater exit distance and therefore the maximal exit distance of the *crossing the river* task is 3.

For this measure `Enforced Hill-climbing` (see the subroutines for the `Search Framework` in Table 2) is used as search algorithm. It acts the same as `Breadth First Search` (`expandCheck` and `updateClosed` are the same in both algorithms), until a state with a better heuristic value than *reference* is found. There it detects progress and re-initializes.

A plan is guaranteed to be found on state spaces without local minimums after considering $\mathcal{O}(h(s_0) \cdot b^{c+1})$ states, where b is the maximal branching factor and c is the maximal distance of a state s to the exit of its corresponding plateau. This is exponential only in the maximal exit distance c. The downside of this approach is that the heuristic has to be precomputed. Another issue is that h^+ might have local minimums for the given task and even if it does not the computation of the delete relaxed heuristic h^+ is NP-hard. It is not essential to use the h^+ heuristic. Any heuristic without local minimums could be used.

Name	`Enforced Hill-climbing`
`progressCheck`	$h(\mathit{candidate}) < h(\mathit{reference})$
`expandCheck`	$\mathit{candidate} \notin \mathit{closed}$
`updateClosed`	insert *candidate* into *closed*

Table 2: `Enforced Hill-climbing` represented compactly.

3.4 Improvability Width

Chen and Giménez Chen and Giménez (2007) introduced an algorithm that is exponential only in the improvability width that finds a satisficing plan for a given task without dead ends. They defined multiple slightly different improvability width notions where all of them are based on the *improvability*. So we define this first.

Definition 3.2 (k-Improvable)**.** Let $\Pi = \langle V, O, I, \gamma \rangle$ be a planning task. A path π **improves** a variable $u \in V$ in state s if:

- for all $v \in \mathit{vars}(\gamma)$: $s(v) = \gamma(v)$ implies $s[\![\pi]\!](v) = \gamma(v)$; and
- if $u \in \mathit{vars}(\gamma)$, then $s[\![\pi]\!](u) = \gamma(u)$

A variable v in state s is **k-improvable** if a path π that improves v and $|\bigcup_{o \in \pi} \mathit{vars}(\mathit{eff}(o))| \leq k$ exists.

Note that a variable v that is k-improvable in state s is also $(k{+}1)$-improvable in s due to the inequality in the definition. The concept k-improvable in a state s is defined for all state variables but it is only interesting for variables $u \in \mathit{vars}(\gamma)$ with $s(u) \neq \gamma(u)$ because the others are trivially 0-improvable with the empty path.

So an improving path increases (or does not change) the number of variables that are in goal position and each variable that was in goal position, in the beginning, is still there after applying the path.

For example, a variable is 2-improvable in state s if there is an improving path from s that only acts on one fixed face of the hypercube and a variable is 3-improvable in s if there is an improving path from s that only acts on one fixed cell of the hypercube (Figure 3).

For the initial state of the *crossing the river* task the state variables *fox* and *carrot* are 3-improvable (and not 2-improvable) because each path from the initial state that ends in a state s where $s(\mathit{fox}) = \gamma(\mathit{fox})$ (analogous for *carrot*) has effects on the variable *boat* and *rabbit*.

With $\mathit{wrong}(s) := \{v \in \text{vars}(\gamma) \mid s(v) \neq \gamma(v)\}$ we denote the variables in s that differ from the goal.

Definition 3.3 (Improvability Width)**.** A planning task has **improvability width** k if it is unsolvable, or for every reachable state s that is not a goal state, every variable $u \in \mathit{wrong}(s)$ is k-improvable in s.

Definition 3.4 (Persistent Improvability Width)**.** A planning task has **persistent improvability width** k if it is unsolvable, or for every reachable state s that is not a goal state, there exists a variable $u \in \mathit{wrong}(s)$ such that u is k-improvable in s.

They defined the Hamming improvability width and persistent Hamming improvability width in a similar fashion based on the k-Hamming improvability.

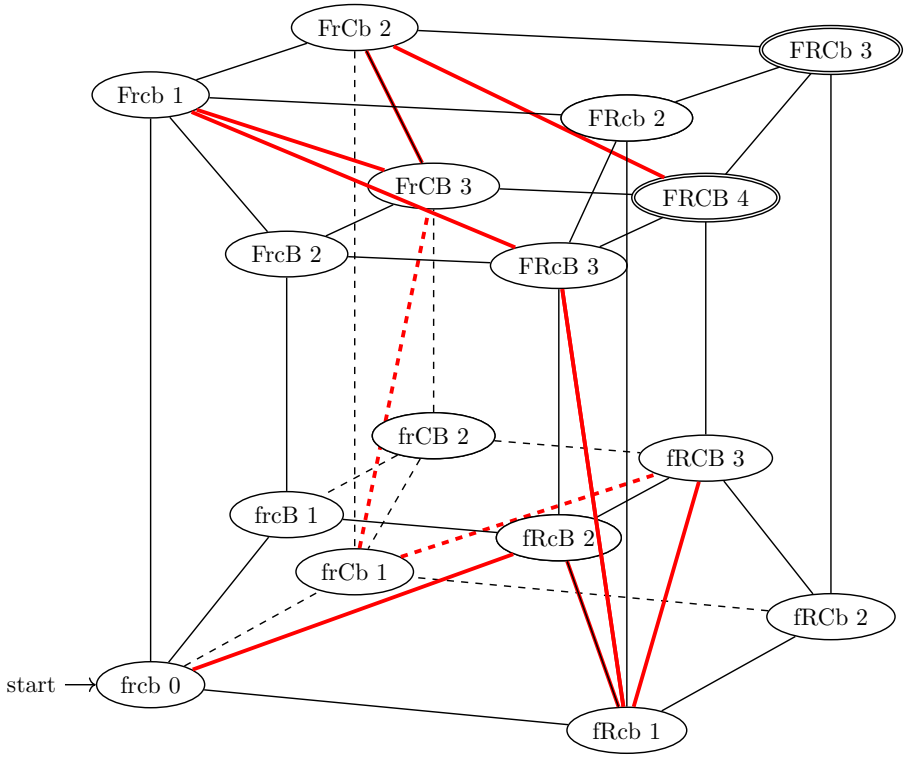


Figure 3: State space for the *crossing the river* task. Note that the black edges are not the state transitions but a visualization of the 4-dimensional hypercube. The number in each state shows the Hamming distance to the initial state *frcb*. The red lines indicate the state transitions.

Definition 3.5 (k-Hamming Improvable)**.** The Hamming distance $d_h(.,.)$ of two states s, s' is defined as $d_h(s,s') = |\{v \in V \mid s(v) \neq s'(v)\}|$.

A state variable u in state s is k-Hamming improvable if an improving path $\pi = o_1, \ldots, o_n$ exists such that $d_h(s, s[\![o_1, \ldots, o_i]\!]) \leq k$ for each $i \in \{1, \ldots, n\}$.

See for an example the Hamming distance from the initial state to each state in the hypercube (Figure 3). A variable is 2-Hamming improvable in state s if there is an improving path from s that only acts on the faces of the hypercube that are connected to s and a variable is 3-Hamming improvable in s if there is an improving path from s that only acts on the cells of the hypercube that are connected to s.

Definition 3.6 (Hamming Improvability Width)**.** A planning task has **Hamming improvability width** k if it is unsolvable, or for every reachable state s that is not a goal state, every variable $u \in \mathit{wrong}(s)$ is k-Hamming improvable in s.

Definition 3.7 (Persistent Hamming Improvability Width)**.** A planning task has **persistent Hamming improvability width** k if it is unsolvable, or for every reachable state s that is not a goal state, there exists a variable $u \in \mathit{wrong}(s)$ such that u is k-Hamming improvable in s.

If we know that a task has a bounded (persistent) (Hamming) improvability width of k, we can use the `Improvability Width Algorithm` (`IWA`). See subroutines in Table 3. It does a `Breadth First Search` but only on the part of the state space graph where each state differs at most in k-many variables from the reference state. The size of this state space graph is exponential only in k. Progress is made if an improving state is found. Progress can be made at most $|\gamma|$ times. Therefore, the search time is exponential only in k. For the `Improvability Width Algorithm` is no precomputation needed. No heuristic guidance is used.

We only show the table of the variant that uses the persistent Hamming improvability width because it is the most general.

Name	`Improvability Width Algorithm`
`progressCheck`	$\mathit{wrong}(\mathit{candidate}) \subsetneq \mathit{wrong}(\mathit{reference})$
`expandCheck`	$\mathit{candidate} \notin \mathit{closed}$ and $d_h(\mathit{candidate}, \mathit{reference}) \leq k$
`updateClosed`	insert *candidate* into *closed*

Table 3: `Improvability Width Algorithm` represented compactly.

3.5 Novelty Width

Lipovetzky and Geffner (2012) described an iterative algorithm that calls `Novelty Width Algorithm` (`NWA`) for the same task and a number k that is incremented each iteration until a plan is found. `Novelty Width Algorithm` (see the subroutines in Table 4) is a modification of `Breadth First Search` to find a plan with complexity that is only exponential in the novelty width. In the case of a unit cost task, this plan is also optimal.

The difference to `Breadth First Search` is the stricter pruning. `Novelty Width Algorithm` prunes not only the states that are duplicates (i.e the states in the *closed* set) but all states with a *novelty* larger than k. The novelty of a state s is the size of the smallest partial assignment p that agrees with s but no other state that was generated previously in the search. The amount of partial assignments of size k is exponential only in k. No partial assignment p can be used twice to make `expandCheck` return true. This is due to the fact that `updateClosed` puts p into *closed* as soon as `expandCheck` returns true. Therefore, the search time is exponential only in k. No precomputation is required and no heuristic guidance is used.

Figure 4 shows the search tree of a `Novelty Width Algorithm` with $k = 3$ on the *crossing the river* task. We see in depth 2 that this `Novelty Width Algorithm` with $k = 1$ halts because the state *fRcb* counts as duplicate. A

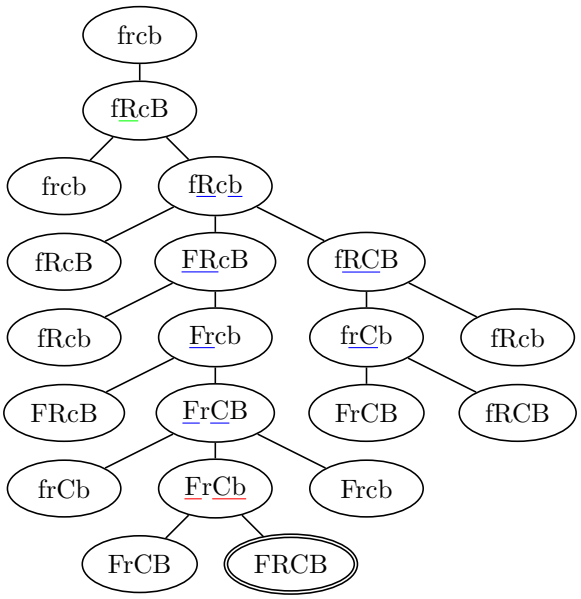


Figure 4: Search tree of a `Novelty Width Algorithm` with $k = 3$. States with novelty 1, 2, and 3 have a smallest new partial assignment underlined in green, blue and red respectively. The tie-breaking is indicated by nodes of the same depth being more to the left if they are generated earlier.

Name	`Novelty Width Algorithm`
`progressCheck`	false
`expandCheck`	$\exists p \subseteq$ *candidate* with $\lvert p\rvert \leq k, p \notin$ *closed*
`updateClosed`	insert each $p \subseteq$ *candidate* with $\lvert p\rvert = k$ into *closed*

Table 4: `Novelty Width Algorithm` represented compactly.

`Novelty Width Algorithm` with $k = 2$ would halt at depth 6 because *FrCb* contains no pair that is new to the search tree, only a new triple.

The novelty width is defined with the graph $\mathcal{G}^i$.

Definition 3.8. For $\Pi = \langle V, I, O, \gamma \rangle$ the graph $\mathcal{G}^i$ is defined inductively by:

- the partial assignment p of size $|p| = i$ is a root in $\mathcal{G}^i$ iff p agrees with I.
- $\langle p, p' \rangle$ is a directed edge in $\mathcal{G}^i$ iff p is in $\mathcal{G}^i$ and for every optimal plan π for $\Pi(p)$ there is an operator $o \in O$ such that π followed by o is an optimal plan for $\Pi(p')$ and $|p'| = i$.

Definition 3.9 (Novelty Width)**.** A formula φ on the state variables V is of **novelty width** 0 if it is true in I. Otherwise its novelty width is the smallest w such that $\mathcal{G}^w$ contains a partial assignment p with $p \models \varphi$.

The novelty width of a planning task Π is the novelty width of the formula that is the conjunction of all elements in γ (the goal formula).

A `Novelty Width Algorithm` can find a sub-optimal plan if k is less than the novelty width. This can happen if the states of the optimal path are categorized as a duplicate due to large novelty while all states on a sub-optimal path provide a lower novelty. We call the smallest k for which a `Novelty Width Algorithm` finds a satisficing plan the *effective* novelty width.

Lipovetzky and Geffner (2012) split multiple IPC domains with N atomic goals into N problems with single goals and tested them for their effective novelty width. Less than 12% of them had an effective novelty width greater than 2. Note that it is possible for two different states that are in the same depth of the search tree to be seen as a duplicate of each other, this makes the effective novelty width dependent on the tie-breaking.

Considering Figure 4 again. All nodes in the example that are not expanded are true duplicates. Therefore, each tie-breaking strategy would not expand them. This implies that the novelty width, as well as the effective novelty width, is 3 for the *crossing the river* task.

3.6 Correlation Complexity

Correlation complexity is another measure for the complexity of planning tasks introduced by Seipp et al. (2016) which is based on potential heuristics. They looked for potential heuristics that are descending and dead-end avoiding (DDA).

Definition 3.10 (Correlation Complexity)**.** The **correlation complexity** of a planning task Π is defined as the minimal dimension of all descending, dead-end avoiding potential heuristics for Π.

Finding a plan with the use of a DDA heuristic is easily done with `Simple Hill-climbing` (`SHC`). The subroutines are shown in Table 5. This algorithm works by starting at the initial state and repeatably updating the current state with a successor that provides progress until a goal state is found. This can be interpreted as wandering downwards in the state space. The number of state expansions is bounded by $\mathcal{L} = h^{pot}(I) - \min_{s \in S} h^{pot}(s)$ if all weights are integers (Seipp et al., 2016).

In contrast to Section 3.3 the heuristic h in `progressCheck` is not fully precomputed. Only the weights for a DDA potential heuristic on the task are precomputed. In `progressCheck` the value of h^{pot} is evaluated which is a sum over the precomputed weights. The amount of summands is exponential only in the dimension of the potential heuristic. Therefore, the search time is exponential only in the correlation complexity.

Name	`Simple Hill-climbing`
`progressCheck`	$h^{pot}(candidate) < hpot(reference)$
`expandCheck`	false
`updateClosed`	-

Table 5: `Simple Hill-climbing` represented compactly.

Seipp et al. investigated multiple IPC planning domains and showed that all of them had a correlation complexity of 2 but it is possible to construct planning tasks with arbitrarily large correlation complexity.

It is impossible to find a potential heuristic for the *crossing the river* task that is descending. To show this we consider the state transition from *frcb* to *fRcB* that brings the boat and the rabbit from the west to the east shore as well as the two state transitions from *fRCB* to *frCb* and *fRCB* to *frCb* that bring the rabbit and the boat from the east to the west shore. The first state transition is critical and has to be descending for each DDA heuristic. For the first state transition to be descending with a 1-dimensional potential heuristic it is required that $w(f) + w(r) + w(c) + w(b) + w(\emptyset) > w(f) + w(R) + w(c) + w(B) + w(\emptyset)$ which implies $w(r) + w(b) > w(R) + w(B)$. However, at least one of the other two state transitions has to be descending as well. For the last state transition to be descending with a 1-dimensional potential heuristic would imply that $w(f) + w(R) + w(C) + w(B) + w(\emptyset) > w(f) + w(r) + w(C) + w(b) + w(\emptyset)$ which implies $w(R) + w(B) > w(r) + w(b)$ (analogous for the other state transition). This causes a contradiction.

The *crossing the river* task has correlation complexity of 2, too. With the weights from Table 6 it is possible to create a descending, dead-end avoiding potential heuristic (see Figure 5). The computation of the weights for a DDA potential heuristic can be expressed as a mixed integer program (Francès et al., 2019).

partial state	weight
f	5
fr	3
Fc	2
cB	2
r	1
fB	1
rB	1

Table 6: Partial states and corresponding weight for a potential heuristic that is DDA (all other partial states have a weight of zero).

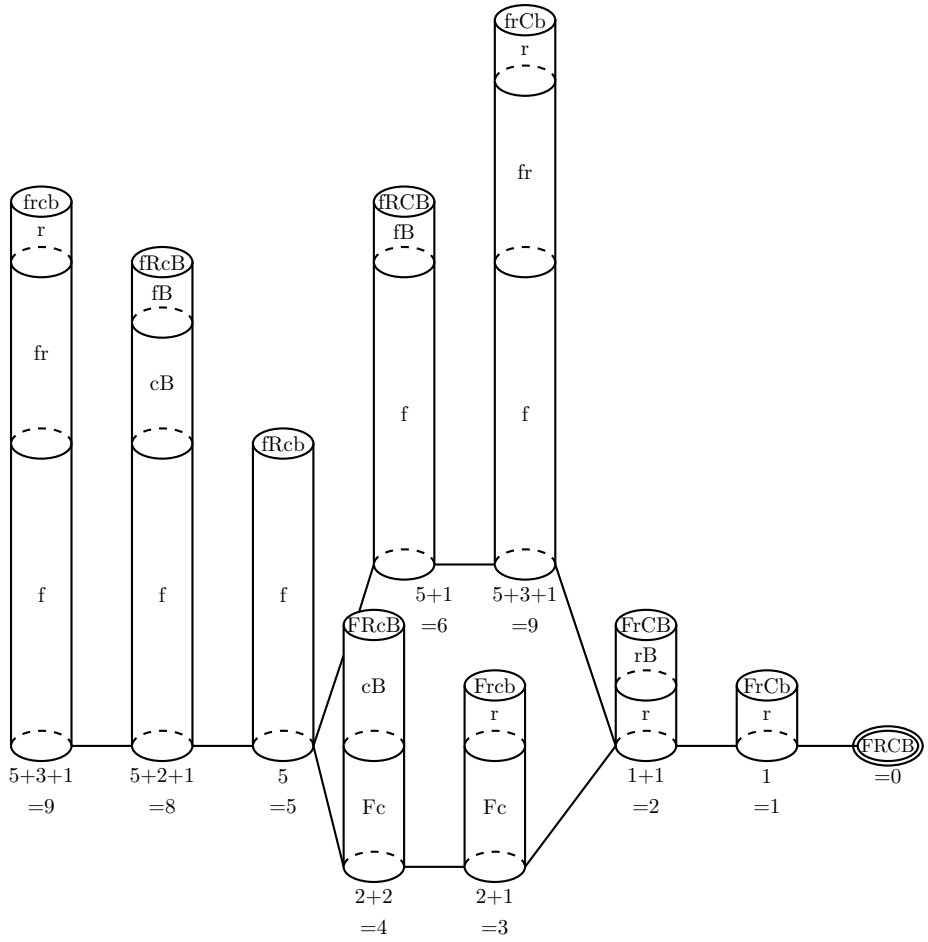


Figure 5: Graphical representation of the potential heuristic with the weights from Table 6. The height of the partial cylinders represents the weight of the partial state.

4 Lower Bounds of the Correlation Complexity

Before we compare the measures we want to understand better what causes a large correlation complexity.

The correlation complexity of a task[4] Π describes the minimal dimension of a potential heuristic h^{pot} such that h^{pot} is DDA. This can be interpreted as a measure of how many facts the agent has to consider at once to find the next best successor. Unless all reachable states are goal states the correlation complexity is at least 1 (the agent has to consider at least *something*). Seipp et al. (2016) introduced two criteria to identify that a task has correlation complexity of at least 2.

