UNIVERSITY OF WATERLOO COMPUTER SCIENCE DEPARTMENT
UNIVERSITY OF WATERLOO COMPUTER SCIENCE DEPARTMENT
UNIVERSITY OF WATERLOO COMPUTER SCIENCE DEPARTMENT

# *The Complexity and Decidability of* SEPARATION™

*Bernard Chazelle*
*Thomas Ottmann*
*Eljas Soisalon-Soininen*
*Derick Wood*

*Data Structuring Group*
*CS-83-34*

*November, 1983*

# THE COMPLEXITY AND DECIDABILITY OF SEPARATION™ [1]

*Bernard Chazelle*[2], *Thomas Ottmann*[3],
*Eljas Soisalon-Soininen*[4], *and Derick Wood*[5]

*ABSTRACT*

We study the difficulty of solving instances of a new family of sliding block puzzles called **SEPARATION**™. Each puzzle in the family consists of an arrangement in the plane of $n$ rectilinear wooden blocks, $n > 0$. The aim is to discover a sequence of rectilinear moves which when carried out will separate each piece to infinity. If there is such a sequence of moves we say the puzzle or arrangement is *separable* and if each piece is moved only once we say it is *one-separable*. Furthermore if it one-separable with all moves being in the same direction we say it is *iso-separable*.

We prove:

(1) There is an $O(n \log n)$ time algorithm to decide whether or not a puzzle is iso-separable, where the blocks have a total of $n$ edges.

(2) There is an $O(n \log^2 n)$ time algorithm to decide whether or not a puzzle is one-separable.

(3) If the puzzle is *loose* (a term made precise in the paper) then it is decidable whether or not the puzzle is separable.

(4) For loose puzzles the decidability of separability is *NP*-hard.

---

[1] The work of the first author was supported under a National Science Foundation Grant No. MCS-8303925, that of the third by a grant from the Alexander von Humboldt Foundation, and that of the fourth by a Natural Sciences and Engineering Research Council of Canada Grant No. A-5692.

[2] Computer Science Department, Brown University, Box 1910, Providence, R.I. 02912, U.S.A.

[3] Institut für Angewandte Informatik und Formale Beschreibungsverfahren, Universität Karlsruhe, Postfach 6380, D-7500 Karlsruhe, W. Germany.

[4] Department of Computer Science, University of Helsinki, Tukholmankatu 2, SF-00250 Helsinki 25, Finland.

[5] Data Structuring Group, Department of Computer Science, University of Waterloo, Waterloo, Ontario N2L 3G1, Canada.

(5) There are puzzles which require time exponential in the number of edges to separate them.

## 1. INTRODUCTION

The Simba puzzle consists of 10 rectangular wooden blocks arranged in a tray one of whose side's has a gap. The purpose of the puzzle is to rearrange the blocks by sliding them north, south, east, or west so that the largest block can escape through the gap. In [GY] the translation problem for rectangles is studied. The aim is to translate the original figure to some new position by moving each rectangle once and only once. Moreover as in Simba the rectangles are not allowed to slide over each other, so it is useful to think of the rectangles as rectangular wooden blocks.

In this paper we consider rectilinear wooden blocks rather than rectangular ones, we restrict movements to be only in the northerly, southerly, easterly and westerly directions as in Simba, we allow, in general, each block to be moved many times, and we concentrate on separating the blocks rather than translating the arrangement, configuration, or puzzle. In Simba the separation of one specific block is the purpose of the puzzle, while in [GY] the translation of a figure yields a sequence of moves, which enable the rectangles to be separated from each other, in the given order. This intuitive notion of separation can be expressed more precisely as moving each block independently to infinity without sliding over any other block (except at infinity). This is the definition of a family of puzzles called **SEPARATION**$^{TM}$.

In Section 2 we consider **SEPARATION**$^{TM}$ in which each piece is only allowed to move once, that is iso- and one-separability. In Section 3 we investigate the decidability status of **SEPARATION**$^{TM}$ when each piece is allowed a finite, but unbounded, number of moves. We show, assuming the initial arrangement is *loose* in a way which is made more precise later, that separability is decidable. In Section 4 we demonstrate that decidability is *NP*-hard and that there are separable puzzles which require exponential time to separate them. Finally we close with a discussion of some further problems and results in Section 5.

