Part III

Order-picking

Order-picking is the most important process in most warehouses because it consumes the most labor and it determines the level of service experienced by the downstream customers.

In low-volume distribution, such as of service parts, customer orders will be small, urgent, and different from each other. Consequently, order-pickers may travel long distances for each pick; and their paths through the warehouse may be quite different, even from order-to-order. In such an environment, the challenge is to reduce travel by finding an efficient route visiting the required locations (Chapter 10).

In high-volume distribution, such as the supply of retail stores, customer orders will typically be large and may be similar. Each order-picker is likely to make many picks per unit of distance traveled; and all order-pickers are likely to follow a common path, such as along an aisle of flow rack. The challenge in such order-picking is to keep the work flowing smoothly by eliminating bottlenecks (Chapter 11).

Chapter 10

Pick-paths

Travel time to retrieve an order is a direct expense. In fact, it is the largest component of labor in a typical distribution center. Furthermore, travel time is waste: It costs labor hours but does not add value. Travel time matters also because it affects customer service. The faster an order can be retrieved, the sooner it is available for shipping to the customer.

Consider McMaster-Carr, a distributor of hardware and tools. They distribute over 450,000 skus from four North American distribution centers. Fast service is very important to the customer, who might have large capital equipment or a construction project waiting for the part or tool. Therefore McMaster-Carr begins picking the order almost immediately on receipt, even though it might be for only 2–4 skus, representing only a few locations from among many. In such a case the problem changes from one of balancing flow, as in Chapter 11 to one of sequencing the locations to be visited so that total travel is small. Such a problem must be solved for each trip an order picker must make into the warehouse, because, unlike the fast-pick area, where the general path of travel is common and known in advance, in this case each trip into the warehouse may follow a different path.

10.1 The problem of pick-path optimization

The problem of visiting a given set of locations as quickly as possible has been nicknamed the "Traveling Salesman Problem" (TSP) and has been much studied [27]. In general, the TSP is difficult in several senses:

- There is no known fast solution technique that works in general.
- Randomly-generated instances, even small ones, can be surprisingly time-consuming to solve.
- Optimum, or even good solutions can be complex and hard to describe.

Order-retrieval in a warehouse presents a special case of the TSP in which travel is constrained by aisles and this special structure makes it possible to find optimal solutions quickly by computer [34, 18, 35]. However, despite marketing claims, most warehouse management systems do not support pick-path optimization. There are several reasons for this. The most important is that any optimum-finding algorithm must know the geometric layout of the warehouse, including distances between all pairs of storage locations; and most WMS's do not maintain this level of information. Such detailed information would not only be time-consuming to gather but would have to be specialized to every site and updated after any change in physical layout.

Finally, even if the WMS does support some kind of pick-path awareness, there remains the problem of communicating the path to the picker. A high-quality path is not useful if the order picker does not follow it. Typically the WMS tells the picker only the sequence of locations, not the actual path to follow. The picker must figure out the shortest path from location to location; and this can be hard to do because order pickers work under pressure and with only local information. Figure 10.1 shows the difficulty.

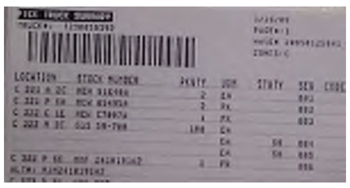
Incidentally, in this regard it can be more effective to pick from a paper pick list than from an RF device. With paper, the order picker can see at a glance the next few locations to be visited and can use his knowledge of the warehouse to plan his path. On the other hand, a typical RF device displays only the very next location to be visited, which makes it impossible for the order picker to improve the picking sequence. This situation may change soon as advanced telecommunications enables the WMS to pass the order pickers actual maps of pick paths to be followed.

10.2 Heuristic methods of generating short pick paths

How can we generate short travel paths that are realizable by an order picker who has no detailed map of the warehouse?

Imagine that a picker must visit *all* the storage locations of a warehouse; and suppose further that we can find an efficient global path to visit all these locations. We have to compute this efficient path only once and then we can use it many times: When a picker travels to retrieve the items of an order, we require that he simply visit the required locations in the same sequence as does the efficient global path. Thus the global path imposes a sequence that will be respected by all travel. When we receive a customer order the WMS simply sorts the pick lines by storage location so that they appear in the same sequence as the efficient global path. The idea is that if the global path is efficient the sub-path induced on each customer order is likely to be efficient as well.

The problem of finding a good global path through the storage locations is known as the "Probabilistic Traveling Salesman Problem" or PTSP and there is a large literature []. For the PTSP problem the main issue is length of the

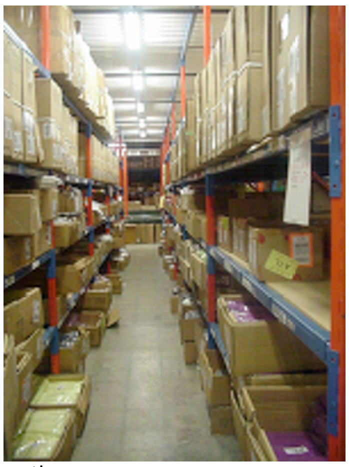


PICK TRUCK SUMMARY
TRUCK#: [illegible]

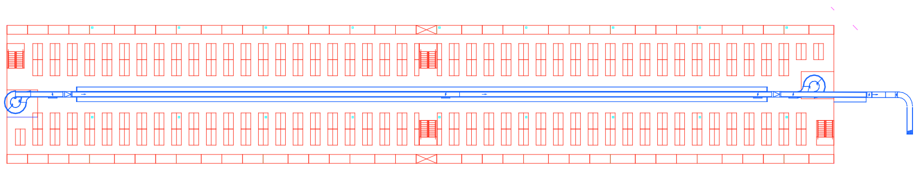
[illegible]
PAGE#: 1
[illegible]
[illegible]

LOCATION	STOCK NUMBER	QTY	UOM	STATY	SEQ	CODE
C 321 A 2C	[illegible]	2	EA		001	
C 321 P 5A	[illegible]	2	PK		002	
C 322 C 1E	[illegible]	4	PK		003	
C 322 A 3C	[illegible]	100	EA			
			EA	50	004	
			EA	50	005	
C 322 P 6E	[illegible]	1	PK		006	

ALTM: [illegible]



(a) Local information



(b) Global information

Figure 10.1: An order picker has only local information, such as a sequence of locations and the view from an aisle (a), from which to determine a pick path that is globally efficient. It is hard for the order picker to know or account for the global layout of the pick area (b).

induced sub-paths. Within the context of order-picking there are additional issues.

10.2.1 Path outlines

A good global path should not only induce short sub-paths on the customer orders, it should also help the picker visualize where the next location and how to travel there most directly. We want a path outline that will induce a short pick path for most orders and yet is simple in structure so that order-pickers can understand it. An effective path outline will account for the physical layout of rack, where the most popular items are stored, and what a typical order looks like. In addition, management may devise simple rules by which the path outline can be adapted for the particular customer orders. By providing the order-pickers with a set of rules to adapt the path, they leverage the intelligence of the work force, rather than embedding the decision-making in the WMS software.