Theorem 4.1. *Let Π be a planning task in normal form, and let o and o' be critical operators of Π that are inverses of each other. Then Π has correlation complexity of at least 2 (Seipp et al., 2016)*

Theorem 4.2. *Let Π be a planning task in normal form, and let o be an operator that is critical and dangerous in Π. Then Π has correlation complexity of at least 2 (Seipp et al., 2016)*

These criteria focus on the operators in the task. We want to add two criteria that look from a different angle and focus on the states instead.

4.1 4 States Criterion

We first look at the Theorem 4.3. Roughly speaking it checks if a given potential heuristic can be translated into a 2-dimensional heuristic. Note that each heuristic (that never evaluates ∞, which is sufficient for our purposes) can be translated into a potential heuristic if the dimension is large enough.

Theorem 4.3. *Let $\Pi = \langle V, I, O, \gamma \rangle$ be a planning task, and let h^{pot} be a potential heuristic. If there exist states a, b, c, d in Π and a partition $\{W, M\}$ of V such that:*

$$\begin{aligned} h^{pot}(a) &> h^{pot}(b), \\ h^{pot}(c) &\geq h^{pot}(d), \\ a^W &= b^W, \\ c^W &= d^W, \\ a^M &= d^M, \\ b^M &= c^M, \end{aligned}$$

then the dimension of h^{pot} is at least 2.

[4] In this work we assume that each task is solvable.

For the proof we use the fact that we can split one (finite) sum over all $x \in X$ with $\{Y, Z\}$ partition of X into two sums where one iterates over $y \in Y$ and the other over $z \in Z$.

$$\sum_{x \in X} f(x) = \sum_{y \in Y} f(y) + \sum_{z \in Z} f(z),$$

with f an arbitrary function.

Proof. Let h^{pot} be a heuristic and a, b, c, d states in Π and $\{W, M\}$ a partition of V such that:

$h^{pot}(a) > h^{pot}(b)$ and $h^{pot}(c) \geq h^{pot}(d)$ and $a^W = b^W$ and $c^W = d^W$ and $a^M = d^M$ and $b^M = c^M$.

Assume that the dimension of the potential heuristic h^{pot} is 1. The assumption implies that $h^{pot}(s) = h^{pot}(s^W) + h^{pot}(s^M) - w(\emptyset)$ for each state s. The weight $w(\emptyset)$ is subtracted because $\emptyset \subseteq s^W$ and $\emptyset \subseteq s^M$. Therefore, it is added twice on the right side of the equation but only once on the left. This provides us with the two inequalities

$$h^{pot}(a^W) + h^{pot}(a^M) - w(\emptyset) > h^{pot}(b^W) + h^{pot}(b^M) - w(\emptyset)$$

$$h^{pot}(c^W) + h^{pot}(c^M) - w(\emptyset) \geq h^{pot}(d^W) + h^{pot}(d^M) - w(\emptyset).$$

Since $a^W = b^W$ and $c^W = d^W$ we can simplify the inequalities to

$$h^{pot}(a^M) > h^{pot}(b^M)$$

$$h^{pot}(c^M) \geq h^{pot}(d^M).$$

We know that $a^M = d^M$ and $b^M = c^M$ so we replace these values in the latest inequality and end up with

$$h^{pot}(a^M) > h^{pot}(b^M)$$

$$h^{pot}(b^M) \geq h^{pot}(a^M).$$

which is a contradiction. So the assumption is not true and therefore the dimension of the potential heuristic h^{pot} is at least 2. □

From Theorem 4.3 follows the 4 states criterion.

Theorem 4.4 (4 States Criterion)**.** *Let $\Pi = \langle V, I, O, \gamma \rangle$ be a planning task. If for each potential heuristic h^{pot} that is DDA on Π there exist states a, b, c, d in Π and a partition $\{W, M\}$ of V such that:*

$$h^{pot}(a) > h^{pot}(b),$$
$$h^{pot}(c) \geq h^{pot}(d),$$
$$a^W = b^W,$$
$$c^W = d^W,$$
$$a^M = d^M,$$
$$b^M = c^M,$$

then the correlation complexity of Π is at least 2.

Note that these states and the partition do not have to be the same for all DDA potential heuristics.

Proof. We know that the correlation complexity of a task Π is the minimal dimension over all potential heuristics that are DDA on Π. The condition of the 4 states criterion says that each DDA potential heuristic holds the condition for Theorem 4.3. Therefore, we can apply Theorem 4.3 on each potential heuristic that is DDA on Π. We conclude that the dimension of each potential heuristic that is DDA on Π is at least 2. Therefore, the correlation complexity of Π is at least 2. □

The 4 states criterion is more general than the criteria from Theorems 4.1 and 4.2. To prove that the 4 states criterion is in fact a generalization of the other two we show that if the criteria from Theorem 4.1 or 4.2 can be used, then the criterion from Theorem 4.3 can be used as well. To do this, we first have to prove that the condition of Theorem 4.1, o and its inverse are both critical, implies that $vars(pre(o))$ is a proper subset of V. For this, we will use the following definition and lemma.

Definition 4.5 (cycle-avoiding successor)**.** Let s be a solvable state and $s' \in succ(s)$. We call s' a **cycle-avoiding successor** of s if it exists a plan from s' that does not visit s.

If at least one plan from s' exists and all plans from s' visit s we call it a **cycle-inducing successor** of s.

In other words, if s' is a successor of s and s is a landmark for s' then s' is a cycle-inducing successor of s.

With this definition, we can categorize each successor of s. The successor $s' \in succ(s)$ is either a cycle-avoiding successor of s or a cycle-inducing successor of s or unsolvable. For a fixed s each successor belongs to exactly one of the three categories. However, the state s' can be a cycle-avoiding successor of s but a cycle-inducing successor of s''. In the *crossing the river* task the state fRcB is a cycle-avoiding successor of frcb but fRcB is a cycle-inducing successor of fRcb.

Lemma 4.6. *Let $\Pi = \langle V, I, O, \gamma \rangle$ be a planning task in normal form, and let o and o' be critical operators of Π that are inverses of each other. Then $vars(pre(o)) \subsetneq V$.*

Proof. Let Π be a planning task in normal form, and let o and o' be critical operators of Π that are inverses of each other.

Assume $vars(pre(o)) = V$. This implies that o is in exactly one state $s = pre(o)$ applicable. Since o is critical we know that one alive state exists in where o is critical. Let z be such a state.

Each plan from z includes o because o is critical in z. This implies that for each state x that is solvable and it exists a path from z to x that does not include o it holds that o is critical in x. Since o is critical in z there must be a path that does not include o from z to a state y with o applicable in y (this

could be the empty path. In that case $z = y$). Since s is the only state where o is applicable we conclude $s = y$. We see that o is critical in s because s is reachable from z with a path that does not include o.

With o is critical in s and s is the only state where o is applicable we conclude that $s[\![o]\!]$ is the only cycle-avoiding successor of s.

Let $s' = \mathit{pre}(o')$. We argue analogously that $s'[\![o']\!]$ is the only cycle-avoiding successor of s'.

With o, o' inverse of each other and $\mathit{vars}(\mathit{pre}(o)) = V$ we conclude that $s = s'[\![o']\!]$ and $s' = s[\![o]\!]$.

We, therefore, see that each cycle-free plan from s visits s' and each cycle-free plan from s' visits s which implies a cycle and thereby a contradiction.

We finally conclude that the assumption $\mathit{vars}(\mathit{pre}(o)) = V$ is not true and therefore $\mathit{vars}(\mathit{pre}(o)) \subsetneq V$ holds. □

Corollary 4.7. *The 4 states criterion is a generalization of the criterion from Theorem 4.1.*

We show that the condition of the criterion from Theorem 4.1 implies the condition of the 4 states criterion.

Proof. The correlation complexity requires by definition a descending, dead-end avoiding heuristic. In this proof, we focus on the descending property.

If $\Pi = \langle V, I, O, \gamma \rangle$ is a planning task in normal form with the operators o and o' that are inverse of each other and critical in Π, then for each plan π there exist states s, $s[\![o]\!]$, s', $s'[\![o']\!]$ that are visited by π (with s visited before $s[\![o]\!]$ and s' visited before $s'[\![o']\!]$ and o applicable in s and o' applicable in s'). For a heuristic h to be descending the inequalities $h(s) > h(s[\![o]\!])$ and $h(s') > h(s'[\![o']\!])$ have to hold.

Since Π is in normal form and o is the inverse of o' we conclude $\mathit{vars}(\mathit{eff}(o)) = \mathit{vars}(\mathit{eff}(o')) =: M$ and $\mathit{pre}(o)^M = \mathit{eff}(o')$ and $\mathit{pre}(o')^M = \mathit{eff}(o)$, because each operator in normal form has $\mathit{vars}(\mathit{eff}(o)) \subseteq \mathit{vars}(\mathit{pre}(o))$.

With o applicable in s and $M \subseteq \mathit{vars}(\mathit{pre}(o))$ we conclude $s^M = \mathit{pre}(o)^M$, analogous $s'^M = \mathit{pre}(o')^M$. A similar argument for the effect, with $M = \mathit{vars}(\mathit{eff}(o))$ we conclude $s[\![o]\!]^M = \mathit{eff}(o)$, analogous $s'[\![o']\!]^M = \mathit{eff}(o')$. This implies that $s^M = s'[\![o']\!]^M$ and $s'^M = s[\![o]\!]^M$.

Let $W := V \setminus M$. As $\mathit{vars}(\mathit{eff}(o)) = \mathit{vars}(\mathit{eff}(o')) = M$ we see that $s^W = s[\![o]\!]^W$ and $s'^W = s'[\![o']\!]^W$ because o and o' act only on variables in M.

Considering that o is critical. This implies that $\emptyset \neq \mathit{eff}(o) = M$. With $\mathit{vars}(\mathit{eff}(o)) \subseteq \mathit{vars}(\mathit{pre}(o))$ and Lemma 4.6 we conclude that $M \subsetneq V$. Therefore, $\{W, M\}$ is a partition of V.

This shows that we can use the 4 states criterion because for each heuristic h that is descending there exist states $s, s[\![o]\!], s', s'[\![o']\!]$ in Π and a partition $\{W, M\}$ of V such that:

$h(s) > h(s[\![o]\!])$ and $h(s') > h(s'[\![o']\!])$ and $s^W = s[\![o]\!]^W$ and $s'^W = s'[\![o']\!]^W$ and $s^M = s'[\![o']\!]^M$ and $s[\![o]\!]^M = s'^M$, which is the condition of the 4 states criterion. □

Corollary 4.8. *The 4 states criterion is a generalization of the criterion from Theorem 4.2.*

We show that the condition of the criterion from Theorem 4.2 implies the condition of the 4 states criterion.

Proof. The correlation complexity requires by definition a descending, dead-end avoiding heuristic.

If $\Pi = \langle V, I, O, \gamma \rangle$ is a planning task in normal form with the operator o that is dangerous and critical in Π then for each DDA heuristic h there exist reachable states s, $s[\![o]\!]$, s', $s'[\![o]\!]$ with s, $s[\![o]\!]$, s' alive and $s'[\![o]\!]$ unsolvable and o applicable in s and s' and $h(s) > h(s[\![o]\!])$ and $h(s'[\![o]\!]) \geq h(s')$.

Let $M := \mathit{vars}(\mathit{pre}(o))$ and $W := V \setminus M$. With Π is in normal form we conclude $s^M = s'^M$ and $s[\![o]\!]^M = s'[\![o]\!]^M$. Because of $\mathit{eff}(o) \subseteq \mathit{pre}(o)$, applying the operator o does not affect any variable in W. Therefore, $s^W = s[\![o]\!]^W$ and $s'^W = s'[\![o]\!]^W$.

It remains to show that $\{M, W\}$ is a partition of V. Therefore, we assume $\mathit{vars}(\mathit{pre}(o)) = V$. This implies that $\mathit{pre}(o)$ is the only state where o is applicable and therefore $s[\![o]\!] = s'[\![o]\!]$ but the one is solvable while the other is unsolvable. We conclude that the assumption is wrong and that $\mathit{vars}(\mathit{pre}(o)) \subsetneq V$. Considering that o is critical. This implies that $\emptyset \neq \mathit{eff}(o)$. Since o is in normal form we know that $\mathit{vars}(\mathit{eff}(o)) \subseteq \mathit{vars}(\mathit{pre}(o))$ and therefore $\emptyset \subsetneq \mathit{vars}(\mathit{pre}(o))$. So with $M = \mathit{vars}(\mathit{pre}(o))$ we conclude $\emptyset \subsetneq M \subsetneq V$. Therefore, $\{M, W\}$ is a partition of V.

This shows that we can use the 4 states criterion, because for each heuristic h that is DDA there exists states $s, s[\![o]\!], s'[\![o]\!], s'$ in Π and a partition $\{W, M\} = V$ such that:

$h(s) > h(s[\![o]\!])$ and $h(s'[\![o]\!]) \geq h(s')$ and $s^W = s[\![o]\!]^W$ and $s'[\![o]\!]^W = s'^W$ and $s^M = s'^M$ and $s[\![o]\!]^M = s'[\![o]\!]^M$ □

We proved that the 4 states criterion is a generalization of the criteria from Theorem 4.1 and Theorem 4.2. So the 4 states criterion detects all cases that the other two detect. Does the 4 states criterion detect a case where none of the old ones does? The answer is yes. We can show that with the following example.

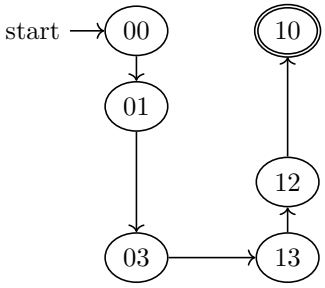


Figure 6: State space of a task with inverse macros. A node with label xy represents the state $\{v_1 \mapsto x, v_2 \mapsto y\}$.

$$
\begin{aligned}
V =& \{v_1, v_2\} \\
dom(v_1) =& \{0, 1\} \\
dom(v_2) =& \{0, 1, 2, 3\} \\
O =& \{\langle \{v_1 \mapsto 0, v_2 \mapsto 0\}, \{v_2 \mapsto 1\} \rangle, \\
& \langle \{v_2 \mapsto 1\}, \{v_2 \mapsto 3\} \rangle, \\
& \langle \{v_1 \mapsto 0, v_2 \mapsto 3\}, \{v_1 \mapsto 1\} \rangle, \\
& \langle \{v_1 \mapsto 1, v_2 \mapsto 3\}, \{v_2 \mapsto 2\} \rangle, \\
& \langle \{v_2 \mapsto 2\}, \{v_2 \mapsto 0\} \rangle \} \\
I =& \{v_1 \mapsto 0, v_2 \mapsto 0\} \\
\gamma =& \{v_1 \mapsto 1, v_2 \mapsto 0\}
\end{aligned}
$$

In Figure 6 it is easy to see that each operator in this task is critical, but no two operators are inverse of each other. Therefore, we cannot use the criteria from Theorem 4.1 and Theorem 4.2 on this task. However, if we do not only look at individual critical operators but also at sequences of operators, called macros, we see that the macro from 00 to 03 and the macro from 13 to 10 are inverse of each other and critical.

If we extend the arguments from Seipp et al. (2016) from only operators to macros we can detect that the correlation complexity of this example is at least 2 as well.

However, there is also another example where the new criterion works but the old ones do not. In Figure 7 we see the state space of a task without any dangerous operator (each is reversible) nor any critical operator. For each state s there are two plans from s that do not share any operator.

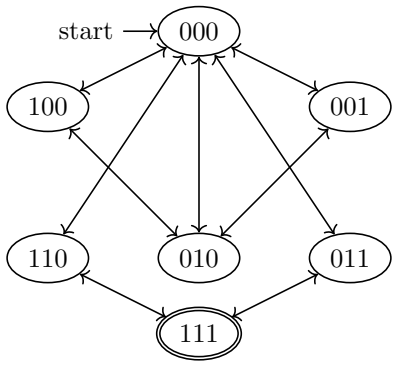


Figure 7: State space of a task that does not contain any dangerous nor critical operator. A node with label xyz represents the state $\{v_1 \mapsto x, v_2 \mapsto y, v_3 \mapsto z\}$. The goal state is labeled 111.

For a heuristic h to be descending it has to hold that $h(000) > h(110) > h(111)$ or $h(000) > h(011) > h(111)$. We assume the latter case. We could argue with the other analogously due to the symmetry of the task. Considering all the plans from the states 100, 010 and 001 we see that the state 000 is visited by all of them. We conclude that $h(100) > h(000)$ and $h(010) > h(000)$ and $h(001) > h(000)$. With transitivity we conclude $h(010) > h(011)$.

Now we can use the 4 states criterion with $a = 001$, $b = 000$, $c = 010$, $d = 011$ and $W = \{v_1, v_2\}$, $M = \{v_3\}$ for the case that the DDA potential heuristics holds $h(000) > h(011) > h(111)$. For the case that the DDA potential heuristics holds $h(000) > h(110) > h(111)$ we have $a = 100$, $b = 000$, $c = 010$, $d = 110$ and $W = \{v_2, v_3\}$, $M = \{v_1\}$.

With the 4 states criterion, we have a sufficient condition for a potential heuristic to have a dimension of at least 2. Is it also a necessary condition? The answer is no. Consider the state space in Figure 8.

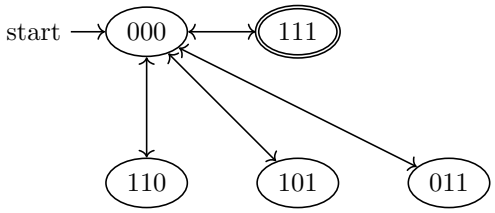


Figure 8: State space of a task with correlation complexity of 2 that is not detected by the criterion from Theorem 4.3. A node with label xyz represents the state $\{v_1 \mapsto x, v_2 \mapsto y, v_3 \mapsto z\}$. The goal state is labeled 111.

We cannot use the criterion from Theorem 4.3 because it would require a partition of $\{W, M\}$ of V. Since $|V| = 3$ either M or W contains 2 variables.

We assume $W = \{v_1, v_2\}$ (we can argue analogously for the assumption $M = \{v_1, v_2\}$). Looking at the states in the task we see that only one pair has the same projection on W. Therefore, there are no 4 states a, b, c, d with $a^W = b^W$ and $c^W = d^W$ (analogously $a^M = d^M$ and $b^M = c^M$). We can argue analogous for all possible partitions $\{W, M\}$ of V due to the symmetry of the task.

4.2 8 States Criterion

We will introduce a concept called canonical upward projections to formulate the 8 states criterion to detect that a task has correlation complexity larger than 2. The canonical upward projection is a special case of upward projection. This is similar to the P^m construction by Haslum (2009) and to the Π^C compilation by Steinmetz and Hoffmann (2018). We discuss the similarities and differences later.

Definition 4.9 (Upward Projection)**.** Let $\Pi = \langle V, I, O, \gamma \rangle$ a planning task, s a state in Π and M a superset of the state variables $M \supsetneq V$.

We call s^M an **upward projection** of s on M if s^M is an assignment of the variables in M and the (downward) projection of s^M on V equals the original state.

$$(s^M)^V = s$$

The upward projection does not specify what the assignments of the variables of $M \setminus V$ are nor what the corresponding domains are. We focus on one specific family of upward projections that project on a canonical extension of the state variables V.

Definition 4.10 (Canonical Extension)**.** Let V be a set of state variables with finite domains and $k \in \mathbb{N}$. We call the set $V^{\leq k}$ the **canonical extension** of size k of V if $V^{\leq k} \supseteq V$ and there exists a bijective mapping f from $\{W \in Pow(V) \mid |W| \leq k\}$ to $V^{\leq k}$ such that

$$dom(f(W)) = \bigtimes_{v \in W} dom(v).$$

where $Pow(V)$ is the power set of V i.e. the set that contains all subsets of V.

