# Heuristic Reasoning for 2D Containment Problems

**Nuno Marques**, **João Bernardo** and **Pedro Capela**
Instituto Nacional de Engenharia de Sistemas e Computadores,
Avenida Duque d'Ávila 23, 1000 Lisboa, Portugal
{nrm,jcb,pmc}@trabi.inesc.pt

**ABSTRACT**

Industry nowadays seeks for automation in processes that, until a few years ago, were mainly dependent on human work and intelligence. Containment problems belong to that family of automation processes. We can define a containment problem as a way of placing a set of shapes into another shape without overlapping. In this paper we apply A. I. concepts such as A* informed search algorithms to solve 2D translational containment problems. A containment problem can be described through a search problem and solved by using a heuristic that makes A* search complete and optimal. Morphological operators define the geometric description of the state space. To implement morphological operations, discrete mathematics is applied, allowing us to change the complexity analysis from the polygonal vertex/edge approach to the discrete unit/area approach.

**Keywords**: Containment Problems, Geometric Algorithms, Heuristic Search, Morphological Operators, Admissible Heuristics.

## INTRODUCTION

Generally, we can describe a solution of a containment/nesting problem as a way of placing a set of objects into a container. We are interested in finding good solutions, in the sense that they have some kind of quality. We will address only one subset of the containment problem universe – the two dimension translational containment problems, where we will try to find the maximal collection of shapes that fit into a given container.

This paper adopts the standard framework of heuristic search, in which the object is to find a sequence of operators leading from a given initial state to any goal state. Containment problems become instances of problem definitions for heuristic search. The next section will describe previous research. The following section defines morphological operators and it's associated properties. Then we will give formal descriptions of the 2D translational containment problem and of a generic search problem. We will also define the containment problem as an instance of a search problem. Next we will describe heuristic search. The completeness and optimality of an A* method is discussed, introducing the admissibility condition on the heuristic. The morphological opening heuristic is presented, and we will demonstrate that it is an admissible heuristic, thus allowing us to have an A* solution that is optimal and complete. Discrete and continuous implementations of the Minkowski operators are compared. Finally, some examples are shown, future work is discussed and we draw the conclusions.

## PREVIOUS WORK

Computational / Combinatorial geometry algorithms found in literature are mainly dependent on the complexity of the polygons. The single polygon containment problem was introduced by Chazelle [Chaze83a], which aim to place a $m$-gon $P$ into a $n$-gon container $C$. The running time bound for convex polygons is $O(mn^2)$ and for non-convex polygons is $O(m^3n^2 \log n)$ which was achieved by Avnaim [Avnai89a]. In multiple polygon case, Milenkovic proved the running time bound of $O((m^2+mn)^{2k} \log n)$ for placing $k$ non-convex $m$-gons into a non-convex $n$-gon [Milen96a]. Heuristic search methods are independent from the geometric description of each set so present a significant advantage when we need to have tight running times bounds. Geometric

algorithms tend to be more independent and capable of understanding specific problem irregularities.

## MORPHOLOGICAL OPERATORS

This chapter will introduce the basic operators and it's properties such as the Minkowski sum and difference operators. Let $A, B \subseteq \Re^2$, $t \in \Re^2$. The *morphological mirror* of $A$, $(-A)$, is

$$(-A) = \{-a : a \in A\}. \quad (1)$$

The *translation* of *A* by *t*, *A+t*, is

$$A + t = \{a + t : a \in A\}. \quad (2)$$

The *Minkowski sum* of *A* with respect to *B*, $A \oplus B$, is

$$A \oplus B = \bigcup_{b \in B} A + b \quad (3)$$

The Minkowski sum has important properties that will help us describing the containment process. Here we list some of these properties:

$$A \oplus B = B \oplus A \quad (4)$$

$$A \oplus (B \oplus C) = (A \oplus B) \oplus C \quad (5)$$

$$A \oplus (B \cup C) = (A \oplus B) \cup (A \oplus C) \quad (6)$$

$$(A \oplus B \subseteq A \oplus C) \Leftrightarrow (B \subseteq C). \quad (7)$$

The relation with translation is defined through the use of the following properties:

$$A \oplus \{t\} = A + t \quad (8)$$

$$A \oplus (B + t) = A \oplus B + t \quad (9)$$

The most useful property determines how translations of shapes are mapped from and into Minkowski sum sets. We have the fundamental property,

$$((B + t) \cap A \neq \varnothing) \Leftrightarrow (t \in A \oplus (-B)) \quad (10)$$