The simplest path outline is that along a single aisle, as shown in Figure 10.2. Such a path outline has the desirable property that any optimal path is consistent with this ordering. This configuration is typically found in a fast-pick module of a distribution center.

A commonly found path outline through static shelving is the so-called *serpentine* pick path illustrated in Figure 10.3. The path induces 1-way direction of travel in each aisle, which means that it may be possible to conserve floor space by using narrow aisles. However, location sequence might not give optimal travel paths, as in this instance where the picker would have to travel needlessly along the lengths of aisles 3 and 6. Unless a typical customer order visits every aisle, such a path can result in wasted travel.

One way of ameliorating this problem is to specify only a partial ordering among the storage locations. For example, Figure 10.4 shows an incompletely-specified serpentine outline that sequences the locations of the first few aisles from the left but thereafter sequences only the aisles themselves and not locations within the aisles. This imposes less *a priori* structure on the eventual pick path and relies on the intelligence of the order picker to adapt it appropriately. This example could be an effective path outline if the early (leftmost) aisles contain the more popular products. In this case the picker can construct a more efficient pick path by skipping aisles 3 and 6.

Because this path outline cannot guarantee in advance in which direction the picker may travel the later (rightmost) aisles, the right-side of the warehouse must allow 2-way travel for passing and so may need wide aisles.

Another common type of pick path is the *branch-and-pick*, which sequences only the aisles and not the locations within an aisle. The pick path passes a reduced set of locations (endcaps), which are where fastest-moving items should be stored (Figure 10.5) and the picker detours as necessary into the aisles. This is typically used when there are shallow aisles with some slow-moving product.

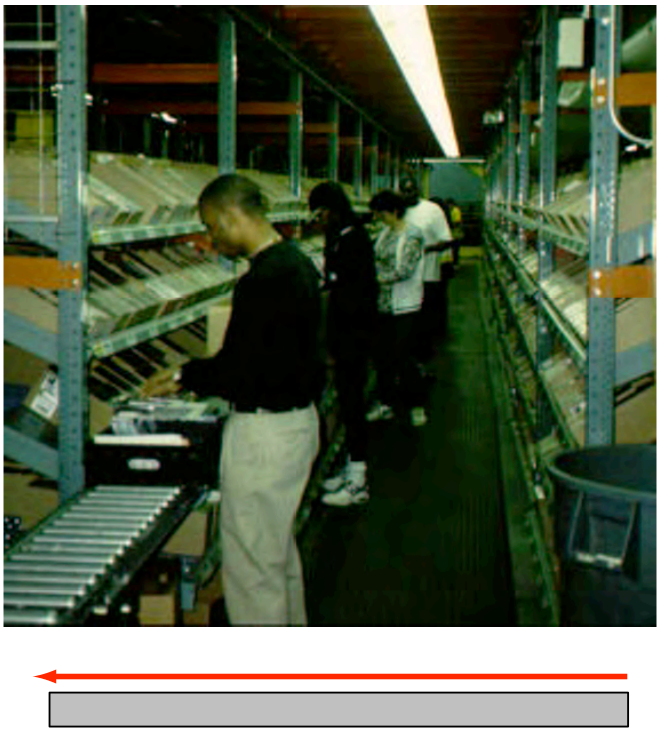


Figure 10.2: This picking area contains only popular skus and so every order is likely to require walking down the aisle. Thus an efficient route of travel is known in advance.

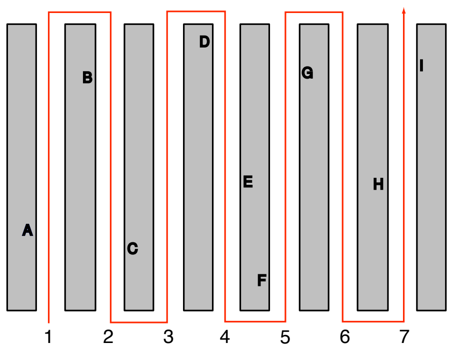


Figure 10.3: A serpentine pick path can result in unnecessary travel (in this case along aisles 3 and 6).

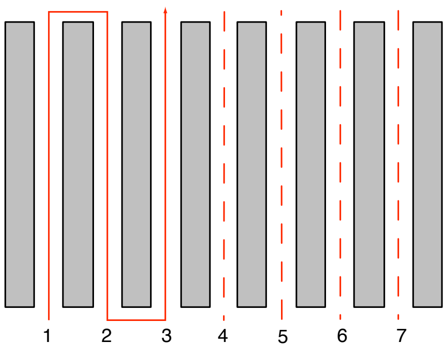


Figure 10.4: A modified path outline that sorts aisles 4–6 but not their locations.

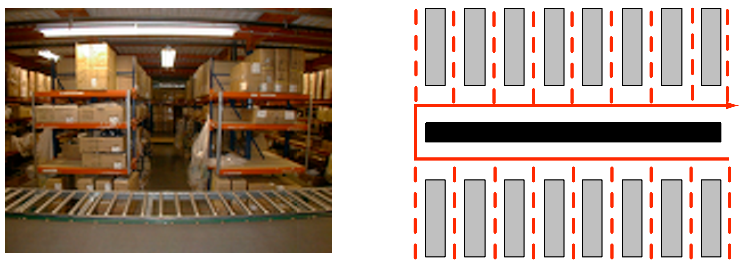


Figure 10.5: Example of branch-and-pick

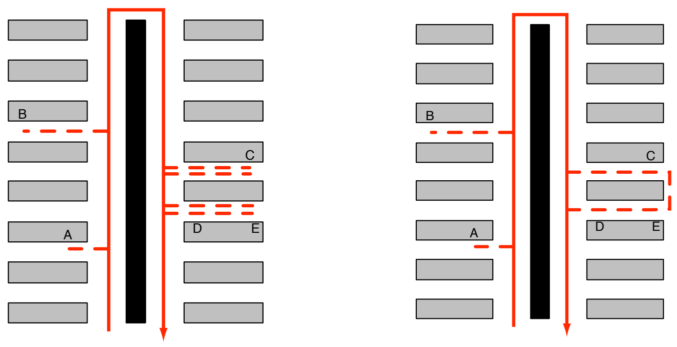


Figure 10.6: Left: Travel path generated by detouring into and back out of aisles. Right: Travel can be reduced by allowing picker to determine entry and exit from each aisle.

It can be useful to further rules with which a picker can generate a travel path from a pick list. For example, the simplest algorithm might be "Visit aisles in sequence; detour into and back out of each aisle as necessary to pick product".

10.2.2 Product placement

To help pickers guess the best way to travel, one must make the geometry and addresses work together. For example, if locations that are close also have similar addresses, and *vice versa*, then the pick list also tells picker something about *where* the next address is in the warehouse. The global (layout of the warehouse) is embedded in the local (list of addresses).

In addition, it is generally better to store the most popular skus close to the path outline, so that they can be reached with little or no detour, as shown in Figure 10.7. Fewer and shorter detours also means that the eventual travel path will be simpler, which means that the savings is more likely to be realized because the order-picker can follow it.

10.3 Pick-path optimization

To compute optimal pick paths requires that the computer know the shortest distance between any two locations in the warehouse (and the corresponding route of travel). As of this writing, no warehouse management systems that we know of manage an explicit geometric model of the layout of the warehouse. Therefore true pick-path optimization is not currently done.

