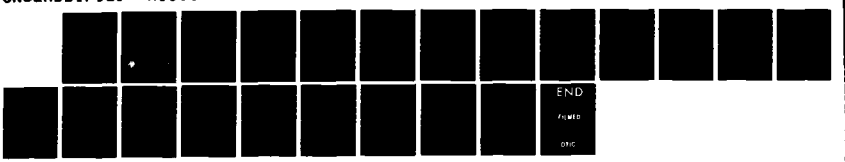
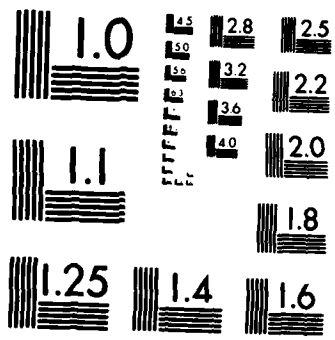
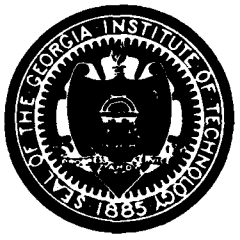
AD-A149 994 RETRIEVAL STRATEGIES FOR A CAROUSEL CONVEYOR(U) GEORGIA INST OF TECH ATLANTA SCHOOL OF INDUSTRIAL AND SYSTEMS ENGINEERING J J BARTHOLDI ET AL. NOV 84 1/1

UNCLASSIFIED N00014-80-K-0709 F/G 13/3 NL





SCHOOL OF INDUSTRIAL AND SYSTEMS ENGINEERING



GEORGIA INSTITUTE
OF TECHNOLOGY
ATLANTA, GEORGIA 30332

85 01 08 011

RETRIEVAL STRATEGIES FOR A CAROUSEL CONVEYOR

John J. Bartholdi, III and Loren K. Platzman

Georgia Institute of Technology
School of Industrial and Systems Engineering
Atlanta, GA 30332
1-404-894-3036

Abstract

We analyze algorithms that sequence the retrieval of items from a carousel conveyor, and show how the appropriate algorithm depends on the load to which the carousel is subjected.

Key words: carousel conveyor, retrieval, automated storage-and-retrieval, warehousing, heuristic algorithms.

This research was supported by the National Science Foundation under grant #ECS-8351313, the Office of Naval Research under grant #N00014-80-k-0709, and by U.S. industry through The Material Handling Research Center at Georgia Institute of Technology.

Submitted to *IIE Transactions* November 1984.

0. Introduction.

A *carousel conveyor* is a length of shelf fashioned into a closed loop that is rotatable (under computer control) in either direction. It is increasingly found in automated warehouses since it can help considerably with the storage and retrieval of small parts. Among the benefits of a carousel is that, rather than have a picker, human or robotic, travel to retrieve an item, the item can travel to the picker. This allows the picker to perform some other activity (e.g. sorting, packaging, labelling, or picking from another carousel) while the item is travelling. This parallelism of activity enables greater throughput and therefore enhanced warehouse operation.

1. Item- and order- density.

An *order* is a set of *items* that must be picked together, as for a single customer. We assume that only one order can be picked at a time, and that we must pick many orders.

How to best retrieve a collection of orders is a two level sequencing problem: how to sequence items within an order, and how to sequence orders. These levels cannot be separated because the time to retrieve the next order depends on where the carousel ended up after retrieving the previous order.

Two parameters measure the density of traffic to which the carousel is subjected. The *order-density* indicates how many orders are received per unit time; the *item-density* indicates the average number of items within each order. Both of these parameters are important to the operation of a carousel since they jointly affect how to best retrieve items and orders.

Both the order-density and the item-density affect how we might choose to solve the problems of sequencing orders and items. For each parameter we consider two extreme situations. For order rate these are

(1) The order rate is small relative to the speed of order retrieval. When the order rate is low, the average time to retrieve an order is smaller than the average time between the arrival of orders. Usually the carousel will retrieve an order, and then wait for the arrival of the next order. Orders will then be retrieved in first-come, first-served sequence, and to be efficient we must retrieve each individual order quickly. Thus our retrieval problem decomposes into a series of problems, each of which is to quickly retrieve a single order.