We denote the variable that corresponds to the set $\{v_{i_1}, v_{i_2}, \ldots, v_{i_m}\}$ as $v_{i_1, i_2, \ldots, i_m}$. Note that the canonical extension is *almost* unique because the cartesian product is not commutative so it makes a difference in which order $\bigtimes_{v \in W} dom(v)$ iterates. However, we assume a fixed order for the state variables to iterate and with this assumption the canonical extension is unique. The upward projection of a state on the canonical extension is still not unique because the assignments of the variables of $M \setminus V$ are ambiguous. The canonical upward projection is an unique upward projection on a canonical extension.

Definition 4.11 (Canonical Upward Projection)**.** Let $V = \{v_1, \ldots, v_n\}$ be a set of state variables with finite domains, $s = \{v_1 \mapsto d_1, \ldots, v_n \mapsto d_n\}$ a state

and $k \in \mathbb{N}$. We call an upward projection of s on $V^{\leq k}$ the **canonical upward projection** if for each $v_{i_1,i_2,\ldots,i_m} \in V^{\leq k}$:

$$s^{V^{\leq k}}(v_{i_1,i_2,\ldots,i_m}) = \langle s(v_{i_1}), s(v_{i_2}), \ldots, s(v_{i_m}) \rangle$$

We use the shorthand notation $s^{\leq k}$. With $\Pi^{\leq k}$ we denote the planning task Π where each state is replaced with its canonical upward projection.

In the *crossing the river* task the set of state variables is

$$V = \{v_{fox}, v_{rabbit}, v_{carrot}, v_{boat}\}$$

the corresponding canonical extension is

$$\begin{aligned} V^{\leq 2} = \{&v_{fox}, v_{rabbit}, v_{carrot}, v_{boat}, \\ &v_{fox,rabbit}, v_{fox,carrot}, v_{fox,boat}, \\ &v_{rabbit,carrot}, v_{rabbit,boat}, v_{carrot,boat}\} \end{aligned}$$

with the domains

$$dom(v_{fox}) = dom(v_{rabbit}) = dom(v_{carrot}) = dom(v_{boat}) = \{\mathit{West}, \mathit{East}\}$$

and $dom(v) = \{\mathit{West}, \mathit{East}\} \times \{\mathit{West}, \mathit{East}\}$ for all other variables in $v \in V^{\leq 2} \setminus V$. For example the canonical upward projection of the state $s = \{v_{fox} \mapsto \mathit{West}, v_{rabbit} \mapsto \mathit{East}, v_{carrot} \mapsto \mathit{West}, v_{boat} \mapsto \mathit{East}\}$ on $V^{\leq 2}$ is

$$\begin{aligned} s^{\leq 2} = \{&v_{fox} \mapsto \mathit{West}, v_{rabbit} \mapsto \mathit{East}, v_{carrot} \mapsto \mathit{West}, v_{boat} \mapsto \mathit{East}, \\ &v_{fox,rabbit} \mapsto \langle \mathit{West}, \mathit{East} \rangle, v_{fox,carrot} \mapsto \langle \mathit{West}, \mathit{West} \rangle, \\ &v_{fox,boat} \mapsto \langle \mathit{West}, \mathit{East} \rangle, v_{rabbit,carrot} \mapsto \langle \mathit{East}, \mathit{West} \rangle, \\ &v_{rabbit,boat} \mapsto \langle \mathit{East}, \mathit{East} \rangle, v_{carrot,boat} \mapsto \langle \mathit{West}, \mathit{East} \rangle\} \end{aligned}$$

The most significant difference between the canonical upward projection and the P^m construction by Haslum (2009) or the $\Pi^{\mathcal{C}}$ compilation by Steinmetz and Hoffmann (2018) is that the canonical upward projection does not describe a valid planning task. The operators are not fitting to the resulting state space (it would be possible to extend the definition in a way that the operators are projected upward as well. However, this would be rather cumbersome and unnecessary since we do not need the operators for our arguments).

The additional variables in a $\Pi^{\mathcal{C}}$ compilation have a binary domain and each additional variable represents a partial assignment. In other words, a conjunction of facts. The additional variables in the canonical upward projection represent a conjunction of variables. The domain of such an additional variable is the cartesian product of the corresponding domains. The set of partial assignments that are considered by a $\Pi^{\mathcal{C}}$ compilation, is not further specified. If $\mathcal{C}$

contains all partial assignments of size $\leq k$, then we can interpret the additional facts from the $\Pi^{\mathcal{C}}$ compilation as the translation into STRIPS of the additional facts from $\Pi^{\leq k}$.

The P^m construction is defined on propositional STRIPS tasks, while the canonical upward projection and the $\Pi^{\mathcal{C}}$ compilation are defined on tasks in finite domain representation. However, the P^m construction considers all partial assignments of size $\leq k$ and is in this regard similar to the canonical upward projection.

With the canonical upward projection we can artificially reduce the dimension of a potential heuristic because we have facts in the canonical upwards projection that encode multiple facts at once.

Theorem 4.12. *Let $\Pi = \langle V, I, O, \gamma \rangle$ be a planning task. If there exists a potential heuristic h^{pot} of dimension k then there exists a potential heuristic h'^{pot} of dimension 1 such that $h^{pot}(s) = h'^{pot}(s^{\leq k})$ for each state s in Π.*

Proof. We remember that $h^{pot}(s) = \sum_{p \in \mathcal{P}}(w(p) \cdot [p \subseteq s])$. For each partial assignment $p = \{v_{i_1} \mapsto d_{i_1}, \ldots, v_{i_m} \mapsto d_{i_m}\}$ that agrees with s with $|p| \leq k$ there exists a corresponding fact $f = (v_{i_1,\ldots,i_m} \mapsto \langle d_{i_1}, \ldots, d_{i_m} \rangle)$ in the canonical upwards projection $s^{\leq k}$ and therefore a partial assignment $p' = \{f\}$ of size 1. Let $\mathcal{P}'$ be the set of all possible partial assignments of the canonical upward projection. By choosing $w(p') = w(p)$ for each p we see that $h^{pot}(s) = \sum_{p \in \mathcal{P}}(w(p) \cdot [p \subseteq s]) = \sum_{p' \in \mathcal{P}'}(w(p') \cdot [p' \subseteq s^{\leq k}]) = h'^{pot}(s^{\leq k})$ □

Looking at the contrapositive of Theorem 4.12 we see that we can use the canonical upward projection on $V^{\leq k}$ to check if a given heuristic is impossible to represent as a potential heuristic of dimension k. We now use this to create our next criterion.

Theorem 4.13. *Let $\Pi = \langle V, I, O, \gamma \rangle$ be a planning task, and let h^{pot} be a potential heuristic on Π. If there exist states a, b, c, d, e, f, g, r in $\Pi^{\leq 2}$ and a partition $\{A, B, C\}$ of $V^{\leq 2}$ such that:*

$$h^{pot}(a) > h^{pot}(b), \quad h^{pot}(c) > h^{pot}(d), \quad h^{pot}(e) > h^{pot}(f), \quad h^{pot}(g) > h^{pot}(r),$$

$$\begin{aligned} a^A = b^A, \quad c^A = d^A, \quad e^A = f^A, \quad g^A = r^A, \\ a^B = d^B, \quad b^B = c^B, \quad e^B = r^B, \quad f^B = g^B, \\ a^C = r^C, \quad b^C = g^C, \quad c^C = f^C, \quad d^C = e^C, \end{aligned}$$

then the dimension of h^{pot} is at least 3.

Proof. Assume there exists a potential heuristic h'^{pot} of dimension 1 on the task $\Pi^{\leq 2}$ and a partition $\{A, B, C\}$ of $V^{\leq 2}$ such that:

$h^{pot}(a) > h^{pot}(b)$ and $h^{pot}(c) > h^{pot}(d)$ and $h^{pot}(e) > h^{pot}(f)$ and $h^{pot}(g) > h^{pot}(r)$ and

$a^A = b^A$ and $c^A = d^A$ and $e^A = f^A$ and $g^A = r^A$ and

$a^B = d^B$ and $b^B = c^B$ and $e^B = r^B$ and $f^B = g^B$ and

$a^C = r^C$ and $b^C = g^C$ and $c^C = f^C$ and $d^C = e^C$. The assumption implies that $h'^{pot}(a) = h'^{pot}(a^A) + h'^{pot}(a^B) + h'^{pot}(a^C) - 2 \cdot w(\emptyset)$ (analogous for b, c, d, e, f, g, r). This provides us four inequalities (the $-2 \cdot w(\emptyset)$ is not written in the following because it cancels out immediately). The first one is:

$$h'^{pot}(a^A) + h'^{pot}(a^B) + h'^{pot}(a^C) > h'^{pot}(b^A) + h'^{pot}(b^B) + h'^{pot}(b^C)$$

with $a^A = b^A$ we can simplify it to

$$h'^{pot}(a^B) + h'^{pot}(a^C) > h'^{pot}(b^B) + h'^{pot}(b^C).$$

The second inequality is:

$$h'^{pot}(c^A) + h'^{pot}(c^B) + h'^{pot}(c^C) > h'^{pot}(d^A) + h'^{pot}(d^B) + h'^{pot}(d^C)$$

with $c^A = d^A$ we can simplify it to

$$h'^{pot}(c^B) + h'^{pot}(c^C) > h'^{pot}(d^B) + h'^{pot}(d^C).$$

We know that $c^B = b^B$ and $d^B = a^B$ so we replace these values

$$h'^{pot}(b^B) + h'^{pot}(c^C) > h'^{pot}(a^B) + h'^{pot}(d^C).$$

The third inequality is:

$$h'^{pot}(e^A) + h'^{pot}(e^B) + h'^{pot}(e^C) > h'^{pot}(f^A) + h'^{pot}(f^B) + h'^{pot}(f^C)$$

with $e^A = f^A$ we can simplify it to

$$h'^{pot}(e^B) + h'^{pot}(e^C) > h'^{pot}(f^B) + h'^{pot}(f^C).$$

We know that $e^C = d^C$ and $f^C = c^C$ so we replace these values

$$h'^{pot}(e^B) + h'^{pot}(d^C) > h'^{pot}(f^B) + h'^{pot}(c^C).$$

The fourth inequality is:

$$h'^{pot}(g^A) + h'^{pot}(g^B) + h'^{pot}(g^C) > h'^{pot}(r^A) + h'^{pot}(r^B) + h'^{pot}(r^C)$$

with $g^A = r^A$ we can simplify it to

$$h'^{pot}(g^B) + h'^{pot}(g^C) > h'^{pot}(r^B) + h'^{pot}(r^C).$$

We know that $g^B = f^B$ and $g^C = b^C$ and $h^B = e^B$ and $r^C = a^C$ so we replace these values

$$h'^{pot}(f^B) + h'^{pot}(b^C) > h'^{pot}(e^B) + h'^{pot}(a^C).$$

By adding the first and the second inequality we get

$$h'^{pot}(a^B) + h'^{pot}(a^C) + h'^{pot}(b^B) + h'^{pot}(c^C)$$

$$>$$

$$h'^{pot}(b^B) + h'^{pot}(b^C) + h'^{pot}(a^B) + h'^{pot}(d^C).$$

Which we simplify to

$$h'^{pot}(a^C) + h'^{pot}(c^C) > h'^{pot}(b^C) + h'^{pot}(d^C).$$

By adding the third and the fourth inequality we get

$$h'^{pot}(e^B) + h'^{pot}(d^C) + h'^{pot}(f^B) + h'^{pot}(b^C)$$
$$>$$
$$h'^{pot}(f^B) + h'^{pot}(c^C) + h'^{pot}(e^B) + h'^{pot}(a^C).$$

Which we simplify to

$$h'^{pot}(d^C) + h'^{pot}(b^C) > h'^{pot}(c^C) + h'^{pot}(a^C).$$

We see that the two inequalities

$$h'^{pot}(a^C) + h'^{pot}(c^C) > h'^{pot}(b^C) + h'^{pot}(d^C)$$
$$h'^{pot}(d^C) + h'^{pot}(b^C) > h'^{pot}(c^C) + h'^{pot}(a^C)$$

form a contradiction. Therefore, the assumption must be false. So we conclude that there exists no potential heuristic of dimension 1 on task $\Pi^{\leq 2}$. With the contrapositive of Theorem 4.12 we conclude that no potential heuristic h^{pot} on Π with dimension 2 exists. Therefore, the dimension of h^{pot} is at least 3. □

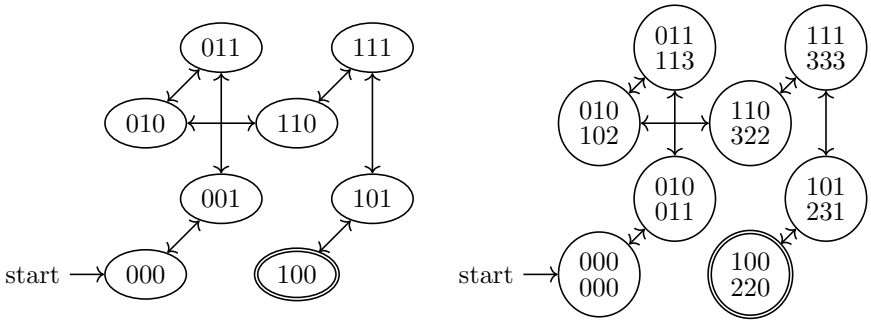


Figure 9: Left: State space of the Gray code counter task of size 3. A node with label xyz represents the state $\{v_1 \mapsto x, v_2 \mapsto y, v_3 \mapsto z\}$. Right: Canonical upward projection of the state space of the Gray code counter task. The numbers in the second row inside a node represent the assignments of the variables $v_{1,2}, v_{1,3}, v_{2,3}$ and 0 is the shorthand notation for $\langle 0, 0\rangle$, 1 for $\langle 0, 1\rangle$, 2 for $\langle 1, 0\rangle$, 3 for $\langle 1, 1\rangle$. (The following colored underlining indicates which original fact correspond to which added fact.) For example the state $\{v_1 \mapsto 1, v_2 \mapsto 0, v_3 \mapsto 1\}$ is represented as the node with the label 101 in the left figure and the canonical upward projection of it is $\{v_1 \mapsto 1, v_2 \mapsto 0, v_3 \mapsto 1, v_{1,2} \mapsto \langle 1, 0\rangle, v_{1,3} \mapsto \langle 1, 1\rangle, v_{2,3} \mapsto \langle 0, 1\rangle\}$ and is represented as the node with the label 101231 in the right figure.

Theorem 4.14 (8 States Criterion)**.** *Let* $\Pi = \langle V, I, O, \gamma \rangle$ *be a planning task. If for each potential heuristic* h^{pot} *that is DDA on* Π *there exist states* a*,* b*,* c*,* d*,* e*,* f*,* g *and* r *in* $\Pi^{\leq 2}$ *and a partition* $\{A, B, C\}$ *of* $V^{\leq 2}$ *such that:*

$$h^{pot}(a) > h^{pot}(b), \quad h^{pot}(c) > h^{pot}(d), \quad h^{pot}(e) > h^{pot}(f), \quad h^{pot}(g) > h^{pot}(r),$$

$$\begin{aligned} a^A = b^A, \quad c^A = d^A, \quad e^A = f^A, \quad g^A = r^A, \\ a^B = d^B, \quad b^B = c^B, \quad e^B = r^B, \quad f^B = g^B, \\ a^C = r^C, \quad b^C = g^C, \quad c^C = f^C, \quad d^C = e^C, \end{aligned}$$

then the correlation complexity of Π *is at least 3.*

Proof. We know that the correlation complexity of a task Π is the minimal dimension over all potential heuristics that are DDA on Π. The condition of the 8 states criterion says that each DDA potential heuristic holds the condition for Theorem 4.13. Therefore, we can apply Theorem 4.13 on each potential heuristic that is DDA on Π. We conclude that the dimension of each potential heuristic that is DDA on Π is at least 3. Therefore, the correlation complexity of Π is at least 3. □

Seipp et al. (2016) investigated the correlation complexity of the *Gray code counter* of size 3. They have proven that the correlation complexity of that task is exactly 3. We can also use the 8 states criterion to prove that the correlation complexity of this task is at least 3.

Looking at Figure 9, we see that we can apply the 8 states criterion with the following states (the colored underlining indicates which digit belongs to which set of the partition) $a = \underline{000000}$, $b = \underline{001011}$, $c = \underline{011113}$, $d = \underline{010102}$, $e = \underline{110322}$, $f = \underline{111333}$, $g = \underline{101231}$ and $r = \underline{100220}$ and the subsets of $V^{\leq 2}$ as $A = \{v_1, v_2, v_{1,2}\}, B = \{v_{1,3}\}, C = \{v_3, v_{2,3}\}$. Each DDA heuristic fulfills the inequalities for the 8 states criterion on these states.

5 Planning and Linear Algebra

In this chapter, we see how we can use linear algebra to get a different view on the planning tasks and thereby create a new criterion for a correlation complexity of at least 2. In Figures 1 and 3 as well as in Figure 9 we see the state space displayed in a space with $|V|$ dimensions. In these examples is the domain size for each variable 2. In this chapter, we restrict ourselves to such tasks. We can translate each SAS$^+$ planning task into an equivalent[5] planning task where each state variable has a domain of size 2. In a domain with exactly two elements, we can arbitrarily substitute one element with the number 0 and the other with the number 1.

For each fact $v \mapsto d$ in the original task we create a state variable $b_{v\mapsto d}$ with $dom(b_{v\mapsto d}) = \{0, 1\}$. As an example, we translate the task from Figure 6 into this form.

$$
\begin{aligned}
V =& \{b_{v_1\mapsto 0}, b_{v_1\mapsto 1}, b_{v_2\mapsto 0}, b_{v_2\mapsto 1}, b_{v_2\mapsto 2}, b_{v_2\mapsto 3}\} \\
dom(b) =& \{0,1\} \text{ for all } b \in V \\
O =& \{\langle \{b_{v_1\mapsto 0} \mapsto 1, b_{v_2\mapsto 0} \mapsto 1, b_{v_2\mapsto 1} \mapsto 0\}, \{b_{v_2\mapsto 0} \mapsto 0, b_{v_2\mapsto 1} \mapsto 1\}\rangle, \\
& \langle \{b_{v_2\mapsto 1} \mapsto 1, b_{v_2\mapsto 3} \mapsto 0\}, \{b_{v_2\mapsto 1} \mapsto 0, b_{v_2\mapsto 3} \mapsto 1\}\rangle, \\
& \langle \{b_{v_1\mapsto 0} \mapsto 1, b_{v_2\mapsto 3} \mapsto 1, b_{v_1\mapsto 1} \mapsto 0\}, \{b_{v_1\mapsto 1} \mapsto 1, b_{v_1\mapsto 0} \mapsto 0\}\rangle, \\
& \langle \{b_{v_1\mapsto 1} \mapsto 1, b_{v_2\mapsto 3} \mapsto 1, b_{v_2\mapsto 2} \mapsto 0\}, \{b_{v_2\mapsto 2} \mapsto 1, b_{v_2\mapsto 3} \mapsto 0\}\rangle, \\
& \langle \{b_{v_2\mapsto 2} \mapsto 1, b_{v_2\mapsto 0} \mapsto 0\}, \{b_{v_2\mapsto 0} \mapsto 1, b_{v_2\mapsto 2} \mapsto 0\}\rangle\} \\
I =& \{b_{v_1\mapsto 0} \mapsto 1, b_{v_1\mapsto 1} \mapsto 0, b_{v_2\mapsto 0} \mapsto 1, b_{v_2\mapsto 1} \mapsto 0, b_{v_2\mapsto 2} \mapsto 0, b_{v_2\mapsto 3} \mapsto 0\} \\
\gamma =& \{b_{v_1\mapsto 1} \mapsto 1, b_{v_2\mapsto 0} \mapsto 1\}
\end{aligned}
$$

Before we dive into linear algebra we want to point out that, instead of iterating the sum over all possible partial assignments, to evaluate the potential heuristic we can instead iterate over the subsets of V.