Eventually warehouse management systems will have geometric models of their warehouses; and advances in telecommunications will make it easy and economical to give picker precise travel instructions. Then there will be no

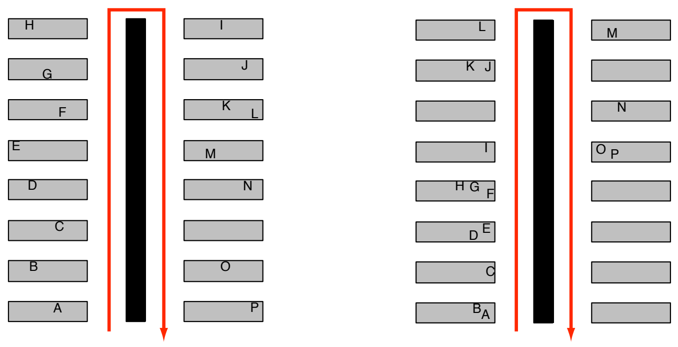


Figure 10.7: A much shorter travel path is possible if popular items are stored close to path outline, as on the right.

reason not to take advantage of pick-path optimization.

The fundamental result in pick-path optimization is due to Ratliff and Rosenthal [34], who gave an algorithm for quickly finding the shortest tour of a set of locations in a warehouse. We will illustrate their ideas by giving a simplified version of their algorithm, which will generate *near*-optimal pick paths. The simplification is to restrict slightly the allowable patterns of travel so that the picker is forbidden to revisit a previously-visited aisle. In other words, we will find the shortest pick path subject to the constraint that the aisles cannot be visited out of their natural sequence.

Because of this restriction the suggested path may be slightly longer than the unconstrained optimum; but

- This algorithm is in the same spirit as the optimum-finding algorithm, but is much simpler and so is easier to program and to explain.
- Any unnecessary travel required because of the no-backtracking restriction is generally small.
- The generated path is simpler in structure than an optimum path and so easier for an order picker to understand and follow.

Following [34] we generate a pick-path by dynamic programming. This takes advantage of the fact that an optimum path can assume only a limited number of patterns at each end of an aisle. Our restriction that the picker can never revisit an aisle reduces the number of possible patterns to only two so that each order-picker can be imagined to follow this rule.

> Pick all the required items from the current aisle and travel forward to the next aisle.

Note that there are two ways of traveling to the next aisle: An order-picker can either

- Travel all the way through the current aisle, picking as necessary; or else
- Enter the aisle only as far as necessary to pick all required items therein and then return to the same end of the aisle from which it was entered.

The decision points are at the ends of each aisle, as suggested in Figure 10.8. Because there is no backtracking the picker must travel left to right across the

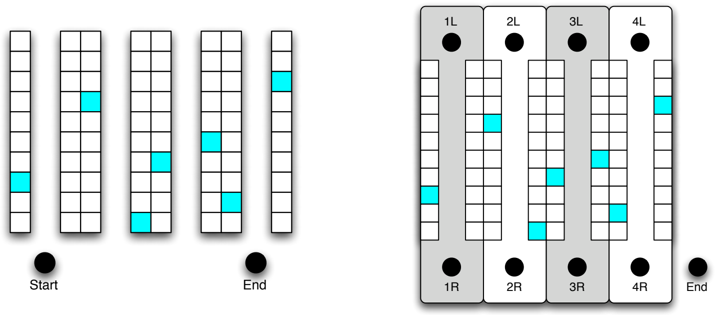


Figure 10.8: To visit the locations on the left, imagine a decision point at the end of each aisle. Because backtracking is forbidden, the order-picker must visit aisle i before visiting aisle $i + 1$.

warehouse and so we can consider sequentially the decisions to be made at each aisle. We can graphically summarize the sequence of decisions to be made as in the series of figures beginning with Figure 10.9.

Figure 10.13 shows the final graph summarizing the sequence of decisions to be made by the order picker. The shortest path in this graph corresponds to an efficient pick path through the warehouse. (The shortest path can be found by elementary algorithms from graph theory or discrete mathematics [1].)

Notice that the connections of the network need be built only once; and for subsequent use only the lengths of the edges need be updated, as shown in Figure 10.14.

This approach can be extended naturally to handle the addition of cross aisles, although the work to solve increases quickly [34].

10.3.1 How to take advantage of optimization

For the effort of optimization to be worthwhile, it must be easy to implement a computed solution. But it can be hard for an order picker to know how to travel in the shortest way if they are told only the sequence of locations and not the paths from location to location. In fact, the shorter the pick path, the less likely it is to make sense to the order picker, who has only local information.

Nevertheless there are some simple ways to help the picker follow optimized pick paths. For example, the WMS could include instructions giving the direction the order-picker should travel after making each pick. Figure 10.15 shows

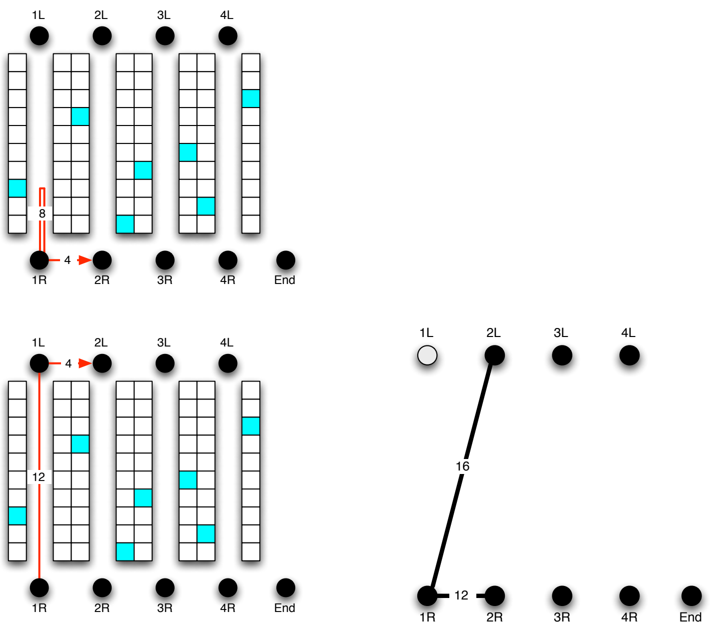


Figure 10.9: Enumeration and summary of paths from Aisle 1 to Aisle 2. Each candidate path is represented by an edge of the same length in the graph.