The *Minkowski difference* of *A* with respect to *B*,

$$A \Theta B = \overline{\overline{A} \oplus B} \quad (11)$$

has some important properties:

$$A \Theta B = \overline{B} \Theta \overline{A} \quad (12)$$

$$A \Theta (B \Theta C) = (A \Theta B) \Theta C \quad (13)$$

$$(A \cap B) \Theta C = (A \Theta C) \cap (B \Theta C) \quad (14)$$

$$A \Theta (B \cup C) = (A \Theta B) \cap (A \Theta C) \quad (15)$$

$$(A \Theta B \subseteq A \Theta C) \Leftrightarrow (B \supseteq C). \quad (16)$$

The following property says that if a shape *B*, positioned at point *t*, is contained into *A* then that point belongs to the Minkowski difference of *A* with the mirror of *B*. By using this property we can get the set of all points *t* in which *B* fits into *A* just by computing the difference.

$$((B + t) \subseteq A) \Leftrightarrow (t \in A \Theta (-B) \neq \varnothing) \quad (17)$$

The morphological *opening operator* is a composed operator, based on the Minkowski sum and difference operators,

$$A o B = (A \Theta (-B)) \oplus B. \quad (18)$$

If *B* fits into *A*, the opening of *A* with respect to *B* consists of all the points of *A* which are reachable by some part of *B* via translation [Serra82a].

## CONTAINMENT CONFIGURATION

The goal of a translational containment algorithm is to find positions $t_1, ..., t_n \in \Re^2$ that translate the objects $P_1, ..., P_n$ into the container $P_0$ [1]. Let $\Omega_n \subseteq 2^{\Re^2}$ be the *n*-superset containing all $P_i$'s[2]. A *translational configuration* $\zeta(\Omega_n)$ is a *(n+1)* tuple

$$\zeta(\Omega_n) = \langle (t_0, P_0), ..., (t_n, P_n) \rangle \quad (19)$$

where $P_0, ..., P_n \in \Omega_n$ and $t_0 = (0,0)$, $t_1, ..., t_n \in \Re^2$.

### Validity of Configurations

A translational configuration is *valid* if and only if the following condition is satisfied:

$$t_j - t_i \in \overline{P_i \oplus (-P_j)} \quad (20)$$

where $t_i, t_j \in \Re^2$ are translations of $P_i$ and $P_j$ [Danls95a]. If a containment configuration is valid then there exist a nesting of $P_1, ..., P_n$ into the container $P_0$.

---

1 In fact, $P_0$ represents the complement of the container.

2 Example: $\Omega_3 = \{P_0, P_1, P_2, P_3\}$.

## DEFINING SEARCH PROBLEMS

To formally describe a general search problem, we need to have the following information: The **initial state**, defining the starting point of the search; A collection of **operators**, that determines the transformation of states; The **goal test**, that checks if the state reached the goal and the **path cost function**, which assigns a cost to a path [Russl94a]. Therefore we need to generate this information to specify the containment problem as a search problem. Heuristic search tries to find a sequence of operators leading from a given initial state to a goal state. The guide to the search is a function *h(n)*, called heuristic function, which estimates the lowest cost of any path from the state *n* to a goal state. We will make the correspondence between general search problem definition and containment problem definition. We have that the initial state is just a state where the container is set and a collection of shapes is ready to be placed. The operators will just select a valid position within the container. The goal test will only check if all shapes are placed.