(2) The order rate is large relative to the speed of order retrieval. If the order rate is high, the problem does not decompose, since new orders may arrive during the retrieval of any order. Thus some orders must wait during the unproductive time spent travelling from the end of one order to the beginning of the next, and we would like for them to not wait +long.

(1') The item-density is large relative to number of storage locations in the carousel. In this case, to retrieve all the items of an order, most, or all of the storage locations must be visited for each order, and so all retrieval problems are (almost) the same. Such re+rieval problems are easy since an efficient route visiting all of the storage locations is trivial to find: simply rotate the carousel clockwise! If merely *most* locations are to be visited, then simply rotating the carousel clockwise still produces an effective (and probably minimal-time) picking sequence. Since all good solutions to such problems look very similar, they are simple to solve.

(2') The item-density is small relative to the number of storage locations in the carousel. In this case each retrieval of an order requires the solution of a routing problem which visits only a small number of the storage locations. The solutions to such problems can be very different, and so not amenable to the simple approach above. In this case we must exercise ingenuity to quickly find effective retrieval routes.

The choice of retrieval strategy will therefore depend on the item- and order-densities. When the order-density is low, retrieval strategies can be simpler, since the problem decomposes into a succesion of independent retrieval problems; when the order-density is high, retrieval strategies must account for the sequencing of orders. When the item-density is low, retrieval strategies must be more complex in order to search for an efficient picking sequence; when it is high, the retrieval strategies can be very simple and still produce efficient picking sequences.

2. Retrieval Strategies.

Consider a carousel serviced by a single picker. The picker is to retrieve a fixed and known collection of items. We would like this to be done as quickly as possible, on the premise that this will improve service. The total time to retrieve all of the items may be expressed as a sum of two terms: the total time during which the carousel is travelling + the total time during which the carousel is stopped for picking. While an item is being picked, neither the carousel nor the picker is available for other activities. At the end of the pick time, both the carousel and the picker become available. All of the pick times are assumed to be constant, approximately equal, and independent of the sequence in which the items are picked. Thus, to shorten the total retrieval time we must shorten the time during which the carousel travels between storage locations.

For definiteness we assume that each order has been preprocessed, and is given as a circularly-linked list of items sorted according to storage location on the carousel, with a pointer to the current relative position of the pick station. To avoid tortured grammar, we will occasionally speak of the picker as moving around a stationary carousel, even though the opposite is true. Also it will sometimes be more convenient to speak of travel distances in place of travel times. (We assume them to be proportional.)

2.1. When the order-density is low.

In this case the retrieval problem decomposes into a series of independent problems, each of which is to quickly retrieve one order.

H.D. Ratliff (private communication) has observed that an optimal pick sequence can be found quickly: In retrieving n items from a carousel, it is never optimal to rotate the conveyor through a complete revolution nor to rotate past the same location more than twice. Consequently some storag location (or else the current position of the pick station) will be the "rightmost"

in an optimal retrieval sequence. Now a location can be rightmost in at most two different ways: the carousel can rotate counter-clockwise to the item and then reverse direction, or the carousel can rotate clockwise and then reverse direction and travel to the item (Figure 1). Since the current location of the pick station can be rightmost in only one way, there are only 2n+1 retrieval sequences that are candidates for optimality.

"OPTIMAL RETRIEVAL" ALGORITHM

Step 1: Evaluate the 2n+1 candidate retrieval sequences that correspond to specifying a "rightmost" location.

Step 2: Choose the best of the candidate solutions from Step 1.

This algorithm determines the very quickest sequence in which to retrieve the items of an order. It requires only O(n) steps (beyond the preprocessing sort).

The intelligence of this algorithm lies in its determining the best place at which to reverse the direction of travel of the carousel. However, when the item-density is large, it is unlikely that the optimal solution will double back so that the computation that determines the best place to double back is likely to be wasted.

