Relative-Order Abstractions for the Pancake Problem

Malte Helmert and **Gabriele Röger**[1]

Abstract. The pancake problem is a famous search problem where the objective is to sort a sequence of objects (*pancakes*) through a minimal number of prefix reversals (*flips*). The best approaches for the problem are based on heuristic search with abstraction (pattern database) heuristics. We present a new class of abstractions for the pancake problem called *relative-order abstractions*. Relative-order abstractions have three advantages over the *object-location abstractions* considered in previous work. First, they are *size-independent*, i. e., do not need to be tailored to a particular instance size of the pancake problem. Second, they are more *compact* in that they can represent a larger number of pancakes within abstractions of bounded size. Finally, they can exploit *symmetries* in the problem specification to allow multiple heuristic lookups, significantly improving search performance over a single lookup. Our experiments show that compared to object-location abstractions, our new techniques lead to an improvement of one order of magnitude in runtime and up to three orders of magnitude in the number of generated states.

1 INTRODUCTION

Many search problems in the AI literature can be thought of as *permutation problems*, where the objective is to transform a given arrangement of a set of objects into a desired goal arrangement through a sequence of actions that permute the current arrangement according to certain rules. We will later give a formal account of permutation problems. Informally, and applying a rather loose definition, the class of permutation problems includes many classical AI search benchmarks such as the 15- and 24-puzzle [15, 18, 17], Rubik's Cube [16] and the TopSpin puzzle [12]. It also includes important application problems such as the genome rearrangement problem [6, 19] and the greenhouse logistics problem [9].

The *pancake problem* [5], which is the main subject of this paper, is another example of a permutation problem. In the n-pancake problem, a stack of n pancakes of different size must be arranged in order of increasing size through a sequence of *flips*, where each flip reverses the order of some pancakes at the top of the stack (e. g., a 4-flip would reverse the order of pancakes in locations 1–4, counting from the top). Figure 1 shows a possible initial arrangement and the goal arrangement for the 6-pancake problem. An optimal (minimal-length) solution for this example is given by the sequence F_5, F_6, F_3, F_4, F_5, where F_k denotes a k-flip.

In addition to the interest it has attracted as a benchmark problem in the AI search community [14, 12, 13, 11, 21, 23, 1, 22, 20] the problem is also of practical relevance due to its strong similarity to the *genome rearrangement problem* [6, 19]. In the genome rearrangement problem, the objective is to assess the evolutionary proximity of two genomes by finding a minimal sequence of evolutionarily plausible gene transformations that transform the one genome into the other. One common way of modelling "evolutionary plausibility" is by only allowing *inversions* [6], which are precisely the kinds of transformations allowed in the *burnt pancake* [7] variant of the pancake problem. Indeed, the only major difference between the state spaces of the burnt pancake problem and the genome rearrangement problem is that in the latter the "pancakes" are not arranged into a stack but into a circular configuration, so that inversions can be applied everywhere rather than just at the top of the stack.

Figure 1. 6-pancake problem

Like many commonly studied permutation problems, the pancake problem is easy to solve suboptimally. It admits a trivial 2-approximation, and increasingly better approximation algorithms have been developed over the last 35 years [7, 10, 2]. However, the original problem formulation [5] asks for *optimal* solutions, which are much harder to find. The standard approach for optimally solving permutation problems, which we also follow in this work, is to use a heuristic search algorithm like A* or IDA* together with an admissible heuristic.

2 PERMUTATION PROBLEMS

We now provide formal definitions for the central notions underlying permutation problems. While we will ultimately apply these definitions to the pancake problem, it is useful to keep the definitions general to make them more widely applicable.

Definition 1 (permutation problem)
A permutation problem *of size n is defined by a 5-tuple $\Pi = \langle L, O, A, s_{\mathrm{I}}, s_{\mathrm{G}} \rangle$ consisting of*

- *a set of* locations *L with $|L| = n$,*
- *a set of* objects *O with $|O| = n$,*
- *a set of* actions *A where each action is a permutation of L, that is a bijective mapping $a : L \to L$,*
- *the initial state $s_{\mathrm{I}} : L \to O$, a bijective mapping from locations to objects, and*
- *the goal state $s_{\mathrm{G}} : L \to O$, another bijective mapping from locations to objects.*

Problem Π is called symmetric *iff A is closed under inversion, i. e., for all permutations $a \in A$, the inverse function a^{-1} is also contained in A.*

[1] Albert-Ludwigs-Universität Freiburg, Germany,
{helmert,roeger}@informatik.uni-freiburg.de

Our definition is rather narrow in requiring that all objects are distinguishable and that there is a unique goal state. Both restrictions can be lifted without affecting much of our discussion. The main reason to postulate these requirements, along with symmetry, is that they are important for the use of so-called *dual heuristic lookups* [22].

