Very High Density Storage Systems

Kevin R. Gue

Department of Industrial & Systems Engineering

Auburn University

Auburn, AL 36849-5346

kevin.gue@auburn.edu

IIE Senior Member

March 14, 2005

Abstract

We introduce and develop models for *very high density* physical goods storage systems, which are characterized by sometimes having to move interfering items in order to gain access to desired items. We describe a simple but effective algorithm to densely fill rectangular storage spaces, subject to a constraint on the number of interfering items. We also prove an upper bound on storage density for any rectangular space, including traditional warehouses.

1 Dense storage

The effective use of space is a goal for most every organization. The high cost and limited availability of real estate near population centers—where most businesses would like to locate—often forces firms to make the most of smaller facilities. In large cities it is common to find multi-level department stores, parking lots, and golf driving ranges, all efforts to make better use of space. Tokyo is even beginning to explore development underground (Wehrfritz and Itoi, 2003). The increasing value of facility space is a natural outcome of business development: as the firm grows, efficient use of space allows it to postpone the purchase or lease of larger facilities.

Effective space utilization has long been a theme in material logistics and transportation as well. For example, ports must make the best use of limited acreage to store empty and full containers and other cargo; shipping and trucking firms must maximize freight per vehicle in order to reduce the total costs of transportation; and distributors strive for high utilization of warehouse space to decrease total facility and operating costs.

Managers and engineers in warehouses and distribution centers have devised many ways to improve storage density. For example, Narrow-Aisle (NA) and Very Narrow-Aisle (VNA) storage systems have less space devoted to aisles, but this often comes at the cost of increased travel and congestion due to one-way travel within the narrow aisles. Another way to increase storage density is through the design of the storage locations themselves, as in "double-deep" pallet rack where each location contains two pallets, one behind the other. Warehouse managers usually configure items such that the pallets (or totes) in a lane contain the same *stock keeping unit* (sku) so that it is rarely necessary to move one item out of the way to access another. To increase storage density for fast-moving, bulky products, managers sometimes put pallets in a *block stacking* area, where pallets for a single sku are stacked on the floor in a lane that could be several pallets deep, and perhaps 2–4 pallets high. Because the lanes are so deep, this type of storage is more dense than single- or double-deep pallet rack. A disadvantage of this technique is *honeycombing*, in which open pallet positions in a lane cannot be used because they would block access to the sku currently occupying that

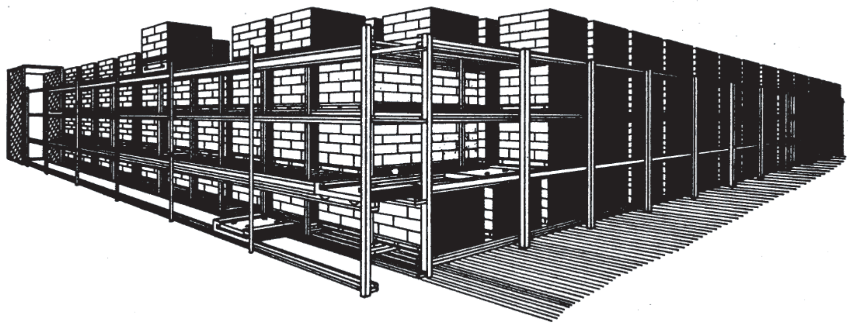


Figure 1: An automated deep bulk storage system. Pallets are stored in long lanes, and are automatically brought to the front aisle where they move to the input-output point.

lane (Heragu, 1997). An automated form of dense storage is *deep bulk storage*, in which pallets are stored several deep in an automated handling system (see Figure 1).

Why choose one form of dense storage over another? For example, why would a distributor choose single- or double-deep pallet rack over deep-bulk storage, when the latter system consumes far less space per storage location? The answer is that less dense storage systems confer an advantage by allowing easier access to items. For example, in single-deep pallet rack, workers never have to move an interfering item to gain access to a desired item. In a deep-bulk system a single retrieval might require several items to be moved, and this adversely affects average retrieval time and system capacity. We should also note that a denser system can be more secure from pilferage or tampering, precisely because access to many items is difficult.

We say that a storage system has *Very High Density* (VHD) when it is sometimes necessary to move interfering items in the system in order to gain access to desired items. Generally speaking, double-deep pallet rack has very high density because one might have to move a pallet to retrieve the one behind it, but single-deep pallet rack does not, because every pallet is accessible directly from an aisle. Deep-bulk systems also have very high density, as do many container yards in ports, where containers are sometimes stored several units deep, and perhaps several units high.

One of the first issues in designing a storage system is where to put the storage locations and aisles, a task usually referred to as the layout problem. This is the subject of our work. Specifically, we ask, for a very high density storage system,

1. How should items be stored within a space to maximize storage density, subject to a maximum *lane depth*; that is, the maximum number of items in a storage lane? and

2. Are certain storage spaces more amenable to dense storage than others? That is, do the dimensions of a storage space affect its potential storage density, and if so, how?

We believe VHD systems merit definition and study for a number of reasons. First, VHD systems are found in industry, and there is little existing research to support them. In addition to the systems we mention above, we know of one automated storage system that keeps pallets in a system similar to that in Figure 1, but it dynamically rearranges pallets to improve retrieval time in anticipation of scheduled requests. The original motivation for our work is a requirement by the U.S. Navy to build ships that can act as a "floating distribution centers." Unlike current supply ships, which store cargo in massive holds for offload in a single event, these future ships must be capable of selecting individual items, preparing them for delivery, and shipping them via aircraft. The ship has competing design goals of needing to store as much cargo as possible in a very confined space and needing high throughput to support forces ashore. Second, we believe that storage density is likely to play a more important role in future storage systems, for two reasons: (a) the high cost of real estate near population centers — where firms would like to locate to reduce transportation costs — will cause firms to look for ways to make better use of existing space, and (b) the increasing cost of labor relative to technology-based systems should make automated, labor-free storage systems more attractive in the future. Such systems are natural candidates for the ideas we present here. Third, we believe there is insufficient theoretical understanding of the relationship between storage density and retrieval time. We contend that these are two basic characteristics of any storage system, and it is important to understand how they interact.

Some authors have looked at the arrangement of aisles and racks in a rectangular warehouse, but to our knowledge none has addressed the objective of maximizing density. Instead, the primary focus has been on reducing labor costs because this is typically the largest operating cost in the warehouse (Frazelle, 2002). Heragu (1997) describes how to determine the length and width of a rectangular warehouse, with the objective of minimizing average travel distance. His method presupposes a way of arranging racks inside the warehouse, and is based on a method in Askin and Standridge (1993). Bassan et al. (1980) compare four different internal configurations of a warehouse, two having racks parallel to the longest dimension of the building and two having racks perpendicular to it. They determine which designs are best, based on expected annual travel distance as a surrogate for labor cost. Note that minimizing travel for workers also increases maximum sustainable throughput because workers can make more picks per time.

Iranpour and Tung (1989) describe a model to lay out spaces in a parking lot. Although this problem is similar to ours, it is complicated with other issues such as turning radii of vehicles, the angle of approach to the space, and the need for sensible traffic flow within the lot.