For larger item-densities simpler, non-optimal procedures begin to work almost as well as the optimal. A natural one is

"NEAREST ITEM" HEURISTIC

Step 1: Always rotate to the nearest item to be retrieved.

This heuristic is simpler than the optimal algorithm, but it is not guaranteed to produce the shortest retrieval route. Nevertheless, it does have provable performance bounds that guarantee on the quality of its solutions.

THEOREM 1: *Under NEAREST ITEM, the total distance travelled to retrieve one order is never greater than one revolution of the carousel. Furthermore, it is is never greater than 2 * the optimal distance.*

Proof: Consider the route produced by NEAREST ITEM, and partition it into maximal segments in which there are no changes of direction; assume there are k such segments. For the *jth* segment, let I_j be the length along the segment to the first item retrieved on that segment; let R_j be the length of the remainder of the segment. (See Figure 1.) Then, since the heuristic always selects the nearest item to pick next,

$$I_j \geq 2*I_{j-1} + R_{j-1}$$

Expanding this gives

$$I_j \geq \sum_{i=1}^{j-1} (I_i + R_i)$$

Thus each I_j, and I_k in particular, is at least as large as the total distance that has preceeded it, so that the total distance travelled is never more than $2*I_k + R_k$. But $2*I_k + R_k$ is never more than one revolution of the carousel which yields the first conclusion.

The second conclusion is obtained as follows. Let I be the shortest interval that spans all of the items in the order. The distance travelled under the optimal sequence must be at least as long as I. If I is greater than or equal to one-half a carousel revolution, then the claim holds, since the heuristic requires no more than one revolution. If I is less than one-half a carousel revolution, then $I \geq I_k + R_k$, and 2*(the distance travelled under the optimal sequence) $\geq 2*I \geq 2*(I_k + R_k) \geq 2*I_k + R_k \geq$ the distance travelled under the heuristic. ■

Thus, NEAREST ITEM can never produce a solution with an excessively large relative error, or with an excessively large absolute error.

When the item-density is high, it is unlikely that an optimal route will double back. In this case we can safely restrict our consideration to only 2 of the n+1 candidate routes for optimality: the 2 routes that do not double back. If we choose the shorter of these 2, then we are very likely to have chosen the optimal route.

"SHORTER DIRECTION" HEURISTIC

Step 1: Evaluate the length of the route that simply rotates clockwise, and the length of the route that simply rotates counter-clockwise.

Step 2: Choose the shorter of the two routes from Step 1.

While this heuristic never produces a route longer than the length of the carousel, it can, in the worst case, produce a route that is many times longer than the optimal (by a factor of one-half the number of carousel locations). Nevertheless, its routes are likely to be optimal when the item-density is large.

Notice that both NEAREST ITEM and SHORTER DIRECTION are suitable for dynamic versions of the retrieval problem, in which the list of requested items changes even while the order is being picked. In contrast, an optimal solution requires more computation to update itself.

When the item-density is large with repect to the number of carousel locations, then almost any reasonable heuristic will produce routes only negligibly longer than optimal, on the average. Thus, in the interests of economical computation, we might as well adopt the simplest such heuristic. For example, we might accept only a slight degradation of expected quality of solution by using the following extremely simple retrieval strategy.

"MONOMANIACAL" HEURISTIC

Step 1: Always rotate to the right and pick items as they are encountered.

This heuristic is absurdly simple yet for large number of item-densities, it becomes nearly optimal.

Thus as the item-density increases, the qualities of solutions of the retrieval strategies become indistinguishable and optimal. When the item-density is large, we might as well use the simplest retrieval strategies.

2.2. When the order-density is high.

When the order-density is high, the order-retrieval problems do not decompose and we must sequence orders, as well as the items within the orders. A typical way to organize retrieval in this case is to solve a "rolling" problem, *i.e.* to specify some collection of orders to be retrieved, and ignore any new arrivals while retrieving them. This converts the dynamic problem to a series of (easier) static problems. Unfortunately, to solve even the static problem optimally can be difficult: J. Spinrad and C. Tovey (private communication) have shown that the problem is NP-complete [1], which strongly suggests that no fast optimum-finding algorithm exists. Nevertheless there exist fast heuristics that give solutions guaranteed to be close to the optimal.