The original motivation for the problems discussed here was the generalization of the results of [GY] to rectilinear polygons, and an interest in moving rectilinear objects through rectilinear passages, see [HJW], [LPW], [OSC], [R], and [SS1-3]. **SEPARATION**$^{TM}$ can also be viewed as the opposite of two-dimensional bin packing, see [BCR], or compaction, see [SLW]; we thought, in fact, of calling it BIN UNPACKING!

## 2. ISO- AND ONE-SEPARABILITY OF SEPARATION$^{TM}$

In this section we sketch the proof of the following theorems.

**Theorem 2.1** *Given a puzzle consisting of $p$ pieces with a total of $n$ edges. Then one-separability is decidable and moreover it can be determined in $O(n \log^2 n)$ time and $O(n \log n)$ space.*

In *one-separability* each piece is only allowed to move once, but it may move in any one of the four directions. To approach an efficient solution to this version of the puzzle we first consider a special case in which the pieces must move in the same direction, that is *iso-separability*.

**Theorem 2.2** *Given a puzzle consisting of $p$ pieces with a total of $n$ edges. Then iso-separability is decidable and moreover it can be determined in $O(n \log n)$ time and space.*

Without loss of generality assume that easterly movement is only allowed. Then an arrangement such as Figure 2.1 is not easterly-separable, although it is iso-separable, while that of Figure 2.2 is not even one-separable, but it is separable.

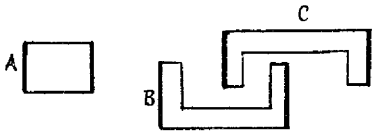


Figure 2.1

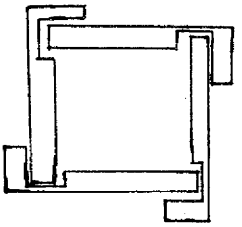

Figure 2.2

Returning to Figure 2.1 a viewer in the far east when looking over the puzzle can see that a leading edge of $B$ (outlined in bold-face) is trapped between a leading and trailing edge of $C$, whereas when it is viewed from the south, see Figure 2.3, the leading edges of $B$ and $C$ are free. We say that $B$ *traps* $C$, and is *trapped by* $C$, with respect to the east-west direction.

Similarly $A$, in Figure 2.1, which is trapped neither by $B$ nor $C$, is *blocked* by $B$ and $C$. In other words $A$ cannot be moved east until both $B$ and $C$ have been so moved. We say a piece is *free* if it is neither trapped nor blocked with respect to the given direction. Note that the relation traps is symmetric, whereas blocks is asymmetric. The relation traps is captured by:

> A piece $A$ traps a piece $B$, in a given puzzle with respect to the east-west direction, if and only if the *EW*-convex hulls of $A$ and $B$ have a non-empty intersection.

We say a piece is *EW-convex* if its intersection with a straight line, in the east-west direction, is either empty or a line segment. The *EW-convex hull* of a piece is the smallest *EW*-convex piece containing the given piece, see Figure 2.4 for an example. Note that the *EW*-convex hull does not affect the leading and trailing edges (or portions thereof).

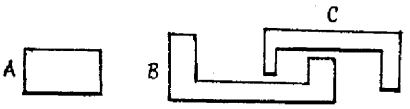


Figure 2.3

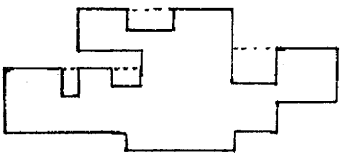

Figure 2.4

These simple observations are the key to the decidability of iso- and one-separability. An efficient algorithm is based on the segment tree, see [BW]. Without more ado we give a high-level algorithm.

**Algorithm** *EASTERLY-SEPARABILITY*;

*Step 1*: Replace each piece with its *EW*-convex hull.