Lemma 5.1. *Let h^{pot} be a potential heuristic of dimension d and s be a state. Then:*

$$h^{pot}(s) = \sum_{p\in\mathcal{P}} (w(p) \cdot [p \subseteq s]) = \sum_{\substack{U\in Pow(V)\\ |U|\leq d}} w(\{u \mapsto s(u) \mid u \in U\}).$$

Proof. The expression $[p \subseteq s]$ evaluates to 1 if and only if $(u \mapsto s(u))$ is an element of p for each $u \in vars(p)$. We define $\mathcal{P}'(s) := \{\{u \mapsto s(u) \mid u \in U\} \mid U \subseteq V\}$ which is the set of partial assignments that agree with the state s. This

[5] There is a bijection from each plan in the original task to each plan in the translated task. However, the size of the state space grows exponentially.

is trivially a subset of all possible partial assignments $\mathcal{P}'(s) \subseteq \mathcal{P}$. We split the sum into two parts

$$\sum_{p \in \mathcal{P}} (w(p) \cdot [p \subseteq s]) = \sum_{p \in \mathcal{P}'(s)} (w(p) \cdot [p \subseteq s]) + \sum_{p \in \mathcal{P} \setminus \mathcal{P}'(s)} (w(p) \cdot [p \subseteq s]).$$

The first part iterates over all partial assignments that agree with s and the second part iterates over all that do not agree with s. Therefore, we can replace $[p \subseteq s]$ with a constant 1 in the first one and a constant 0 in the second. There we can ignore the second part as it sums up to 0 and the remaining part we simplify

$$\sum_{p \in \mathcal{P}} (w(p) \cdot [p \subseteq s]) = \sum_{p \in \mathcal{P}'(s)} (w(p) \cdot 1) + \sum_{p \in \mathcal{P} \setminus \mathcal{P}'(s)} (w(p) \cdot 0) = \sum_{p \in \mathcal{P}'(s)} w(p).$$

We know for each $p \in \mathcal{P}'(s)$ that $p = \{u \mapsto s(u) \mid u \in \mathit{vars}(p)\}$ and for each subset $U \subseteq V$ there is exactly one partial assignment p with $\mathit{vars}(p) = U$ that agrees with s and that is the projection s^U. So we conclude

$$\sum_{p \in \mathcal{P}'(s)} w(p) = \sum_{U \in Pow(V)} w(\{u \mapsto s(u) \mid u \in U\}).$$

We now consider the dimension d of the heuristic and split the sum into two parts. The first iterates over the subsets of V of size less or equal to d and the second iterates over the others, the subsets of size greater than d.

$$h^{pot}(s) = \sum_{\substack{U \in Pow(V) \\ |U| \leq d}} w(\{u \mapsto s(u) \mid u \in U\}) + \sum_{\substack{U \in Pow(V) \\ |U| > d}} w(\{u \mapsto s(u) \mid u \in U\})$$

With the definition of the dimension of a heuristic, we know that the weight of a partial assignment $w(p) = 0$ for all $p \in \mathcal{P}$ with $|p| > d$. Therefore, the second part sums up to 0. We conclude:

$$h^{pot}(s) = \sum_{\substack{U \in Pow(V) \\ |U| \leq d}} w(\{u \mapsto s(u) \mid u \in U\}).$$

□

Corollary 5.2. *Let h^{pot} be a potential heuristic of dimension 1 and s be a state. Then:*

$$h^{pot}(s) = \sum_{v \in V} w(\{v \mapsto s(v)\}) + w(\emptyset)$$

Proof. As h^{pot} is of dimension 1 we know from Lemma 5.1 that

$$h^{pot}(s) = \sum_{\substack{U \in Pow(V) \\ |U| \leq 1}} w(\{u \mapsto s(u) \mid u \in U\})$$

We split this sum into two parts. The first iterates over all subsets of V of size 1 and the second over all subsets of size 0. As $\emptyset$ is the only subset of V of size 0 we get

$$h^{pot}(s) = \sum_{\substack{U \in Pow(V) \\ |U|=1}} w(\{u \mapsto s(u) \mid u \in U\}) + w(\emptyset).$$

With $|U| = 1$ we conclude that each U in the sum is of the form $\{v\}$ with $v \in V$. Since $\{u \mapsto s(u) \mid u \in \{v\}\} = \{v \mapsto s(v)\}$ we conclude

$$h^{pot}(s) = \sum_{v \in V} w(\{v \mapsto s(v)\}) + w(\emptyset).$$

□

5.1 Vectors between Points and Transitions between States

Looking at the Figure 1, we see that the transition system of the task is represented in a 4-dimensional space. Each dimension corresponds to a state variable and each state corresponds to a single point in this space.

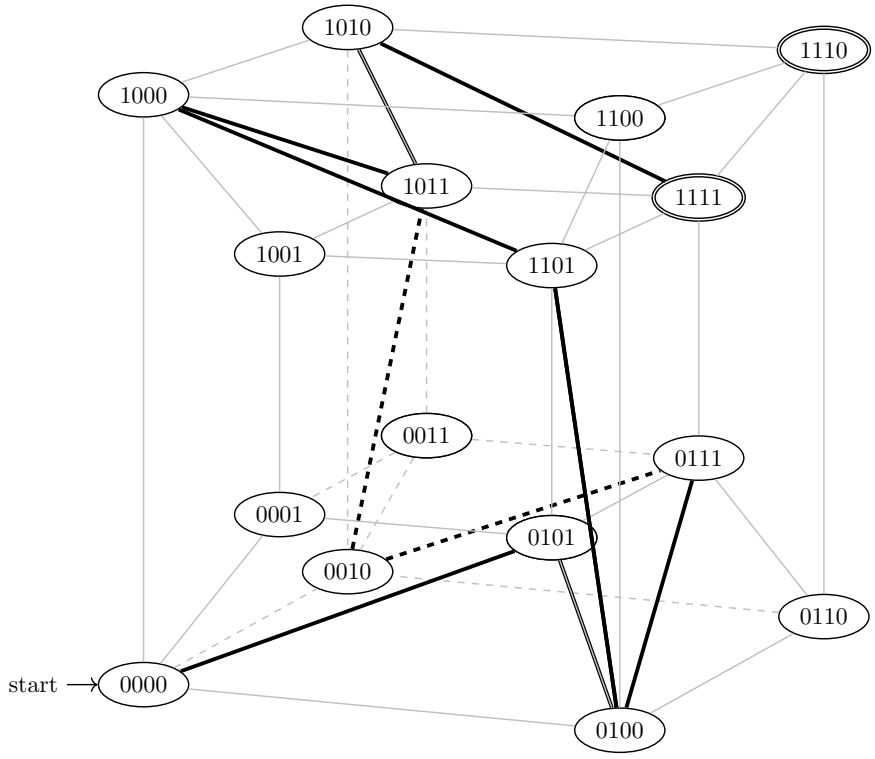


Figure 10: State space for the *crossing the river* task with substituted domain values. (Compare Figure 1)

Looking at the representation of the *crossing the river* task in Figure 10, we see that the states 0000 and 0101 are connected. By viewing the states as points in the 4-dimensional space, we can interpret the transitions as vectors. So the vector $\overrightarrow{t_1} = (0, 1, 0, 1)^\top$ is the vector from state 0000 to 0101.

Note that we are not only talking about state transitions (arcs in the state space that are induced by an operator), but the transition from any state to any state. Independent of whether or not the one is even reachable from the other. The vector $\overrightarrow{t_2} = (0, -1, 1, 0)^\top$ is the vector from state 0101 to 0011.

We call the translation of two states into a vector the vectorization. We define it formally:

Definition 5.3 (Vectorization)**.** Let $\Pi = \langle V, I, O, \gamma \rangle$ a planning task with only binary domains. The vector $\overrightarrow{t} \in \mathbb{R}^{|V|}$ is the **vectorization** from the state s to the state s' where

$$\overrightarrow{t}[i] := s'(v_i) - s(v_i)$$

for each $i \in \{1, \ldots, |V|\}$.

As notation we use $\overrightarrow{t_{s,s'}}$ for the vectorization from state s to state s'. This definition implies an order for the state variables. We can choose an arbitrary one for that. We denote this order with an index function *idx*. As shorthand notation we use $\overrightarrow{t}[v] = \overrightarrow{t}[idx(v)]$. The vector represents the change that a transition from s to s' induces.

5.2 Weight of Vectors and State Variables

For a potential heuristic $h^{pot}(s) = \sum_{p \in \mathcal{P}} (w(p) \cdot [p \subseteq s])$, we define the weight of a state variable $v \in V$ as

$$w(v) := w(\{v \mapsto 1\}) - w(\{v \mapsto 0\}).$$

Based on the weight of a state variable, we define the weight of a vector $\overrightarrow{t}$ as

$$w(\overrightarrow{t}) := \sum_{v \in V} w(v) \cdot \overrightarrow{t}[v].$$

Note that the weight function w is now overloaded and accepts partial assignments, state variables, and vectors as input.

Theorem 5.4. *Let h be a potential heuristic of dimension 1 and $\overrightarrow{t_{s,s'}}$ the vectorization from s to s' with s, s' being states of a planning task with only binary domains. Then:*

$$w(\overrightarrow{t_{s,s'}}) = h(s') - h(s).$$

Proof. With h being of dimension 1, we know from Corollary 5.2 that

$$h(s') - h(s) = \sum_{v \in V} w(\{v \mapsto s'(v)\}) - \sum_{v \in V} w(\{v \mapsto s(v)\}).$$

Both sums iterate over the same set so we can combine them.

$$h(s') - h(s) = \sum_{v \in V} w(\{v \mapsto s'(v)\}) - w(\{v \mapsto s(v)\}).$$

Considering an individual summand

$$w(\{v \mapsto s'(v)\}) - w(\{v \mapsto s(v)\}) = \begin{cases} 0 & \text{if } s'(v) = s(v) \\ w(v) \cdot 1 & \text{if } s'(v) = 1 \text{ and } s(v) = 0 \\ w(v) \cdot (-1) & \text{if } s'(v) = 0 \text{ and } s(v) = 1 \end{cases}$$

Looking at the individual cases:

- if $s'(v) = s(v)$ then $s'(v) - s(v) = 0$
- if $s'(v) = 1$ and $s(v) = 0$ then $s'(v) - s(v) = 1$
- if $s'(v) = 0$ and $s(v) = 1$ then $s'(v) - s(v) = -1$

With these implications of the conditions, we conclude that

$$w(v \mapsto s'(v)) - w(v \mapsto s(v)) = w(v) \cdot (s'(v) - s(v)) = w(v) \cdot \overrightarrow{t_{s,s'}}[v].$$

Plugging this equivalent summand into the equation, we get

$$h(s') - h(s) = \sum_{v \in V} w(v) \cdot \overrightarrow{t_{s,s'}}[v] = w(\overrightarrow{t_{s,s'}}).$$

□

The vectorization of any two states is an element of $\mathbb{R}^{|V|}$. The function w over the vectors in $\mathbb{R}^{|V|}$ is preserving over addition and scalar multiplication. In the words of linear algebra: the function $w : \mathbb{R}^{|V|} \to \mathbb{R}$ is a linear map (Fischer, 2010).

Definition 5.5 (Linear Map)**.** Let $n \in \mathbb{N}$. A function $f : \mathbb{R}^n \to \mathbb{R}$ is a **linear map** if for each two vectors $\overrightarrow{x}, \overrightarrow{y} \in \mathbb{R}^n$ and scalar $k \in K$ the conditions

$$f(\overrightarrow{x} + \overrightarrow{y}) = f(\overrightarrow{x}) + f(\overrightarrow{y})$$

and

$$f(k \cdot \overrightarrow{x}) = k \cdot f(\overrightarrow{x})$$

are satisfied.

In our case the function w maps from the vector space $\mathbb{R}^{|V|}$ to the vector space $\mathbb{R}$.

Theorem 5.6. *The weight function, $w : \mathbb{R}^{|V|} \to \mathbb{R}$, is a linear map.*

Proof. Let $\overrightarrow{x}, \overrightarrow{y}$ be vectors in $\mathbb{R}^{|V|}$ and $k \in \mathbb{R}$. This chain of equations shows the first condition:

$$\begin{aligned} w(\overrightarrow{x} + \overrightarrow{y}) &= \sum_{v \in V} (w(v) \cdot (\overrightarrow{x}[v] + \overrightarrow{y}[v])) \\ &= \sum_{v \in V} (w(v) \cdot \overrightarrow{x}[v] + w(v) \cdot \overrightarrow{y}[v])) \\ &= \sum_{v \in V} w(v) \cdot \overrightarrow{x}[v] + \sum_{v \in V} w(v) \cdot \overrightarrow{y}[v] \\ &= w(\overrightarrow{x}) + w(\overrightarrow{y}) \end{aligned} \tag{1}$$

This chain of equations shows the second condition:

$$\begin{aligned} w(k \cdot \overrightarrow{x}) &= \sum_{v \in V} (w(v) \cdot (k \cdot \overrightarrow{x}[v])) \\ &= k \cdot \sum_{v \in V} (w(v) \cdot (\overrightarrow{x}[v])) \\ &= k \cdot w(\overrightarrow{x}) \end{aligned} \tag{2}$$

□

In linear algebra, the span of a set S of vectors is defined as the set of all linear combinations of vectors from S (Fischer, 2010). For our purposes, we want to focus on the linear combinations with non-negative coefficients. This is called the convex cone *cone*(S) (in our case it is a polyhedral cone because we generate it with a finite amount of vectors).

Definition 5.7 (convex cone)**.** Let $n \in \mathbb{N}$ and S be a set of vectors in $\mathbb{R}^n$. We call the set

$$cone(S) := \{\sum_{i=1}^{n} \lambda_i \cdot \overrightarrow{x_i} \mid \overrightarrow{x_i} \in S, \lambda_i \in \mathbb{R}_{\geq 0}\}$$

the **convex cone** of S and the elements of S are called the **generators** of the convex cone. (Dattorro, 2005)

Definition 5.8 (overlapping)**.** We say the convex cone of the sets of vectors S and S' are **overlapping** if

$$cone(S) \cap cone(S') \supsetneq \{\overrightarrow{0}\}.$$

The convex cone is always a subset of the span. For example the span of $S = \{(0,1)^\top, (-1,0)^\top\}$ is the entire two-dimensional plane but the convex cone of S is only the 2rd quadrant (including the edges). The convex cone of $S' = \{(0,2)^\top, (0.3,0)^\top\}$ is overlapping with *cone*(S) because the vector $(0,1)^\top$ is in both convex cones.

Lemma 5.9. *Let X' and Y' be sets of vectors in $\mathbb{R}^n$ where cone(X') and cone(Y') are overlapping and $X \supseteq X'$ and $Y \supseteq Y'$. Then cone(X) and cone(Y) are overlapping.*

Proof. With the definition of the convex cone we see that $cone(X) \supseteq cone(X')$ and $cone(Y) \supseteq cone(Y')$ Since $cone(X')$ and $cone(Y')$ are overlapping we know that there exists a element $\overrightarrow{z} \in cone(X') \cap cone(Y')$ with $\overrightarrow{z} \neq 0$. This $\overrightarrow{z}$ is also element in $cone(X)$ because it is a superset of $cone(X')$. Analogous for $cone(Y)$. We conclude $\overrightarrow{z} \in cone(X) \cap cone(Y)$ and therefore $cone(X)$ and $cone(Y)$ are overlapping. □

This means that if we want to show that two convex cones C and G are overlapping it is sufficient to prove that at least one convex cone C' that is contained in C overlaps with at least one convex cone G' that is contained in G.

A convex cone is fully defined by its generators. With the following Theorem and Corollaries, we see that if the sign of a linear mapping to $\mathbb{R}$ is the same for all generators then it is the same for all vectors in the convex cone (except the zero vector).

Theorem 5.10. *Let S be a set of vectors in $\mathbb{R}^n$ and the function $f : \mathbb{R}^n \to \mathbb{R}$ a linear mapping. If $f(\overrightarrow{x}) > 0$ for each $\overrightarrow{x} \in S$, then $f(\overrightarrow{y}) > 0$ for all $\overrightarrow{y} \in cone(S) \setminus \{\overrightarrow{0}\}$.*

Proof. With $\overrightarrow{y} \in cone(S)$ we know that it is of the form

$$\overrightarrow{y} = \sum_{i=1}^{n} \lambda_i \cdot \overrightarrow{x_i}$$

with $n = |S|$, $\overrightarrow{x_i} \in S$ and $\lambda_i \geq 0$ for all $i \in \{1, \ldots, n\}$. Since f is a linear mapping we know that

$$f(\overrightarrow{y}) = \sum_{i=1}^{n} \lambda_i \cdot f(\overrightarrow{x_i}).$$

As $f(x_i) > 0$ and $\lambda_i \geq 0$ for each $i \in \{1, \ldots, n\}$ we know that each summand is non-negative $\lambda_i \cdot f(\overrightarrow{x_i}) \geq 0$. Since $\overrightarrow{y} \in cone(S) \setminus \{\overrightarrow{0}\}$ we know that $\overrightarrow{y} \neq \overrightarrow{0}$ and therefore there is at least one $\lambda_i \neq 0$. This implies that at least one summand is strictly positive. A sum of non-negative and strictly positive summands is strictly positive, therefore $f(\overrightarrow{y}) > 0$. □

Corollary 5.11. *Let S be a set of vectors in $\mathbb{R}^n$ and the function $f : \mathbb{R}^n \to \mathbb{R}$ a linear mapping. If $f(\overrightarrow{x}) < 0$ for each $\overrightarrow{x} \in S$ then $f(\overrightarrow{y}) < 0$ for all $\overrightarrow{y} \in cone(S) \setminus \{0\}$.*

Proof. From the condition we know that $f(\overrightarrow{x}) < 0$ for each $\overrightarrow{x} \in S$. Therefore, $-f(\overrightarrow{x}) > 0$ for each $\overrightarrow{x} \in S$. With $-f$ being a linear mapping and Theorem 5.10 we conclude that $-f(\overrightarrow{y}) > 0$ for each $\overrightarrow{y} \in cone(S) \setminus \{0\}$. We conclude $f(\overrightarrow{y}) < 0$ for each $\overrightarrow{y} \in cone(S) \setminus \{0\}$. □

Corollary 5.12. *Let S be a set of vectors in $\mathbb{R}^n$, the function $f : \mathbb{R}^n \to \mathbb{R}$ a linear mapping. If $f(\overrightarrow{x}) \geq 0$ for each $\overrightarrow{x} \in S$ then $f(\overrightarrow{y}) \geq 0$ for all $\overrightarrow{y} \in cone(S) \setminus \{0\}$.*

Proof. With $\overrightarrow{y} \in cone(S)$ we know that $\overrightarrow{y}$ is of the form

$$\overrightarrow{y} = \sum_{i=1}^{n} \lambda_i \cdot \overrightarrow{x_i}$$

with $n = |S|$, $\overrightarrow{x_i} \in S$ and $\lambda_i \geq 0$ for all $i \in \{1, \ldots, n\}$. Since f is a linear mapping we know that

$$f(\overrightarrow{y}) = \sum_{i=1}^{n} \lambda_i \cdot f(\overrightarrow{x_i}).$$

As $f(x_i) > 0$ and $\lambda_i \geq 0$ for each $i \in \{1, \ldots, n\}$ we know that each summand is non-negative $\lambda_i \cdot f(\overrightarrow{x_i}) \geq 0$. Therefore, the sum is non-negative and hence $\overrightarrow{y} = \sum_{i=1}^{n} \lambda_i \cdot \overrightarrow{x_i} \geq 0$. □

Whit this corollary we can detect that the correlation complexity of the task in Figure 8 is at least 2. The vector $\overrightarrow{o_1} = (1, 1, 1)^\top$ encodes the transition from the initial state to the goal state. Each of the vectors $\overrightarrow{o_2} = (1, 1, 0)^\top$, $\overrightarrow{o_3} = (1, 0, 1)^\top$ and $\overrightarrow{o_4} = (0, 1, 1)^\top$ encode a transition that corresponds to a dangerous operator. They encode the transitions from the initial state to the dead-ends.