As the definition shows, we define states of permutation problems as mappings from locations to objects, indicating which object is contained at which location. For example, if we number the six locations in Fig. 1 from top to bottom as $l_1, \ldots, l_6$ and the six pancakes in increasing size as $o_1, \ldots, o_6$, then the figure depicts the state $\{l_1 \mapsto o_3, l_2 \mapsto o_2, l_3 \mapsto o_5, l_4 \mapsto o_1, l_5 \mapsto o_6, l_6 \mapsto o_4\}$.

We represent actions in such a way that $a(l_{\text{to}}) = l_{\text{from}}$ means that the object at location l_{from} in the state where a is applied moves to location l_{to} in the resulting state. Defining actions in this way, rather than the opposite way, is convenient because it means that the effect of applying an action a on a state s can be expressed as a simple function composition: $app_a(s) := s \circ a$.

To define formal semantics for permutation problems and to give a precise definition of abstractions in the next section, we need to define the state space of a permutation problem.

Definition 2 (state space of a permutation problem)
Let $\Pi = \langle L, O, A, s_{\text{I}}, s_{\text{G}} \rangle$ be a permutation problem. The state space *of Π is the labeled directed graph $\mathcal{S}(\Pi) = \langle S, A, T \rangle$ where*

- *the vertices are the set of* states $S = \{s \mid s : L \to O, s \text{ bijective}\}$,
- *the edge labels are the actions A of Π, and*
- *the (directed) labeled edges are the* transitions $T = \{\langle s, a, s \circ a \rangle \mid s \in S, a \in A\}$.

The algorithmic problem we want to solve is that of determining, given a permutation problem Π, whether the state space of Π contains a path from the initial state to the goal state, and if this is the case, to report the sequence of labels along a shortest such path. For the pancake problem in particular, which we define next, a solution always exists, so tree search algorithms like IDA* can be used without taking special care to ensure termination on unsolvable inputs.

Definition 3 (pancake problem)
A pancake problem of size n is a permutation problem $\langle L, O, A, s_{\text{I}}, s_{\text{G}} \rangle$ where

- *the locations are $L = \{l_1, \ldots, l_n\}$,*
- *the objects are the* pancakes $O = \{o_1, \ldots, o_n\}$,
- *the actions are the* flips $\{F_2, \ldots, F_n\}$, *where $F_i : L \to L$ is defined as*
$$F_i(l_j) = \begin{cases} l_{i+1-j} & \text{if } j \leq i \\ l_j & \text{otherwise} \end{cases}$$
- *s_{I} is an arbitrary state $s : L \to O$, and*
- *s_{G} is the* sorted state *given by $s_{\text{G}}(l_i) = o_i$ for all $1 \leq i \leq n$.*

3 ABSTRACTIONS

A common method for defining admissible heuristics for search problems is to map the original state space into a smaller, homomorphic *abstract state space* and to estimate the distance from state s to the goal state s_{G} by the length of a shortest path from the abstract state corresponding to s to the abstract state corresponding to s_{G}. If the abstract state space is sufficiently small, these estimates can be precomputed for all abstract states prior to search and saved in a lookup table. For the class of abstractions most commonly used in the search literature, these lookup tables are known as *pattern databases* (PDBs), and the corresponding heuristics are called pattern database heuristics [3]. Pattern database heuristics have two very convenient properties. First, the table lookup is very cheap, which allows evaluating the heuristic very quickly for a large number of states. Second, as long as the goal state remains fixed, a pattern database can be computed once and then reused for many instances of the same search problem. We now give a general definition of abstractions that subsumes the class of abstractions that underlie pattern databases. While our definitions are given in terms of permutation problems, we remark that they can equally well be applied to arbitrary search spaces.

Definition 4 (abstraction)
Let Π be a permutation problem with goal state s_{G} and state space $\mathcal{S}(\Pi) = \langle S, A, T \rangle$. An abstraction *of Π is a mapping $\alpha : S \to S'$, where S' is an arbitrary set called the set of* abstract states.

The abstract state space *induced by α is defined as $\alpha(\mathcal{S}(\Pi)) = \langle S', A, T' \rangle$, where $T' = \{\langle \alpha(s_1), a, \alpha(s_2) \rangle \mid \langle s_1, a, s_2 \rangle \in T\}$.*

The abstraction heuristic *induced by α is the function $h^\alpha : S \to \mathbb{N}_0 \cup \{\infty\}$ which maps each state $s \in S$ to the length of a shortest path from $\alpha(s)$ to $\alpha(s_{\text{G}})$ in the abstract state space.*

It is not hard to prove that the heuristic h^α is admissible and consistent for any abstraction α [8]. Of course, the choice of abstraction influences the informativeness of the heuristic. In practice, there is usually a tradeoff between the time and memory resources required to construct an abstraction and evaluate h^α on the one hand and the accuracy of the heuristic on the other hand.