*Step 2*: Sort the leading and trailing edges of the pieces in ascending order

according to their $x$-projection.

*Step 3*: Construct a skeletal segment tree based on the $y$-fragments determined by the $y$-projections of the leading and trailing edges.

*Step 4*: Insert the leading and trailing edges into the segment tree in $x$-sorted order.

*Step 5*: Attempt to peel the segment tree.

**end** *EASTERLY-SEPARABILITY*.

Step 4 ensures that each nodelist (of the edges which cover or mark a node) is sorted from east-to-west. Therefore, in Step 5, a necessary condition for the separation of a piece is that all appearances of its leading and trailing edges are in the first and possibly, second positions of its associated nodelists. This is because two leading edges of different pieces having the same $x$-projection must have disjoint $y$-projections (since they cannot overlap).

In order to begin to peel the segment tree (Step 5) we have to find a piece which is free, that is none of its edges are trapped or blocked. For this purpose we add further information to each node of the segment tree. Let $cover(u)$ denote the set of edges which mark or cover the node $u$. This is usually implemented as a doubly-linked list called the nodelist. Now let $easternmost(u)$ denote the set of first appearances in $cover(v)$, for all proper descendants $v$ and $u$ [†]. We say an edge of a piece is *blocked* if it lies to the west of the edge of some piece and their $y$-projections overlap, otherwise it is *free*. As pointed out above, an edge $e$ which appears in the first position in its nodelist is a candidate for freedom. It might not be free because either there is a larger or smaller blocking edge to its east. If the blocking edge is larger it will appear in the cover set of some proper ancestor of the nodes covered by $e$, while if it is smaller it will appear in the cover set of some proper descendant. Let $e$ appear in $cover(u)$ for some node $u$. Then the first case can be determined by examining the cover sets of the root-to-$u$ path. A larger blocking edge will appear in the first position of one of these nodelists. The second case involves the use of $easternmost(u)$, since a smaller blocking edge must appear in it, since it too must appear first in some nodelist of a descendant of $u$. Indeed $e$ is blocked by a smaller edge if and only if the most easterly of the edges in $easternmost(u)$ is to the east of $e$. This uses the *maxeast* operation, that is the *easternmost* sets can be organized as priority queues.

Now to prepare for peeling the segment tree we keep with each piece not only the number of its nodelist appearances, but also the number of free appearances. Initially, that is after Step 4, no appearances are free, hence a

---

[†] Actually since a leading and trailing edge of the same piece may both cover the same node this should really be the set of first, and possibly second, appearances.

traversal of the tree is made and for each node $u$, the first appearance in $cover(u)$ $e$, say, is tested for freedom. This involves examining $O(\log n)$ nodes. Since the *easternmost* sets are also constructed during Step 4, this traversal requires $O(n \log^2 n)$ time. If, after the traversal, no piece has its free count equal to its appearance count, then no piece can be separated from the others in an easterly direction. However if there is a free piece, each of its appearances is deleted and the *cover* and *easternmost* sets updated at all affected nodes. Removing an edge $e$ from $cover(u)$ for some node $u$ is straightforward, as is its removal from $easternmost(v)$ for all proper ancestors $v$ of $u$. It is more difficult, however, to update freedom information for the remaining pieces. The edge $e$ may block either smaller edges at descendants of $u$, or larger edges at ancestors of $u$. The latter situation is the more straightforward one - simply re-consider the freedom of the first appearances in $cover(v)$ for all ancestors $v$ of $u$. Indeed unless the edge furthest east in $easternmost(v)$ belongs to the deleted piece, freedom cannot be affected. In the former situation the edges in $easternmost(v)$, for all ancestors of $u$ and $u$ itself are the only ones which may be affected. It appears that the freedom of all of them needs to re-considered. To avoid this we modify the definition of $easternmost(u)$ by not including appearances in it which are already blocked below $u$, thus $easternmost(root)$ is the set of appearances which are blocked, at worst, by $cover(root)$. Let $\lambda u$ and $\rho u$ denote the left and right child of node $u$, and for a node $u$, define left and right sets $L$ and $R$, respectively, by:

$$L = \begin{cases} maxeast(cover(\lambda u)), & \text{if } cover(u) \neq \emptyset \text{ and } maxeast(cover(\lambda u)) \\ & \text{is east of } maxeast(easternmost(\lambda u)), \text{and} \\ easternmost(\lambda u), & \text{otherwise.} \end{cases}$$