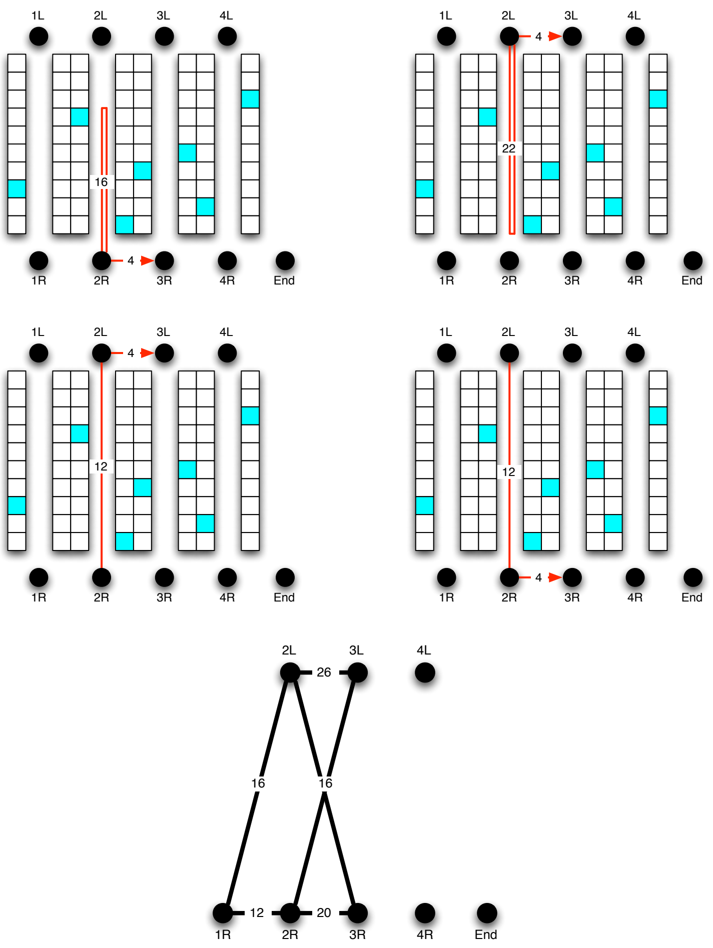


Figure 10.10: Enumeration and summary of paths from Aisle 2 to Aisle 3

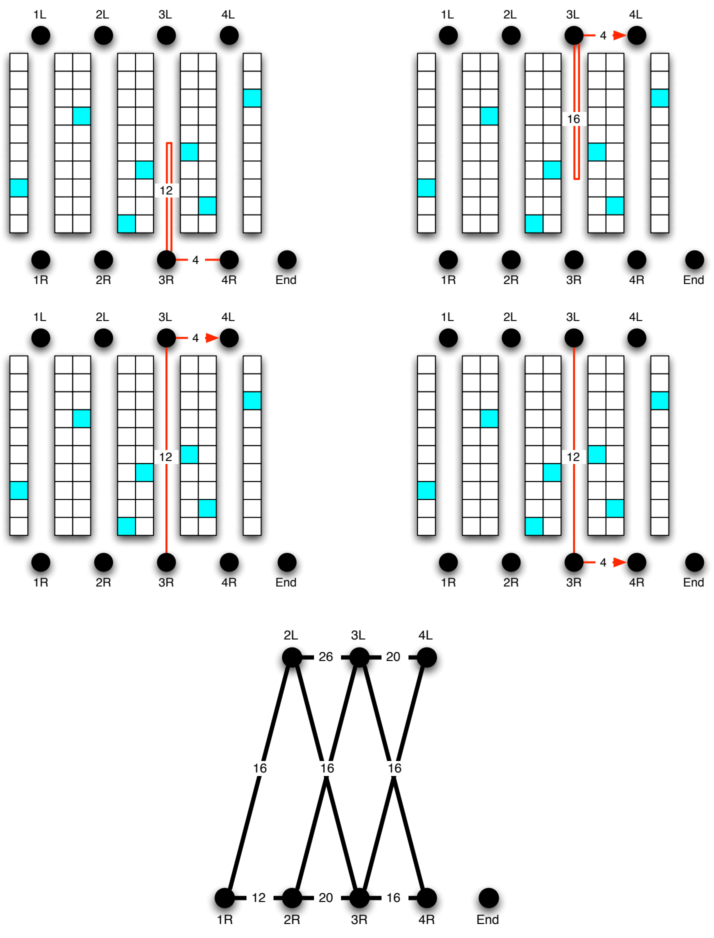


Figure 10.11: Enumeration and summary of paths from Aisle 3 to Aisle 4

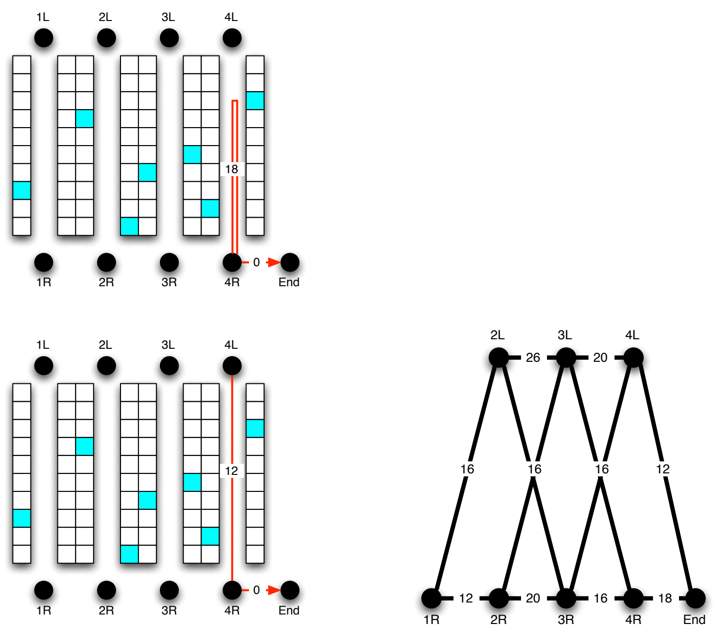


Figure 10.12: Enumeration and summary of paths from Aisle 4 to completion.

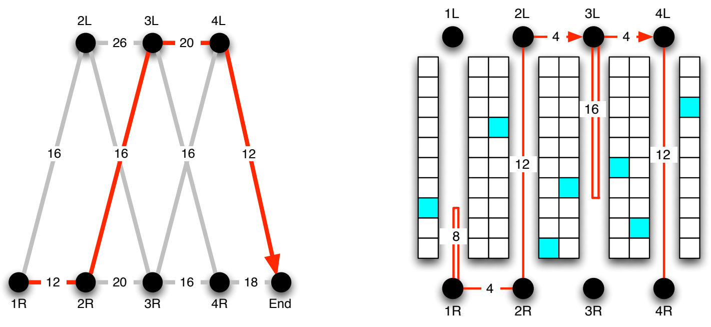


Figure 10.13: The shortest path on the associated graph gives an efficient pick path in warehouse

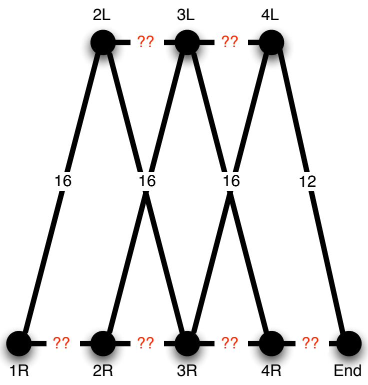


Figure 10.14: Only the lengths of the edges along each side of the graph need be updated to reflect new customer orders.