Container storage in ports is an example of dense storage, and has been addressed in several papers. de Castilho and Daganzo (1993) examine temporary storage areas for containers in a marine terminal. Their work addresses retrieval times in a storage system that requires moving interfering items. They develop formulas for the expected number of moves required to retrieve a container, and for the expected moves required to retrieve a set of containers, given an operating strategy. They do not address questions of how to lay out the space, and because the crane moves in a third dimension, there is no need for aisles. Taleb-Ibrahimi et al. (1993) address requirements for handling and storage space in a terminal that stages containers for export shipment. Kim and Kim (2002) present a model to determine the amount of storage space and the number of cranes in an import container operation, and Kim and Park (2003) show how to allocate temporary storage space to outbound containers to reduce handling costs and improve throughput. Generally speaking, the focus of these papers is improving throughput in a dense storage system by wisely sequencing stows and retrievals and allocating temporary storage space to groups of containers. In practice, storage density in a container terminal is an artifact of ship schedules and available storage space, rather than a product of layout and design choices.

In the following section we propose models to allocate items within a space, and we describe storage space dimensions that tend to favor the highest densities. In Section 3 we make conclusions and suggest ways that our results might be used to develop VHD storage systems.

2 Maximizing storage density

Rather than model a specific, existing storage system, we model an abstract system which we believe captures the "physics" of existing (and future) systems. Our model is based on a grid of storage locations, which could represent pallet locations in a warehouse or automated handling system, or even locations in a container yard (except that each cell in our grid is square, rather than rectangular).

Consider a rectangular grid of dimensions $m \times n$, where $m \leq n$. In each cell we can store an item, and there is a transport mechanism or vehicle that moves items from their current locations into unoccupied cells. We assume that the transporter itself must move among unoccupied cells; that is, it is co-planar with the items, rather than being able to move freely underneath or above them. The grid represents an enclosed space with a single input-output (I/O) point lying on the boundary of the grid. Therefore, cells comprising the aisles must form a connected network with access to the I/O point, and items may not be retrieved from any boundary location other than the I/O point.

In our model, aisle cells are the same size as cells containing items, and so aisles have the same width as items. For many automated systems this is a reasonable assumption, but for other systems—those involving forklifts in particular—aisles are wider than items to allow the transporter to maneuver. Should wider aisles be necessary, there are at least two

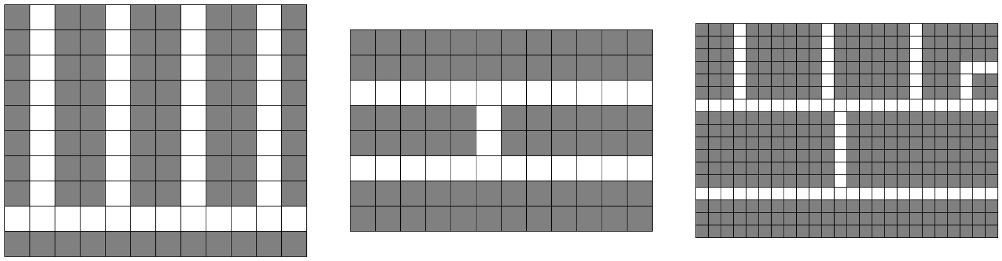


Figure 2: Example layouts: single-, two-, and three-deep, from left to right. Shaded cells represent stored items; empty cells aisles. Any aisle cell on boundary may serve as in input-output point.

options. If aisle width is approximately an integer multiple of the width of a storage cell, we can redefine "cell" to mean the integer multiple and proceed as before. An alternative is simply to "stretch" the aisles to required width and then fill in newly created storage space with items. We choose not to address these implementation-specific details, but to focus instead on abstract systems with equal aisle and cell widths.

2.1 A simple algorithm

We say that an arrangement of items (a layout) with maximum *lane depth* k is k-deep, if every item is accessible by moving no more than $k-1$ other items. Thus, in a 1- or single-deep layout every item is accessible directly from an aisle; in a 2-deep layout every item is accessible by moving no more than one other item, and so on. (Note that a single-deep layout is not technically a VHD system because it is not necessary to move an item to retrieve any other. Nevertheless, we include it as an important base case which is frequently found in practice.) Figure 2 shows three example layouts.

The first question is, given an $m \times n$ grid, what is the maximum number of items possible in a k-deep layout? Our algorithm, which we call Fill-and-Rotate, is based on a simple labeling procedure and a series of rotations of the grid (see Figure 3):

1. For an $m \times n$ grid, fill the bottom k rows with items and mark the next row as an

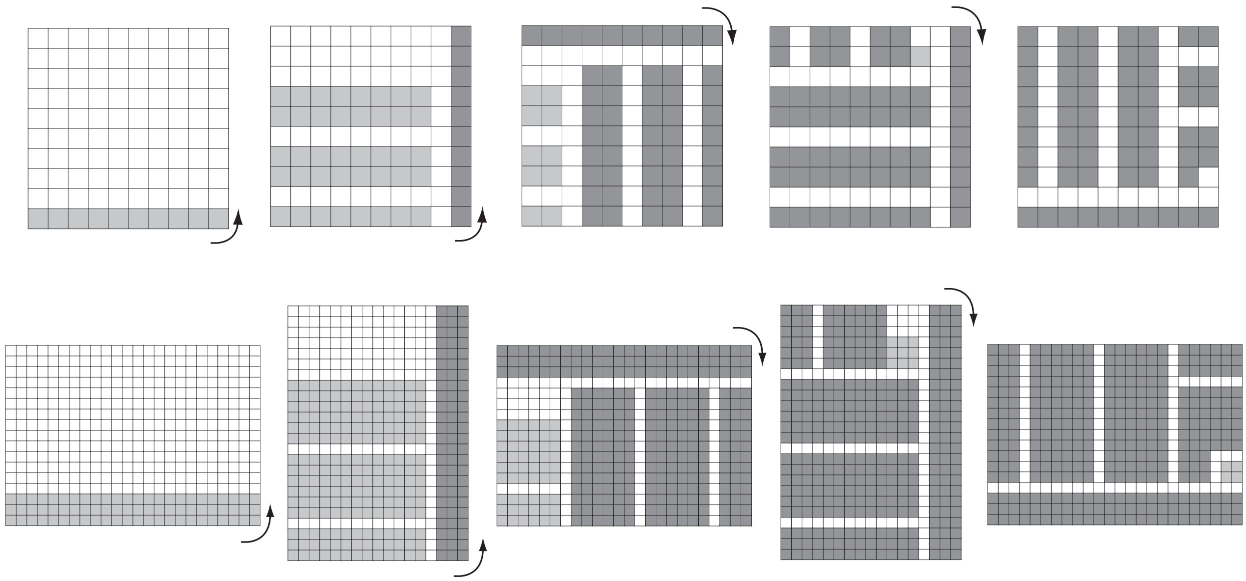


Figure 3: The Fill-and-Rotate algorithm applied to a 10×10 grid with $k = 1$ and to a 17×24 grid with $k = 3$. The 10×10 solution was shown to be optimal by Fujie (2004).

aisle. Rotate the grid counterclockwise.