$R$ is defined similarly. Now let:

$$easternmost(u) \text{ be } L \cup R.$$

Recall that an edge $e$ at a node $u$ only *directly* blocks appearances of edges at nodes below it if they appear in $easternmost(u)$. Now $easternmost(u)$ after removal of $e$ requires no further updating. Therefore consider $easternmost(\pi u)$, that is the parent of $u$. We need to add to $easternmost(\pi u)$ those appearances from $easternmost(u)$ which were blocked by $e$ but are no longer blocked at $u$. These appearances can be found by a range query of $easternmost(u)$ using the $x$-coordinates of $e$ and $e' = maxeast(cover(u) - \{e\})$. Note that if this query has a non-empty result then $e'$ is blocked from below and otherwise $e'$ is the only new addition to $easternmost(\pi u)$. Now consider $\pi\pi u$, the newly added appearances in $easternmost(\pi u)$ must be divided into those which should be added to $easternmost(\pi\pi u)$, and those which are blocked at $\pi u$. But this is similar to the previous reduction. The newly-freed appearances are those added to $easternmost(root)$ which are to the east of $maxeast(cover(root))$.

Observe that an appearance can be added to at most $O(\log n)$

*easternmost* sets and each addition requires $O(\log n)$ time. Thus each appearance contributes at most $O(\log^2 n)$ time during updating.

Fortuitiously this modified *easternmost* set is sufficient for the earlier stages of the algorithm, hence there are no major changes to be considered.

Now Theorem 3.1 follows because deletion and insertion of an edge can affect $O(\log n)$ nodes and, thus, require the updating of $O(\log n)$ priority queues each of size $O(n)$. Although we have only discussed the deletion of an appearance when it is in the first position of a nodelist, it is straightforward to modify this to deal with an appearance at any position (the segment tree has a dictionary of appearances for each edge, providing access, in constant time, to each appearance). We keep four segment trees, one for each direction, at any stage we check if there is a piece having all its edges free with respect to one of the directions. If there is we delete its edges from all four trees, and repeat the process until there are either no free pieces or no pieces at all.

Clearly we can apply Theorem 2.1 to solve the iso-separability problem as well, but we can improve the solution by way of:

**Theorem 2.3** *A puzzle is easterly-separable if and only if the EW-convex hulls of its pieces are disjoint.*

**Proof:** Straightforward. □

Clearly a puzzle is easterly-separable if and only if it is westerly-separable. Now it can be determined in $O(n \log n)$ time and $O(n)$ space whether or not two pieces intersect, by way of a simple extension to the algorithm in [BW] for rectangle intersections. Thus Theorem 2.2 follows.

## 3. THE DECIDABILITY OF SEPARATION™

In this section we sketch the proof that separability is decidable when we assume the initial arrangement is *loose*. First observe that the number of moves can be independent of the size of the puzzle. Consider the puzzle consisting of four pieces in Figure 3.1. The only way that $A$ and $B$ can be separated is by moving the two $U$-shaped pieces out of $A$. The two $U$-shaped pieces can only be moved alternately *a distance dependent on the narrowness of the $U$ s*. This distance can be made smaller than any $\epsilon > 0$, hence the total number of moves to achieve separation is independent of the number of edges.

We now introduce the notion of an *EW-obstacle*. Define the *adjacency graph* $G_{EW}$ of a puzzle as follows. The pieces in the puzzle are the nodes of $G_{EW}$, and the directed edges of $G_{EW}$ are determined by:

For all pieces $p$ and $q$ in the puzzle, there is an edge $(p,q)$ in

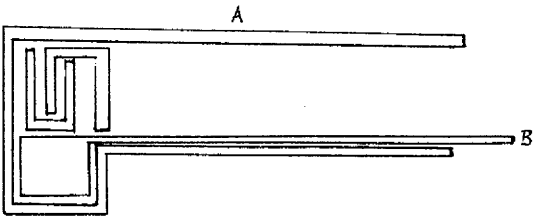


Figure 3.1

$G_{EW}$, if and only if $p$ and $q$ have a common segment $e$ that is they abutt, and $p$ is to the west, locally, of $e$ and $q$ is to the east, locally, of $e$.