Aisle	Bay	Shelf	Sku	Qty	Continue in direction. . .
2A	5	1	ABC324	1	- away from conveyor
2B	8	2	CBD286	1	- toward conveyor
2B	1	2	DAB332	3	- toward conveyor
3A	4	1	ACF119	2	- away from conveyor
⋮	⋮	⋮	⋮	⋮	

Figure 10.15: A pick list with travel directions suitable for use on a branch-and-pick pick-path outline

an example pick list that would give sufficient information to specify routes, not just sequences of picks, for a pick path that followed a branch-and-pick outline (no backtracking to a previously passed aisle). Of course the effectiveness of this depends on how disciplined the pickers are. There is a trade off here: The more complex paths allowed, the more information that must be passed to the order pickers.

10.3.2 How much is optimization worth?

Finally, it must be asked how much pick-path optimization matters. The answer is that it depends. There are some situations in which pick path improvement does not matter much at all. Both are extreme cases: First, if the locations of the most popular items are concentrated in a small area then the length of any reasonable pick path could not be much longer than the shortest possible pick path. And if each order requires visits to very many locations then the optimal pick path must traverse most aisles and could not be much shorter than any reasonable pick path. This means that one can reduce any need for pick-path improvement by batching orders or by placing product so that the most frequently requested items are stored near each other.

On the other hand, the warehouses that are most likely to benefit from pick-path optimization are those that have many items, most of which are slow-moving. Examples include warehouses distributing hardware, building supplies or aftermarket auto parts.

10.4 Summary

Despite advertising claims, most WMS's do not support pick-path optimization. Instead they simply sort each pick list according to storage address (path outline). To make this method work best:

- Define path outlines that will generate short, understandable routes.
- Give pickers local rules to help them adapt the path outline.
- Place product to work with the path outline.

Pick-path optimization is in principle doable now but is rarely implemented for several reasons.

- The WMS must have a geometric model of the warehouse layout to compute shortest paths between pairs of locations.
- Limited communications bandwidth makes possible to tell picker where to go next but not the route by which to travel there.
- It does not always generate significant savings.

10.5 Questions

Question 10.1 *Consider the problem of finding the shortest pick path through a warehouse with parallel aisles but no cross aisle. Give an example of when a shorter path is possible if backtracking is allowed.*

Question 10.2 *Devise a worst-case example to show how much longer a picker might be required to travel if backtracking is forbidden in a warehouse with parallel aisles (no cross aisle).*

Question 10.3 *The word "detour" generally connotes the requirement to go out of your way. In what sense is it a detour to enter a side aisle to pick an item as part of a larger order in a branch-and-pick system?*

Question 10.4 *In which of the following scenarios might pick-path optimization be economically justified?*

- *A very busy unit-load warehouse*
- *A warehouse in which orders arrive intermittently and each is picked immediately (no batching). A typical order is for 1–2 skus.*
- *A distributor of recorded music, with only a very few, very popular skus.*
- *A warehouse that does most of its picking from a few single-aisle pick modules onto conveyor.*
- *All of the above*
- *None of the above*

Question 10.5 *Which of the following best describes the effect of adding crossover aisles to a warehouse? (That is, aisles that run orthogonally across the main direction of aisles.)*

- *It reduces travel because it creates shortcuts.*
- *It complicates travel-planning because it creates more possible paths.*
- *It reduces storage capacity by taking aisle space.*
- *None of the above*
- *All of the above*

Question 10.6 *To pick a customer order requires that the order-picker visit the set of locations shown in Figure 10.16. Label the corresponding network with the appropriate distances so that solving the shortest path problem on the network generates a shortest tour visiting all the locations.*

.

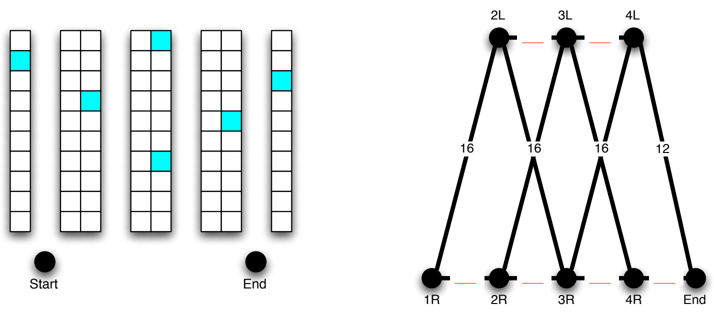


Figure 10.16: An order-picker must start at the leftmost dot, visit the shaded locations by traveling the aisles of the warehouse, and finish at the right-most dot.

Question 10.7 (Harder) *Generalize the algorithm to find shortest pick-paths when there are no restrictions on travel (other than the requirement to follow aisles; in particular, backtracking is allowed).*

Question 10.8 (Exploration) *Generalize the algorithm to find shortest pick-paths to account for a single "cross-aisle" running horizontally through the middle of the warehouse. How does worst-case computational effort depend on the number of cross-aisles?*

Chapter 11

Flow and balance: Piece-picking by "bucket brigade"

Self-organizing systems do not require a centralized authority to manage them. Instead, they achieve global coördination spontaneously through the interaction of many simple components.

When workers are organized into "bucket brigades" they can function as a self-organizing system that spontaneously achieves its own optimum configuration without conscious intention of the workers, without guidance from management, without any model of work content, indeed without any data at all. The system in effect acts as its own computer.

11.1 Self-organization

A self-organizing system is one in which global organization spontaneously evolves from myriad local interactions of the pieces. Here is an example: Consider a hive of honeybees. Each day they face a logistics problem of how to coördinate their efforts to harvest nectar. The measure of success is a social one: the good of the colony. But bees have no blueprint, no mechanism of central planning. Instead, each bee follows a simple "algorithm" that determines what she does next; and when many bees follow the same algorithm, an allocation of foragers evolves that is close to the best imaginable. In effect the colony acts as a computer that finds the (nearly) optimal allocation of effort [25].

Among the advantages of this self-organization are that:

- It requires no central planning or higher organizational entity. There is no management function because each entity simply follows a local rule.
- It is adaptive: It will spontaneously reallocate effort in response to changes

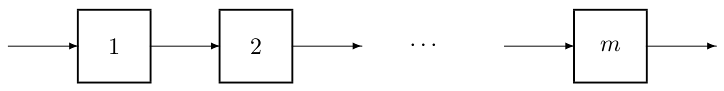


Figure 11.1: A simple flow line in which each item requires processing on the same sequence of work stations.

in the environment.

Exploring these simple ideas has led to some practical applications within management science/industrial engineering. Here is one in warehousing.

11.2 Order-assembly by bucket brigade

Bucket brigades are a way of coördinating workers who are progressively assembling product along a flow line in which there are fewer workers than stations (work stations in the context of manufacturing; storage locations in the context of order-picking). Each worker follows this simple rule: “Carry work forward, from station to station, until someone takes over your work; then go back for more”. When the last worker completes a product (or customer order), he walks back upstream and takes over the work of his predecessor, who then walks back and takes over the work of his predecessor, and so on, until the first worker begins a new product (customer order) at the start of the line. No unattended work-in-process is allowed in the system.

Note that workers are not restricted to any subset of stations; rather they are to carry each product as far toward completion as possible. Note also that a worker might catch up to his successor and be blocked from proceeding; the bucket brigade rule requires that the blocked worker remain idle until the station is available.