### Defining a State

A state is just a valid nesting and a collection of shapes to place. Therefore a state is a translational valid configuration (Eq. 19) and information about the shapes to place. A state contains information about the search process, which guides us in finding the path. We have

$$S(\Omega_d, \Phi_k) = \langle \zeta(\Omega_d), \Phi_k \rangle \tag{21}$$

where $\zeta(\Omega_d)$ is a translational valid configuration of size *d* and $\Phi_k$ is a collection of *k* objects to be placed. The *initial state*

$$S(\Omega_0, \Phi_n) = \langle \zeta(\Omega_0), \Phi_n \rangle \tag{22}$$

where $\Omega_0 = \{P_0\}$ and $\Phi_n = \{P_1, ..., P_n\}$ just sets the initial configuration, where only the container is part of it.

### The Allocation Operator

The *allocation operator* just sets the transformation of one state into another. If we have a shape to place and a containment state, the allocation operator returns a new state. Formally,

$$Allocate(S(\Omega_n, \Phi_k), Q, t) = S(\Omega_{n+1}, \Phi_{k-1}) \tag{23}$$

where $Q \in \Phi_k$, $\Omega_{n+1} = \Omega_n \cup \{Q\}, \Phi_{k-1} = \Phi_k \setminus \{Q\}$.

The new state is $S(\Omega_{n+1}, \Phi_{k-1}) = \langle \zeta(\Omega_{n+1}), \Phi_{k-1} \rangle$, where

$$\begin{cases} \zeta_i(\Omega_{n+1}) = \zeta_i(\Omega_n), \forall_{i=0..n} \\ \zeta_{n+1}(\Omega_{n+1}) = (t, Q) \end{cases} \tag{24}$$

for $t \in Strg(\zeta(\Omega_n), Q)$, where *Strg* is any nesting strategy[3].

### The Path Cost Function

Before choosing an adequate path cost function, we must decide what kind of measuring unit is used. Since one of the main goals is to maximise the area of the placed shapes inside the container, it makes sense to use some kind of surface measuring unit[4]. We can now define the path cost function as the amount of container area that the new shape will retain when placed. The path cost function for the containment problem

$$\text{cost}(S(\Omega_d, \Phi_k), S(\Omega_{d+1}, \Phi_{k-1})) = Area(Q) \tag{25}$$

where $Q \in \Phi_k$, $\Omega_{d+1} = \Omega_d \cup \{Q\}$, $\Phi_{k-1} = \Phi_k \setminus \{Q\}$.

## ADMISSIBILITY CONDITION

The purpose of A* search is to minimise the total path cost. A* sums both the heuristic function and the path cost function in order to estimate that path. We have that the estimated cost of the cheapest solution through *n* is the sum of the path cost from the start node to node *n* (we will call it *g(n)*), and of the **heuristic** function *h(n)*, which is the estimated cost of the cheapest path from *n* to the goal. A* is complete and optimal, given a certain restriction on the *h* function – The **admissibility** property. A search strategy is complete if is guaranteed to find a solution if one exists. It holds the optimality property if the highest-quality solution is found when several different solutions can be reached. If *h* never overestimates the cost to reach the goal, *h* is admissible. In other words, *h* must be optimistic.

### Morphological Heuristic

We will now build *g* and *h* functions for the containment search problem, such that they are coherent and *h* is admissible. The *g* function is just the sum of the path cost function along the path from the root. $g(S(\Omega_d, \Phi_k))$ is just

3 A nesting strategy is an operator that given a container and a shape, returns a position to place the shape inside the container.