It is often convenient to define abstractions in terms of an equivalence relation $\sim$ over states, where two states are mapped to the same abstract state iff they are equivalent under $\sim$.

Definition 5 (abstraction induced by equivalence relation)
Let S be the set of states of a permutation problem, and let $\sim \subseteq S \times S$ be an equivalence relation over S. The equivalence class for a given state is defined as $[s]_\sim := \{r \in S \mid s \sim r\}$. When the equivalence relation is clear from the context, we can write $[s]$ instead of $[s]_\sim$.

The abstraction induced by $\sim$ *is the function $h^\sim : s \mapsto [s]_\sim$.*

Object-Location Abstractions

The predominant approach for defining heuristics for permutation problems is to use so-called *domain abstractions* [20]. The most commonly used domain abstractions are *projections* based on a set of *distinguished objects*, which we will now introduce. We remark that more general domain abstractions exist [4], but we are not aware of any work that uses them in the context of permutation problems.

Definition 6 (object-location abstraction)
Let Π be a permutation problem with objects O, and let $O' \subseteq O$.

The object-location abstraction *for O' is defined by the following equivalence relation $\sim^{\text{OL}}_{O'}$:*

$$s \sim^{\text{OL}}_{O'} r \iff \text{for all } o \in O' \text{: } s^{-1}(o) = r^{-1}(o)$$

We call these abstractions *object-location abstractions* because they faithfully preserve the location of (certain) objects, unlike the abstractions we will consider later. In the case of the pancake problem, one intuitive interpretation of the object-location abstraction for a set of pancakes O' is that in the abstract state space, all pancakes that do not belong to the set O' are indistinguishable from each other. Otherwise, the abstract state space is the same as the real state space.

Figure 2 illustrates an object-location abstraction applied to the initial and goal states of Fig. 1. The distinguished objects are $O' =$

Figure 2. Abstract states of the 6-pancake problem in Figure 1 with object-location abstraction for $O' = \{o_2, o_3, o_6\}$

Figure 3. Two states that are equivalent under the relative-order abstraction for $O' = \{o_1, o_2, o_4, o_5, o_6\}$ but not under any nontrivial object-location abstraction

$\{o_2, o_3, o_6\}$. In this example, the abstraction provides a heuristic value of 3 for the initial state, corresponding to the action sequence F_5, F_6, F_4 which solves the abstract task.

Relative-Order Abstractions

All previous work on optimal solutions for the pancake problem that we are aware of is based on object-location abstractions [14, 12, 13, 11, 21, 23, 1, 22, 20]. In the following, we introduce *relative-order abstractions* and suggest that they are a more useful class of abstractions for this problem.

Definition 7 (relative-order abstraction)
Let Π be a permutation problem with objects O, and let $O' \subseteq O$. We assume that there is a total order $\prec$ on the locations of Π.

The relative-order abstraction *for O' is defined by the following equivalence relation $\sim^{\text{ORD}}_{O'}$:*

$$s \sim^{\text{ORD}}_{O'} r \iff \textit{for all } o, o' \in O': \\ ((s^{-1}(o) \prec s^{-1}(o')) \textit{ iff } (r^{-1}(o) \prec r^{-1}(o'))).$$

In words, we treat two states as equivalent for purposes of the abstraction if the *relative order* of the objects in O' is the same in both states. In the case of the pancake puzzle, the total order $\prec$ on the locations is the natural order where $l_i \prec l_j$ iff $i < j$.

Figure 3 illustrates the definition with an example of two states which are equivalent under the relative-order abstraction for $O' = \{o_1, o_2, o_4, o_5, o_6\}$. The two states are equivalent because the relative order of all pancakes in the set O' is identical in both states. Observe that these two states are not equivalent under any object-location abstraction except for the trivial $\sim^{\text{OL}}_{\emptyset}$.

Relative-order abstractions for the pancake problem have a very useful property: the abstract space induced by $\sim^{\text{ORD}}_{O'}$ is *isomorphic* to the *real state space* of the $|O'|$-pancake problem. The following theorem makes this more precise.

Theorem 1 (Isomorphism for relative-order abstractions)
Let $n, k \in \mathbb{N}_1$ with $n \geq k$. Let S_n be the set of states for an n-pancake problem, let $\mathcal{S}_k = \langle S_k, A_k, T_k \rangle$ be the state space of a k-pancake problem, and let $\mathcal{S}' = \langle S', A', T' \rangle$ be the abstract state space induced on an n-pancake problem by a relative-order abstraction with distinguished objects O', where $O' = \{o_{i_1}, \ldots, o_{i_k}\}$, $i_1 < i_2 < \cdots < i_k$.