2. Beginning at the bottom, fill k rows and mark an aisle, then fill up to $2k$ rows and mark an aisle repeatedly until doing so would result in an infeasible layout (this happens if there are $[k+1, 2k]$ unmarked rows remaining). If there are no unmarked cells, STOP; otherwise, rotate the grid counterclockwise.

3. Beginning at the bottom, fill k rows; then mark an aisle and fill $2k$ rows repeatedly until there are fewer than $2k + 1$ unmarked rows remaining.

4. If there are more than k rows remaining, mark an aisle, fill the remaining rows, and STOP. Otherwise, rotate the grid *clockwise*.

5. Fill the k bottom rows and mark an aisle. If all cells are marked or filled, STOP; otherwise, rotate the grid clockwise.

6. Fill the remaining rows from the bottom, leaving an aisle in the top row; STOP.

A formal statement of the algorithm is in Appendix A. Note that FILL-AND-ROTATE returns a different layout for an $m \times n$ grid than for an $n \times m$ grid.

Figure 4 shows several example solutions, for $k = 3$. Note that there are certain combinations of m and n that seem to yield very regular structure and others that seem to yield slightly irregular structure. For example, layouts for $n = 21, 25, 26, 27, 28$ have a single horizontal aisle connected to several vertical aisles, but layouts with $n = 22 \ldots 24$ have more complicated structure, especially when $m = 15 \ldots 17$. Does this change in structure have an effect on the density? That is, for a given density coefficient k, are there certain values of m and n that produce more dense designs?

To investigate this question, imagine replacing each layout in Figure 4 with the denser of an $m \times n$ or $n \times m$ layout and then expanding the figure for many values of m and n. Now replace each layout with a grayscale cell, where the shade of the cell represents the relative storage density of the layout (higher densities get *lighter* shades). Figure 5 shows the results for $k = 1$ through 6. A cell in each density plot in the figure represents a combination of m and n as follows: The bottom left cell in each plot represents the relative density of the FILL-AND-ROTATE algorithm applied to a grid of size $(2k+2) \times (2k+2)$, the next cell to the right represents the same for the denser of a $(2k+2) \times (2k+3)$ or $(2k+3) \times (2k+2)$ grid, and so on. (Choosing the more dense of the two orientations makes the plots symmetric about the $m = n$ cells.) The top right cell in each plot represents a 50×50 grid. Note that there are definite regions—combinations of m and n—for which densities are higher, and these occur when m and n are close or equal to, but do not exceed, a multiple of $2k + 1$. Figure 4 offers insight: as n approaches a multiple of $2(3) + 1 = 7$ (in this case, $n = 25 \rightarrow 28$) each additional column (increment of n) adds $m - 1$ items and only 1 aisle space, for a marginal increase in density of $(m - 1)/m$; but when n increases to one greater than a multiple of $2k + 1$ (e.g., $n = 21 \rightarrow 22$) the structure is broken, and additional aisle spaces (beyond one) must be added to maintain the k-deep constraint, thus reducing the density. A similar argument explains why m should be close or equal to a multiple of $2k + 1$ for maximum density.

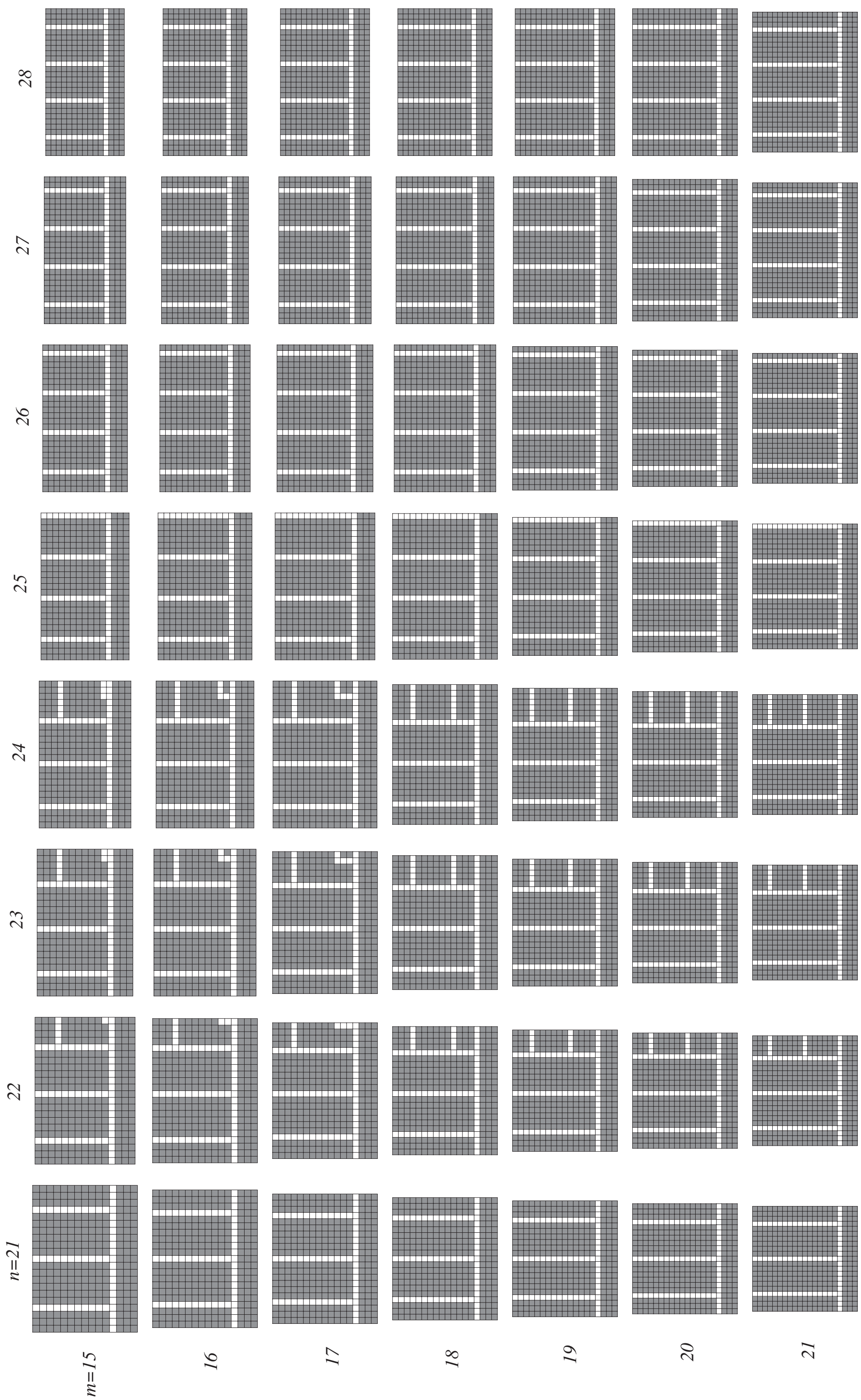


Figure 4: Results of FILL-AND-ROTATE for several grids. Rotate the page clockwise to view.