4 Such as square millimetres.

$$g(S(\Omega_d, \Phi_k)) = \sum_{i=1}^{d} Area(P_i) \quad (26)$$

for $P_i \in \Omega_d, \forall_{i=1..d}$. The heuristic should estimate the area occupied when we reach the goal. In order to have an admissible heuristic we should be optimistic. If we just check if it is possible to place the remaining shapes, weight that possibility and sum each of those areas, we can build an admissible heuristic. If we apply the Minkowski difference operator to all of the remaining non-placed shapes to check if the nesting is still possible for each one, and avoid states where reaching the goal is impossible, we get the estimate of the shortest path to the goal[5]. Therefore

$$h(S(\Omega_d, \Phi_k) = \sum_{i=d}^{m} (\delta(\zeta(\Omega_d), P_i)) \quad (27)$$

where $\zeta(\Omega_d)$ belongs to $S(\Omega_d, \Phi_k)$, and $P_i \in \Phi_k$. The delta function measures the possibility of placing each of the remaining shapes into the container. If such nesting is possible returns a weighted shape's area. The cost raises as the amount of the inaccessible region increases. If not a maximum cost is returned. $\delta(\zeta(\Omega_d), P_i)$ is defined by

$$\begin{cases} \dfrac{(Area(C) - Area(C o P_i))Area(P_i)}{Area(C)}, & \text{if } C\Theta(-P_i) \neq \varnothing \\ Area(\overline{P_0}), & \text{if } C\Theta(-P_i) = \varnothing \end{cases}$$

where

- $C = \overline{(P_0 + t_0) \cup (P_1 + t_1) \cup ... \cup (P_d + t_d)}$
- $(t_0, P_0), ..., (t_d, P_d) \in \zeta(\Omega_d)$. (28)

**Proving Admissibility**

We know that when we reach the goal, the cost of the nesting is the sum of all the placed shapes.

$$\text{cost}(S(\Omega_0, \Phi_n), S(\Omega_n, \Phi_0)) = \sum_{i=1}^{n} Area(P_i) \quad (29)$$

So if we are in a path towards the goal, the Minkowski difference operator define sets that are not empty for all of the shapes to place,

$$\underset{Q \in \Phi_k}{\forall} C\Theta Q \neq \varnothing \quad (30)$$

and therefore the heuristic will return a weighted path-cost to reach the goal.

$$\delta(\zeta(\Omega_d), Q) = \frac{(Area(C) - Area(C o Q))Area(Q)}{Area(C)} \quad (31)$$

[5] We can avoid bad paths and keep following the right ones.

Since the weight is always positive and less than one, the estimated path cost never overestimates the real cost.

$$Area(Q) \geq \frac{(Area(C) - Area(C o Q))Area(Q)}{Area(C)} \quad (32)$$

If placing any shape is impossible then we will punish the path raising the cost each time reaching the goal is impossible.

$$\delta(\zeta(\Omega_d), Q) = Area(\overline{P_0}) \quad (33)$$

Recall that the completeness property states that if there is a solution A* will find it and the optimality property states that find the highest-quality solution when there are several different solutions. So if the initial collection of shapes cannot be placed into the container then no solution is found, simply because there is no solution.

## DISCRETE AND CONTINUOUS GEOMETRIC OPERATORS

Up to now we have discussed only one component of the containment problems. We should bear in mind that we have two separate components, both very important if we seek for a practical algorithm. We have the

- **Geometric Component**, where we aim to find one efficient geometrical representation of the shapes and of the container, in the sense that it should save memory resources and it can compute the basic geometric operations in a minimum amount of time;
- **Combinatorial Component**, where we need to build the maximum of configurations, with good (practical) time efficiency and good space efficiency (it doesn't use more than the system memory resources).

The second component was already discussed in this article. Let's focus on the first component by selecting a suitable geometric representation. Up to now the representation space we have used is $\Re^2$. While it is a widely applied representation scheme, it presents some computational disadvantages, such as

- Floating point operations;
- The complexity of the operators is strongly dependent on the geometrical primitives.

If we can overcome the first disadvantange by using better algorithms, the second one is inherent to the choice we make. For example, if we adopt a polygonal representation of the real world, all of the geometrical operations are vertex/edge dependent. Using a discrete coordinate system, we can complement the real representation by adding some new funcionalities. Once we have discrete coordinate systems, the complexity depends only on

the dimension of the shapes, and is much more controllable. So we should try to use a dual representation, where each element has a

- **Continuous Representation**, that allows a close approximation to the real world;
- **Discrete Representation**, that allows an approximate calculation of geometric operations, in which we can determine the precision.