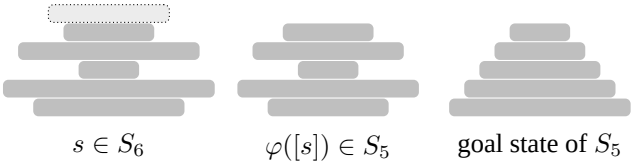

Figure 4. Abstract states of the 6-pancake problem in Figure 1 with relative-order abstraction for $O' = \{o_1, o_2, o_4, o_5, o_6\}$, illustrating Theorem 1. Left panel: original state $s \in S_6$ with "undistinguished" pancake o_3 highlighted; middle panel: state in S_5 corresponding to the abstract state for s; right panel: goal state in S_5.

Then the unlabeled directed graphs *for $\mathcal{S}_k$ and $\mathcal{S}'$, omitting self-loops, are isomorphic via isomorphism $\varphi : S' \to S_k$ defined as:*

Given $[s'] \in S'$, choose any representative $s \in [s']$, restrict s to those locations which map to objects in O', then renumber the locations to $l_1, \ldots, l_k$ and the objects to $o_1, \ldots, o_k$ in a way that preserves their relative order.

Proof sketch: To show that φ is well-defined (i. e., $\varphi([s'])$ does not depend on the choice of representative $s \in [s']$), we exploit that by the definition of relative-order abstractions, all states equivalent to s have the objects in O' in the same order.

To show that φ is an isomorphism, we must show that abstract transitions in $\mathcal{S}'$ correspond to flips in $\mathcal{S}_k$. This is true because abstract transitions in $\mathcal{S}'$ are derived from flips in $\mathcal{S}_n$, and they are still flips (although for a possibly smaller number of objects) when restricting to O'. ∎

Figure 4 illustrates the theorem on the running example from Fig. 1, using $O' = \{o_1, o_2, o_4, o_5, o_6\}$. The heuristic value in this case is 5: an optimal solution to transform the 5-pancake state depicted in the middle into the 5-pancake goal state depicted at the right is given by the action sequence F_2, F_4, F_5, F_2, F_4. This heuristic estimate is significantly better than with the object-location abstraction discussed earlier, yet the two heuristics require PDBs of the same size, as both abstract state spaces consist of 120 states. We will discuss the issue of PDB compactness in some more detail in the following subsection.

Comparison of the Two Classes of Abstractions

Before we turn to an experimental comparison of object-location abstractions and relative-order abstraction, we discuss some of their theoretical properties. In particular, we will make the following three observations:

1. Relative-order abstractions are *size-independent*. Object-location abstractions are size-dependent.
2. Relative-order abstractions permit *symmetric lookups*. Object-location abstractions do not.
3. Relative-order abstractions are *more compact* than object-location abstractions in the sense that for large problems, the PDB size required to achieve any given desired level of heuristic accuracy is smaller for them. (This is equivalent to saying that for large enough problems, they offer larger heuristic values than object-location abstractions for a given PDB size bound.)

Size-Independence and Symmetric Lookups. One drawback of object-location abstractions is that they are not reusable for permutation problems of different size. For example, a PDB constructed for

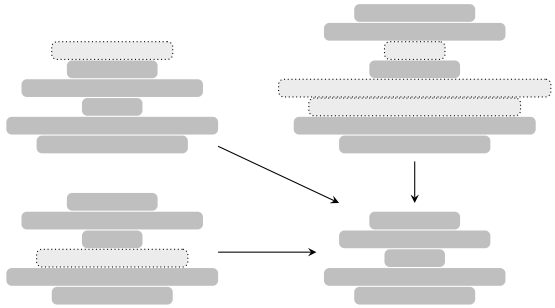

Figure 5. Size-independence and symmetric lookups for relative-order abstractions: three different abstractions map to the same state in S_5.

the abstraction of the 6-pancake problem shown in Fig. 2 cannot be applied to the 7-pancake problem. Even if we fix the problem size but consider different distinguished object sets O'_1 and O'_2, a PDB for $\sim^{\text{OL}}_{O'_1}$ cannot be used for $\sim^{\text{OL}}_{O'_2}$. Indeed, several papers [23, 22, 20] state that symmetric lookups are impossible in the pancake problem because "there are no symmetries in the Pancake problem that enable different abstractions to make use of the same PDB" [20].

While this is indeed true for the object-location abstractions considered in these papers, it is not true for abstractions of the pancake problem in general. In particular, the isomorphism property of Theorem 1 implies that we can use the same PDB for all relative-order abstractions where $|O'|$ has the same size, independent of the precise choice of O', and independent of the size of the original pancake problem that is abstracted. This is illustrated in Fig. 5 for three different relative-order abstractions: at the top left, the relative-order abstraction with $O' = \{o_1, o_2, o_4, o_5, o_6\}$ for the 6-pancake problem, at the bottom left, the relative-order abstraction with $O' = \{o_1, o_2, o_3, o_5, o_6\}$ for the 6-pancake problem, and at the top right, the relative-order abstraction with $O' = \{o_2, o_3, o_4, o_5, o_7\}$ for the 8-pancake problem.