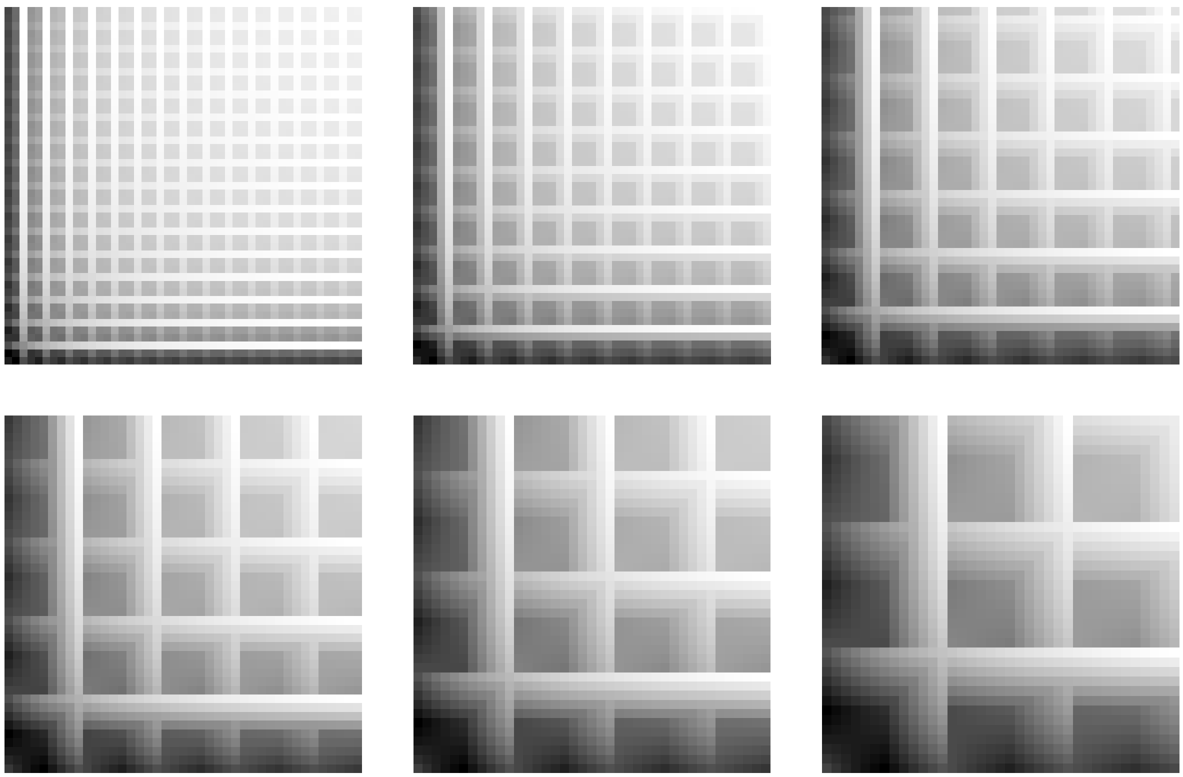


Figure 5: Relative densities of layouts produced by Fill-and-Rotate for $k = 1$ through 6 (left to right, top to bottom). Lighter cells indicate more dense designs. Grayscale shades reflect relative densities within plots, not between plots.

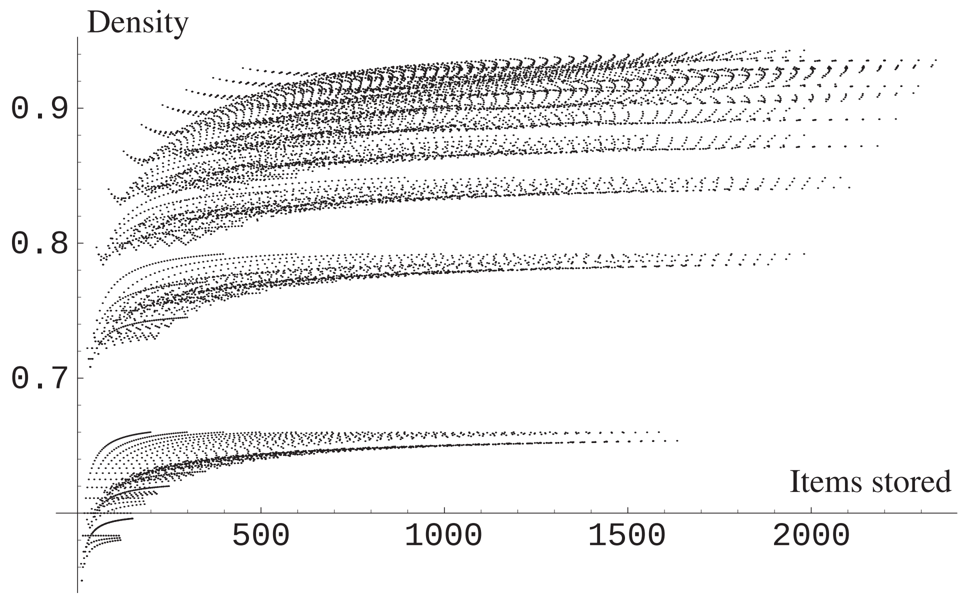


Figure 6: Density versus number of items stored for 14,770 configurations with m and n varying between 4 and 50, and k from 1 to 10.

Figure 6 shows the density of 14,770 configurations with m and n varying between 4 and 50, and k varying from 1 to 10. The plot shows storage density (vertical axis) versus number of items stored in the grid (horizontal axis), and gives us a way to compare similar configurations. The plot shows $k = 1$ configurations grouped at the bottom, $k = 2$ configurations in the next group up, and so on. Points along the upper "frontier" for each group represent dominating configurations; that is, to store a certain number of items with a specified value of k, configurations along the upper frontier are those that would consume the least amount of space.

Points corresponding to $k = 1$ configurations appear to lie along many concave curves, and those curves are in two major groups, one lying below the other. The top group of curves corresponds to configurations with the horizontal dimension of the grid (n) being an exact multiple of $2k + 1$. Layouts in this group have very few, long aisles connected by a single cross-aisle. Individual "curves" correspond to the horizontal dimension n taking on a particular multiple of $2k + 1$. In the $k = 1$ case, the top left curve is comprised of

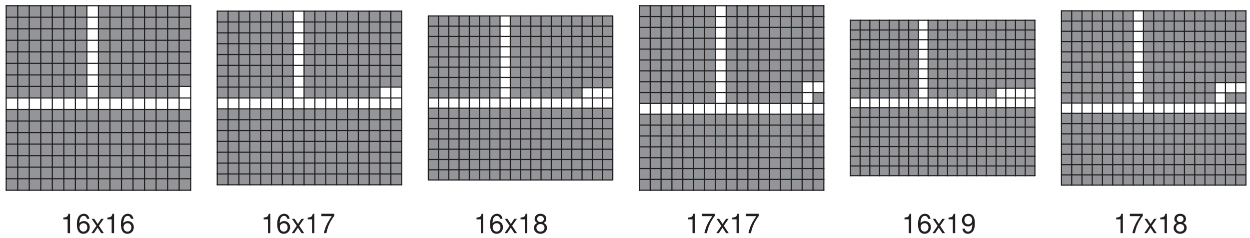


Figure 7: Small, high density designs for $k = 7$. Density decreases to the right.

configurations having $n = 6$, the next curve $n = 9$, and so on. (We omit the trivial case with m or $n = 2k + 1$ and require $m, n > 2k + 1$.) In the $k = 2$ case, from the upper-left, the curves represent configurations having $n = 10$, $n = 15$, and so on. This observation suggests that the most dense designs are long and narrow, with the smallest dimension equal to $2(2k + 1)$ or $3(2k + 1)$.