With $0.5 \cdot \overrightarrow{o_2} + 0.5 \cdot \overrightarrow{o_3} + 0.5 \cdot \overrightarrow{o_4} = \overrightarrow{o_1}$ we see that $\overrightarrow{o_1}$ is an element of $cone(\{\overrightarrow{o_2}, \overrightarrow{o_3}, \overrightarrow{o_4}\})$. For each DDA heuristic of dimension 1 on this task the weight of $\overrightarrow{o_1}$ has to be negative and the weights of $\overrightarrow{o_2}$, $\overrightarrow{o_3}$ and $\overrightarrow{o_4}$ have to be non-negative which contradicts Corollary 5.12.

We now introduce the red-blue-split (RB-split), which categorizes all vectorized transitions of a state space topology. The blue set B contains all vectorized transitions where the transition improves the heuristic value and the red set R all the others. The RB-split helps us to connect the convex cones with 1-dimensional potential heuristics.

Definition 5.13 (RB-split)**.** Let $\Pi = \langle V, I, O, \gamma \rangle$ be a planning task with only binary domains, $\langle \mathcal{S}, h \rangle$ be a state space topology of Π with $\mathcal{S} = \langle S, E, G, s_0 \rangle$. We call the tuple $\langle R, B \rangle$ the **RB-split** of Π with h if $R = \{\overrightarrow{t_{s,s'}} \mid s, s' \in S, h(s') - h(s) \geq 0\}$ and $B = \{\overrightarrow{t_{s,s'}} \mid s, s' \in S, h(s') - h(s) < 0\}$.

Theorem 5.14. *Let h be a potential heuristic on a planning task $\Pi = \langle V, I, O, \gamma \rangle$ in normal form with binary domains and $\langle R, B \rangle$ the RB-split of Π with h. If $dim(h) = 1$ then $cone(R)$ and $cone(B)$ are not overlapping.*

Proof. With h being of dimension 1 and Theorem 5.4 we know that $h(s') - h(s) = w(\overrightarrow{t_{s,s'}})$ for each $s, s' \in S$. Therefore, we know that $w(\overrightarrow{r}) \geq 0$ for each $\overrightarrow{r} \in R$ and $w(\overrightarrow{b}) < 0$ for each $\overrightarrow{b} \in B$. With Corollary 5.2 and Corollary 5.11 we know that $w(\overrightarrow{r'}) \geq 0$ for each $\overrightarrow{r'} \in cone(R) \setminus \{0\}$ and $w(\overrightarrow{b'}) < 0$ for each $\overrightarrow{b'} \in cone(B) \setminus \{0\}$. Assume $\overrightarrow{x} \in cone(R) \cap cone(B)$ with $\overrightarrow{x} \neq 0$. This implies that $w(\overrightarrow{x}) \geq 0$ because $\overrightarrow{x} \in cone(R)$ and $w(\overrightarrow{x}) < 0$ because $\overrightarrow{x} \in cone(B)$. This is a contradiction. We conclude that $cone(R) \cap cone(B) = \{0\}$ and therefore $cone(R)$ and $cone(B)$ are not overlapping. □

With Lemma 5.9 and the contrapositive of Theorem 5.14 we create an even more general criterion.

Theorem 5.15. *Let h be a potential heuristic on the planing task Π, $\langle R, B\rangle$ the RB-split of Π with h. Let $R' \subseteq cone(R)$ and $B' \subseteq cone(B)$. If the convex cones $cone(R')$ and $cone(B')$ are overlapping then the dimension of h is at least 2.*

Proof. The overlapping of the convex cones $cone(R')$ and $cone(B')$ implies with $cone(R') \subseteq cone(R)$, $cone(B') \subseteq cone(B)$ and Lemma 5.9 that the convex cones of R and B are overlapping. This implies with the contrapositive of Theorem 5.14 that $dim(h) \neq 1$. We conclude that the dimension of h is at least 2. □

Theorem 5.16 (RB-spilt criterion)**.** *Let Π be a planning task. If for each potential heuristic h^{pot} that is DDA on Π there exists a $\overrightarrow{x} \neq \overrightarrow{0}$ such that $\overrightarrow{x} \in cone(R) \cap cone(B)$, where $\langle R, B\rangle$ is the RB-split of Π with h^{pot}, then the correlation complexity of Π is at least 2.*

Proof. The existence of $\overrightarrow{x} \neq \overrightarrow{0}$ such that $\overrightarrow{x} \in cone(R) \cap cone(B)$ implies that $cone(R)$ and $cone(B)$ are overlapping. This implies, with the contrapositive of Theorem 5.14 that $dim(h) \neq 1$. We conclude that the dimension of each h^{pot} that is DDA on Π is at least 2. Therefore, the correlation complexity of Π is at least 2. □

Corollary 5.17. *The RB-split criterion is a generalization of the 4 states criterion.*

We show that the condition of the 4 states criterion from Theorem 4.1 implies the condition of the RB-split criterion.

Proof. Let $\Pi = \langle V, I, O, \gamma\rangle$ be a planning task. Without loss of generality, we assume that Π is a planning task with only binary domains. If for each potential heuristic h^{pot} that is DDA on Π there exist states a, b, c, d in Π and a partition $\{W, M\}$ of V such that:

$$\begin{aligned} h^{pot}(a) &> h^{pot}(b),\\ h^{pot}(c) &\geq h^{pot}(d),\\ a^W &= b^W,\\ c^W &= d^W,\\ a^M &= d^M,\\ b^M &= c^M. \end{aligned}$$

We show that the vector $\overrightarrow{t_{a,b}}$ is not the zero vector $\overrightarrow{0}$ and in both cones $cone(R)$ and $cone(B)$.

The inequalities imply that the vector $\overrightarrow{t_{a,b}}$ is an element of B and the vector $\overrightarrow{t_{d,c}}$ is an element of R for each RB-split with a potential heuristic h^{pot} that is DDA on Π. The strict inequality implies $\overrightarrow{t_{a,b}} \neq \overrightarrow{0}$.

The vector $\overrightarrow{t_{a,b}}$ is defined as $\overrightarrow{t_{a,b}}[v] = b(v) - a(v)$ and $\overrightarrow{t_{d,c}}[v] = c(v) - d(v)$ for all $v \in V$.

We know from $a^W = b^W$ and $c^W = d^W$ that $\overrightarrow{t_{a,b}}[v] = \overrightarrow{t_{d,c}}[v] = 0$ for each $v \in W$. With $a^M = d^M$ and $b^M = c^M$, we conclude that $\overrightarrow{t_{a,b}}[v] = \overrightarrow{t_{d,c}}[v]$ for all $v \in M$. This implies $\overrightarrow{t_{a,b}} = \overrightarrow{t_{d,c}}$.

Therefore $\overrightarrow{t_{a,b}} \in \mathit{cone}(R) \cap \mathit{cone}(B) \setminus \{\overrightarrow{0}\}$ for each RB-split of Π with any DDA potential heuristic.

□

The RB-split criterion detects a correlation complexity of at least 2 in cases where the 4 states criterion does not. We consider Figure 8 again. Earlier, we showed that the 4 states criterion does not detect that the correlation complexity of this task is at least 2.

Using the vectorization, we see that the vector $\overrightarrow{o_1} = (1,1,1)^\top$ is element of B and the vectors $\overrightarrow{o_2} = (1,1,0)^\top$, $\overrightarrow{o_3} = (1,0,1)^\top$ and $\overrightarrow{o_4} = (0,1,1)^\top$ are elements of R for each RB-split with a DDA heuristic.

We conclude that $\overrightarrow{o_1} \in \mathit{cone}(R) \cap \mathit{cone}(B)$ for each DDA heuristic. So we can use the RB-split criterion to detect that the task of Figure 8 is at least 2.

5.3 Linear Constraints

For a given planing task Π and a family $\mathcal{H}$ of heuristics on Π (for example the family of heuristics that are DDA), we assume we have a black box. That black box provides us with two sets of vectors R' and B' such that for each RB-split $\langle R, B\rangle$ of Π with $h \in \mathcal{H}$ it holds that each $\overrightarrow{r} \in R'$ is an element of R and each $\overrightarrow{b} \in B'$ is an element of B. (For the family of heuristics that are DDA the black box could for example use action landmarks.)

To compute whether or not $\mathit{cone}(R')$ and $\mathit{cone}(B')$ do overlap we determine if a hyperplane that contains $\overrightarrow{0}$ and separates the elements for R' and B' exists. Such a hyperplane implies that all vectors in $\mathit{cone}(R')$ are on one side of the hyperplane and all vectors in $\mathit{cone}(B')$ on the other.

To construct such a hyperplane we use a linear program (LP) without objective function. We represent that LP as

$$A\overrightarrow{w} \geq \begin{pmatrix} \overrightarrow{0} \\ \overrightarrow{1} \end{pmatrix}$$

where A is a matrix of size $n \times m$ and $\overrightarrow{w}$ a vector in $\mathbb{R}^n$. The relation $\geq$ is element-wise and $\begin{pmatrix} \overrightarrow{0} \\ \overrightarrow{1} \end{pmatrix}$ is the vector with $|R'|$-many 0's followed by $|B'|$-many 1's.

For our purpose we put $m = |R'| + |B'|$ and for each $\overrightarrow{r} \in R'$ there is a row $\overrightarrow{r}^T$ in A and for each $\overrightarrow{b} \in B'$ there is a row $-\overrightarrow{b}^T$ in A.

Solving such an LP problem (or detecting the unsolvability) can be done with time complexity polynomial in n. If no solution to the LP exists, then we

know, from Theorem 5.15 that the convex cones are overlapping and therefore each $h \in \mathcal{H}$ has to have a dimension of at least 2. The solution vector $\overrightarrow{w}$ is the normal vector of the hyperplane that separates the cones. We can also extract a 1-dimensional potential heuristic $h^{\blacksquare}$ out of the vector $\overrightarrow{w}$ that assigns to each vector $\overrightarrow{r} \in R'$ a weight $w(\overrightarrow{r}) \geq 0$ and to each vector $\overrightarrow{b} \in B'$ a weight $w(\overrightarrow{b}) < 0$. We extract $h^{\blacksquare}$ by assigning each partial assignment of size 1 the weight $w(\{v \mapsto 0\}) = 0$ and $w(\{v \mapsto 1\}) = \overrightarrow{w}[v]$.

We look at this process in detail for the family $\mathcal{H}$ of DDA heuristics on the following example task and provide graphical interpretations.

$$
\begin{aligned}
V =& \{v_1, v_2, v_3\} \\
dom(v) =& \{0, 1\} \text{ for all } v \in V \\
O =& \{\langle \{v_1 \mapsto 0, v_3 \mapsto 0\}, \{v_3 \mapsto 1\} \rangle, \\
& \langle \{v_1 \mapsto 0, v_2 \mapsto 0, v_3 \mapsto 0\}, \{v_1 \mapsto 1\} \rangle, \\
& \langle \{v_1 \mapsto 1, v_2 \mapsto 0, v_3 \mapsto 0\}, \{v_2 \mapsto 1, v_3 \mapsto 1\} \rangle, \\
& \langle \{v_1 \mapsto 0, v_2 \mapsto 1, v_3 \mapsto 0\}, \{v_2 \mapsto 0, v_3 \mapsto 1\} \rangle, \\
& \langle \{v_1 \mapsto 0, v_2 \mapsto 1, v_3 \mapsto 1\}, \{v_3 \mapsto 0\} \rangle, \\
& \langle \{v_1 \mapsto 0, v_2 \mapsto 1, v_3 \mapsto 1\}, \{v_1 \mapsto 1\} \rangle, \\
& \langle \{v_1 \mapsto 1, v_2 \mapsto 1, v_3 \mapsto 1\}, \{v_3 \mapsto 0\} \} \\
I =& \{v_1 \mapsto 0, v_2 \mapsto 0, v_3 \mapsto 0\} \\
\gamma =& \{\{v_2 \mapsto 1, v_3 \mapsto 0\}\}
\end{aligned}
$$

In this example the black box provides us the sets

$$R' = \{(0, -1, 0)^\top, (0, -1, 1)^\top, (-1, -1, -1)^\top\}$$

and

$$B' = \{(1, 0, 0)^\top, (0, 0, -1)^\top, (1, 1, 0)^\top\}.$$

Figure 11 shows a visual representation of this task, R', and B'. We have $(0, -1, 0)^\top \in R'$ because it represents the transition from 111 to 101 which is the transition from a landmark to a dead-end. The vectors $(1, 0, 0)^\top$ and $(0, 0, -1)^\top$ are in B' because they represent action landmarks. For each DDA heuristic the initial state has to have a greater value than the landmarks and one goal state. Therefore $(-1, -1, -1)^\top \in R'$ and $(1, 1, 0)^\top \in B'$. The vector $(0, -1, 1)^\top$ is the sum of $(0, -1, 0)^\top \in R'$ and the inverse of $(0, 0, -1)^\top \in B'$. Therefore $(0, -1, 1)^\top \in R'$.

Based on R' and B' we create the matrix A where each row corresponds to a vector from R' or B'. $A = \begin{pmatrix} 0 & -1 & 0 \\ 0 & -1 & 1 \\ -1 & -1 & -1 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \\ -1 & -1 & 0 \end{pmatrix}$

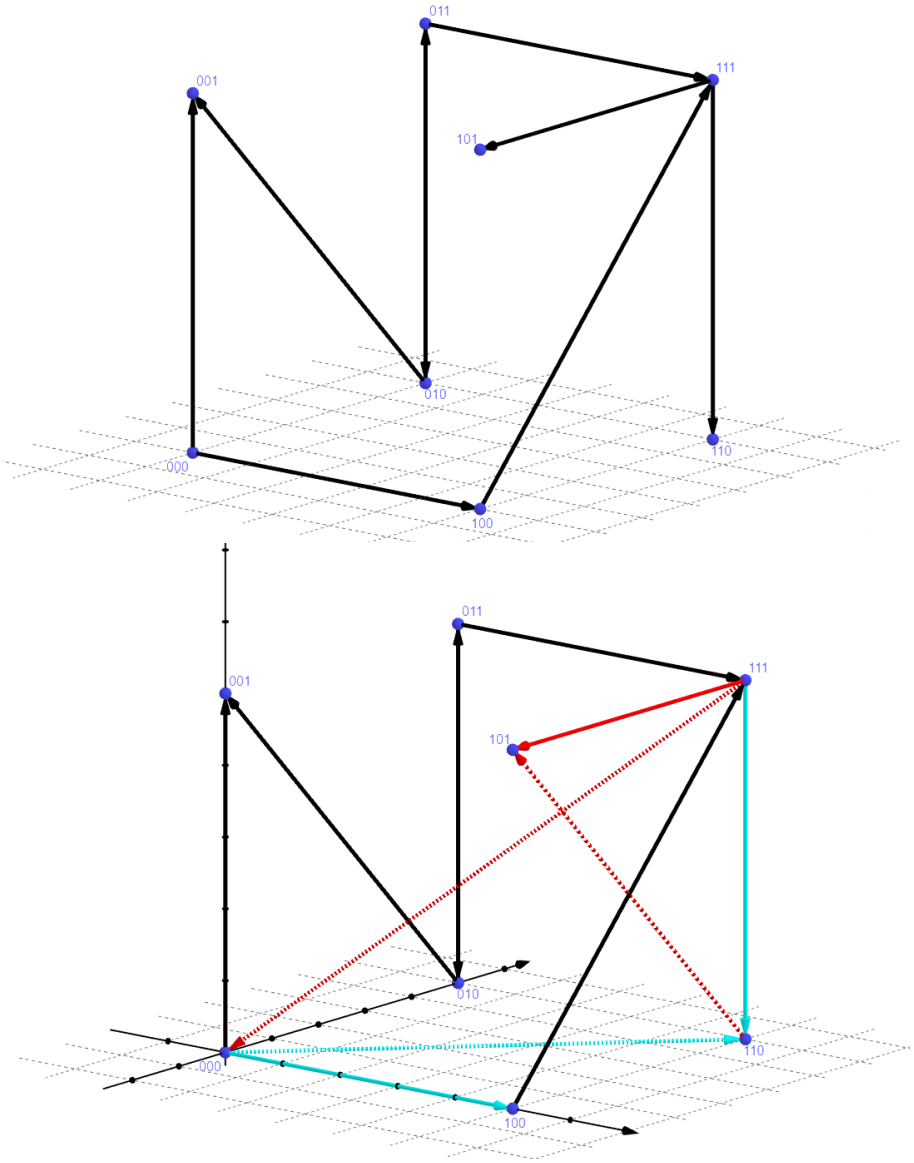


Figure 11: Top: 3D representation of the state space. Bottom: same state space representation with vectors that are colored if detected by the black box. Red color indicates vectors in R', blue color indicates vectors in B', dotted vectors indicate that they are not corresponding to an arch in the state space.
(Graphic created in GeoGebra (Hohenwarter et al., 2013).)

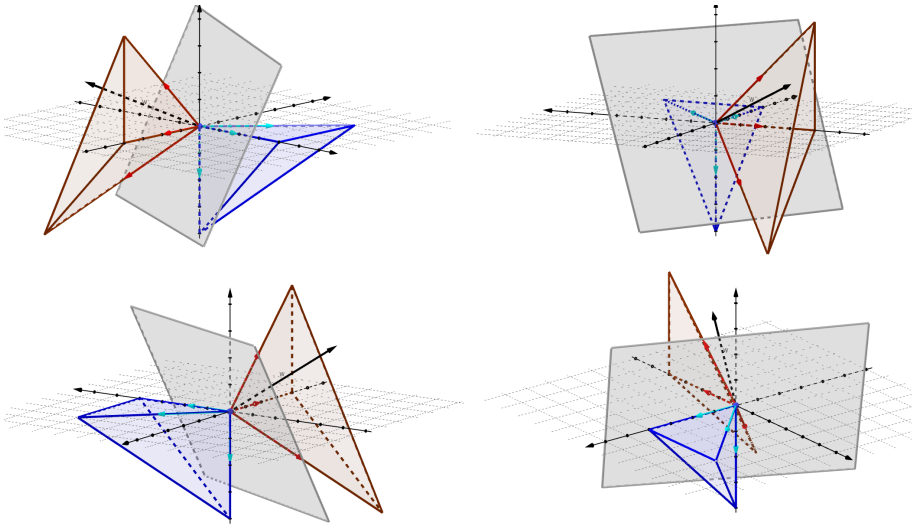


Figure 12: All 4 pictures show the same collection of objects from different points of view. The red and blue vectors are the same as in Figure 11 but the tail of each is at the origin. In orange, we see the convex cone of R' and in dark blue the convex cone of B'. The gray plane is the hyperplane that separates the two cones and the black vector labeled w is the normal of this hyperplane. Dashed lines indicate that the line is behind a different object.
(Graphic created in GeoGebra (Hohenwarter et al., 2013).)

The inequality $A\overrightarrow{w} \geq \begin{pmatrix} \overrightarrow{0} \\ \overrightarrow{1} \end{pmatrix}$ corresponds to this system of linear inequalities:

$$\begin{aligned}
0 \cdot \overrightarrow{w}[1] + \quad & -1 \cdot \overrightarrow{w}[2] + \quad & 0 \cdot \overrightarrow{w}[3] \quad & \geq 0 \\
0 \cdot \overrightarrow{w}[1] + \quad & -1 \cdot \overrightarrow{w}[2] + \quad & 1 \cdot \overrightarrow{w}[3] \quad & \geq 0 \\
-1 \cdot \overrightarrow{w}[1] + \quad & -1 \cdot \overrightarrow{w}[2] + \quad & -1 \cdot \overrightarrow{w}[3] \quad & \geq 0 \\
-1 \cdot \overrightarrow{w}[1] + \quad & 0 \cdot \overrightarrow{w}[2] + \quad & 0 \cdot \overrightarrow{w}[3] \quad & \geq 1 \\
0 \cdot \overrightarrow{w}[1] + \quad & 0 \cdot \overrightarrow{w}[2] + \quad & 1 \cdot \overrightarrow{w}[3] \quad & \geq 1 \\
-1 \cdot \overrightarrow{w}[1] + \quad & -1 \cdot \overrightarrow{w}[2] + \quad & 0 \cdot \overrightarrow{w}[3] \quad & \geq 1
\end{aligned}$$

The vector $\overrightarrow{w} = (-1, -2, 1)^\top$ is a solution for this system of inequalities. This vector is the normal vector of a hyperplane that separates the convex cones of R' and B'. A graphical interpretation of this is shown in Figure 12.