An important consequence of this isomorphism between abstract state spaces is that we can perform so-called "symmetric lookups" for different sets of distinguished objects during search, a technique that has been shown to be very beneficial in a number of search domains [23]. Our experimental results in Section 4 will confirm that significant gains can be reaped in the pancake problem by making use of symmetric lookups.

Compactness. For our running example, we used an object-location abstraction with 3 distinguished objects and contrasted it with a relative-order abstraction with 5 distinguished objects. This choice was not arbitrary: both abstractions give rise to abstract state spaces of the same size, namely 120 abstract states, so their PDBs have identical storage requirements. More generally, the size of the abstract state space induced by an object-location abstraction with k distinguished objects on a size n problem is

$$N^{\text{OL}}(n,k) = n \cdot (n-1) \cdot \ldots \cdot (n-k+1),$$

while the size of a relative-order abstraction with k distinguished objects on a problem of any size is

$$N^{\text{ORD}}(k) = k!.$$

We now bound the average heuristic values obtained by these abstractions, which we denote by $\bar{h}^{\text{OL}}(n,k)$ and $\bar{h}^{\text{ORD}}(k)$, and relate them to the required PDB size. For both abstractions, the average heuristic value for a state picked uniformly from the real state space is simply the average abstract goal distance of all abstract states, as each abstract state is the image of an equal number of real states.

For object-location abstractions, it is easy to see that $\bar{h}^{\text{OL}}(n,k) \leq 4k$, since this an upper bound on the heuristic value of *any* state: an abstract pancake problem with k distinguished objects can always be suboptimally solved with at most $4k$ actions. In the special case where the distinguished pancakes are the k largest pancakes, as suggested by Zahavi et al. [22], we can prove the tighter bound $\bar{h}^{\text{OL}}(n,k) \leq 2k$, and in this case it is also not hard to show that for any fixed value of k, $\bar{h}^{\text{OL}}(n,k)$ converges to $2k$ as n approaches infinity. (We believe that a tighter bound than $4k$ also holds in the general case, but have not yet been able to show this.)

For relative-order abstractions, the isomorphism result implies that $\bar{h}^{\text{ORD}}(k)$ is the average goal distance of all states of the k-pancake problem. To bound this number, the argument that Gates and Papadimitriou [7] use to prove that the *maximal* goal distance for the k-pancake problem is at least k can be generalized to prove that $\bar{h}^{\text{ORD}}(k) \geq k-2$. (To also provide an upper bound, Chitturi et al. have recently shown that $\bar{h}^{\text{ORD}}(k) \leq \frac{18}{11}k + O(1)$ [2].)

We can combine these results for abstract state space size and average heuristic value to determine the PDB size required, in the n-pancake problem, to achieve a given average heuristic value h. We denote these quantities with $\mathit{PDB}^{\text{OL}}(n,h)$ and $\mathit{PDB}^{\text{ORD}}(n,h)$, respectively, and obtain:

$$\mathit{PDB}^{\text{OL}}(n,h) \geq n \cdot (n-1) \cdot \ldots \cdot (n - \lceil h/4 \rceil + 1)$$
$$\mathit{PDB}^{\text{ORD}}(n,h) \leq (h+2)!$$

These inequalities suggest that relative-order pattern databases offer significantly better heuristic values for the n-pancake problem when memory is limited and n grows. For example, for any fixed desired value of h, we have $\mathit{PDB}^{\text{OL}}(n,h) = \Omega(n^{h/4})$ while $\mathit{PDB}^{\text{ORD}}(n,h)$ is a constant. However, this constant is large, and hence this in-the-limit comparison does not necessarily imply a practical difference for problem sizes that can be solved in reasonable time by current search algorithms. Indeed, for the problem sizes and abstractions typically considered in the literature ($n \approx 17$, PDB size between 10^8 and 10^{10}), neither abstraction technique provides significantly better heuristic values than the other.

As an example, Fig. 6 shows the average heuristic values of an object-location heuristic with $k = 5$ as a function of the problem size n. The distinguished pancakes are the largest five pancakes, following Zahavi et al. [22]. Compared to this (dashed and dotted curves) are the average heuristic values of relative-order abstractions of the same size. Since it is not always possible to generate a relative-order PDB of exactly the same size, we report both the average values for the next larger PDB (upper curve) and the next smaller PDB (lower curve). For low values of n, the two abstractions are comparable, while relative-order abstractions begin to dominate for values of n around 30. In the limit, the difference between the two classes of abstraction becomes unbounded as the average heuristic value for the object-location abstractions approaches 10 while the average for relative-order abstractions grows without bound.