An interesting part of the plot is the groups of points that protrude off the frontiers for groups with high values of k. (These appear as "tails" on the upper-left parts of the curves.) The corresponding configurations are square or nearly-square in shape and have a single horizontal and single vertical aisle (forming an inverted T). Configurations at the very tips of these tails have $m, n = 2k + 2$. Figure 7 shows the five leftmost configurations in the tip for $k = 7$.

The configurations in Figure 7 highlight an important practical characteristic of layouts produced by Fill-and-Rotate: as long as $m, n > 2k + 1$, every layout has a cross-aisle of length at least k. This means that any item in the grid may be retrieved without having to move any interfering items out of the grid; that is, all interfering items may be temporarily repositioned within the grid while retrieving a specific item.

2.2 A related problem in graph theory

The Maximum Leaf Spanning Tree Problem (MLSTP) in graph theory asks, for a graph G, construct a spanning tree in G with the maximum number of leaf nodes (nodes having degree one). The MLSTP is also known as the Minimum Connected Dominating Set Problem (Duckworth, 2002; Alzoubi et al., 2002), where a connected dominating set in graph G is a connected set of nodes such that every node in G is connected to that set by one arc. It is easy to see that the set with minimum cardinality also gives a solution to the MLSTP.

Several authors have addressed the MLSTP on general graphs, which has applications in communications networks. Lu and Ravi (1998) describe an approximation algorithm for the MLSTP that gives a lower bound on the maximum number of leaves, which in our case is a lower bound on the maximum number of items in a storage space. Solis-Oba (1998) improves the algorithm of Lu and Ravi to provide solutions that are within a factor of 2 of optimal. Fujie (2003) describes a branch-and-bound procedure to solve MLSTP exactly. MLSTP is known to be NP-Hard on general graphs (Garey and Johnson, 1979); Galbiati et al. (1994) showed that there does not exist a polynomial time approximation scheme for the problem unless P = NP. Its complexity is unknown on grid graphs (Fujie, 2003).

Proposition 1 *The VHD storage problem with $k = 1$ is equivalent to the Maximum Leaf Spanning Tree Problem (MLSTP) on a grid graph.*

Proof The transformation from the single-deep VHD storage problem to MLSTP is: cells in the storage grid are nodes in the graph; permitted physical paths between neighboring cells (up, down, left, and right) are arcs in the graph. It is easy to see that any $m \times n$ storage grid can be transformed into and $m \times n$ grid graph. Stored items in the VHD storage problem correspond to leaf nodes in a solution to MLSTP; aisles in the VHD problem correspond to arcs not extending to leaves (see Figure 8). The last part of the transformation is to show that the constraint in the VHD storage problem that there be at least one I/O point (for the MLSTP, a non-leaf node on the boundary of the grid) does not cut off an optimal solution to MLSTP. It is easy to show every solution to MLSTP contains a non-leaf node on the boundary: each corner node must either be a leaf or non-leaf node. If a corner is a non-leaf node, then it may act as an I/O point in our problem; if it is a leaf node, then by definition one of its only two neighbors, which are both boundary nodes, must be a non-leaf node and may act as an I/O point in our problem. $\square$

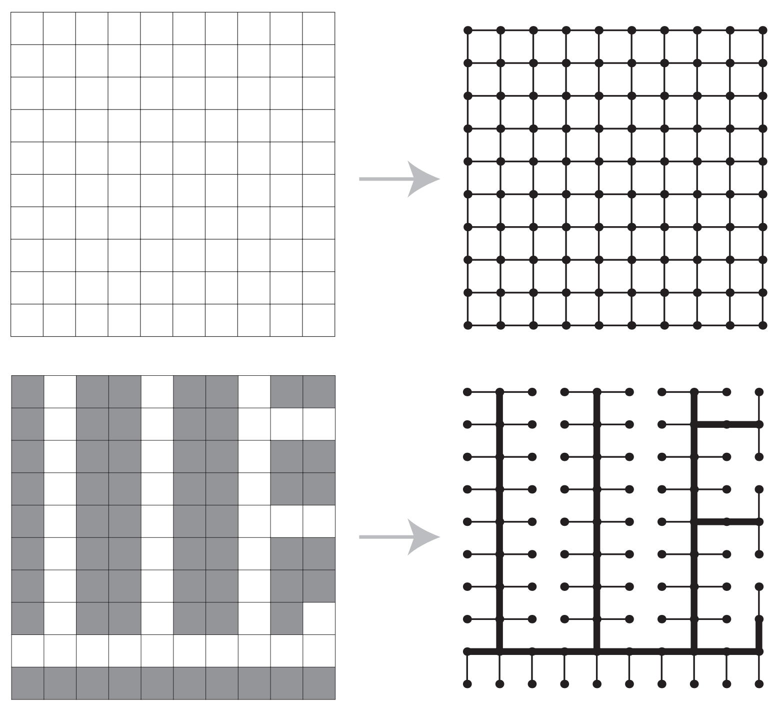


Figure 8: The single-deep VHD storage problem represented as a Maximum Leaf Spanning Tree Problem (MLSTP), here on a 10×10 grid graph. Any storage grid can be represented as a grid graph; a solution to the single-deep VHD problem is a solution to MLSTP.

m	n	OPT	FR1	FR2	m	n	OPT	FR1	FR2
3	3	6	6	6	5	7	20	20	20
3	4	8	8	7	5	8	23	23	23
3	5	10	10	9	5	9	27	27	26
3	6	12	12	11	6	6	22	22	22
3	7	14	14	13	6	7	26	25	26
3	8	16	16	15	6	8	30	28	30
3	9	18	18	17	6	9	34	33	34
4	4	9	9	9	7	7	29	29	29
4	5	11	11	11	7	8	33	33	33
4	6	14	14	14	7	9	39	39	38
4	7	16	16	16	8	8	38	38	38
4	8	18	18	18	8	9	45	45	43
4	9	21	21	21	9	9	51	51	51
5	5	14	14	14	10	10	61	61	61
5	6	18	18	17					

Table 1: Performance of Fill-and-Rotate when $k = 1$. Values indicate the number of items stored in a grid using each method: OPT values are from Fujie (2003, 2004); FR1 and FR2 values are from Fill-and-Rotate applied to an $m \times n$ and $n \times m$ grid, respectively.