We extract a potential heuristic $h^{\blacksquare}$ out of the vector $\overrightarrow{w}$ by choosing $w(\{v_1 \mapsto 1\}) = \overrightarrow{w}[1] = -1$, $w(\{v_2 \mapsto 1\}) = \overrightarrow{w}[2] = -2$ and $w(\{v_1 \mapsto 1\}) = \overrightarrow{w}[3] = 1$ ($w(\{v_1 \mapsto 0\}) = w(\{v_2 \mapsto 0\}) = w(\{v_3 \mapsto 0\}) = 0$). Calculating the value of $h^{\blacksquare}(s)$ for each state s in the task reveals that the heuristic is in fact DDA. We see the individual heuristic values in Figure 13.

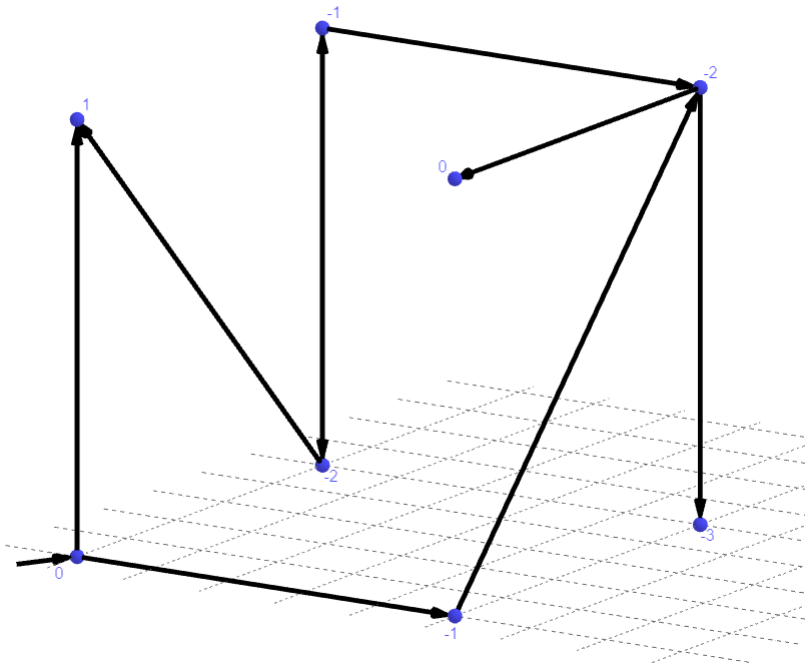


Figure 13: State space from Figure 11 with heuristic values of $h^{\blacksquare}$, extracted from the vector $\overrightarrow{w} = (-1, -2, 1)^\top$.
(Graphic created in GeoGebra (Hohenwarter et al., 2013).)

The heuristic $h^{\blacksquare}$ is not guaranteed to be DDA. For example, the vector $\overrightarrow{w'} = (-1, 0, 1)^\top$ is also a valid solution for the given system of linear inequalities but the potential heuristic extracted from it is not DDA.

For this reason, we introduce a new measure similar to the correlation complexity and present an algorithm to calculate a potential heuristic that is not guaranteed to be DDA but using it guarantees to find the goal with `Simple Hill-climbing`.

6 Comparison

In this chapter, we first introduce the Basel measure. Afterward, we look at the bounds that the correlation complexity, improvability width, novelty width, and Basel measure provide on each other.

6.1 Basel Measure

The correlation complexity of a task is defined as the minimal dimension of a potential heuristic that is DDA. This measure considers all alive states. This can lead to non-intuitive results. We consider the *crossing the river* task but with the modified initial state $I = \{F, R, C, B\}$ that already agrees with the goal γ. This task is intuitively easier than the original and has a novelty width of 0 but the correlation complexity is 2. It did not change because the set of alive states did not change.

To avoid this kind of non-intuitive results and be more comparable to the novelty width, while also being comparable to the correlation complexity of the task, we define the Basel measure (BM). To give an intuition for the Basel measure consider the following, hypothetical scenario. You are in Basel swimming in the Rhine and a duck is floating towards you and asks you for direction to Cologne. You tell it to simply go down the river because the Rhine flows from Basel to Cologne. A swan comes by and asks you for the direction to Nijmegen. You remember that the Rhine splits before reaching Nijmegen (see Figure 14). So you cannot simply tell it to float down the river. It has to take the correct turn at the branching point. A second duck arrives and asks for the direction to the North sea. You know that all branches of the Rhine lead to the North sea so you can give it the same answer as the first duck. You tell it to simply go down the river because it does not matter if it ends up in the IJssel, the Nederrijn-Lek, or the Waal. All of them lead to the goal of the second duck.

It is important that the duck does go down the river until it reaches its goal. If it leaves the river and waddles some on the shore and decides to go to Cologne then it might not work anymore to just go downwards. It might end up at a local minimum. The suggestion to simply go downwards was based on the starting position of the duck. The suggestion does not work for all positions the duck might reach.

In this analogy the bird talking to you correspond to the agent that solves a planning task, the latitude and longitude of a position corresponds to the state and the elevation of the position corresponds to the heuristic value of the state. The position where you are swimming corresponds to the initial state and all positions in Cologne (respectively all positions in Nijmegen or all positions in the North sea) correspond to the goal states.

If we have a descending dead-end avoiding heuristic, we can tell the agent to simply go down the state space topology to reach their goal. We do not know which exact path the agent will take but we know they will end up in a goal state. For the correlation complexity, we look for a heuristic such that we can tell the agent to simply go down the state space topology, no matter at which

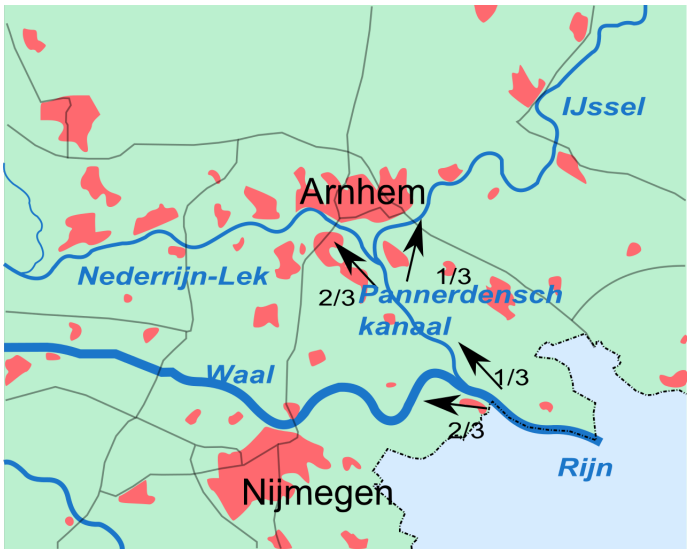


Figure 14: Map of the Rhine (Rijn) branching into the IJssel, the Nederrijn-Lek and the Waal (Wikimedia Commons, 2014).

alive state they are. However, for the Basel measure, it is sufficient if it just works for the state the agent is currently in.

We define the Basel measure as follows.

Definition 6.1 (Basel Measure)**.** The **Basel Measure** of an alive state s in a planning task Π is defined as the minimal dimension of all potential heuristics for Π that guarantee to find a plan from s with `Simple Hill-climbing`. By the Basel measure of a task Π, we mean the Basel measure of the initial state I in this task.

The set of all potential heuristics for Π that guarantee to find a plan from s with `Simple Hill-climbing` contains all DDA heuristics. We call a heuristic on a planning task $\Pi = \langle V, I, O, \gamma \rangle$ that is guaranteed to find a plan from state s with `SHC` a **practically descending and dead-end avoiding** (PDDA) heuristic from s. A heuristic is PDDA for Π if it is PDDA from the initial state I.

Analogous to the RB-split criterion for DDA heuristics and the correlation complexity, we can create a second RB-split criterion for PDDA heuristics and the Basel measure.

Theorem 6.2 (2nd RB-spilt criterion)**.** *Let Π be a planning task. If for each potential heuristic h^{pot} that is PDDA on Π there exists a $\overrightarrow{x} \neq \overrightarrow{0}$ such that $\overrightarrow{x} \in cone(R) \cap cone(B)$, where $\langle R, B \rangle$ is the RB-split of Π with h^{pot}, then the Basel measure of Π is at least 2.*

Proof. The existence of $\overrightarrow{x} \neq \overrightarrow{0}$ such that $\overrightarrow{x} \in cone(R) \cap cone(B)$ implies that

cone(R) and *cone*(B) are overlapping. This implies, with the contrapositive of Theorem 5.14 that $dim(h) \neq 1$. We conclude that the dimension of each h^{pot} that is PDDA on Π is at least 2. Therefore, the Basel measure of Π is at least 2. □

The set of PDDA heuristics for Π contains the set of DDA heuristics for Π. The BM is therefore upper bounded by the correlation complexity and the correlation complexity is lower bounded by the BM. The maximal BM over all alive states is not guaranteed to be the correlation complexity of the task, we consider the following planning task as a counterexample. A binary counter that counts from 11 down to 00 but it is first decided if a little endian or big endian counting is used.

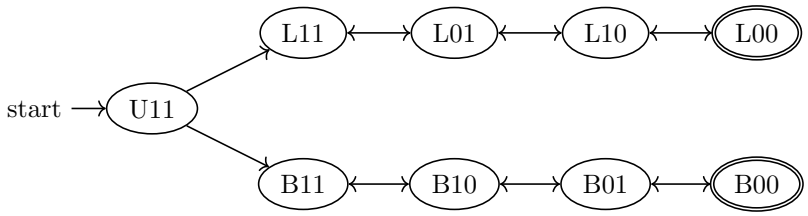


Figure 15: State space of the little/big endian binary counter. The first symbol inside the node indicates the assignment of v with U for *undecied*, L for *little endian* and B for *big endian*. The second symbol shows the assignment of b_0 and the third of b_1.

$$
\begin{aligned}
V =& \{v, b_0, b_1\} \\
dom(v) =& \{\text{undecided}, \text{little endian}, \text{big endian}\} \\
dom(b_0) =& dom(b_1) = \{0, 1\} \\
O =& \{\langle \{v \mapsto \text{undecided}\}, \{v \mapsto \text{little endian}\}\rangle, \\
& \langle \{v \mapsto \text{undecided}\}, \{v \mapsto \text{big endian}\}\rangle, \\
& \langle \{v \mapsto \text{big endian}, b_0 \mapsto 1, b_1 \mapsto 1\}, \{b_1 \mapsto 0\}\rangle, \\
& \langle \{v \mapsto \text{big endian}, b_0 \mapsto 1, b_1 \mapsto 0\}, \{b_1 \mapsto 1\}\rangle, \\
& \langle \{v \mapsto \text{big endian}, b_0 \mapsto 1, b_1 \mapsto 0\}, \{b_0 \mapsto 0, b_1 \mapsto 1\}\rangle, \\
& \langle \{v \mapsto \text{big endian}, b_0 \mapsto 0, b_1 \mapsto 1\}, \{b_0 \mapsto 1, b_1 \mapsto 0\}\rangle, \\
& \langle \{v \mapsto \text{big endian}, b_0 \mapsto 0, b_1 \mapsto 1\}, \{b_1 \mapsto 0\}\rangle, \\
& \langle \{v \mapsto \text{big endian}, b_0 \mapsto 0, b_1 \mapsto 0\}, \{b_1 \mapsto 1\}\rangle, \\
& \langle \{v \mapsto \text{littel endian}, b_0 \mapsto 1, b_1 \mapsto 1\}, \{b_0 \mapsto 0\}\rangle, \\
& \langle \{v \mapsto \text{littel endian}, b_0 \mapsto 0, b_1 \mapsto 1\}, \{b_0 \mapsto 1\}\rangle, \\
& \langle \{v \mapsto \text{littel endian}, b_0 \mapsto 0, b_1 \mapsto 1\}, \{b_0 \mapsto 1, b_1 \mapsto 0\}\rangle, \\
& \langle \{v \mapsto \text{littel endian}, b_0 \mapsto 1, b_1 \mapsto 0\}, \{b_0 \mapsto 0, b_1 \mapsto 1\}\rangle, \\
& \langle \{v \mapsto \text{littel endian}, b_0 \mapsto 1, b_1 \mapsto 0\}, \{b_0 \mapsto 0\}\rangle\} \\
& \langle \{v \mapsto \text{littel endian}, b_0 \mapsto 0, b_1 \mapsto 0\}, \{b_0 \mapsto 1\}\rangle\} \\
I =& \{v \mapsto \text{undecided}, b_0 \mapsto 1, b_0 \mapsto 1\} \\
\gamma =& \{b_0 \mapsto 0, b_1 \mapsto 0\}
\end{aligned}
$$

Figure 15 provides a visualization of the task.

For all states s with $(v \mapsto \text{little endian})$ the weights $w(b_0 \mapsto 1) = 1$ and $w(b_1 \mapsto 1) = 2$ provide a potential heuristic of dimension 1 that is PDDA from s.

For all other states s' the weights $w(v \mapsto \text{undecided}) = 1$, $w(b_0 \mapsto 1) = 2$ and $w(b_1 \mapsto 1) = 1$ provide a potential heuristic of dimension 1 that is PDDA from s'.

However, the correlation complexity of the task is at least 2 since the operators $\langle \{v \mapsto \text{big endian}, b_0 \mapsto 1, b_1 \mapsto 0\}, \{b_0 \mapsto 0, b_1 \mapsto 1\}\rangle$ and $\langle \{v \mapsto \text{littel endian}, b_0 \mapsto 0, b_1 \mapsto 1\}, \{b_0 \mapsto 1, b_1 \mapsto 0\}\rangle$ are both critical but inverse of each other.

To evaluate the Basel measure of a given task in practice, we use the theoretical results from Chapter 5. There, we assumed to have a black box for a family $\mathcal{H}$ of heuristics. This black box provides us with two sets of vectors R' and B' such that for each RB-split $\langle R, B\rangle$ of Π with $h \in \mathcal{H}$ it holds that each $\overrightarrow{r} \in R'$ is an element of R and each $\overrightarrow{b} \in B'$ is an element of B.

For the family of heuristics that are PDDA for Π, this black box could use action landmarks.

In some tasks there are no action landmarks. The task in Figure 7 is an example for that. Therefore, we investigate how we can deal with disjunctive

action landmarks (DALMs).

Definition 6.3 (Disjunctive Action Landmark)**.** A set D of operators is called a disjunctive action landmark of a task if each plan of this task contains at least one of the operators in D.

A given DALM $D = \{o_1, o_2, \ldots, o_t\}$ induces the constraint $\bigvee_{i=1}^{t} \overrightarrow{w} \cdot \overrightarrow{o_i} < 0$. The interpretation of this constraint is that at least one operator of D has to strictly improve the heuristic value. To express such constraints, we use a mixed integer program (MIP), instead of an LP, as we did previously because a MIP is more expressive. However, solving a MIP is NP-complete (Schrijver, 1998).

This approach can find a valid $\overrightarrow{w}$ where the extracted potential heuristic h^w does not guarantee to be PDDA fro Π. To tackle this problem, we refine the solution in a similar fashion as Francès et al. (2019). The main difference is that they stop to refine further as soon as they find a plan. We refine until we find a PDDA heuristic (or detect that no PDDA heuristic exists).

Algorithm 6: `Local Optimum Search`

Data: planning task $\langle V, I, O, \gamma \rangle$, heuristic h
Result: descending path π to a local optimum, set of operators $\hat{O}$ that are applicable in $I[\![\pi]\!]$

```
1  open := [makeRootNode(I)]
2  while open is not empty do
3  |  node := pop last element of open
4  |  s := node.state
5  |  N := {o ∈ O | h(s[[o]]) < h(s), o applicable in s, s[[o]] ⊈ γ}
6  |  if N = ∅ then
7  |  |  return extractPath(node), {o ∈ O | o applicable in s}
8  |  end
9  |  foreach o ∈ N do
10 |  |  n' := makeNode(s, o, s[[o]]) append n' to open
11 |  end
12 end
13 return None
```

To refine the solution, we do a search with Algorithm 6 (`Local Optimum Search`) for a path to a local optimum $\hat{s}$ that is reachable from the initial state by a path π, where π consists only of operators that improve the heuristic value and π does not visit any goal state. The existence of such a state implies that the heuristic h^w is not PDDA. A necessary condition for h^w to be PDDA is that $\hat{s}$ has a successor that improves the heuristic value or at least one of the operators in π does not improve the heuristic. We express that as the constraint

$$\bigvee_{o \in \pi} \overrightarrow{w} \cdot \overrightarrow{o} \geq 0 \vee \bigvee_{o \text{ applicable in } \hat{s}} \overrightarrow{w} \cdot \overrightarrow{o} < 0.$$

We add this constraint to the MIP and update $\overrightarrow{w}$. This way, we eventually end up with h^w being PDDA or the MIP becomes unsolvable which implies that no potential heuristic of dimension 1 that is PDDA exists for this task.

6.2 Basel Measure vs. Novelty Width

The Novelty Width of a task provides an upper bound for the Basel measure, too. For the constructive proof, we use discriminating facts.

Definition 6.4 (Discriminating Facts)**.** Let s, s' be states. If $s \neq s'$ then there exists at least one fact $(v \mapsto d') \in s'$ with $(v \mapsto d') \notin s$. We call the set of such facts the **discriminating facts** from s to s'. We denote the set of discriminating facts as $\delta(s, s') := s' \setminus s$.

Theorem 6.5. *The Basel measure is upper bounded by the novelty width +1.*

Proof. We look at the search tree of a `Novelty Width Algorithm` with input width k that found the plan π for a planning task $\Pi = \langle V, I, O, \gamma \rangle$. That plan π traverses through the states $\pi_0, \pi_1, \ldots, \pi_L$ with $\pi_0 = I$.

To guarantee that a `Simple Hill-climbing` traverses through the same states we create a potential heuristic h were each successor $s \in succ(\pi_{i-1}) \setminus \{\pi_i\}$ holds $h(\pi_{i-1}) \leq h(s)$ and $h(\pi_{i-1}) > h(\pi_i)$ for $i \in \{1, \ldots, L\}$.

Let p_i^* be one arbitrarily chosen partial assignment that was novel in π_i with $|p_i^*| = k$. We can split $succ(\pi_{i-1})$ into the disjoint sets $\{\pi_i\}, \{\pi_{i-1}\} \cap succ(\pi_{i-1}), S_i, C_i, Z_i$ with:

- $S_i := \{s_i \in succ(\pi_{i-1}) \mid p_i^* \subseteq s_i, s_i \neq \pi_i)\}$
- $C_i := \{c_i \in succ(\pi_{i-1}) \mid p_i^* \not\subseteq c_i, p_{i-1}^* \subseteq c_i, c_i \neq \pi_{i-1}\}$
- $Z_i := \{z_i \in succ(\pi_{i-1}) \mid p_i^* \not\subseteq z_i, p_{i-1}^* \not\subseteq z_i\}$

The set S_i contains the successors that also agree with the novel assignment p_i^*. The set C_i contains the successors that also agree with the assignment p_{i-1}^* that was novel in the π_{i-1} and do not agree with the novel assignment p_i^*. The set Z_i contains the successors that do not agree with p_i^* nor with p_{i-1}^*.

Let $\delta^*(s, s')$ be one arbitrarily chosen element from the discriminating facts $\delta(s, s')$. Let $F_i := \{\delta^*(\pi_i, s_i) \mid s_i \in S_i\}$. If a successor x of π_{i-1} agrees with a fact in F_i, then we know that $x \neq \pi_i$. Let $T_i := \{\delta^*(\pi_{i-1}, c_i) \mid c_i \in C_i\}$. If a successor x of π_{i-1} agrees with a fact in T_i, then we know that $x \neq \pi_{i-1}$.