4 EXPERIMENTAL RESULTS

To experimentally evaluate relative-order abstractions, we compare them to object-location abstractions using the approach that performed best in the experiments reported by Zahavi et al. [22]: Dual IDA* (DIDA*) search with BPMX, using the largest pancakes as the set of distinguished objects.

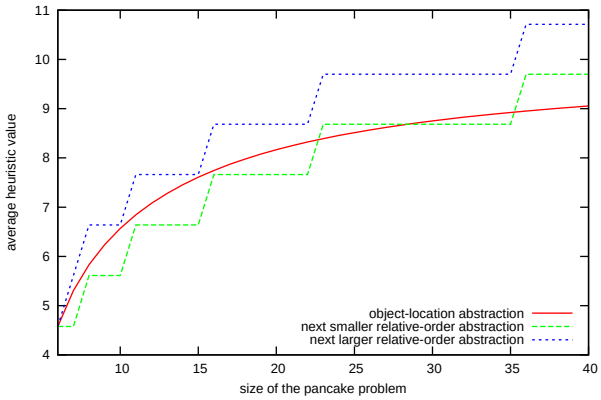

Figure 6. Average heuristic values for abstractions of similar size

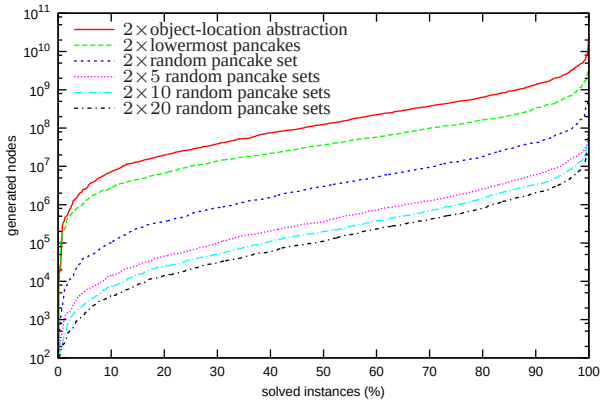

Figure 7. Solved instances plotted against generated notes. For example, a point at $(20\%, 10^5)$ indicates that 20% of the instances required at most 10^5 generated nodes. All results based on DIDA*.

Table 1. 17-pancake results, averaged over 1000 random instances

#	Search	Heuristic	Nodes	Time (s)
1	DIDA*	2×object-location abstraction (7 largest pancakes)	539,179,741	286.79
2	DIDA*	2×lowermost 12 pancakes	131,794,939	150.21
3	DIDA*	2×1 random pancake set	17,183,911	54.26
4	DIDA*	2×5 random pancake sets	2,373,225	36.56
5	DIDA*	2×10 random pancake sets	1,400,455	42.45
6	DIDA*	2×20 random pancake sets	827,976	49.54
7	DIDA*	2×50 random pancake sets	411,830	60.39
8	IDA*	lowermost 12 pancakes	584,871,396	293.25
9	IDA*	fixed + dual lookup	94,090,827	108.87
10	IDA*	1 random pancake set	52,927,944	82.20
11	IDA*	5 random pancake sets	5,044,945	38.35
12	IDA*	10 random pancake sets	2,449,822	36.82
13	IDA*	20 random pancake sets	1,375,713	40.91
14	IDA*	50 random pancake sets	673,340	49.30
15	DIDA*	2×4 random lookups + 2×object-location abstraction	2,362,899	31.62

For object-location abstractions, we reused the PDB generator of Zahavi et al., but reimplemented the search algorithm to make it comparable to our implementation for relative-order abstractions. Our implementation is about a factor of 2 faster than the one by Zahavi et al. on our evaluation machine, which is equipped with a 2.3 GHz AMD Opteron CPU. In all experiments we used the largest PDB that fit into 512 MB of space with one byte of memory per entry.

Unless stated otherwise, we report results for 1000 randomly generated instances of the 17-pancake problem. The average solution length for these instances is 15.770. For object-location abstraction, the largest PDB that fits into the 512 MB bound uses 7 distinguished pancakes and gives an average heuristic value of 10.20. For relative-order abstraction, the largest PDB that fits into the memory bound uses 12 pancakes and gives an average heuristic value of 10.71.

Overall Results. Table 1 shows the average runtime and number of generated nodes across the 1000-instance benchmark set for a number of different search algorithms and heuristics.

Rows 1–7 and 15 give results for Dual IDA*, rows 8–14 for IDA*, all using BPMX. We also experimented with algorithms not using BPMX, which universally performed worse. Entries of the form $2\times k$ in the rows for DIDA* indicate that the heuristic value is computed by maximizing over k regular lookups and k dual state lookups.

Row 1 shows the result for the best-performing configuration of Zahavi et al. described above, using object-location abstraction. This row serves as our reference result. Rows 2–14 all describe configurations using relative-order abstraction, while row 15 describes a configuration using both kinds of abstraction.