A lower bound on the cost to pick an order is the length of the shortest interval containing all the items of the order. The *minimal spanning interval* of the n items in an order may be found in O(n) steps by omitting the largest gap between neighboring items of the order. We distinguish the endpoints of this interval by referring to them as the "endpoints of the order". If there are m orders to be picked, then there are 2m endpoints of orders.

Assume m orders are to be retrieved. Any way of picking the orders must visit the two endpoints of each order and so must travel at least the lengths of the shortest spanning intervals. Without loss of generality, we may assume that the picker always begins retrieving an order at one of the endpoints of the order and finishes at the other. Thus any way of picking orders determines a matching among the endpoints of orders.

Suppose we compute the minimum cost matching among the endpoints of orders; then we will have determined a way of pairing the endpoints of orders so that the total travel distance is as small as possible. However, the minimum cost matching may fail to give an uninterrupted sequence in which to pick the orders, since the endpoints of orders may be paired in such a way as to yield disjoint circuits rather than a single path traceable by the picker. We can account for this by connecting the disjoint circuits in an efficient way. Thus we can heuristically determine a sequence in which to retrieve orders by hierarchically decomposing the problem, solving each level well (minimal spanning intervals, minimum cost matching), and assembling a solution from the pieces (connection of disjoint circuits and path). This is formalized as

"HIERARCHICAL" HEURISTIC

Step 1: Construct the minimal spanning interval of each order.

Step 2: Construct a minimum cost maximal matching among the 2m endpoints of orders and the start position of the picker.

(The matching organizes the order intervals into a set of (possibly disjoint) circuits and a path from the starting point of the picker.)

Step 3: Pick all orders in the circuit or path containing the current position of the picker. Pick these orders, and the items within them, in the sequence of their appearance along the circuit or path.

Step 4: Rotate the carousel clockwise until an unpicked item is encountered; go to step 3.

Figure 2 illustrates the performance of HIERARCHICAL.

Notice that HIERARCHICAL specifies a sequence of retrieval for the orders and, in addition, specifies a sequence of retrieval for the items within each order.

THEOREM 2: *Under HIERARCHICAL, the carousel will never travel more than 1 revolution beyond optimum.*

Note that this guarantee is *independent* of the number of orders and the number of items to be retrieved! Consequently HIERARCHICAL is asymptotically optimal.

Proof: Let $\sum I_j$ be sum of the lengths of the shortest spanning intervals of the orders, and let M be the length of the minimal-cost matching. Then any way of retrieving the orders requires total retrieval distance at least $\sum I_j + M$. HIERARCHICAL requires additional travel of at most one revolution of the carousel due to step 3. ■

An additional result of the above argument is that step 2 of HIERARCHICAL can be implemented with *any* of the candidate matchings with non-overlapping edges. In this case the carousel would never make more than 2 revolutions beyond the optimal.

THEOREM 3: *HIERARCHICAL can be implemented to run in $O(\sum n_j)$ steps, where n_j is the number of items in order j.*

Proof: The minimal spanning interval of order j can be found in $O(n_j)$ steps from the preprocessing sort, so that $O(\sum n_j)$ steps suffice to find all of the minimal spanning intervals.

In the matching exactly one point is not matched at all since there are an odd number of points (the starting point of the picker plus the 2 endpoints of each order). Furthermore, the matching contains no overlapping edges since each point is matched to one of its two nearest neighbors. Each specification of an unmatched point uniquely determines a matching with no overlapping edges, and all such matchings are determined by specifying an unmatched point. If m is the number of orders, we need evaluate only the costs of the $2m+1$ matchings corresponding to all possible choices of unmatched points, since one of them must be optimal. Thus to determine the minimum matching requires only O(m) steps. ■

Thus, for relatively little effort, we can determine a provably near-optimal sequence in which to retrieve a collection of orders. Moreover, as the number of orders increases, the *relative* amount of unnecessary retrieval time quickly becomes negligible.