Now any (directed) cycle in $G_{EW}$ is called an *EW-obstacle*. Clearly *NS-obstacles* can be defined similarly from the corresponding graph $G_{NS}$. Figure 3.2(a) illustrates an *EW*-obstacle and $G_{EW}$ while Figure 3.2(b) gives a similar situation which is not an *EW*-obstacle. Informally an *EW*-obstacle represents a cluster of pieces none of which can be moved in the *EW*-direction *at all*. Moreover there is no hope that they can be moved in the *EW*-direction unless some movement in the *NS*-direction is first made. Clearly an *EW*-obstacle which is also a *NS*-obstacle, called an *obstacle*, can never be moved. In Figure 3.3 $\{A,B\}$ and $\{D,E\}$ are obstacles.

We now have:

**Definition** A puzzle is *loose* if it has neither *EW*- nor *NS*-obstacles.

To show that separability is decidable for loose puzzles we make a number of preliminary observations.

First, observe that a puzzle is separable if and only if it can be transformed by a sequence of moves into a puzzle in which all pieces are no closer than some distance $d$ in the *EW*- or *NS*-direction, where $d$ is greater than the maximum height and length of the pieces.

Second it is useful to form an abstraction of a puzzle called a *scheme*, as follows. A scheme is a pair $(L_{EW},L_{NS})$ of lists giving the *EW* and *NS* order, respectively, of the vertices in the puzzle, where each vertex is specified as a pair $(i,j)$ to designate the $j$-th vertex of the $i$-th piece. Conversely we may say that a scheme is a pair $(L_{EW},L_{NS})$ of permutations of the vertices of some pieces, which can be *realized* by some puzzle formed from them. We say two puzzles $P_1$ and $P_2$ formed from the same pieces are *connected* if they have the same scheme and there exists a move sequence taking $P_1$ to $P_2$ (and, hence, vice versa). Similarly we say that a set of

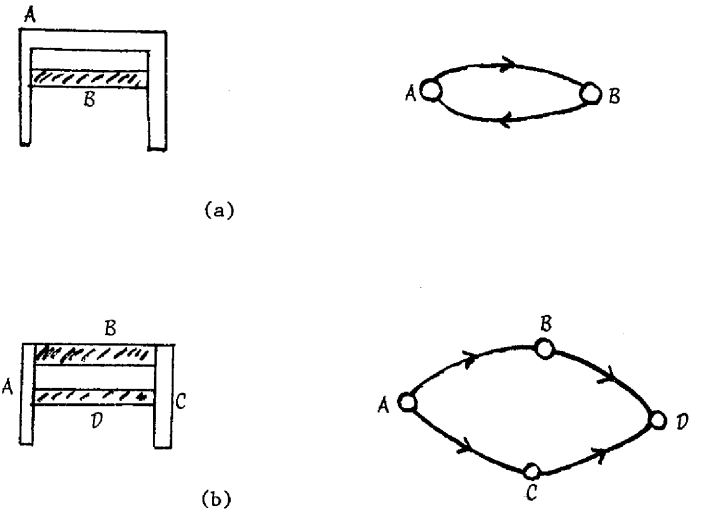


Figure 3.2

puzzles $P$ is *connected* if all $P_1$, $P_2$ in $P$ are connected.

We first prove:

**Lemma 3.1** *Let $(L_{EW}\, L_{NS})$ be a scheme for a given set of pieces, which is realized by a loose puzzle. Then $P$, the set of all puzzles which realize $(L_{EW}, L_{NS})$, is connected.*

**Proof Sketch:** We say $P_1$ *and* $P_2$ *are x-similar* (*y-similar*) if they both realize the same scheme and all corresponding vertices have the same $x$-coordinates ($y$-coordinates).