The final requirement of bucket brigades is that the workers be sequenced from slowest to fastest along the direction of material flow. These protocols, taken together, make the bucket brigade line a perfect *pull system*.

11.2.1 A model

Consider a flow line in which each of a set of items (customer orders) requires processing on the same sequence of m work stations (storage locations), as in Figure 11.1. A station can process at most one item at a time, and exactly one worker is required to accomplish the processing.

The Normative Model, suggested in [21], is given in the following assumptions. We call this model “normative” because it represents the ideal conditions for bucket brigades to work well. However, it is not necessary that these assumptions hold exactly: The behavior of a bucket brigade will resemble that predicted by the Normative Model to the degree that the assumptions of the Normative Model hold. Accordingly implementations should try to make these conditions hold as much as possible—but it is not necessary that they hold exactly, or even to any great extent.

The assumptions are:

Assumption 11.1 (Insignificant Walkback Time) *The total time to assemble a product is significantly greater than the total time for the workers to hand off their work and walk back to get more work.*

Assumption 11.2 (Total Ordering Of Workers By Velocity) *Each worker i can be characterized by a work velocity v_i.*

Assumption 11.3 (Smoothness, Predictability Of Work) *The work-content of the product is spread continuously and uniformly along the flow line (the length of which we normalize to 1).*

The assumption of Insignificant Walkback Time is uncontroversial; it claims simply that it takes longer to assemble a product than it does to walk the line; and, furthermore, it is easy to hand off work.

The assumption of Total Ordering Of Workers By Velocity is likely to hold in an mass-production environment, where work has been "Be-skilled" so that velocity is based on a single dimension, such as motivation or eye-hand coördination.

There is clearly some license in the assumption of Smoothness And Predictability Of Work; nevertheless, this assumption is reasonable in many instances, detailed elsewhere [21]. Suffice it to remind the reader that management and engineering strive to remove variance from work and eliminate bottlenecks, a result of which is to move practice closer to the Normative Model. Still, this assumption is at least less clear than the others and accounting for this is part of the art of implementing bucket brigades.

To what extent do the conclusions of the Normative Model hold when there is variation in the work-content? In short, the behavior of a bucket brigade remains qualitatively similar to behavior predicted by the Normative Model, with this caveat: the faithfulness of the replication depends on the degree of randomness. This means that, except in degenerate cases, it remains preferable to sequence the workers from slowest to fastest and one can expect a high production rate from bucket brigades.

Bartholdi and Eisenstein (1996a) have described the behavior of bucket brigade production lines under the Normative Model [21, 4]. Their main results, slightly simplified, are as follows.

Theorem 11.1 *No matter where a given set of workers start,*

- *There is a unique balanced partition of the effort wherein worker i performs the interval of work:*

$$\textit{from } \frac{\sum_{j=1}^{i-1} v_j}{\sum_{j=1}^{n} v_j} \textit{ to } \frac{\sum_{j=1}^{i} v_j}{\sum_{j=1}^{n} v_j}, \tag{11.1}$$

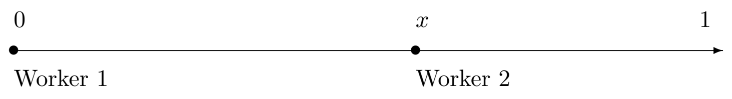


Figure 11.2: Positions of the worker 2 immediately after having completed the k-th order and walked back to take over the order of worker 1 (who has walked back to the start of the line to begin a new customer order).

so that each worker invests the same clock time in each item produced.

- *If the workers are sequenced from slowest to fastest then, during the normal operation of the line, work is spontaneously and constantly reallocated to reach this balance; and the production rate converges to*

$$\sum_{i=1}^{n} v_i \text{ items per unit time,}$$

  which is the maximum possible for the given set of workers.

- *If the workers are* not *sequenced from slowest to fastest, then the line will "sputter": that is, it will produce erratically and at suboptimal rate. Furthermore, the line can behave in counterintuitive ways, such as production rate* decreasing *when a worker increases his velocity.*

Before proving the result in general, we first argue that it is true for the case with two workers. Imagine that we are taking a series of photographs of the line at those times when the workers have just made their hand-offs and the first, slowest worker is beginning a new product. We will study how these photographs change.

Let x be the percent completion of the product held by worker 2 in the k-th photograph (that is, after a total of k items have been completed), as in Figure 11.2. Then the next order will be completed after an elapsed time of $t = (1 - x)/v_2$. During that time, worker 1 will have traveled forward distance $v_1 t$ and so the next hand-off will occur at position $(v_1/v_2)(1 - x)$. We can therefore summarize the changes in the locations of the hand-offs as the following dynamics function:

$$f(x) = (v_1/v_2)(1 - x).$$

This function is linear with negative slope, as illustrated in Figure 11.3. Furthermore—and importantly—because the workers have been sequenced from slower to faster, $v_1 < v_2$ and so $v_1/v_2 < 1$. Thus the slope of the dynamics function is of absolute value less than one and so is a *contraction map*, which means roughly that subsequent hand-offs get closer to each other [2]. Figure 11.3 traces a sequence of hand-offs at positions $x^{(0)}, x^{(1)} = f\left(x^0\right), x^{(2)} = f\left(x^{(1)}\right) = f\left(f\left(x^{(0)}\right)\right), \ldots$, which converges to the fixed point, the intersection of the dynamics function with the identity, where $f(x) = x$.

Here is a proof of convergence for two workers. Let $x^{(k)}$ denote the fraction of work completed on the k-th item as it is handed from worker 1 to worker 2.

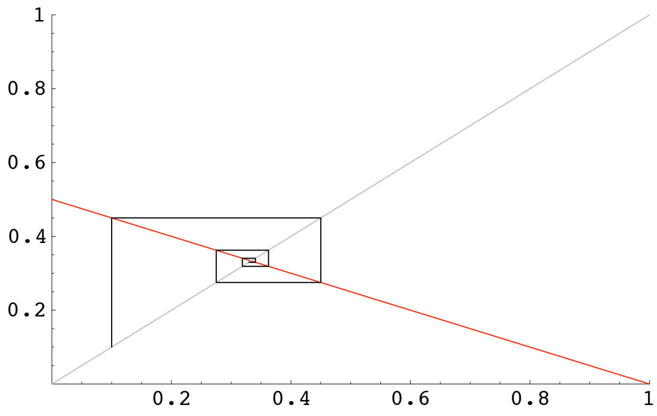


Figure 11.3: Because the slope is of absolute value less than 1, the dynamics function converges to a globally attracting fixed point, where $f(x) = x$. In other words, the assembly line balances itself.

Proof Let $r = v_1/v_2$ and note that

$$\begin{aligned} x^{(1)} &= r\left(1 - x^{(0)}\right); \\ x^{(2)} &= r\left(1 - x^{(1)}\right) \\ &= r\left(1 - r\left(1 - x^{(0)}\right)\right) \\ &= r - r^2 + r^2 x^{(0)}; \text{ and} \\ x^{(3)} &= r - r^2 + r^3 - r^3 x^{(0)}. \end{aligned}$$

By induction

$$x^{(k)} = r - r^2 + r^3 + \cdots + (-1)^k - (-1)^k x^{(0)}.$$

Because $r < 1$,

$$\lim_{k \leftarrow \infty} x_k = \frac{r}{1+r} = \frac{v_1}{v_1 + v_2}.$$

■

The more general proof, for n workers, is based on the same idea.