When the item-density is larger, then in the optimal retrieval route, the subroute to retrieve any individual order will tend to be one that does not double back. In such a situation, a much simpler retrieval strategy produces near-optimal routes.

"NEAREST ORDER" HEURISTIC

Step 1: Construct the minimal spanning interval of each order.

Step 2: Travel to the closest endpoint of an unpicked order and pick that order.

NEAREST ORDER also requires $O(\sum n_j)$ steps, but it is much easier to program. Furthermore, the quality of its solutions can also be guaranteed: if the carousel has S storage locations, then

THEOREM 4: *Under NEAREST ORDER, the carousel will never travel more than $\log_2 S$ revolutions beyond optimum.*

This guarantee is not as good as that for HIERARCHICAL; nevertheless, it too is independent of the number of orders and the number of items to be retrieved, and so is aymptotically optimal.

Proof: First note that once begun, each order is retrieved as quickly as is possible. Therefore any travel beyond optimum must be due to travel *between* orders. Call the travel from the end of one order to the beginning of the subsequent order a "gap". Gaps never partially overlap: Any gap is either not contained in any subsequent gaps, or else is contained entirely within a subsequent gap that is at least twice as long. This holds since otherwise the carousel must not have rotated to the nearest endpoint, in violation of the heuristic rule. Thus the carousel cannot have rotated past any point idle more than $\log_2$ S times. Furthermore, between subsequent rotations past any point the gaps sum to length no more than one carousel revolution. Therefore the total length of all gaps cannot exceed $\log_2$ S revolutions. ■

The above argument in fact establishes a more general result: under the heuristic which always travels next to the order whose closest unpicked item is closest of all, the total distance travelled between orders is never greater than $\log_2$ S. This bound is *independent* of the sequences in which the items are retrieved within the orders. This permits the observation that using NEAREST ITEM within each order and also between orders never requires travel beyond $\log_2$ S * 2 * optimum, and so is asymptotically within twice optimum.

When the item-density is high, most carousel locations must be visited, so that the optimal retrieval route is largely determined. This enables an extremely simple heuristic to perform close to optimal, on the average.

"MONOMANIACAL" HEURISTIC #2

Step 1: Always turn clockwise and pick the next encountered order.

In the worst-case this greedy procedure can suggest retrieval routes that are quite poor. Nevertheless, certain weak performance guarantees can be established.

THEOREM 5: *The total travel time under MONOMANIACAL #2 is never more than twice the minimum possible when always rotating the carousel in one direction.*

Proof: (See the appendix.) ■

Again, as the item-density becomes large, this simplest heuristic produces routes which are negligibly longer (on the average) than optimal routes.

3. Conclusions.

The appropriate choice of heuristic for retrieval depends on both the order-density and the item-density. As these increase, we are justified in using simpler heuristics to sequence retrievals. These simpler heuristics may be capable of undesirable worst-case performance, but a high volume of items within orders will protect them from exhibiting pathological behavior. In this case they may be expected to produce retrieval routes that are negligibly longer (on the average) than optimal routes. Advantages of the simpler heuristics include ease of implementation and speed of execution. Moreover, they are naturally suited for incorporation in distributed control systems. Finally, since they typically are "greedy" algorithms with little or no look-ahead, they can quickly modify tentative solutions. Thus they can operate in a dynamic environment in which the problem is changing even as it is being solved.

Practical choice of appropriate retrieval algorithms must be done by simulation. In general, it depends on (in addition to the item- and order-densities) the probability for each item of being in a typical order, the locations in which those items are stored, and the speed at which the carousel rotates.

Informal testing suggests that, across a wide range of conditions NEAREST ITEM and NEAREST ORDER are the algorithms of choice. They can perform quite close to the optimal (on the average) for even fairly low item-densities; they produce solutions that are guaranteed to be not "too far" from optimal, independently of the number of items and orders they process; they are fast in execution; they are simple to program; and they are suitable for distributed and dynamic control.

