Trash Compaction

Hugo Akitaya[*] Greg Aloupis[*] Maarten Löffler[†] Anika Rounds[*]

Abstract

Let P be a set of n objects on a square grid. A *push* is a transformation of P that involves sweeping a horizontal or vertical line by one unit, starting from the hull of P, displacing objects in the direction of the sweep. For example, when pushing to the right, all the leftmost objects are displaced one unit to the right. This in turn displaces other objects further right. Given P, we want to find a sequence of pushes that will produce a rectangle of a given height and width. We show that deciding whether a square can be produced is NP-hard, but it takes polynomial time to decide if a rectangle of height 2 can be produced.

1 Introduction

There is a rich history in computational and combinatorial geometry on packing unit squares as tightly as possible into various domains. In 1975, Erdös and Graham asked how many unit squares may be packed into a square of given dimensions, allowing arbitrary rotations [6]. This sparked a string of results [8], as well as new questions and variations. Of particular interest is the version where unit squares are not allowed to rotate [7]. The problem has recently been shown to be NP-complete when the domain is a rectilinear polygon with half-integer side lengths [5, 10].

Pushing objects is a vital task in certain motion planning settings with obstacles. Dhagat and O'Rourke first considered the problem of pushing square obstacles on a grid [4]. Many versions of the problem are NP-hard [3], in particular various popular games involving pushing blocks. Recently, the motion planning community has shown interest in controlling configurations of objects using only global interactions, motivated from swarm robotics [1, 2].

In this paper, we consider the *trash compaction* problem: given a set P of n objects (pieces of trash) at integer coordinates in the plane, can we we push them into a more compact configuration using only axis-aligned global *push* operations? In each of the four cardinal directions, we can perform a *push* operation on P. This involves sweeping a line in the given direction by one unit. Any object swept by the line is displaced, thus moving to the next integer coordinate. If another object occupies that coordinate, it is also displaced by one unit, etc. An example of a *left* push is shown in Figure 1. We focus on pushing objects into rectangular configurations. Trivially, this cannot always be done, as shown in Figure 1c.

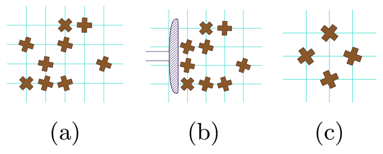


Figure 1: (a) Configuration of objects. (b) Configuration after a left push. (c) Configuration that cannot be pushed into an axis-aligned square.

1.1 General observations

There are some characteristics of this problem that we use in the proof of the main results in this paper. Due to space constraints, some proofs are omitted; they can be found in the full version.

If we are interested in pushing a configuration into a rectangle with dimensions $k \times \frac{n}{k}$, then trivially we require that k divides n.

Observation 1 *Suppose that a push causes j objects to be displaced within a row. This is equivalent to moving the first object to the closest available empty space in the row, j positions away.*

Observation 2 *Pushing horizontally* (resp. *vertically) cannot decrease the number of objects in any column* (resp. *row).*

Observation 3 *If any column contains more than k objects, or any row contains more than $\frac{n}{k}$ objects, then it is not possible to produce a $k \times \frac{n}{k}$ rectangle.*

In general, pushing horizontally then vertically is not equivalent to pushing vertically then horizontally (consider switching the last two pushes in Figure 2).

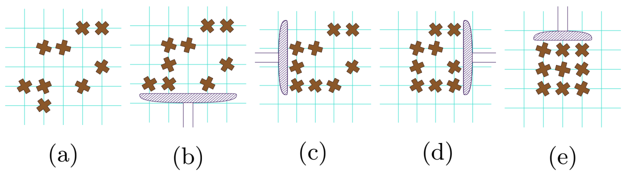


Figure 2: Four pushes to create a square.

[*]Tufts University, Medford, MA, USA
[†]Utrecht University, The Netherlands

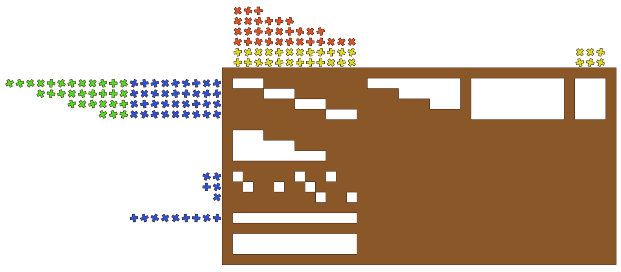
Lemma 1 *Pushing left then right is equivalent to pushing right then left.*

Lemma 2 *Suppose a configuration occupies exactly k rows (not necessarily consecutive). Then if a rectangle of dimensions $k \times \frac{n}{k}$ can be created, one way to do so is to push left until the configuration occupies $\frac{n}{k}$ consecutive columns, then push down until the configuration occupies k consecutive rows.*

2 Pushing into any rectangle

We show hardness of a restricted version of the problem, where only top and left pushes are allowed. Then we extend the result for the original question. Our reduction is from Exact-Hitting-Set (which is NP-hard [9]): Given a set $S = \{1, \ldots, n\}$, and m sets $S_1, \ldots, S_m$, each a subset of S, decide if there is a subset $S' \subset S$ of exactly k elements, such that each S_i has exactly one element in common with S'.