Fujie (2003) applies his branch and bound algorithm to several instances of grid graphs, allowing us to compare the performance of Fill-and-Rotate with optimal solutions when $k = 1$ (see Table 1). If we apply Fill-and-Rotate to both orientations of the grid ($m \times n$, $n \times m$) and choose the more dense solution, we achieve the optimal solution for all 29 cases. Although these results are far from a proof of optimality, they do suggest that Fill-and-Rotate performs well, at least on small problems. Also, Fill-and-Rotate is extremely fast (solutions take less than a second), while Fujie's algorithm is slow even for medium-sized grids, as we would expect from a branch-and-bound technique. In personal correspondence, he indicated that it took three days to solve the 10×10 case.

Table 1 shows that sometimes it is better to apply Fill-and-Rotate to the $m \times n$ grid (long side down) and sometimes to the $n \times m$ grid (short side down). In the table, it is best to use long-side down except for 6×7, 6×8, and 6×9 grids, when it is best to begin with the short side down. Is there a general rule? To answer this question we compared short- and

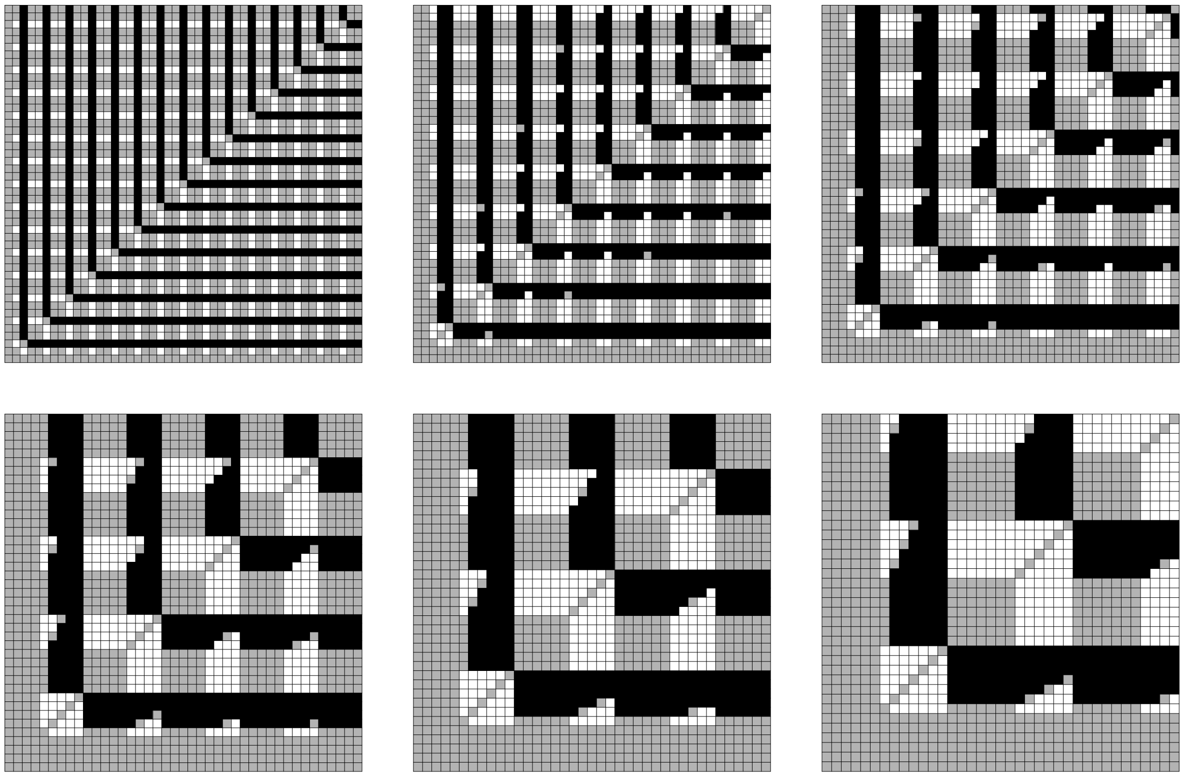


Figure 9: Plots showing which beginning orientation of the grid is best. Black cells indicate short-side down is best; white cells indicate long-side down is best; gray cells indicate both orientations produce the same density. Cells correspond to configurations as in Figure 5.

long-side down solutions for many grids and values of k (see Figure 9). In the figure, black cells represent grids for which short-side down produces a more dense layout; white cells those for which long-side down is superior; and gray those for which the layouts are equally dense. Cells in these plots correspond to grid sizes, as in Figure 5. The figure suggests that short side down is best when m is close or equal to, but does not exceed, a multiple of $2k+1$. When both m and n are multiples of $2k+1$, as our results would recommend to the designer, configurations with short-side down are superior.

2.3 Bounds

We have been unable to prove or disprove the optimality of Fill-and-Rotate, but we can show a performance bound. First, we give an upper bound on any layout:

Theorem 1 *The density of any k-deep layout is less than or equal to $\frac{2k}{2k+1}$.*

Proof See Appendix B. $\square$

Data points in Figure 6 conform to this result. Points in the $k = 1$ group have densities "converging" to $2(1)/(2(1) + 1) = 2/3$. The next group of points converges to $2(2)/(2(2) + 1) = 4/5$, and so on.

We can show that for some configurations, the bound in Theorem 1 is tight in the limit:

Corollary 1 *Let D_{mnk} be the density of an $\{m, n, k\}$ configuration produced by* Fill-and-Rotate. *If m is an integer multiple of $2k + 1$,*

$$\lim_{n\to\infty} D_{mnk} = \frac{2k}{2k+1}.$$

Proof Imagine any grid with $m = b(2k + 1)$ and $n > m$, where b is a positive integer. Fill-and-Rotate (applied with initial grid orientation short-side down) produces a single m-long vertical aisle and b horizontal aisles of length $(n - k - 1)$, forming a configuration with density

$$\frac{2nbk - bk}{nb(2k + 1)} = \frac{k(2n - 1)}{(2k + 1)n} \overset{n\to\infty}{=} \frac{2k}{2k + 1}.$$

$\square$

We can now show that

Theorem 2 *The density of a k-deep layout produced by algorithm* Fill-and-Rotate *is within $\frac{1}{n} + \frac{1}{r}$ of optimal, where $r = \min\{m, n\}$.*

Proof See Appendix B. $\square$

Corollary 2 *The density of layouts produced by algorithm* Fill-and-Rotate *converges to the maximum possible as* $m, n \to \infty$.

The proof follows directly from Theorem 2. Whether or not Fill-and-Rotate is optimal we must leave as an open question.

3 Conclusions

The best layouts for VHD storage spaces have a definite structure: items should be arranged in several aisles connected by a single cross aisle. For certain combinations of length (n), width (m), and lane depth (k) it is best to arrange a slightly different region on one side to compensate for the lack of divisibility between the parameters.

When possible, rectangular storage spaces should be designed such that the length and width are equal to or slightly less than multiples of $2k+1$. These dimensions tend toward the highest storage densities because it is easy to construct aisles with k-deep storage sections on either side. We showed that long, narrow storage spaces with the narrow dimension equal to a small multiple of $2k+1$ are particularly dense.