Rows 2 and 3 are directly comparable to the reference result in row 1 in that they also perform a single heuristic lookup per state, plus the dual lookup. In row 2, we report results where the set of distinguished pancakes in each state consists of the 12 pancakes in the lowermost positions. (In preliminary experiments, this gave better results than, e. g., picking the pancakes from the topmost positions or from positions spread evenly throughout the problem.) In row 3, the set of distinguished pancakes is chosen uniformly randomly for each search state. Both versions clearly outperform the object-location abstraction, with the randomized heuristic (#3) improving node generations by a factor of 31.4 and runtime by a factor of 5.3.

These results can be improved significantly by using symmetric lookups, as shown in rows 4–7. We see that more lookups dramatically reduce the number of generated nodes, down to an average of 411,830 in row 7 when using 50 regular and 50 dual lookups. This is an improvement by a factor of 1309 compared to the reference result with object-location abstractions. However, after a certain point the overhead for additional lookups dominates the improvements in terms of search nodes, so the best results in terms of runtime are obtained with a smaller number (2×5) of symmetric lookups (row 4).

It is interesting to note that we obtain comparable results when using regular (non-dual) IDA* (rows 8–14) with BPMX as long as we perform a sufficient number of symmetric lookups. Our best overall runtime results are obtained with DIDA* and 2×5 lookups and with IDA* with 10 lookups. The average runtimes for these configurations (36.56 seconds vs. 36.82 seconds) is close enough to allow us to draw the conclusion that duality offers no compelling advantage in our best configurations using relative-order abstractions.

As a final data point, row 15 shows that we can further improve our results by maximizing over both kinds of abstraction. The average runtime of 31.62 seconds reported for this configuration improves over the reference result by a factor of 9.1.

Results for Individual Instances. An important observation that is obscured by the summarizing results in Table 1 is that the expan-

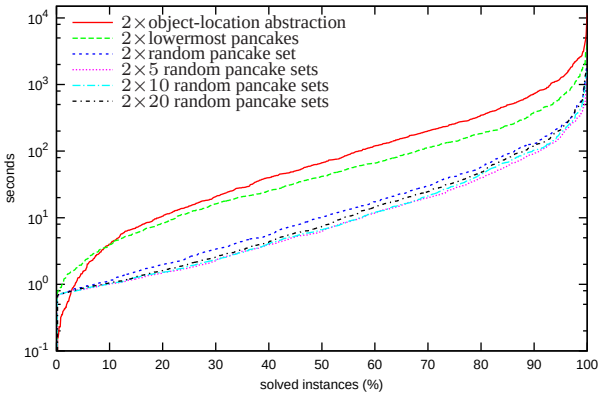

Figure 8. Solved instances plotted against runtime, using the same notation as in Fig. 7. All results based on DIDA*.

Table 2. Results for 18- and 19-pancake over 10 instances

n	Heuristic	Nodes	Time (s)
18	2×object-location abstraction	3,104,195,668	1,638.19
18	2×5 random lookups	13,439,483	209.81
18	2×10 random lookups	7,638,885	236.10
19	2×5 random lookups	1,236,838,871	20,409.90
19	2×10 random lookups	689,598,292	22,576.39

sion and runtime results vary tremendously across different instances of the 17-pancake problem. For instance, while average solution time for the reference configuration was less than 5 minutes, there were six configurations that required more than one hour to solve, and one configuration that required more than 3 hours. Figures 7 and 8 provide detailed results that show how many of the 1000 instances were solved within any given bound on the number of expansions or runtime. The general trend of the figures is similar to Table 1, but we clearly see that there are some exceptionally hard instances.

Larger Problems. We conclude this section with a brief discussion of larger instances of the pancake problem, using 18 and 19 pancakes. As these instances require much more time to solve than the 17-pancake problem, we only evaluated on 10 instances of each size and only compare the reference approach by Zahavi et al. and two of our best approaches, using DIDA* with 2×5 and 2×10 lookups per state. The results are shown in Table 2. For the 18-pancake problem, we again see a considerable improvement of relative-order abstraction over object-location abstraction. For 19 pancakes, we do not report results for object-location abstraction because one of the instances was not solved within our timeout of 50 hours.

5 CONCLUSION

Relative-order abstractions are a new class of abstractions for the pancake problem that offer several advantages over the previously considered *object-location abstractions*: they are *size-independent*, they are more *compact*, and they permit *symmetric lookups*. Our experimental evaluation shows that by exploiting this last property we can improve on the performance of the best object-location abstractions of Zahavi et al. [22] by three orders of magnitude in terms of node generations and one order of magnitude in terms of speed.

In future work, we intend to apply the idea of relative-order abstractions to other search domains. Of particular interest in this regard is the *genome rearrangement problem* [6, 19], whose state space closely resembles the one of the burnt pancake problem, so that it appears likely that similar techniques will be efficient in this domain.