Let $Q_i := \{p_i^* \cup \{f_i\} \mid f_i \in F_i\}$ and $G_i := \{p_{i-1}^* \cup \{t_i\} \mid t_i \in T_i\}$. The idea is to give p_i^* a negative weight to have a low heuristic value for the state π_i but also giving all $p_i^* \cup \{f_i\}$ in Q_i a higher positive weight to steer `Simple Hill-climbing` away from the states in S_i.

It could be the case that one $c_i \in C_i$ agrees with more p_j with $j < i-1$ than π_{i-1} does. This would lead to more negative summands in the evaluation of the potential heuristic, which could lead to $h(c_i) < h(\pi_{i-1})$. This would allow

`Simple Hill-climbing` to take the wrong successor. To compensate that we give all $p^*_{i-1} \cup \{t_i\}$ in G_i a high weight.

Let $n := |V|$ be the number of state variables in the considered task and the auxiliary constant $\Omega := \binom{n}{k+1} + 2$. We chose the weights for the potential heuristic with $w(p^*_i) := -\Omega^{2 \cdot i}$, $w(q_i) := +\Omega^{2 \cdot i+1}$ for all $q_i \in Q_i$ and $w(g_i) := +\Omega^{2 \cdot i-1}$ for all $g_i \in G_i$. (For all other partial assignments the weight is 0.)

Looking at the partial assignments p^*_j for $j \in \{1, \ldots, L\}$. We know p^*_j does not agree with any π_i, $s_i \in S_i$, $c_i \in C_i$ or $z_i \in Z_i$ where $i < j$ because p^*_j is novel in π_j. We know p^*_j does not agree with any $c_i \in C_i$ where $i = j$ due to the definition of C_i. We know p^*_j does not agree with any $z_i \in Z_i$ where $i = j$ or $i = j + 1$ due to the definition of Z_i.

Each $s_i \in S_i$ and π_i agrees with the partial assignments p^*_j where $i = j$. Each $c_i \in C_i$ agrees with the partial assignments p^*_j where $i = j + 1$.

Looking at the partial assignments $q_j \in Q_j$ for $j \in \{1, \ldots, L\}$ we know that all states that agree with q_j agree with p^*_j, too. We know $q_j \in Q_j$ does not agree with any π_i, $c_i \in C_i$ or $z_i \in Z_i$ where $i < j$ because p^*_j is novel in π_j. We know q_j does not agree with π_i where $j = i$ because q_j contains one fact that disagrees with π_i. We know q_j does not agree with any $c_i \in C_i$ where $i = j$ because q_j agrees with p^*_i and c_i does not. We know q_j does not agree with any $z_i \in Z_i$ where $i = j$ because q_j agrees with p^*_i and z_i does not. Each $s_i \in S_i$ agrees with at least one $q_j \in Q_j$ if $i = j$.

Looking at the partial assignments $g_j \in G_j$ for $j \in \{1, \ldots, L\}$ we know that all states that agree with g_j agree with p^*_{j-1}, too. We know $g_j \in G_j$ does not agree with any π_i, $s_i \in S_i$, $c_i \in C_i$ or $z_i \in Z_i$ where $i < j - 1$ because p^*_{j-1} is novel in π_{j-1}. We know g_j does not agree with π_i where $i = j - 1$ because g_j contains one fact from T_j that disagrees with π_i. We know g_j does not agree with any $c_i \in C_i$ where $i = j - 1$ because g_j agrees with p^*_i and c_i does not. We know g_j does not agree with any $z_i \in Z_i$ where $i = j - 1$ because g_j agrees with p^*_i and z_i does not. Each $c_i \in C_i$ agrees with at least one $g_j \in G_j$ if $i = j$.

For any state s the heuristic is evaluated with

$$h(s) = \sum_{j=0}^{L} -\Omega^{2j} \cdot [p^*_j \subseteq s] + \sum_{j=0}^{L} \sum_{q \in Q_j} \Omega^{2j+1} \cdot [q \subseteq s] + \sum_{j=0}^{L} \sum_{g \in G_j} \Omega^{2j-1} \cdot [g \subseteq s].$$

Since p^*_j does not agree with any π_i where $j > i$ we know that

$$h(\pi_i) = \sum_{j=0}^{i} -\Omega^{2j} \cdot [p^*_j \subseteq \pi_i] + \sum_{j=0}^{i} \sum_{q \in Q_j} \Omega^{2j+1} \cdot [q \subseteq \pi_i] + \sum_{j=0}^{i} \sum_{g \in G_j} \Omega^{2j-1} \cdot [g \subseteq \pi_i].$$

For the lower bound, we consider the case that π_i agrees with all partial assignments with a negative weight and none of the partial assignments with a positive weight. We conclude the lower bound:

$$\sum_{j=0}^{i} -\Omega^{2j} + 0 + 0 \leq h(\pi_i).$$

For the upper bound we considering that p_i^* agrees with π_i and there are $\binom{n}{k+1}$ partial assignments of size $k+1$ that agree with a given state. We conclude the upper bounds:

$$h(\pi_i) \leq -\Omega^{2\cdot i} + \binom{n}{k+1} \cdot \Omega^{2\cdot(i-1)+1} + 0 \leq 0,$$

$$\begin{aligned} h(\pi_i) &\leq -\Omega^{2\cdot i} + \binom{n}{k+1} \cdot \Omega^{2\cdot(i-1)+1} + 0 \\ &\leq -(\Omega \cdot \Omega^{2\cdot i-1}) + \binom{n}{k+1} \cdot \Omega^{2\cdot i-1} + 0 \\ &< -\Omega^{2\cdot i-1} \text{ for } \Omega > \binom{n}{k+1} + 1 \\ &\leq \sum_{j=0}^{i-1} -\Omega^{2\cdot j} \\ &\leq h(\pi_{i-1}). \end{aligned} \tag{3}$$

For any $s_i \in S_i$ there is at least one $q \in Q_i$ that agrees with s_i. This implies the lower bound:

$$\begin{aligned} h(s_i) &\geq \sum_{j=0}^{i} -\Omega^{2\cdot j} + \Omega^{2\cdot i+1} + 0 \\ &\geq -2 \cdot \Omega^{2\cdot i} + \Omega^{2\cdot i+1} \text{ for } \Omega > 2 \\ &\geq 0 \text{ for } \Omega > 2 \\ &\geq h(\pi_{i-1}). \end{aligned} \tag{4}$$

For any $c_i \in C_i$ there is at least one $g \in G_i$ that agrees with c_i. The partial assignment p_i^* does not agree with c_i. This implies the lower bound:

$$\begin{aligned} h(c_i) &\geq \sum_{j=0}^{i-1} -\Omega^{2\cdot j} + 0 + \Omega^{2\cdot i-1} \\ &\geq -2 \cdot \Omega^{2\cdot i-2} + \Omega^{2\cdot i-1} \text{ for } \Omega > 2 \\ &\geq 0 \text{ for } \Omega > 2 \\ &\geq h(\pi_{i-1}). \end{aligned} \tag{5}$$

The partial assignments p_i^* and p_{i-1}^* do not agree with any $z_i \in Z_i$. We know from (3) that $\sum_{j=0}^{i-1} -\Omega^{2\cdot j} \geq h(\pi_i)$. This implies the lower bound:

$$\begin{aligned} h(z_i) &\geq \sum_{j=0}^{i-2} -\Omega^{2\cdot j} + 0 + 0 \\ &\geq h(\pi_{i-1}). \end{aligned} \tag{6}$$

We conclude that each successor $s \in succ(\pi_{i-1}) \setminus \{\pi_i\}$ holds $h(\pi_{i-1}) \leq h(s)$ and $h(\pi_{i-1}) > h(\pi_i)$. This implies that π_i is the only successor of π_{i-1} that passes the `progressCheck` of `SHC` with *reference* $= \pi_{i-1}$. The heuristic h forces `SHC` to follow the plan evaluated by `NWA`. This implies that h guarantees to find a plan from I with `SHC`. The partial assignments with non-zero weights are of size k and $k+1$. We conclude that the Basel measure of Π is less or equal to $k+1$.

□

This approach of creating a PPDA heuristic is not practical. It requires calculating a plan beforehand and the weights are exponential to the length of the plan.

Note that `NWA` can find a plan with k smaller than the novelty width. We remember, the smallest k for which a `NWA` finds a satisficing plan is the effective novelty width. In the proof of Theorem 6.5 it is not used that k is the novelty with but only that `NWA` with input width k found a (satisficing) plan π.

Corollary 6.6. *The Basel measure is upper bounded by the effective novelty width* $+1$.

Proof. Same as the proof for Theorem 6.5 □

We showed a way to produce a potential heuristic that leads with `Simple Hill-climbing` to a goal state for a task Π with a dimension that is 1 larger than the (effective) novelty width of Π. So the BM of a task is upper bounded by the (effective) novelty width $+1$. However, we did not answer if this bound is a non-strict inequality. Is there a task where the BM is larger than the (effective) novelty width?

The answer is yes. We show this with the bit shift task as an example task with a BM of 2 and (effective) novelty width of 1.

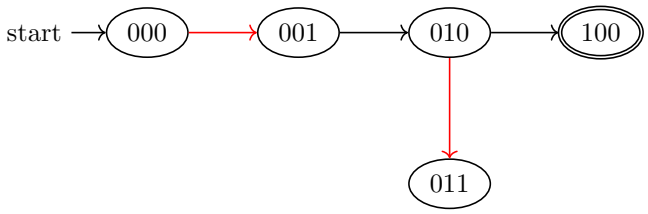


Figure 16: State space of the bit shift task. A node with label xyz represents the state $\{b_2 \mapsto x, b_1 \mapsto y, b_0 \mapsto z\}$. The transitions corresponding to the operator that is dangerous and critical is indicated in red.

$$
\begin{aligned}
V =& \{b_0, b_1, b_2\} \\
dom(b_0) =& dom(b_1) = dom(b_2) = \{0, 1\} \\
O =& \{\langle \{b_0 \mapsto 0, b_2 \mapsto 0\}, \{b_2 \mapsto 1\} \rangle, \\
& \langle \{b_1 \mapsto 0, b_2 \mapsto 1\}, \{b_1 \mapsto 1, b_2 \mapsto 0\} \rangle, \\
& \langle \{b_0 \mapsto 0, b_1 \mapsto 1, b_2 \mapsto 0\}, \{b_0 \mapsto 1, b_1 \mapsto 0\} \rangle \} \\
I =& \{b_0 \mapsto 0, b_1 \mapsto 0, b_2 \mapsto 0\} \\
\gamma =& \{b_2 \mapsto 1\}
\end{aligned}
$$

In Figure 16 it is easy to see that the operator indicated in red is critical and dangerous. Each PDDA potential heuristic h^{pot} for this task has to fulfill the inequalities $h^{pot}(000) > h^{pot}(001)$ and $h^{pot}(010) \leq h^{pot}(011)$. This implies that the vector $\overrightarrow{x} = (0, 0, 1)^\top$ an element of R and of B of the RB-split $\langle R, B \rangle$. We conclude, with the 2nd RB-split criterion that the BM of this task is at least 2.

With weights $w(b_0 \mapsto 1) = -1$, $w(b_1 \mapsto 1) = -2$, $w(b_2 \mapsto 1) = -4$, and $w(b_0 \mapsto 1, b_1 \mapsto 1) = +3$ (all other weights are 0), we see that the BM is exactly 2.

Starting from the state 000 each successor provides a novel fact, except the state 011. Since this state space has only one state with branching and one of them is pruned by `NWA` and all successor, except this pruned one, provide novel facts we conclude that the (effective) novelty width of the bit shift task is 1.

The (effective) novelty width is not bounded by the Basel measure. For example, the binary counter task has the (effective) novelty width equal to the number of bits used but the correlation complexity and Basel measure is 1.

6.3 Basel Measure vs. Persistent Hamming Improvability Width

To compare the novelty width with the Basel measure, we adjusted the weights in a way that forces `SHC` to follow the plan that a `NWA` found. This approach is problematic for an `IWA` because an `IWA` can produce plans with cycles. Consider the example in Figure 17.

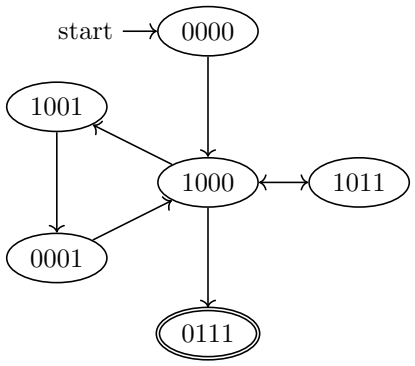


Figure 17: State space of a task that produces a cyclic plan with **IWA**. A node with label $wxyz$ represents the state $\{b_0 \mapsto w, b_1 \mapsto x, b_2 \mapsto y, b_3 \mapsto z\}$.

The initial state of this task is $\{b_0 \mapsto 0, b_1 \mapsto 0, b_2 \mapsto 0, b_3 \mapsto 0\}$ and $\gamma = \{b_1 \mapsto 1, b_2 \mapsto 1, b_3 \mapsto 1\}$. The persistent Hamming improvability width of this task is 2. From the initial state only the state 1000 is reachable. This state does not pass the `progressCheck` so the successors of 1000 that differ only on 2 variables to 0000 are considered. There is only 1001 because 1011 and 0111 differ on 3 variables to 0000. On 1001 the `progressCheck` passes. The next state 0001 does not pass the `progressCheck` so we visit 1000 again. The successors of 1000 (that are not duplicates in this moment) are 1011 and 0111. However, 0111 differs on 3 variables to 1001 so it is not considered. Leaving us with 1011, which differs only on one variable to 1001. So we get to 1011, which passes the `progressCheck`. The only successor of 1011 is, yet again, 1000. The available successors are now 0111 and 1001 (it is not a duplicate anymore because if the `progressCheck` passes the *closed* set refreshes). However, 1001 does not pass the `progressCheck` leaving us with 0111. This state is a goal state and therefore the search ends.

The BM for this task is 1. Consider the weights $w(\{b_0 \mapsto 1\}) = -1$, $w(\{b_1 \mapsto 1\}) = -3$, $w(\{b_3 \mapsto 1\}) = 1$, and 0 for all other weights.

There is also an example task where the BM is 1 greater than the persistent Hamming improvability width.

The example task in Figure 18 is a modification of the Gray code counter. With $\gamma = \{v_0 \mapsto 1, v_1 \mapsto 1\}$ we see that the persistent Hamming improvability width is 2. The state 0110 is reachable without visiting a state that has a Hamming distance greater than 2 to 0000. The state 0110 passes the `progressCheck`. The goal state 1110 is reachable from 0110 without visiting a state that has a Hamming distance greater than 2 to 0110. Therefore, the persistent Hamming improvability width is 2. We see that the task contains the Gray code counter of size 3 and we already know that this task has correlation complexity of 3. Since there is no branching (besides going back), the BM for this task is 3, as well.

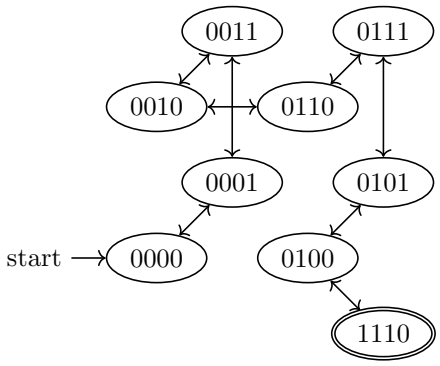


Figure 18: State space of the modified Gray code counter task of size 3. A node with label $wxyz$ represents the state $\{v_0 \mapsto w, v_1 \mapsto x, v_2 \mapsto y, v_3 \mapsto z\}$.

6.4 Improvability Width vs. Novelty Width

We saw that the BM has an upper bound based on the novelty width. Unfortunately, we cannot find such a bound for general tasks with the novelty width and improvability width. To support this claim, we show that it is possible to produce a task with arbitrary (Hamming) improvability width but fixed novelty width and vice versa.

$$\begin{aligned} V =& \{b_1, \ldots, b_n\} \\ dom(b_i) =& \{0, 1\} \text{ for all } i \in \{1, \ldots, n\} \\ O =& \{\langle \{b_1 \mapsto 0, \ldots, b_n \mapsto 0\}, \{b_1 \mapsto 1, \ldots, b_n \mapsto 1\} \rangle\} \\ I =& \{b_1 \mapsto 1, \ldots, b_n \mapsto 1\} \\ \gamma =& \{b_1 \mapsto 1\} \end{aligned}$$

This task has only two reachable states. The distance between these states is n so the (persistent) (Hamming) width is n. Each fact in the successor is novel so the novelty width is 1.

Looking at Figure 19 we see a state space of the planning task with only one goal fact described below.

$$
\begin{aligned}
V =& \{c_1, \ldots, c_m, b_1, b_2, b_3\} \\
dom(b_i) =& \{0, 1\} \text{ for all } i \in \{1, 2, 3\} \\
dom(c_i) =& \{R, W\} \text{ for all } i \in \{1, \ldots, m\} \\
O =& \{\langle \{c_1 \mapsto R, c_2 \mapsto R, \ldots, c_{m-1} \mapsto R, c_m \mapsto R, b_1 \mapsto 0, b_2 \mapsto 0, b_3 \mapsto 0\}, \\
& \quad \{c_1 \mapsto R, c_2 \mapsto R, \ldots, c_{m-1} \mapsto R, c_m \mapsto R, b_1 \mapsto 0, b_2 \mapsto 1, b_3 \mapsto 1\}\rangle, \\
& \quad \langle \{c_1 \mapsto R, c_2 \mapsto R, \ldots, c_{m-1} \mapsto R, c_m \mapsto R, b_1 \mapsto 0, b_2 \mapsto 0, b_3 \mapsto 0\}, \\
& \quad \{c_1 \mapsto W, c_2 \mapsto R, \ldots, c_{m-1} \mapsto R, c_m \mapsto R, b_1 \mapsto 0, b_2 \mapsto 1, b_3 \mapsto 0\}\rangle, \\
& \quad \langle \{c_1 \mapsto R, c_2 \mapsto R, \ldots, c_{m-1} \mapsto R, c_m \mapsto R, b_1 \mapsto 0, b_2 \mapsto 0, b_3 \mapsto 0\}, \\
& \quad \{c_1 \mapsto R, c_2 \mapsto W, \ldots, c_{m-1} \mapsto R, c_m \mapsto R, b_1 \mapsto 0, b_2 \mapsto 1, b_3 \mapsto 0\}\rangle, \\
& \quad \vdots, \\
& \quad \langle \{c_1 \mapsto R, c_2 \mapsto R, \ldots, c_{m-1} \mapsto R, c_m \mapsto R, b_1 \mapsto 0, b_2 \mapsto 0, b_3 \mapsto 0\}, \\
& \quad \{c_1 \mapsto R, c_2 \mapsto R, \ldots, c_{m-1} \mapsto W, c_m \mapsto R, b_1 \mapsto 0, b_2 \mapsto 1, b_3 \mapsto 0\}\rangle, \\
& \quad \langle \{c_1 \mapsto R, c_2 \mapsto R, \ldots, c_{m-1} \mapsto R, c_m \mapsto R, b_1 \mapsto 0, b_2 \mapsto 0, b_3 \mapsto 0\}, \\
& \quad \{c_1 \mapsto R, c_2 \mapsto R, \ldots, c_{m-1} \mapsto R, c_m \mapsto W, b_1 \mapsto 0, b_2 \mapsto 1, b_3 \mapsto 0\}\rangle, \\
& \quad \langle \{c_1 \mapsto R, c_2 \mapsto R, \ldots, c_{m-1} \mapsto R, c_m \mapsto R, b_1 \mapsto 0, b_2 \mapsto 0, b_3 \mapsto 0\}, \\
& \quad \{c_1 \mapsto R, c_2 \mapsto R, \ldots, c_{m-1} \mapsto R, c_m \mapsto R, b_1 \mapsto 0, b_2 \mapsto 0, b_3 \mapsto 1\}\rangle, \\
& \quad \langle \{c_1 \mapsto R, c_2 \mapsto R, \ldots, c_{m-1} \mapsto R, c_m \mapsto R, b_1 \mapsto 0, b_2 \mapsto 0, b_3 \mapsto 1\}, \\
& \quad \{c_1 \mapsto R, c_2 \mapsto R, \ldots, c_{m-1} \mapsto R, c_m \mapsto R, b_1 \mapsto 0, b_2 \mapsto 1, b_3 \mapsto 0\}\rangle, \\
& \quad \langle \{c_1 \mapsto R, c_2 \mapsto R, \ldots, c_{m-1} \mapsto R, c_m \mapsto R, b_1 \mapsto 0, b_2 \mapsto 1, b_3 \mapsto 0\}, \\
& \quad \{c_1 \mapsto R, c_2 \mapsto R, \ldots, c_{m-1} \mapsto R, c_m \mapsto R, b_1 \mapsto 1, b_2 \mapsto 0, b_3 \mapsto 0\}\rangle, \\
I =& \{c_1 \mapsto R, c_2 \mapsto R, \ldots, c_{m-1} \mapsto R, c_m \mapsto R, b_1 \mapsto 0, b_2 \mapsto 0, b_3 \mapsto 0\} \\
\gamma =& \{b_1 \mapsto 1\}
\end{aligned}
$$

This task has m facts that have to stay untouched during the only plan of this task.