We showed that the storage density in a space can be no greater than $2k/(2k+1)$, and that this bound is asymptotically tight as the storage space gets large. Engineers might use this as a rule of thumb when designing storage spaces that will be faced with the need for high density. For example, designers of the Navy's new sea based warehousing ships can compute a simple upper bound on the number of pallets or vehicles that can be stored in a space of given dimensions. Such insight at the design stage could shape the allocation of space between storage and other operations onboard the ship. (We have encountered program managers within the Department of Defense using "stowage factors" — storage densities — that were, in fact, infeasible according to our results.) Moreover, the result has a natural application to warehousing systems: the ubiquitous single-deep pallet rack can provide no greater storage density than 2/3, and double-deep pallet rack no greater than 4/5.

This simple result can be used in other contexts as well: For many parking lots the width of an aisle is approximately the length of a parking space, making the $k = 1$ result (maximum density: 2/3) a rough cut upper bound on the fraction of the area that could be devoted to parking spaces. A similar rule of thumb might be developed for long-term storage yards for shipping containers.

Our overriding goal is to understand the interaction of storage density and throughput for very high density systems. In this work we have constrained ourselves to questions of layout and dimensions for VHD systems. Still to consider are the throughput characteristics for these configurations. For example, in a 2-deep system, which is better from a throughput perspective: a 10×10 configuration, or a 5×20?

A Algorithm

Algorithm 1 Fill-and-Rotate

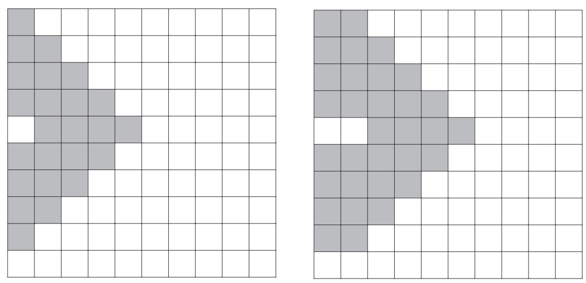
Require: An $m \times n$ grid and a density coefficient k.
Ensure: The grid is oriented with the n-long axis at the bottom.

Assign k rows of items on the bottom, and mark an aisle; rotate the grid counter-clockwise; $\ell_1 \leftarrow n$.
Assign k rows of items on the bottom, and mark an aisle; $\ell_1 \leftarrow \ell_1 - (k+1)$.
while $\ell_1 \geq 2k+1$ **do**
  Assign $2k$ rows and mark an aisle; $\ell_1 \leftarrow \ell_1 - (2k+1)$
end while
if $\ell_1 \leq k$ **then**
  Assign ℓ_1 rows and STOP.
else $\{k < \ell_1 \leq 2k\}$
  Rotate the grid counter-clockwise; $\ell_2 \leftarrow m - (k+1)$
  Assign k rows; $\ell_2 \leftarrow \ell_2 - k$
  while $\ell_2 > 2k+1$ **do**
    Mark and aisle and assign $2k$ rows; $\ell_2 \leftarrow \ell_2 - (2k+1)$
  end while
  if $\ell_2 > k$ **then**
    Mark an aisle, assign $\ell_2 - 1$ rows, and STOP.
  else $\{\ell_2 \leq k\}$
    Rotate the grid clockwise.
    Assign k rows of width ℓ_2 and mark and aisle.
    if $\ell_2 > 1$ **then**
      Assign $\ell_2 - 1$ rows of items and mark an aisle, STOP.
    end if
  end if
end if

B Proofs

Proof of Theorem 1. We say that an aisle cell *covers* another cell if an item in that cell may be retrieved from the aisle cell; that is, there are no more than $k-1$ cells between the two. (For the purposes of the proof, we allow the most general case that items may move to an empty cell in any direction. This may or may not be possible from a mechanical design perspective.) In any feasible solution, every storage cell must be covered by at least one aisle cell. Suppose that the complete aisle network contains A cells. It suffices to show that the total number of items covered is less than $2kA$.

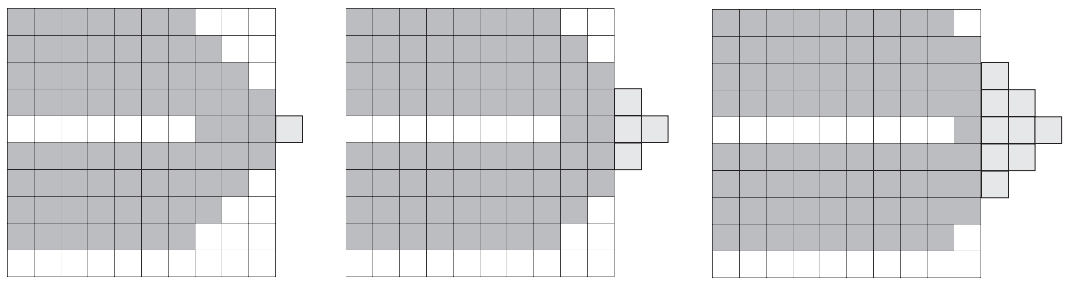
Now imagine any solution to the VHD storage problem, and consider the I/O point, which is an aisle cell on the boundary. The I/O cell has one access point (edge) occupied by the boundary, and at most three others that can provide coverage. The two access points adjacent to the boundary provide at most k coverage each, and the access point opposite the boundary provides an additional $\sum_{i=1}^{k} 2k-1 = k^2$ coverage for a total of k^2+2k items, as shown in the left figure below for $k=4$ (a different accounting of which access point covers which item is possible). Beginning from the I/O point (which every aisle network must



contain), it is possible to "grow" the entire aisle network by connecting aisle cells to the existing network. Note that each additional aisle cell adds at most $2k$ to the total coverage, as shown in the right figure above.

An alternative statement of our problem is to grow the aisle network such that the covered area equals or exceeds the required $m \times n$ grid. Notice that growing the network in certain ways results in fewer than $2k$ additional items covered. We say that such an incremental growth incurs a *loss* equal to the difference (e.g., an aisle cell that adds only $2k-2$ to the total coverage incurs a loss of 2). The figure below illustrates how losses occur when the aisle approaches a boundary. Because the I/O cell has a total coverage of $2k+k^2$, and each additional aisle cell adds no more than $2k$ coverage, it suffices to show that any feasible solution must incur at least k^2 in losses.