(A) To show that $P$ is connected we demonstrate that for two arbitrarily-chosen puzzles $P_1$ and $P_2$ in $P$, $P_1$ and $P_2$ are connected. To prove this we show that there is a puzzle $P$, which is in $P$, is $x$-similar to $P$, is $y$-similar to $P_2$, and there is a move sequence taking $P_1$ to $P$. Showing there is a move sequence taking $P$ to $P_2$ is a similar step. Note that $P$ is indeed in $P$, since if it contained overlapping pieces this would contradict the orders $L_{EW}$ and $L_{NS}$.

(B) To show that there is a move sequence taking $P_1$ to $P$, we further subdivide the problem. Let $H$ be a horizontal line below both $P_1$ and $P$. We define a new puzzle $Q$ which lies above $H$, has at least one

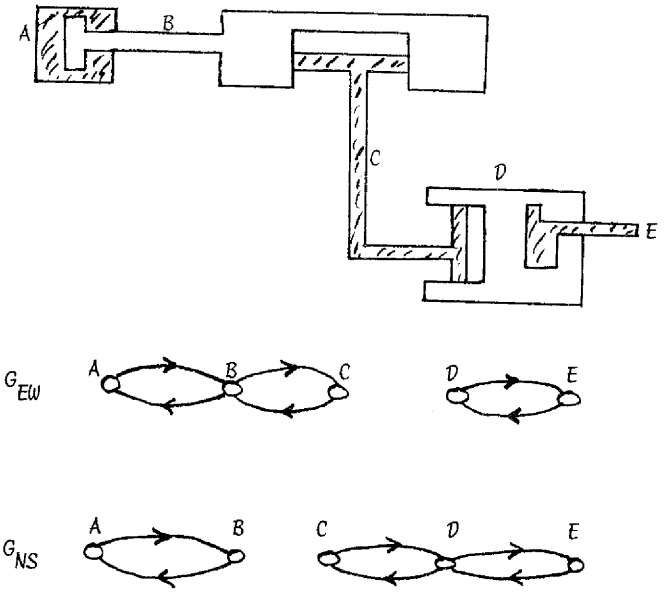


Figure 3.3

piece of $Q$ abutting $H$, has been allowed to "drop" as far as possible without crossing $H$, and is $x$-similar to both $P_1$ and $P$. Thinking of the pieces having weight and of them dropping under gravity is what is meant here.

If we show that $Q$ can be obtained from $P_1$, then, clearly, $P$ can be obtained from $Q$.

(C) To show that there is a move sequence taking $P_1$ to $Q$ we carry out the following algorithm.

For each piece $p_1, \ldots, p_k$ in turn:

Move $p_i$ downwards as far as possible, without crossing $H$. This destroys edges of the kind $(p_i, q)$ and possibly creates a new edge of the form $(q, p_i)$. Moreover it preserves $x$-similarity. If $p_i$ reaches $H$ then it is *frozen* at $H$, and is never moved again. Moreover freezing

propagates - if $p_i$ abutts a frozen $p_j$ then $p_i$ is also frozen.

The above process is iterated until a frozen puzzle is obtained, which is $Q$. We claim it is both unique and independent of $P_1$, in the sense that any puzzle $R$ in $P$ $x$-similar to $P_1$ would give rise to $Q$. We must also show that the process converges and is finite. We omit the details of this proof, simply remarking that it can be shown that each non-frozen piece can be moved in at most $k$ iterations of the process. Finiteness follows, essentially, by observing that each piece, when it is moved, is moved at least distance $\delta$, where $\delta$ is the minimum non-zero $y$-distance between successive (with respect to $L_{NS}$) vertices in $P_1$. Uniqueness is straightforward. □

**Theorem 3.2** *Let $P_1$ be a loose puzzle. Then it is decidable whether or not $P_1$ is separable.*

**Proof Sketch:**

(A) Let $P_2$ be an ordered horizontal placement of the pieces in $P_1$ such that there is a north-south line which can be drawn between any two pieces $p_i$ and $p_j$, with $p_i$ wholly to its west and $p_j$ wholly to its east or vice versa. Then $P_1$ is separable if and only if there is a move sequence taking $P_1$ to $P_2$.