So all of the morphological operations are discrete, thus we work in $Z^2$. We can adjust this result using the continuous / polygonal information.

**Scaling and Discretization**

Using discrete coordinate systems in a practical way is just a matter of adjusting the correct approximation error and therefore the complexity of the geometric algorithms. Going back and forth from a continuous to a discrete coordinate system is done easily. All we have to do is to align both referencials (including the origins) and define an adequate invertible scaling factor between representations. Choosing the scaling factor is equivalent to assign a dimension to each discrete unit. So we have to build an operator that allows switching from representations. The operator should be invertible, so it should be able to transform a continuous position into a discrete position and vice-versa. Formally we have

$$Transform(P) = A \tag{34}$$

such as $P \subseteq \Re^2$, $A \subseteq Z^2, P \subseteq A$. The $P \subseteq A$ condition is essential to maintain the integrity when switching from a discrete representation to a continuous one. For example, if such condition is not preserved, calculating the nofit polygon of two shapes (using the Minkowski sum operator) can lead us to a set of non-valid positions in the continuous space.

**RESULTS**

Several tests where performed using a Pentium II processor running at 350Mhz. All algorithms were developed in C++. (Fig. 1), (Fig. 3) and (Fig. 5) show the input data and (Fig. 2), (Fig. 4) and (Fig. 6) show the results. The container has a bounding box of approximately 3000mm by 3000mm.

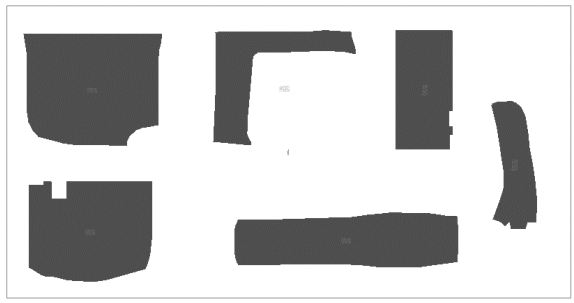

Figure 1: Input data for A* algorithm. (Ex. 1)

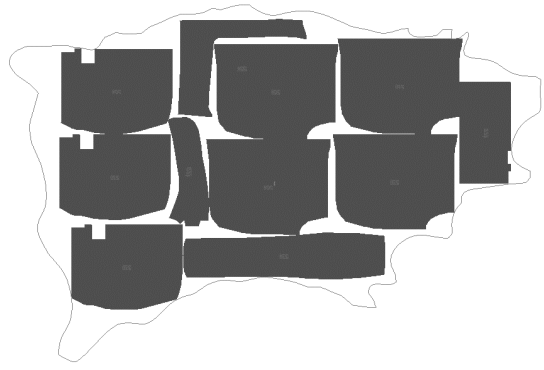

Figure 2: A* algorithm applied to input data. (Ex. 1)

In our first example we will try to nest 4 copies of each shape shown in (Fig. 1). Automatic containment was performed in **22m17s** having an occupation of **72,4%**.

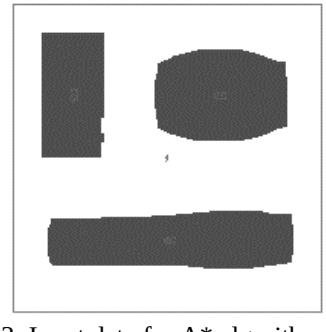

Figure 3: Input data for A* algorithm. (Ex. 2)

In our second example we will try to nest 10 copies of each shape shown in (Fig. 3). Automatic containment was performed in **32m53s** having an occupation of **73,3%**.