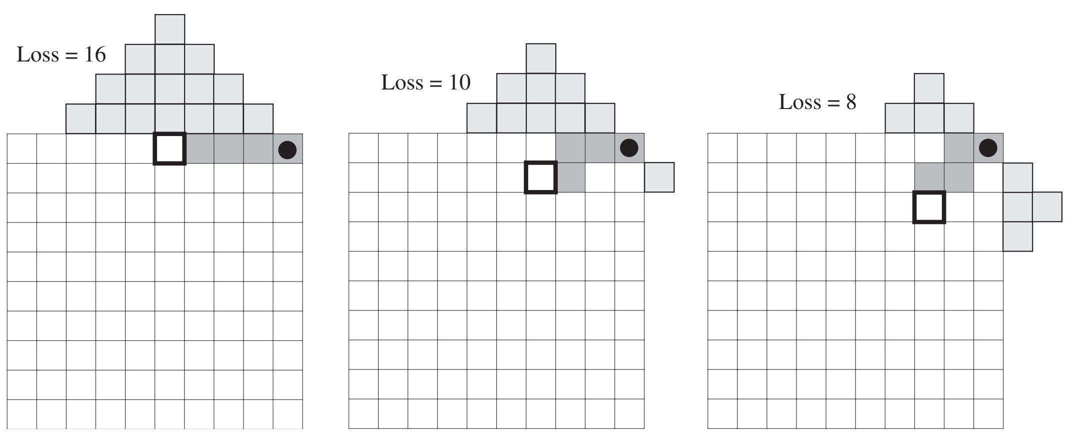
Losses occur in at least three ways: when the network contains an L or a T, and when an aisle cell has an access point in the direction of, and is closer than k cells to, a boundary.



In the latter case, some of the "covered cells" are outside the grid.

When growing the aisle network, the I/O cell covers at most one corner cell in the grid because $m, n > 2k + 1$. Thus at least three corners are uncovered. We will show that to cover another corner cell incurs a loss of at least $\lceil k^2/2 \rceil$, and so the total loss is at least k^2, completing the proof.

The figure below illustrates the task of covering a corner cell. For $k = 4$, we see that



one of the indicated cells (in bold outline) must be an aisle cell. It is easy to see that the minimum loss occurs when the cell nearest the diagonal is the aisle cell, and that an aisle cell in that position incurs a loss of $k^2/2$ for k even, and $(k^2 + 1)/2$ for k odd. Therefore the minimum loss to cover a single corner is $\lceil k^2/2 \rceil$. Applying this result to opposite corners (so there is no overlapping of losses) produces a total loss of at least k^2. □

Proof of Theorem 2. Begin by establishing an upper bound on the number of aisle spaces A in an $m \times n$ grid, where m refers to the number of rows, and n to the number of columns. Every layout contains an n-long aisle as part of Step 1 of the algorithm. Steps 2–5 create vertical aisles: by inspection we see that there are $\lfloor n/(2k+1) \rfloor$ vertical aisles, each having length $m-k-1$, giving $n+(m-k-1)\lfloor n/(2k+1) \rfloor$ aisle cells through Step 5.

The rest of the algorithm adds at most $m-k-1$ additional aisle cells. This can be derived algebraically from the algorithm, and seen visually in Figure 4. Therefore, the algorithm gives a layout with

$$A \leq n + (m-k-1)\left(\left\lfloor \frac{n}{2k+1} \right\rfloor + 1\right).$$

Now we can say that for an $m \times n$ grid,

$$\begin{aligned}
\text{Density} &= \frac{mn-A}{mn} \\
&\geq \frac{mn - \left(n + (m-k-1)\left(\left\lfloor \frac{n}{2k+1} \right\rfloor + 1\right)\right)}{mn} \\
&\geq \frac{mn - n - (m-k-1)\left(\frac{n}{2k+1} + 1\right)}{mn} \\
&= \frac{2kmn - kn - 2km - m + 2k^2 + 3k + 1}{mn(2k+1)} \\
&\geq \frac{2kmn - 2km - kn - m}{mn(2k+1)} \\
&\geq \frac{2kmn - 2km - kn - km}{mn(2k+1)}.
\end{aligned}$$

If $m \leq n$, then

$$\frac{2kmn - 2km - kn - km}{mn(2k+1)} \geq \frac{2kmn - 2km - 2kn}{mn(2k+1)} = \frac{2k}{2k+1}\left(1 - \frac{1}{n} - \frac{1}{m}\right);$$

if $m \geq n$,

$$\frac{2kmn - 2km - kn - km}{mn(2k+1)} \geq \frac{2kmn - 4km}{mn(2k+1)} = \frac{2k}{2k+1}\left(1 - \frac{2}{n}\right).$$

Let $r = \min\{m, n\}$ and note Theorem 1 to complete the proof. □

Acknowledgements

The author thanks the Office of Naval Research for supporting this work. John Bartholdi and Don Eisenstein offered helpful comments, and Samir Amiouny produced an example layout (for which he was awarded the handsome sum of $100) that led to a refinement of the Fill-and-Rotate algorithm. Joe Skufca provided significant insights into the proof of Theorem 1.

References

Alzoubi, K. M., Wan, P.-J., and Frieder, O. (2002). Distributed heuristics for connected dominating sets in wireless ad hoc networks. *Journal of Communications and Networks*, 4(1):1–8.

Askin, R. G. and Standridge, C. R. (1993). *Modeling and Analysis of Manufacturing Systems*. John Wiley & Sons, New York, New York.

Bassan, Y., Roll, Y., and Rosenblatt, M. J. (1980). Internal Layout Design of a Warehouse. *IIE Transactions*, 12(4):317–322.

de Castilho, B. and Daganzo, C. F. (1993). Handling strategies for import containers at marine terminals. *Transportation Research*, 27B(2):151–166.

Duckworth, W. (2002). Minimum connected dominating sets of random cubic graphs. *The Electronic Journal of Combinatorics*, 9:1–13.

Frazelle, E. H. (2002). *World-Class Warehousing and Material Handling*. McGraw Hill, New York, New York.

Fujie, T. (2003). An exact algorithm for the maximum leaf spanning tree problem. *Computers & Operations Research*, 30(13):1931–1944.

Fujie, T. (2004). Private correspondence with the author.

Galbiati, G., Maffioli, F., and Morzenti, A. (1994). A short note on the approximability of the maximum leaves spanning tree problem. *Information Processing Letters*, 52:45–49.

Garey, M. R. and Johnson, D. S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman and Company, New York.

Gue, K. R. (2004). Estimating Throughput in Very High Density Storage Systems. Working paper, Auburn University.

Heragu, S. (1997). *Facilities Design.* PWS Publishing Company, Boston, Massachusetts, first edition.

Iranpour, R. and Tung, D. (1989). Methodology for optimal design of a parking lot. *Journal of Transportation Engineering*, 115(2):139–160.

Kim, K. H. and Kim, H. B. (2002). The optimal sizing of the storage space and handling facilities for import containers. *Transportation Research Part B*, 36:821–835.

Kim, K. H. and Park, K. T. (2003). A note on a dynamic space-allocation method for outbound containers. *European Journal of Operational Research*, 148:92–101.

Lu, H. and Ravi, R. (1998). Approximating maximum leaf spanning trees in almost linear time. *Journal of Algorithms*, 29:132–141.

Solis-Oba, S. (1998). 2-approximation algorithm for finding a spanning tree with maximum number of leaves. *Lecture notes in computer science*, 1461:441–452.

Taleb-Ibrahimi, M., de Castilho, B., and Daganzo, C. F. (1993). Storage space vs handling work in container terminals. *Transportation Research*, 27B(1):13–32.

Wehrfritz, G. and Itoi, K. (2003). Subterranean City. *Newsweek*.