Proof As before, let $x_i^{(k)}$ be the percent completion of the order held by worker i immediately after completion of order k and having walked back to start the next order. (See Figure 11.4).

Then the clock time separating workers i and $i + 1$ is

$$t_i^{(k)} = \frac{x_{i+1}^{(k)} - x_i^{(k)}}{v_i};$$

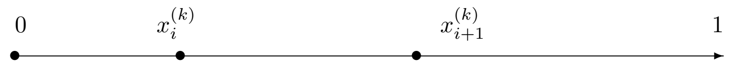


Figure 11.4: Positions of the workers after having completed k products.

and the next item will be completed after time

$$t_n^{(k)} = \frac{1 - x_n^{(k)}}{v_n}.$$

In the next, $k+1$-st photograph, the clock-time separating workers i and $i+1$ becomes

$$\begin{aligned} t_i^{(k+1)} &= \frac{x_{i+1}^{(k+1)} - x_i^{(k+1)}}{v_i} \\ &= \frac{\left(x_i^{(k)} + v_i t_n^{(k)}\right) - \left(x_{i-1}^{(k)} + v_{i-1} t_n^{(k)}\right)}{v_i} \\ &= \left(\frac{v_{i-1}}{v_i}\right) t_{i-1}^{(k)} + \left(1 - \frac{v_{i-1}}{v_i}\right) t_n^{(k)}. \end{aligned}$$

Because the workers are sequenced from slowest-to-fastest $(v_{i-1}/v_i) < 1$, and so we may interpret these equations as describing a finite state Markov Chain that is irreducible and aperiodic. By the Markov Chain Theorem the $t_i^{(k)}$ and therefore the $x_i^{(k)}$ converge; and the specific claims follow by simple algebra. ■

Figure 11.5 shows an example of how the movement of the workers stabilizes, with the faster workers eventually allocated more work. This figure was generated by a simulation of three workers of velocities $\mathbf{v} = (1, 2, 3)$.

11.2.2 Improvements that are not

It is tempting to try to improve the performance of bucket brigade lines by modifying the protocol; however, the variants that come first to mind actually perform *worse*. For example, an appealing but flawed variation of the bucket brigade protocol is to allow any worker, when blocked, to leave his partially-completed item in a buffer before the busy station and walk back to take over the work of his predecessor. This variant protocol will increase work-in-process inventory and can even *reduce* the production rate! This can be seen in simulations, where workers tend to collect in the region of the line preceding any station that is frequently busy. This increases the production rate of the preceding segment of the line, which only accelerates the accumulation of in-process inventory immediately preceding the highly-utilized station. This, in turn, decreases overall production rate of the line for two reasons:

- Fewer workers remain to staff the final segment of the line so each tends to assume a larger share of work and the time between product completions increases.

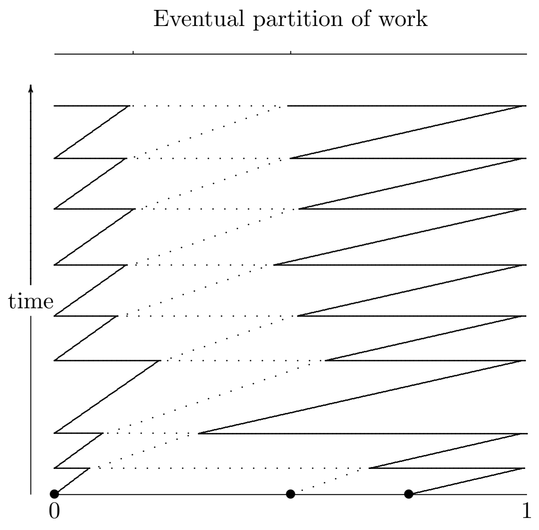


Figure 11.5: A time-expanded view of a bucket brigade production line with three workers sequenced from slowest to fastest. The solid horizontal line represents the total work content of the product and the solid circles represent the initial positions of the workers. The zigzag vertical lines show how these positions change over time and the rightmost spikes correspond to completed items. The system quickly stabilized so that each worker repeatedly executes the same portion of work content of the product.

- Because no one waits in front of the frequently busy station, it is idle every time a worker leaves it, which is contrary to the principal of keeping bottleneck stations constantly busy.

Eschewing buffers seems to contradict conventional wisdom that it is important to have buffers near a bottleneck—until one realizes that in bucket brigade production one must buffer both work-in-process *and* a worker, which is done by requiring the blocked worker to remain waiting at the bottleneck station.

One might also think that the bucket brigade protocol could be improved by requiring the workers to *circle* through the work stations. This avoids any delay in handing off work but it requires that every worker perform every task. There are several objections to be made to this. First, when real workers are finally assigned to the line they will not be of identical skill levels and so the production rate will eventually be determined by that of the slowest worker, behind whom all the others will accumulate. The production rate will remain suboptimal even if faster workers are allowed to preëmpt a slower worker and pass him: The slower worker would have to remain idle until his work station became free again and so the line could not keep all workers busy. Moreover, when workers are asked to perform every task on the line then the learning effect and so realized production rate will be reduced.

11.2.3 Some advantages of bucket brigades

As a way of coördinating workers on an assembly line, bucket brigades have many attractive properties, including:

- It is a pure pull system, so work-in-process inventory is strictly controlled.
- It requires no special material handling system because the workers themselves carry the items from station to station.
- Because the line can be made self-balancing, it does not require accurate measurement of task times and so can avoid some of the expense of time-motion studies.
- It is consistent with other trends in manufacturing: For example, it exploits the advantages of work teams and the grouping of technology into cells.
- The protocol is simple and identical for each worker: Workers are not confused as to what task to perform next and management need not intervene to keep work flow balanced and production rate high.

Bucket brigade manufacturing seems most appropriate when:

- *All the work is based on a single skill.* This ensures that workers can move among the stations to where the work is, without worrying about whether they can do the work there. It also allows workers to be ranked by a single score, their velocity along the production line, so that the line can be made self-balancing.

Economic forces ensure tend to move production lines in this direction, in which the primary worker skills are simple dexterity and enthusiasm.

- *A worker can move easily among stations and can easily take over work in process.* This ensures that the bucket brigade protocol does not introduce additional wasted time to pass work.

- *Demand for the products varies significantly.* Bucket brigade manufacturing can more easily track changeable demand because cross-training of workers and low work-in-process inventory mean flexibility of configuration, and short production lead times. In addition, a bucket brigade line can be configured quickly: The assignment of tasks to stations need not be carefully balanced because the movement of the workers balances the line; this reduces the time required to lay out a new line and so shortens changeovers. Finally, because the line is self-balancing, production rates are easily adjustable by simply adding or removing workers from a team.