Construction. We build a rectangle with dimensions $\Theta(nm) \times \Theta(n+m)$. We fill this rectangle with objects, except for several empty *holes*. We then place the same number of objects as is needed to fill all the holes, above and to the left of the construction, in such a way that any sequence of top and left pushes that correctly fills the holes must correspond to a solution for Exact-Hitting-Set. To make the proof easier to follow, we color the objects that start outside the main rectangle. Figure 3 is an example for $n{=}4$, $m{=}3$, $k{=}2$. $S_1{=}\{1,3,4\}$, $S_2{=}\{1,2,3\}$, $S_3{=}\{3,4\}$.


(a) $n{=}4, m{=}3, k{=}2$. $S_1{=}\{1,3,4\}, S_2{=}\{1,2,3\}, S_3{=}\{3,4\}$

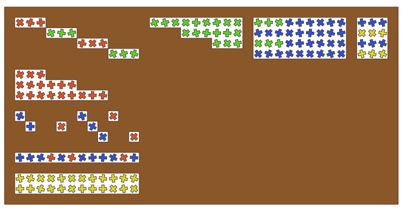

(b) $S'{=}\{2,4\}$

Figure 3: Reduction from Exact-Hitting-Set. For a larger copy with labeled regions, see the full version.

For our description, we use the convention described in Observation 1. There are eight main regions of the rectangle in which we insert holes. The upper-left *Staircase* region contains n horizontal holes, each of length m. To the right of the Staircase are three big holes: *Green-triangle*, *Green-overflow* and *k-check*. To the bottom of the Staircase is a big hole called *Red-triangle*, followed by a region of small holes called *Hit-check* and two big holes: *Red-overflow* and *Yellow-buffer*. The Staircase is nm wide and n tall. We consider that S is arbitrarily ordered and the i-th element ($i \in \{1, \ldots, n\}$) is represented by a hole of height 1 in the i-th row of the Staircase ranging from column $(i{-}1)m{+}1$ to im. The Green-triangle and Red-triangle are $(n{-}1)m$ wide and $n{-}1$ tall and the difference of the number of empty columns between adjacent rows is m. The Green-overflow, k-check, Red-overflow and Yellow-buffer are rectangular holes of dimensions $n \times (n{-}k{+}1)m$, $n{\times}m$, $1{\times}nm$ and $k{\times}nm$, resp. In the Hit-check region, make an $n{\times}m$ matrix of holes encoding the sets S_j. Each row represents a set. One element is represented by m columns. If the i-th element of S is in S_j, row i has a $1{\times}1$ hole in column $(i{-}1)m{+}j$. Outside the rectangle, right above the Staircase, place a $k \times nm$ rectangle of yellow objects. Right above, place n rows of red objects, each containing m less objects than the previous (starting with nm). Above the k-check region, place a $k \times m$ rectangle of yellow objects. To the left of the Staircase, place $n \times (n{-}k{+}1)m$ blue objects. To the left of the i-th row of blue objects, place $nm{-}(i{-}1)m$ green objects. To the left of the i-th row of the Hit-check region add $|S_i| - 1$ blue objects. Finally, to the left of the Red-overflow, add $1 \times (n{-}k{+}1)m$ blue objects.

Correctness. First consider that the hitting set problem has a positive solution S'. We convert S' into a sequence of moves that result in a rectangle. For $i \in \{1, \ldots, n\}$, if the i-th element of S is in S', left-push m times then down-push once. If the i-th element of S is not in S', down-push once then left-push m times. Left-push $(n{-}k{+}1)m$ times then down-push k times. This sequence will fill the holes in the Staircase corresponding to elements in S' with green objects and other holes with red objects. The Green-triangle and Red-triangle are filled with green and red objects respectively. The Green-overflow will be filled mostly with blue objects. Since $|S'| = k$, exactly $n{-}k$ rows of the Green-overflow will contain m green objects and, therefore, the k-check hole is filled with $n{-}k$ rows of blue and k rows of yellow objects. Because S' hits each subset exactly once, the objects to the left of the Hit-check and Red-Overflow will fill the holes that are not filled with red objects. Thus, this sequence compacts all objects into a rectangle.