ACKNOWLEDGEMENTS

This work was supported by the German Research Council (DFG) by DFG grant NE 623/10-2 and as part of the Transregional Collaborative Research Center "Automatic Verification and Analysis of Complex Systems" (AVACS).

REFERENCES

[1] Kenneth Anderson, Robert Holte, and Jonathan Schaeffer, 'Partial pattern databases', in *Proc. SARA 2007*, pp. 20–34, (2007).
[2] Bhadrachalam Chitturi, William Fahle, Zhaobing Meng, Linda Morales, Charles O. Shields Jr., Ivan Hal Sudborough, and Walter Voit, 'An $(18/11)n$ upper bound for sorting by prefix reversals', *Theoretical Computer Science*, **410**(36), 3372–3390, (2009).
[3] Joseph C. Culberson and Jonathan Schaeffer, 'Pattern databases', *Computational Intelligence*, **14**(3), 318–334, (1998).
[4] Carmel Domshlak, Jörg Hoffmann, and Ashish Sabharwal, 'Friends or foes? On planning as satisfiability and abstract CNF encodings', *JAIR*, **36**, 415–469, (2009).
[5] Harry Dweighter, 'Elementary problem e2569', *The American Mathematical Monthly*, **82**(10), 1010, (1975).
[6] Esra Erdem and Elisabeth R. M. Tillier, 'Genome rearrangement and planning', in *Proc. AAAI 2005*, pp. 1139–1144, (2005).
[7] William H. Gates and Christos H. Papadimitriou, 'Bounds for sorting by prefix reversal', *Discrete Math*, **27**, 47–57, (1979).
[8] Malte Helmert, Patrik Haslum, and Jörg Hoffmann, 'Flexible abstraction heuristics for optimal sequential planning', in *Proc. ICAPS 2007*, pp. 176–183, (2007).
[9] Malte Helmert and Hauke Lasinger, 'The Scanalyzer domain: Greenhouse logistics as a planning problem', in *Proc. ICAPS 2010*, pp. 234–237, (2010).
[10] Mohammad H. Heydari and I. Hal Sudborough, 'On the diameter of the pancake network', *Journal of Algorithms*, **25**(1), 67–94, (1997).
[11] Robert Holte, Ariel Felner, Jack Newton, Ram Meshulam, and David Furcy, 'Maximizing over multiple pattern databases speeds up heuristic search', *AIJ*, **170**(16–17), 1123–1136, (2006).
[12] Robert Holte, Jack Newton, Ariel Felner, Ram Meshulam, and David Furcy, 'Multiple pattern databases', in *Proc. ICAPS 2004*, pp. 122–131, (2004).
[13] Robert C. Holte, Jeffery Grajkowski, and Brian Tanner, 'Hierarchical heuristic search revisited', in *Proc. SARA 2005*, pp. 121–133, (2005).
[14] Robert C. Holte, Maria B. Perez, Robert M. Zimmer, and Alan J. MacDonald, 'Hierarchical A*: Searching abstraction hierarchies efficiently', in *Proc. AAAI 1996*, pp. 530–535, (1996).
[15] Richard E. Korf, 'Depth-first iterative-deepending: An optimal admissible tree search', *AIJ*, **27**(1), 97–109, (1985).
[16] Richard E. Korf, 'Finding optimal solutions to Rubik's Cube using pattern databases', in *Proc. AAAI 1997*, pp. 700–705, (1997).
[17] Richard E. Korf and Peter Schultze, 'Large-scale parallel breadth-first search', in *Proc. AAAI 2005*, pp. 1380–1385, (2005).
[18] Richard E. Korf and Larry A. Taylor, 'Finding optimal solutions to the twenty-four puzzle', in *Proc. AAAI 1996*, pp. 1202–1207, (1996).
[19] Tansel Uras and Esra Erdem, 'Genome rearrangement and planning: Revisited', in *Proc. ICAPS 2010*, pp. 250–253, (2010).
[20] Fan Yang, Joseph Culberson, Robert Holte, Uzi Zahavi, and Ariel Felner, 'A general theory of additive state space abstractions', *JAIR*, **32**, 631–662, (2008).
[21] Uzi Zahavi, Ariel Felner, Robert Holte, and Jonathan Schaeffer, 'Dual search in permutation state spaces', in *Proc. AAAI 2006*, pp. 1076–1081, (2006).
[22] Uzi Zahavi, Ariel Felner, Robert C. Holte, and Jonathan Schaeffer, 'Duality in permutation state spaces and the dual search algorithm', *AIJ*, **172**(4–5), 514–540, (2008).
[23] Uzi Zahavi, Ariel Felner, Jonathan Schaeffer, and Nathan Sturtevant, 'Inconsistent heuristics', in *Proc. AAAI 2007*, pp. 1211–1216, (2007).