For each partial assignment of size less than $|V| - 1$ that agrees with the 3rd state of the plan,

$$p \subsetneq \{c_1 \mapsto R, \ldots, c_m \mapsto R, b_1 \mapsto 0, b_2 \mapsto 1, b_3 \mapsto 0\}, |p| < |V| - 1$$

there exists a state $s \in succ(I)$ such that $p \subsetneq s$. The smallest partial assignment that is novel in the 3rd state is $\{c_1 \mapsto R, \ldots, c_m \mapsto R, b_2 \mapsto 1, b_3 \mapsto 0\}$ (missing only the fact $b_1 \mapsto 0$). Therefore, the smallest partial assignment that is novel in the 3rd state has size $|V| - 1 = m + 2$. So the novelty width of the task is $m + 2$. Each state in the plan differs only on one fact from the initial state. Therefore, the (persistent) Hamming improvability width is 1. The plan changes only 3 facts, therefore the (persistent) improvability width is 3.

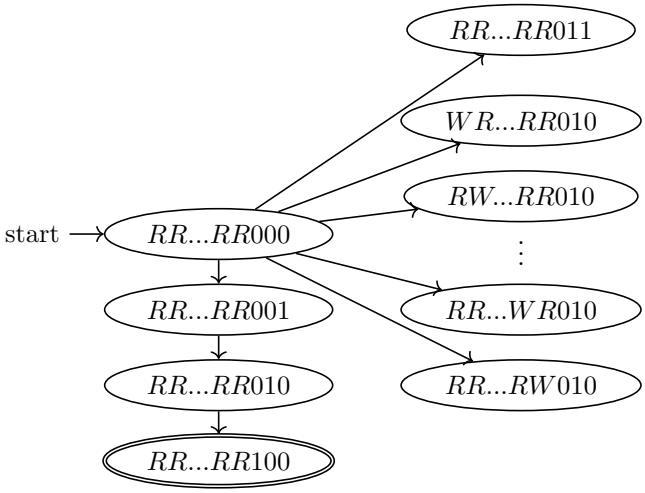


Figure 19: State space of a task with arbitrarily large novelty width but constant (persistent) (Hamming) improvability width. A node with label $wx...yzabc$ represents the state $\{c_1 \mapsto w, c_2 \mapsto x, \ldots, c_{m-1} \mapsto y, c_m \mapsto z, b_1 \mapsto a, b_2 \mapsto b, b_3 \mapsto c\}$.

7 Experimental Results

We implemented the `Local Optimum Search` in the Fast Downward[6] planning system (Helmert, 2006) and tested how often the initial heuristic has to be refined until a PDDA heuristic was found or the MIP was unsolvable. The domains we investigated are Gripper (IPC 1998), VisitAll (IPC 2011), Blocks (IPC 2000), Pegsol (IPC 2008), and Movie (IPC 1998). The first 3 listed domains have a correlation complexity of at most 2 for each task (Seipp et al., 2016).

We investigated each instance of these domains. First with additional initial constraints that were created with DALMs from the implementation of Büchner et al. (2021a,b) based on h^m landmarks (Keyder et al., 2010) and RHW landmarks (Richter et al., 2008) and second without any previous knowledge of the task. To evaluate the MIP, we used IBM CPLEX 20.1. All calculations were performed on an Intel Xeon Silver 4114 processor running at 2.2 GHz at sciCORE[7] scientific computing center at the University of Basel. We set the time limit to 1800 seconds (30 minutes) and the memory limit to 3.5 GiB.

We list the number of binary variables, continuous variables, and iterations used to detect whether or not the BM of the task is 1 as well as the runtime. In Table 7 and Table 8, we see the individual results.

The number of continuous variables in the MIP is the number of facts in the task and the number of weights for the potential heuristic.

We see for the domains Gripper and Blocks that they have, for all tested instances, a Basel measure of 2. In the VisitAll domain, we get for `problem03--half.pddl` a Basel measure of 2. The task has an agent that starts in the middle of a 3 by 3 grid. Goal is to visit location $0-0$, $0-2$, $1-1$, $2-0$, and $2-1$. A potential heuristic of dimension 1 with the weight $w(\textit{location-x-y-is-visited} \mapsto \textit{true}\}) = -1$ for each location would provide a PDDA heuristic. However, Fast Downward removes in the translation the *location-x-y-is-visited* variables if they are not goal variables. The translator removes them because they are not part of the goal nor in any precondition and therefore seemingly irrelevant. However, the translated task has a BM of 2 while the original one has a BM of 1.

The Basel measure is 1 for all tested instances of the Movie domain. This aligns with the intuition of this simple domain. Here we see that the initial constraints created with DALMs, are beneficial for the number of binary variables as well as the number of iterations.

In most cases, the usage of the constraints from the DALMs reduces the number of iterations needed to detect whether or not the BM of the task is 1. However, they cause, in most cases, the usage of more binary variables.

The difference in the runtime shows that the constraints from the DALMs are beneficial in most cases. In the cases where the calculation was faster without initial constraints, the difference is less than 35%. The exception to this is `p02.pddl` from the Pegsol domain. The calculation with the additional constraints did not finish in the 1800 seconds limit and the calculation without

[6] http://www.fast-downward.org
[7] http://scicore.unibas.ch/

additional constraints took 1578 seconds. Therefore, we do not know if the difference there is also in this 35% margin.

In some cases of the Blocks domain, the constraints from the DALMs reduced the runtime by orders of magnitude.

task	binary variables		continuous variables	iterations		BM	runtime (s)	
	DALMs	none		DALMs	none		DALMs	none
`gripper:`								
`prob01.pddl`	62	**27**	24	25	**22**	≥ 2	191	**130**
`prob02.pddl`	91	**39**	34	**39**	44	≥ 2	510	**434**
`prob03.pddl`	120	**51**	44	**57**	74	≥ 2	853	**785**
`prob04.pddl`	156	**63**	54	**78**	112	≥ 2	**1279**	1759
`prob05.pddl`	**173**	-	64	**83**	-	≥ 2	**1428**	-
`prob07.pddl`	**226**	-	84	**99**	-	≥ 2	**1598**	-
`visitall-opt11-strips:`								
`problem02-full.pddl`	**19**	20	10	**5**	9	1	**2**	18
`problem02-half.pddl`	8	8	6	**3**	4	1	**1**	2
`problem03-full.pddl`	**80**	-	25	**34**	-	1	**192**	-
`problem03-half.pddl`	**55**	56	17	37	**35**	≥ 2	673	**513**
`blocks:`								
`probBLOCKS-4-0.pddl`	54	**48**	30	**16**	51	≥ 2	**125**	712
`probBLOCKS-4-1.pddl`	48	**39**	30	**23**	28	≥ 2	218	**175**
`probBLOCKS-4-2.pddl`	58	**48**	30	**26**	40	≥ 2	**270**	529
`probBLOCKS-5-0.pddl`	**24**	57	42	**3**	85	≥ 2	**2**	1160
`probBLOCKS-5-1.pddl`	**60**	66	42	**26**	107	≥ 2	**267**	1419
`probBLOCKS-5-2.pddl`	76	**57**	42	**70**	85	≥ 2	**1199**	1297
`probBLOCKS-6-0.pddl`	99	**78**	56	**74**	94	≥ 2	**1275**	1454
`probBLOCKS-9-0.pddl`	**64**	-	110	**4**	-	≥ 2	**2**	-
`probBLOCKS-9-2.pddl`	**56**	-	110	**4**	-	≥ 2	**2**	-
`probBLOCKS-10-0.pddl`	**58**	-	132	**3**	-	≥ 2	**2**	-
`probBLOCKS-10-2.pddl`	**73**	-	132	**4**	-	≥ 2	**2**	-
`pegsol-08-strips:`								
`p01.pddl`	26	**22**	60	**10**	11	1	32	**26**
`p02.pddl`	-	**90**	100	-	**66**	≥ 2	-	**1578**

Table 7: Number of binary variables, continuous variables, and iterations used to detect the Basel measure as well as the seconds of runtime. The columns for binary variables, iterations, and runtime are split to compare the usage of initial constraints based on the disjunctive action landmarks (DALMs) in contrast to using no initial constraints (none).

task	binary variables		continuous variables	iterations		BM	runtime (s)	
	DALMs	none		DALMs	none		DALMs	none
movie:								
prob01.pddl	**10**	17	14	**2**	15	1	**3**	7
prob02.pddl	**10**	17	14	**2**	39	1	**4**	17
prob03.pddl	**10**	17	14	**2**	37	1	**4**	17
prob04.pddl	**10**	17	14	**2**	18	1	**4**	9
prob05.pddl	**10**	17	14	**2**	48	1	**5**	44
prob06.pddl	**10**	17	14	**2**	35	1	**5**	19
prob07.pddl	**10**	17	14	**2**	38	1	**6**	42
prob08.pddl	**10**	17	14	**2**	11	1	**6**	9
prob09.pddl	**10**	17	14	**2**	11	1	**6**	9
prob10.pddl	**10**	17	14	**2**	22	1	**7**	28
prob11.pddl	**10**	17	14	**2**	13	1	**7**	10
prob12.pddl	**10**	17	14	**2**	28	1	**7**	36
prob13.pddl	**10**	17	14	**2**	38	1	**8**	45
prob14.pddl	**10**	17	14	**2**	44	1	**8**	60
prob15.pddl	**10**	17	14	**2**	10	1	**9**	11
prob16.pddl	**10**	17	14	**2**	11	1	**9**	12
prob17.pddl	**10**	17	14	**2**	11	1	**9**	12
prob18.pddl	**10**	17	14	**2**	33	1	**9**	64
prob19.pddl	**10**	17	14	**2**	11	1	**10**	13
prob20.pddl	**10**	17	14	**2**	27	1	**10**	76
prob21.pddl	**10**	17	14	**2**	11	1	**10**	14
prob22.pddl	**10**	17	14	**2**	23	1	**11**	22
prob23.pddl	**10**	17	14	**2**	24	1	**11**	41
prob24.pddl	**10**	17	14	**2**	30	1	**12**	86
prob25.pddl	**10**	17	14	**2**	7	1	**12**	14
prob26.pddl	**10**	17	14	**2**	30	1	**13**	91
prob27.pddl	**10**	17	14	**2**	14	1	**13**	32
prob28.pddl	**10**	17	14	**2**	25	1	**13**	45
prob29.pddl	**10**	17	14	**2**	11	1	**15**	17
prob30.pddl	**10**	17	14	**2**	11	1	**14**	18

Table 8: Number of binary variables, continuous variables, and iterations used to detect the Basel measure as well as the seconds of runtime. The columns for binary variables, iterations, and runtime are split to compare the usage of initial constraints based on the disjunctive action landmarks (DALMs) in contrast to using no initial constraints (none).

8 Conclusion

8.1 Discussion

In this thesis, we investigated the correlation complexity by Seipp et al. (2016). We introduced the 4 states criterion as well as the 8 states criterion, these criteria are useful to detect a lower bound of the correlation complexity of a task.

By translating the state space into a vector space, we showed the relation of 1-dimensional potential heuristics and linear maps. With this approach, we connected classical planning with linear algebra. This way we are able to deduce the relation of the heuristic value of two states by the relations of the heuristic value of other states. We formulated the RB-split criterion with this connection. It is useful to detect a correlation complexity of at least 2. These criteria are more general than the criteria introduced by Seipp et al. (2016).

We also introduced the Basel measure, a new measure for the complexity of a classical planning task that is a lower bound to the correlation complexity. It measures how large the partial assignments of a potential heuristic for a task have to be such that `Simple Hill-climbing` always finds a goal. The comparison to the novelty width (Lipovetzky and Geffner, 2012) showed that the Basel measure is upper bounded by the novelty width +1.

Our experiment shows that the Basel measure is at least 2, for some IPC tasks. The experiment showed that the Basel measure is 1 for some small IPC tasks by calculating a potential heuristic that finds a plan with `Simple Hill-climbing`. However, the problem of finding the weights for such a potential heuristic efficiently remains open.

8.2 Future Work

In Chapter 4, we introduced the 4 states criterion to detect a correlation complexity of at least 2 as well as the 8 states criterion to detect a correlation complexity of at least 3. This seems like the first instances of a more general pattern. Further investigation might reveal a 2^n states criterion that detects a correlation complexity of at least n for any $n \in \mathbb{N}$.

We translated the state space into a vector space. This invites to translate the operator effects into transformation matrices. Investigation in this direction could reveal more connections from planning to linear algebra.

The experiment used more binary variables for the MIP if the constraints created with DALMs were used. The number of binary variables could be reduced by only using some of the DALMs. This could reduce the time to calculate the MIP.

Seipp et al. (2016) suggested studying the correlation complexity in a wider set of benchmark domains to improve the understanding of what makes planning hard a what makes easy planning easy. With the RB-split criterion, we have a new strong tool in the arsenal to do so.

9 Acknowledgments

I would like to thank Malte Helmert for giving me the opportunity to write this thesis as well as Thomas Keller and Augusto B. Corrêa for their help and supervision.

Calculations were performed at sciCORE (http://scicore.unibas.ch/) scientific computing center at University of Basel.

References

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

Bonet, B. and Geffner, H. (2001). Planning as heuristic search. *Artificial Intelligence*, 129(1-2):5–33.

Büchner, C., Keller, T., and Helmert, M. (2021a). Code, benchmarks and experiment data for the ICAPS 2021 paper "Exploiting Cyclic Dependencies in Landmark Heuristics". https://doi.org/10.5281/zenodo.4604735.

Büchner, C., Keller, T., and Helmert, M. (2021b). Exploiting Cyclic Dependencies in Landmark Heuristics. In *Proceedings of the Thirty-First International Conference on Automated Planning and Scheduling (ICAPS 2021)*. AAAI Press. To appear.

Bylander, T. (1994). The computational complexity of propositional STRIPS planning. *Artificial Intelligence*, 69(1-2):165–204.

Bylander, T. (1997). A linear programming heuristic for optimal planning. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI 1997)*, pages 694–699. AAAI Press.

Chen, H. and Giménez, O. (2007). Act Local, Think Global: Width Notions for Tractable Planning. In *Proceedings of the Seventeenth International Conference on Automated Planning and Scheduling (ICAPS 2007)*, pages 73–80. AAAI Press.

Dattorro, J. (2005). *Convex Optimization & Euclidean Distance Geometry*. Meboo Publishing.

Domshlak, C., Hoffmann, J., and Katz, M. (2015). Red–black planning: A new systematic approach to partial delete relaxation. *Artificial Intelligence*, 221:73–114.

Fikes, R. E. and Nilsson, N. J. (1971). STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2(3-4):189–208.

Fischer, G. (2010). *Lineare Algebra. Eine Einführung für Studienanfänger (17. Auflage)*. Vieweg+Teubner.

Francès, G., Corrêa, A. B., Geissmann, C., and Pommerening, F. (2019). Generalized Potential Heuristics for Classical Planning. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI 2019)*, pages 5554–5561. AAAI Press.

Haslum, P. (2009). $h^m(P) = h^1(P^m)$: Alternative Characterisations of the Generalisation From $h^{\max}$ To h^m. In *Proceedings of the Nineteenth International Conference on Automated Planning and Scheduling (ICAPS 2009)*, pages 354–357. AAAI Press.

Helmert, M. (2006). The Fast Downward planning system. *Journal of Artificial Intelligence Research*, 26:191–246.

Hoffmann, J. (2001). Local Search Topology in Planning Benchmarks: An Empirical Analysis. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI 2001)*, pages 453–458. Morgan Kaufmann.

Hoffmann, J. (2002). Local Search Topology in Planning Benchmarks: A Theoretical Analysis. In *Proceedings of the Sixth International Conference on Artificial Intelligence Planning and Scheduling (AIPS 2002)*, pages 92–100. AAAI Press.

Hoffmann, J. and Nebel, B. (2001). The FF Planning System: Fast Plan Generation through Heuristic Search. *Journal of Artificial Intelligence Research*, 14:253–302.

Hohenwarter, M., Borcherds, M., Ancsin, G., Bencze, B., Blossier, M., Delobelle, A., Denizet, C., Éliás, J., Fekete, A., Gál, L., Konečný, Z., Kovács, Z., Lizelfelner, S., Parisse, B., and Sturr, G. (2013). GeoGebra 4.4. http://www.geogebra.org.

Keyder, E., Richter, S., and Helmert, M. (2010). Sound and Complete Landmarks for And/Or Graphs. In *Proceedings of the Nineteenth European Conference on Artificial Intelligence (ECAI 2010)*, pages 335–340. IOS Press.

Lipovetzky, N. and Geffner, H. (2012). Width and Serialization of Classical Planning Problems. In *Proceedings of the Twentieth European Conference on Artificial Intelligence (ECAI 2012)*, pages 540–545. IOS Press.

Papadimitriou, C. H. (1977). The Euclidean traveling salesman problem is NP-complete. *Theoretical Computer Science*, 4(3):237–244.

Pommerening, F., Helmert, M., Röger, G., and Seipp, J. (2015). From Non-Negative to General Operator Cost Partitioning. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI 2015)*, pages 3335–3341. AAAI Press.

Richter, S., Helmert, M., and Westphal, M. (2008). Landmarks Revisited. In *Proceedings of the Twenty-Third International Conference on Automated Planning and Scheduling (ICAPS 2007)*, pages 975–982. AAAI Press.

Russell, S. and Norvig, P. (2010). *Artificial Intelligence: A Modern Approach (3rd Edition)*. Pearson Education.

Schrijver, A. (1998). *Theory of linear and integer programming.* John Wiley & Sons.

Seipp, J., Pommerening, F., Röger, G., and Helmert, M. (2016). Correlation Complexity of Classical Planning Domains. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI 2016)*, pages 3242–3250. AAAI Press.

Steinmetz, M. and Hoffmann, J. (2018). LP Heuristics over Conjunctions: Compilation, Convergence, Nogood Learning. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI 2018)*, pages 4837–4843. AAAI Press.

Wikimedia Commons (2014). Nederlands: Afvoerverdeling rijn. https://commons.wikimedia.org/wiki/File:Afvoerverdeling_rijn.svg.

Declaration on Scientific Integrity
Erklärung zur wissenschaftlichen Redlichkeit

includes Declaration on Plagiarism and Fraud
beinhaltet Erklärung zu Plagiat und Betrug

Author — Autor
Simon Dold

Matriculation number — Matrikelnummer
15-050-461

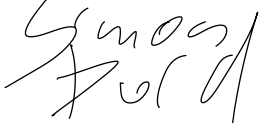
Title of work — Titel der Arbeit
Correlation Complexity and Different Notions of Width

Type of work — Typ der Arbeit
Master's Thesis

Declaration — Erklärung
I hereby declare that this submission is my own work and that I have fully acknowledged the assistance received in completing this work and that it contains no material that has not been formally acknowledged. I have mentioned all source materials used and have cited these in accordance with recognised scientific rules.

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.

Basel, May 28, 2021



Signature — Unterschrift