Now, consider that there exists a sequence that results in a rectangle. The k-check hole can only be filled by yellow and blue objects and, since yellow objects can only be pushed into the rectangle after all red objects, there must be $n-k$ rows completely filled with blue objects in the k-check hole. That implies that each hole in the Staircase can only be filled with objects of the same color (either green or red) and that exactly k such holes are filled with green objects. Holes in the Hit-check region can only be filled with red or blue objects. A hole in such a region is filled with red objects only if the hole directly above it in the Staircase is filled with green objects. Since there are $|S_i|-1$ blue objects to the left of the i-th line, exactly one hole in each line must be filled with a red object. Therefore, the set S' of holes filled with green objects in the Staircase corresponds to a solution to the Exact-Hitting-Set instance.

Four-sided trash compaction. The reduction can be adapted to the model that allows pushes from all four cardinal directions. We remove one object from the top-left of the rectangle and add a *blocker* object at the bottom-right of the construction, which will prevent us from pushing from the right or bottom, because it would cause a row or column to become too long. However, once we resolve all pushes from the top and left, one top push and one left push incorporate the *blocker* filling the gap at the top-left corner.

Theorem 3 *Given a configuration P of n objects, and two integers w and h, deciding whether P can be pushed into a $w \times h$ configuration is NP-hard.*

Corollary 4 *Deciding if a configuration can be pushed into a $\sqrt{n} \times \sqrt{n}$ configuration is NP-hard.*

3 Pushing into a rectangle of height 2

It is trivial to decide if P can be pushed into a single row. This can be done if and only if each column contains at most one element. To decide if P can be pushed into a rectangle of height 2 requires more effort, even if P initially occupies only three rows. Let r_1, r_2, and r_3 denote the number of elements in the top, middle, and bottom rows respectively. We can trivially check in linear time whether any row contains more than half of the objects; if so, we report that no solution is possible. We now assume there are at most $n/2$ objects per row. Suppose that P occupies m columns and assume that the input is not trivial. (See Observation 3 and Lemma 2.) Since we want to compress the configuration into a rectangle of width $\frac{n}{2}$, we perform at most $m-\frac{n}{2}$ (i.e., at most $\frac{n}{2}$) horizontal pushes. Notice that we will push vertically exactly once. As soon as that happens, by Lemma 2, it is trivial to check if a rectangle can be formed. Without loss of generality, we may assume that we are never pushing up, since we perform at most one vertical push. We need to determine the number of left pushes and the number of right pushes that should be performed before the single vertical push. By Lemma 1, we can perform those left and right pushes in any order. Thus we characterize each potential solution by its *push signature* (ℓ, r), where ℓ is the number of left pushes and r is the number of right pushes. Since $\ell + r \leq n$, we can simply try all possible push signatures with these constraints, of which there are a quadratic number. Checking if a push signature is feasible takes amortized constant time by maintaining, for each row, a pointer to the last empty space in the configuration, so this would yield a total time complexity of $O(n^2)$.

Consider a push signature $G = (\ell, r)$, after performing ℓ left pushes and r right pushes. G is *feasible* if there is no column containing 3 objects. G is *conforming* (resp. *sub-conforming*) if the number of columns where there are two objects in the top two rows is equal to (resp. less than) $(r_1 + r_2 - r_3)/2$.

Lemma 5 *A configuration of n objects can be compressed into a $2 \times \frac{n}{2}$ rectangle iff there exists a push signature (ℓ, r) that is both feasible and conforming.*

Now, we introduce the *feasibility diagram*. It is a 2-dimensional matrix where the number of left pushes and the number of right pushes are on the axes (see Figure 4). We mark the feasible cells.

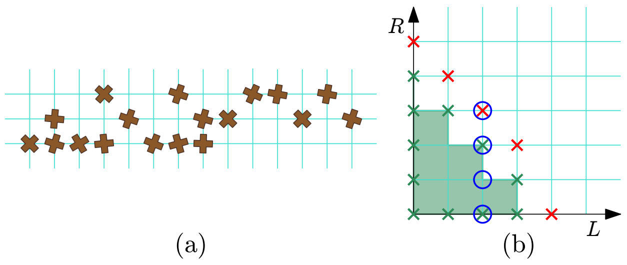


Figure 4: (a) A configuration (n_1=5, n_2=6,n_3=7) and (b) its feasibility diagram. The vertical axis is the number of right pushes, and the horizontal is the number of left pushes. The $\times$'s represent a computed feasibility push signature, where green is 'feasible' and red is 'not feasible'. The green shaded region shows the region of feasibility, which is all points below the staircase outlined by the computed feasible points. The blue $\circ$'s mark a conforming push signature.