ACKNOWLEDGEMENTS

The authors thank R.C. Wilson for first interesting us in automated storage-and-retrieval. We also thank H.D. Ratliff and John A. White for support and helpful discussions.

APPENDIX

Proof of Theorem 5: We first establish some terminology and prove a lemma.

Imagine that the first item to be picked from each order is specified. By implication then, the last item to be picked from each order is determined. We will say that an order is "active" at all carousel locations between and including the locations of its first and last items. The specified first and last items of an order determine an "interval of activity" for that order. For a specification of first and last items of each order, let A(j) be that set of orders active at carousel location j. This set has cardinality |A(j)|, and since only one order may be picked at a time, to retrieve all orders we must rotate the carousel past location j at least |A(j)| times, once for each active order. We refer to the largest of the |A(j)| as the "maximum overlap".

To each carousel position i, there corresponds a particularly important way of specifying the interval of activity for each order k. Let F(i,k) denote the interval of activity beginning with the first item in order k that is encountered when rotating the carousel clockwise from location i. (See Figure 3.)

Let r be the smallest number of carousel revolutions required to retrieve all orders.

LEMMA: *For the intervals of activity F(i,k), |A(j)| ≤ 2*r for all carousel locations j.*

Proof: For a specific location j, consider the intervals of activity F(i,k) that are active at location j. Now the optimal picking sequence may choose different first items for each order than chosen by F(i,k). This induces a partition of the orders which are active at location j. One group includes those orders k for which the optimal picking sequence chooses the same first item as F(i,k). Such orders can be no more than r in number, since the optimal picking sequence requires only r rotations of the carousel. The remaining orders must all be active at location i in the optimal picking sequence, and so must be no more than r in number. Thus the maximum number of orders active at location j within F(i,k) can be no more than r+r. ■

Now to complete the proof of the theorem: For any specification of intervals of activity for the orders, define a "critical interval" to be a maximal sequence of consecutive carousel locations for which the |A(j)| are all equal to the maximum overlap. At the initial location of a critical interval, there must be the first item of some order, since otherwise the critical interval would not be maximal. Thus, under NEAREST ORDER, the carousel would never rotate idle past the start of a critical interval without picking one of the orders in that interval. Also, if any order ends in a critical interval, then it must end in the last location of the interval, for otherwise the |A(j)| would not all be equal throughout the interval. Thus the heuristic must pick one order throughout each pass through a critical interval.

Now consider the intervals of activity F(i,k). With each revolution of the carousel, the heuristic reduces by 1 the overlap of the currently critical intervals. But, by the preceding lemma, no overlap exceeds 2*r and so no more than 2*r carousel revolutions are necessary to retrieve all orders. ■

REFERENCES

[1] M. Garey and D. Johnson (1980). *Computers and Intractability: A Guide to the Theory of NP-completeness* W.H. Freeman and Co.