(B) Let $(L^i_{EW}, L^i_{NS})$ be the scheme of $P_i$, $i = 1,2$. Now let $G$ be a graph whose nodes are (realizable) schemes of the set of pieces of $P_1$. For all (realizable) schemes $S_1$ and $S_2$, there is an edge $(S_1, S_2)$ if and only if there is a puzzle $P$ realizing both $S_1$ and $S_2$. This is possible only if $P$ contains two co-linear edges. Such a $P$ represents the point of change between two schemes. During a move sequence taking $P_1$ to $P_2$ (if it exists) there will be time instants when such points of change are crossed, while at all other instants the current scheme is unchanged. Thus $P_1$ can be transformed into $P_2$ if and only there is a path in $G$ from a scheme of $P_1$ to a scheme of $P_2$, that is separability has been reduced to reachability in a graph. Since reachability is easily determined, separability is decidable if and only if $G$ can be constructed. $G$ can be constructed if it is decidable whether or not a given scheme is realizable. But this corresponds to determining whether or not the linear system of equalities and inequalities $L(S)$ of a given scheme $S$ has a solution. The equalities are given by the interdistances between vertices of each piece, whereas the inequalities are given by the lists $(L_{EW}, L_{NS})$ of $S$. The simplex method can be used to solve $L(S)$, hence separability is decidable. □

## 4. COMPLEXITY OF SEPARATION$^{TM}$

We sketch the proofs of two results in this section, namely **SEPARATION**$^{TM}$ is shown to be *NP*-hard by reducing the partition problem to it and that there are separable puzzles which require exponential time.

For the first proof sketch note that we only consider the predicate: *can the given instance of* **SEPARATION**$^{TM}$ *be separated?* The *partition problem* is: given $n$ weighted objects partition them into two equally-weighted subsets. The reduction is illustrated in Figure 4.1.

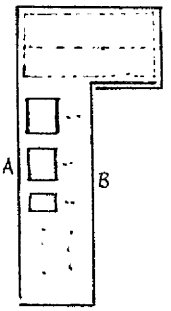


Figure 4.1

The puzzle is so tightly defined that none of the $P_i$ can be separated unless $B$ is moved west as far as $A$, when $B$ can be separated by sliding it south. However this is possible if and only if the set of $P_i$s can be partitioned and placed above $B$. Observe that there is enough working space to the west of $B$ to manipulate the $P_i$s. Thus we have:

**Theorem 4.1** **SEPARATION**$^{TM}$ *is NP-hard.*

In Section 2 we demonstrated that there are separable puzzles which require a number of moves independent of the size of the puzzle. However the moves required to separate the puzzle are self-evident. We close this section with one further example which is separable, but non-trivially, see Figure 4.2.