Lemma 6 *Any cells below and to the left of a feasible cell are also feasible.*

Proof. Let a feasible cell have coordinates (ℓ, r). If $(\ell-1, r)$ were not feasible, we could take that configuration and push left to produce a feasible configuration. This means that we would have decreased the number of elements in a column, which is not possible by Observation 2. Therefore $(\ell-1, r)$ must also be feasible. The same applies to $(\ell, r-1)$. □

This property implies the Pareto-maximal cells form a staircase. We also mark the conforming cells, which form a x, y-monotone path. Finally, we check whether there is any feasible conforming cell.

Computing the feasible cells. A column of height 3 can only be created where an empty space is, and therefore, by keeping track of these spaces, we do not need to search the entire configuration every time we push. Using this fact, we can compute the feasibility of all push signatures that consist of only left or only right pushes in linear time, which will give us a maximum number of right and left pushes that are feasible. Given the maximum number R of right pushes that yield a feasible solution, we can then begin constructing the staircase; refer to Figure 4. We know that $(0, R)$ is feasible. If pushing left from the resulting configuration still yields a feasible solution, we note that $(1, R)$ is also feasible and continue. If pushing left is non-feasible, we decrement the number of right pushes we make, and see if pushing right that many times and then pushing left once yields a feasible solution. The crucial point is that we can test this in constant time, even though we cannot "unpush", by using the above observation. We continue in this way until we have drawn out a staircase in our diagram. By Lemma 6, we will characterize all feasible cells in the diagram in linear time.

Computing the conforming cells. After each push we add one element to one column for each row, which means that we can keep track of the number of columns we add elements to in constant time per push. At each push, we maintain the number of columns with two elements in the following way. If there is a column with one element in the middle row and one in the bottom row, then any push down will not change the number of elements in the resulting rows for this column. If there is a column with one element in the top row and one in the bottom row, then the same is true. However, if there is a column with one element in the top row and one element in the middle row, then when we push downward, the element from the middle row will be displaced into the bottom, and the element from the top row will be in the resulting top row. At each push, we maintain the number of columns that have a top-middle configuration, as well as the number of columns with the bottom-middle and top-bottom configurations. The number of top-middle configurations tells us how many objects will be added to the bottom row upon pushing, which will give us the point at which a push signature is conforming.

Any push-signature representing a solution will lie in the intersection of the conforming and feasible regions of the diagram. We compute both sets in linear time, and check if their intersection is empty.

Theorem 7 *Given any configuration P of n objects that occupy at most 3 rows, we can decide in $O(n)$ time whether P can be pushed into a $2 \times \frac{n}{2}$ rectangle.*

3.1 Rectangles of height k

The brute-force $O(n^2)$-time algorithm easily extends to rectangles of height $k > 2$. A solution now consists of exactly $k-2$ vertical pushes, and a number of left and right pushes between pairs of vertical pushes. We can encode this by a sequence of $k-2$ push signatures; since there are at most $n^{2(k-2)}$ such sequences.

Theorem 8 *If P has n objects occupying at most k rows, we can decide in $O(n^{2(k-2)})$ time whether P can be pushed into a $2 \times \frac{n}{2}$ rectangle.*

References

[1] A. Becker, E. Demaine, S. Fekete, and J. McLurkin. Particle computation: Controlling robot swarms with only global signals. In *ICRA*, pages 6751–6756, 2014.

[2] A. Becker, E. Demaine, S. Fekete, S. M. Shad, and R. Morris-Wright. Tilt: The video. designing worlds to control robot swarms with only global signals. In *Proc. 25th Multimedia Exp. Comp. Geom.*, 2015.

[3] E. D. Demaine, M. L. Demaine, M. Hoffmann, and J. O'Rourke. Pushing blocks is hard. *CGTA*, 26(1):21–36, 2003.

[4] A. Dhagat and J. O'Rourke. Motion planning amongst movable square blocks. In *Proc. 4th CCCG*, pages 188–191, 1992.

[5] D. El-Khechen, M. Dulieu, J. Iacono, and N. van Omme. Packing 2x2 unit squares into grid polygons is NP-complete. In *Proc. CCCG*, pages 17–19, 2009.

[6] P. Erdös and R. Graham. On packing squares with equal squares. *J. Comb. Theory*, 19:119–123, 1975.

[7] R. Fowler, M. Paterson, and S. Tanimoto. Optimal packing and covering in the plane are NP-complete. *Information Processing Letters*, 12(3):133–137, 1981.

[8] E. Friedman. Packing unit squares in squares: A survey and new results. *The Electronic Journal of Combinatorics*, 2009.

[9] M. R. Garey and D. S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness.* W. H. Freeman & Co., New York, NY, USA, 1990.

[10] A. van Renssen and B. Speckmann. The 2x2 simple packing problem. In *Proc. 23rd CCCG*, 2011.