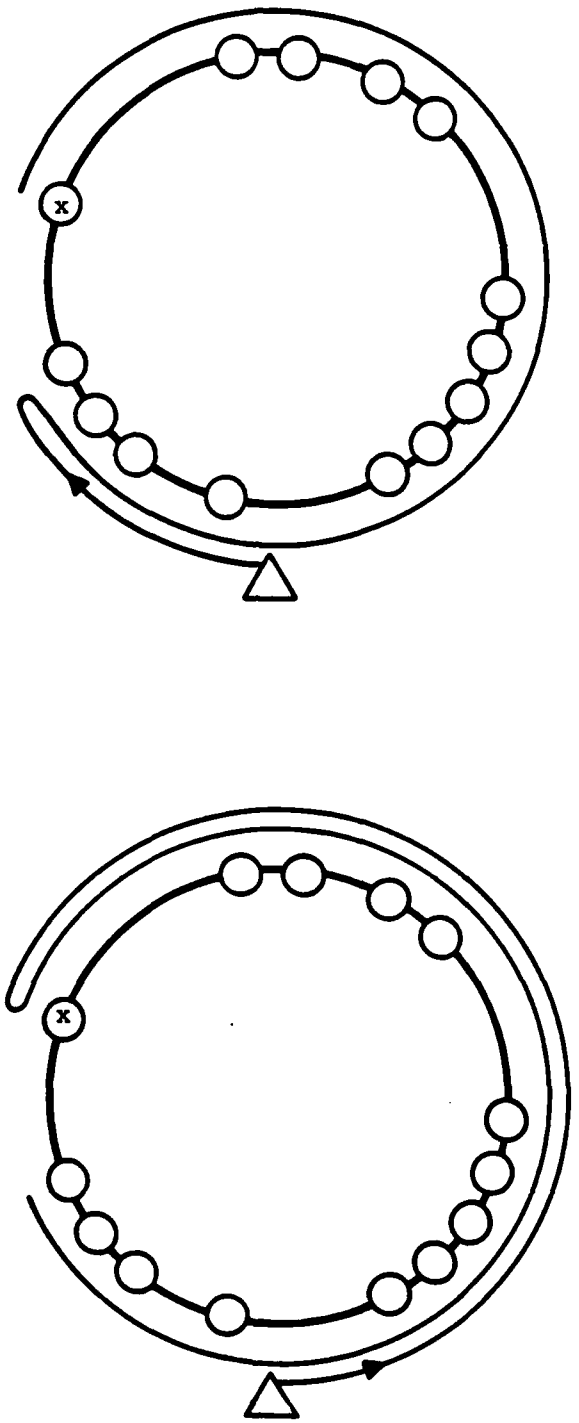


Figure 1: The two different candidate retrieval sequences in which the marked storage location is rightmost.

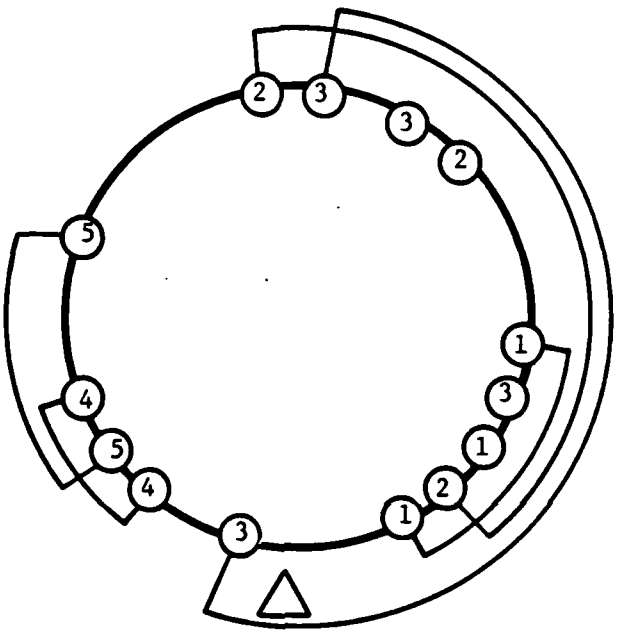


Figure 2A: The minimal spanning intervals of a collection of orders.

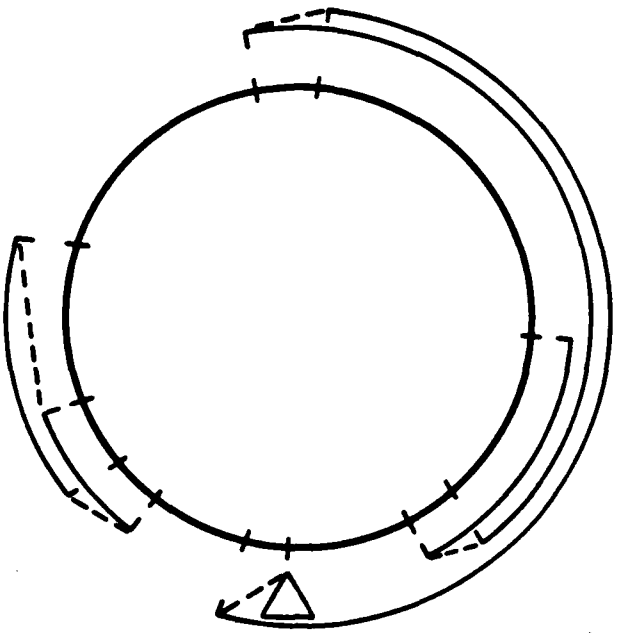


Figure 2B: The minimum-cost maximal matching on the endpoints of the orders and the start position of the picker.