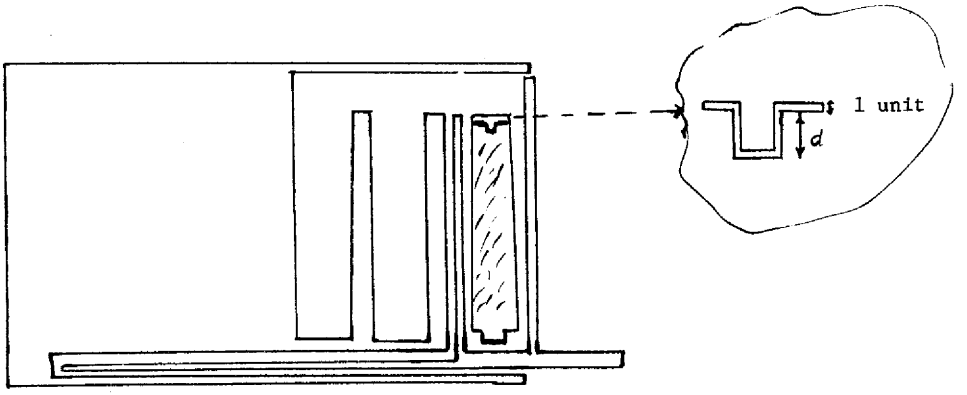


Figure 4.2

The piece $B$ can be viewed as a bolt, while the pieces $T$ are discs with different sized holes, which form a Towers of Hanoi. Note that the thickness of $m$ discs, if arranged in sorted order, is $m+d$ units, where each disc has a thickness of 1 unit and its hole is $d$ units deep. In unsorted order they form a tower which is $md$ units high.

Now to release the bolt $B$ all discs in $T$ need to be moved. They can only be moved into the two wells and the connecting passage, but because of the considerable difference in height between sorted and unsorted order, this forces the discs to be moved almost according to the standard Towers of Hanoi sequence, especially when $d = m$. Without belabouring the details we have:

**Theorem 4.2** *The puzzle illustrated in Figure 4.2 requires an exponential number of moves to separate it.*

## 5. DISCUSSION

We have introduced, **SEPARATION**$^{TM}$, a new family of sliding block puzzles and investigated some aspects of its complexity and decidability. Clearly much remains to be done. For example in Simba, one designated piece must be separated first, clearly this requirement and variants of it can be placed in our general framework. Again how efficiently can two-separability be decided? And, in general, given a $k$, now efficiently can $k$-separability be determined? Finally is **SEPARATION**$^{TM}$ decidable when the

puzzle is not loose? We conjecture that this is the case but its proof is elusive.

## REFERENCES

[BCR] Baker, B.S., Coffman Jr., E.G., and Rivest, R.L., Orthogonal Packings in Two Dimensions, *SIAM Journal on Computing 9* (1980), 846-855.

[BW] Bentley, J.L., and Wood, D., An Optimal Worst-Case Algorithm for Reporting Intersections of Rectangles, *IEEE Transactions on Computers, C-29* (1980), 571-577.

[GY] Guibas, L.J., and Yao, F.F., On Translating a Set of Rectangles, *Proceedings of the Tenth Annual ACM-SIGACT Symposium on Theory of Computing* (1980), 154-160.

[HJW] Hopcroft, J.E., Joseph, D.A., and Whitesides, S.H., On the Movement of Robot Arms in 2-Dimensional Bounded Regions, *Proceedings of the 23rd Annual Symposium on Foundations of Computer Science*, (1982), 280-289.

[LPW] Lozano-Perez, T., and Wesley, M., An Algorithm for Planning Collision-Free Paths among Polyhedral Obstacles, *Communications of the ACM 22* (1979), 560-570.

[OSC] O'Dúnlaing, C., Sharir, M., and Yap, C.K., Retraction: A New Approach to Motion Planning, *Proceedings of the Fifteenth Annual ACM Symposium on Theory of Computing* (1983), 207-220.

[R] Reif, J., Complexity of the Mover's Problem and Generalizations, *Proceeding of the 20th Annual Symposium on Foundations of Computer Science* (1979), 421-427.

[SLW] Schlag, M., Liao, Y.Z., and Wong, C.K., An Algorithm for Optimal Two-Dimensional Compaction Layouts, IBM Research Center, Yorktown, Research Report RC 9739, 1982.

[SS1] Schwartz, J.T., and Sharir, M., On the Piano Mover's Problem: I. The Special Case of a Rigid Polygonal Body Moving amidst Polygonal Barriers, *Communications on Pure and Applied Mathematics* (1983), to appear.

[SS2] Schwartz, J.T., and Sharir, M., On the Piano Mover's Problem: II. General Techniques for Computing Topological Properties of Real Alagebraic Manifolds, *Advances in Applied Mathematics* (1983), to appear.

[SS3] Schwartz, J.T., and Sharir, M., On the Piano Mover's Problem: III. Coordinating the Motion of Several Independent Bodies: The Special Case of Circular Bodies Moving amidst Polygonal Barriers, New York University Courant Institute Computer Science Technical Report, 1983.