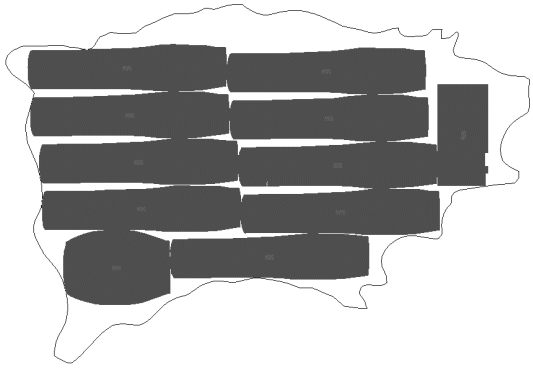

Figure 4: A* algorithm applied to input data. (Ex. 2)

In the third example we will try to nest 5 copies of each shape shown in (Fig. 5).

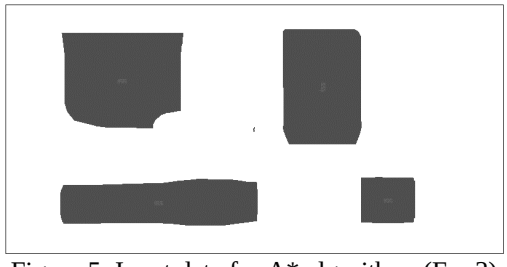

Figure 5: Input data for A* algorithm. (Ex. 3)

Automatic containment was performed in **32m53s** having an occupation of **73,3%**.

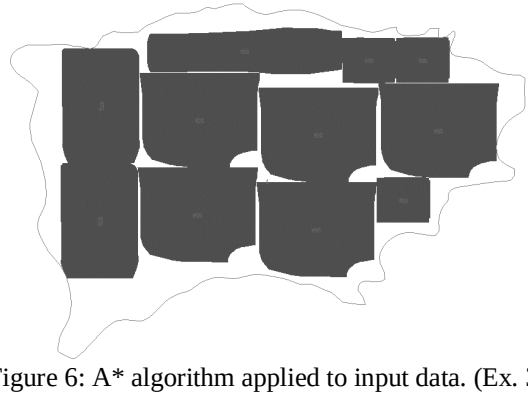

Figure 6: A* algorithm applied to input data. (Ex. 3)

## FUTURE WORK

We are currently working in extending the current concepts so that we can apply rotations to the shapes to be placed. Rotational containment present an additional degree of freedom that makes the branching factor of the A* tree increase substantially. We will try to apply A* to fill in the gaps, and solve most of the containment through the use of an iterative algorithm. We are also working on the mixed discrete/polygonal representational issues, trying to increase the speed of the calculations and minimise the distortion between both representations.

## CONCLUSIONS

Containment problems were, are and it will be an area of investigation in the quest for the "Holy Grail". For small problem spaces, A* search, or one of it's memory bounded equivalents, are applicable to solve difficult problems. It is the case of translational containment. A* can solve this family of problems until the problem space becomes too large to be computed. Implementing the geometric description using the mixed discrete/polygonal can allow A* to be a useful tool in containment problems where we have to make a trade off between running time and nesting efficiency.


## ACKNOWLEDGEMENTS

We would like to express our thanks to all people involved in F.A.C.A.P. project that have encouraged our current research, for their suggestions and critical comments.


## REFERENCES


[Avnai89a] Avnaim, F. *Placement et déplacement de formes rigides ou articulés*. PhD thesis, Université de Franche-Comté, France, 1989.

[Chaze83a] Chazelle, B. The Polygon Containment Problem. In Preparata, editor, Advances in Computing Research, Volume 1: Computational Geometry, pages 1-33. JAI Press, Greenwich, Connecticut 1983.

[Danls95a] Daniels, K.: *Containment Algorithms for Nonconvex Polygons with Applications to Layout.* Harvard University; Cambridge, Massachusetts. pages 41-44, 1997.

[Milen96a] Milenkovic, V. *Translational Polygon Containment and Minimal Enclosure using Linear Programming Based Restriction*. In Proc. 28th Annu. ACM Sympos. Theory Comput. (STOC96), pages 109-118, 1996.

[Russl94a] Russell, S. Norvig, P. *Artificial Intelligence – A Modern Approach*. Prentice Hall, pages 55-111 , 1994.

[Serra82a] Serra, J. *Image Analysis and Mathematical Morphology*, Vol. 1. Academic Press, New York, 1982.