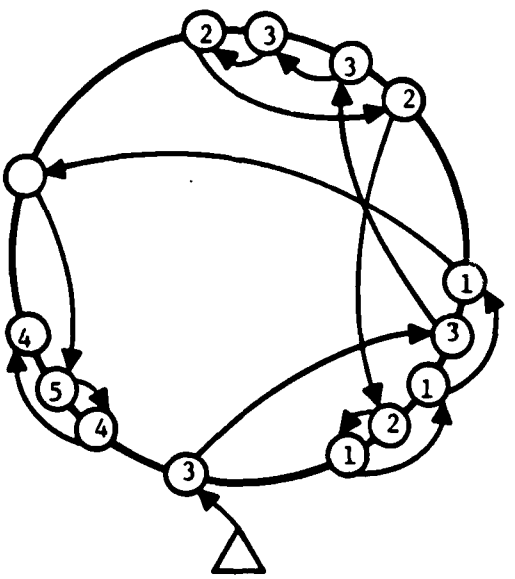


Figure 2C: The sequence of picks determined by the HIERARCHICAL heuristic. It is guaranteed to be within 1 revolution of optimum.

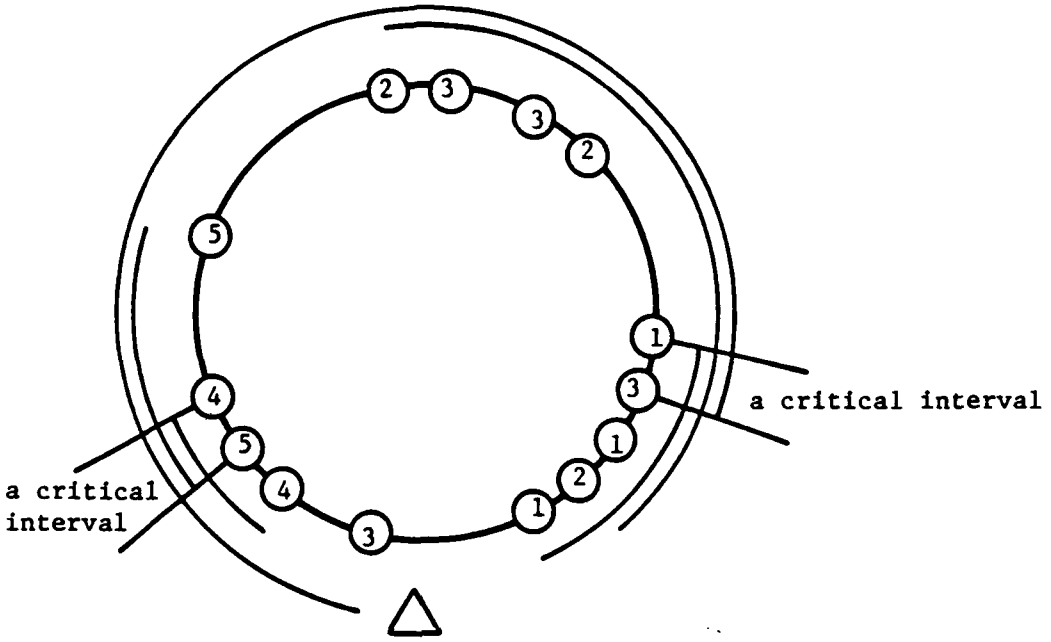


Figure 3: The intervals of activity F (start position, •). The maximum overlap has value 3, as shown in the two critical intervals, so that all of the orders can be retrieved within 3 revolutions of the carousel.

END

FILMED

2-85

DTIC