11.3 Bucket brigades in the warehouse

In many high-volume distribution warehouses, fast moving items are picked from cases stored in a type of shelving called *flow rack*. Within each bay (section of storage) are shelves with rollers and the shelves are tilted to bring the cases forward.

The bays of flow rack are arranged in aisles and a conveyor system runs down each aisle. The *start of an aisle* is the end that is upstream with respect to the movement of the conveyor. For clarity we will describe a single-aisle of flow rack. (Even when there are multiple aisles of flow rack, each aisle is generally operated as an independent module within the warehouse.)

An *order* is a list of items for a single customer together with quantities to be picked. It is typical that orders are released in a batch each day to the picking operation. Then each order is picked by "progressive assembly": The order is picked by no more than one person at a time and the items are accumulated as the order is picked (rather than picking all orders simultaneously and sorting the items afterward).

Paperwork describing orders to be picked waits at the start of the aisle. Each order sheet lists the items and quantities to be picked in the sequence in which items will be encountered along the aisle. The first picker takes the next order sheet, opens a cardboard carton, and slides it along the passive lane of the conveyor as he moves down the aisle picking the items for that order. At some point the second picker takes over and continues picking that order while the first picker returns to the start to begin the next order. When the order is complete the carton(s) are pushed onto the powered portion of the conveyor, which takes them to the packing and shipping department.

There are several ways of coördinating the pickers. Under *zone-picking*, the bays are divided into regions and each picker works within an assigned region: Worker 1 is responsible for picking all items lying within bays $1, \ldots, b_1$; worker 2 is responsible for picking all items lying within bays $b_1 + 1, \ldots, b_2$; and so on.

In designing such order-picking systems managers try to balance the expected work among the pickers during the each picking period. The trouble with this is that it balances the work only *on the average over the picking period*, which means only that everyone will have performed the same total number of picks—yet the line can have been significantly out of balance from order to order!

The order-picking system will constantly seek balance if configured as a bucket-brigade with pickers sequenced from slowest to fastest. However, there is an important difference here: Unlike manufacturing the "items" produced on this line (that is, orders picked) are *not identical* and in fact are best modeled as "random". For example, one might think of each sku i in the warehouse as being in the next order with probability p_i independently of all other skus. Because of this, the system converges to a state of balance in a stochastic sense. This is still an improvement over a static balance because:

- It constantly seeks balance from order to order and so will be out of balance much less often and therefore it will be more productive.
- It spontaneously adapts to disruptions and seasonalities.
- It does not require anyone to compute a balance.

These advantages have been dramatically illustrated in the national distribution center of a major chain retailer that implemented a bucket brigade style of order-picking. After changing to the bucket brigade protocol, their productivity, measured in average number of picks per person-hour, increased over 30% [20], while reducing need for management intervention (Figure 11.6). This was achieved at essentially no cost, and in particular, with no change to the product layout, equipment, or control system (except to render parts of the latter unnecessary).

Previously, work on this line had been assigned by a computer-based model of work content that was run each night preceding picking. Such a model cannot be accurate because

- It cannot economically account for all the relevant detail that determines work content, such as:
  - Location, which might be at waist level or on an inconveniently high shelf.
  - Shape and weight, which might make an item easy to grab or hard to handle.
  - Velocities of the workers, who can range from 50–150% of standard.

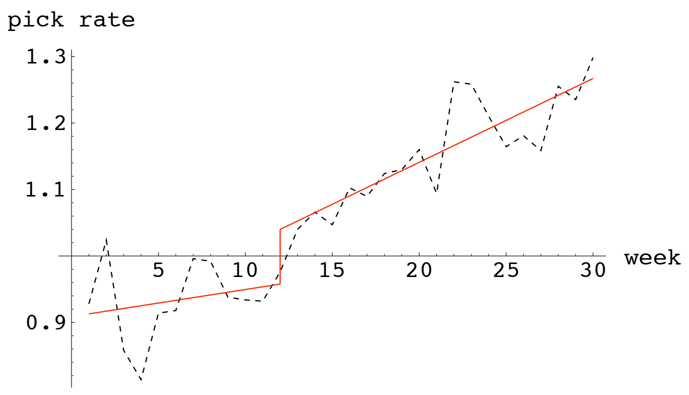


Figure 11.6: Average pick rate as a fraction of the work-standard. Zone-picking was replaced by bucket brigade in week 12. (The solid lines represent best fits to weekly average pick rates before and after introduction of the bucket brigade protocol.)

  - Distribution of locations: One worker might have her picks distributed over three bays while another has as many picks distributed over five bays.
  - Additional work such as disposing of empty containers, sealing a full tote and opening another, prepping an sku, reaching to pull additional stock to the front of the flow rack, and so on.
  - Economies of scale: picking two units is often less than twice the work of picking one unit.

- Even though it might appear balanced on average, the allocation of work can nevertheless be quite unbalanced for every order.
- A static balance cannot adjust to unforeseen events such as equipment malfunction, employee absences, and so on.

Because the model of work content was inaccurate, as all such must be, considerable management time was devoted to adjusting the allocation of work during the day. (In fact, the retailer dedicated a manager to this.) The bucket brigade protocol has made this centralized managerial effort unnecessary—yet still results in better performance.

Figure 11.7 shows average pick rates at 2-hour intervals at a major US distributor of recorded music. After conversion to bucket brigades the average pick rate increased by 20% and the variance of pick rate decreased by 90%; thus bucket brigades were both more productive and more predictable, which made it easier to staff.

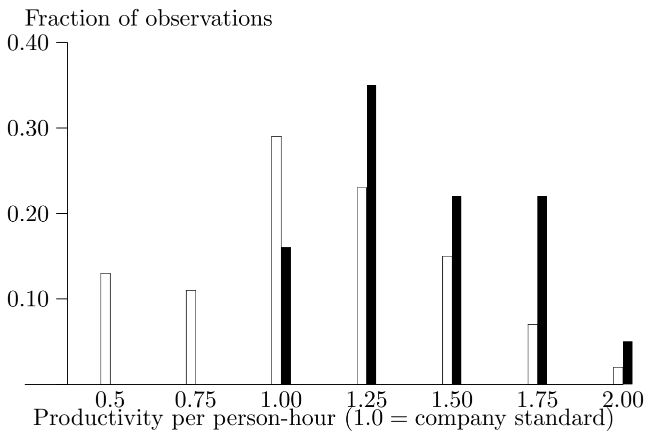


Figure 11.7: Distribution of average pick rate, measured in 2-hour intervals before (clear bars) and after bucket brigades (shaded bars). Under bucket brigades production increased 20% and standard deviation decreased 20%.)

11.4 Summary

One of the advantages of bucket brigades is that they are so flexible. As long as you are careful not to destroy the self-balancing mechanism they can be adapted to many different situations [20].

The ideas behind bucket brigades are simple:

- Abolish rigid assignments of work, which prevent people from going to where the work is.
- Sequence the workers from slowest to fastest to make a “pull system” that is self-organizing.
- Amplify the technical improvements of bucket brigades by emphasizing teamwork.

The result is to make the assembly line into an analog computer. We program this computer by sequencing the workers from slowest-to-fastest. There is no need to input data to this computer because the task times are “read” by doing them. The output is the optimal allocation of work.