A Note on the Linearity of Ratliff and Rosenthal's Algorithm for Optimal Picker Routing

Katrin Heßler[a], Stefan Irnich*,[a]

[a] *Chair of Logistics Management, Gutenberg School of Management and Economics, Johannes Gutenberg University Mainz, Jakob-Welder-Weg 9, D-55128 Mainz, Germany.*

Abstract

Ratliff and Rosenthal (1983) state that their dynamic programming algorithm for optimal picker routing has linear complexity in the number of aisles. Indeed, solving the dynamic program is linear in the number of aisles. However, computing the cost coefficients of the dynamic program certainly requires the consideration of all picking positions, whose number is independent of the number of aisles. A straightforward approach sorts the set of picking positions lexicographically by aisle and distance from the bottom cross-aisle, which gives a superlinear (log-linear) algorithm. We show that a better overall linear time complexity can be achieved for a given unsorted sequence of picking positions. The proposed algorithm is linear in the sum of the number of aisles and the number of picking positions.

Keywords: Picker routing problem, dynamic programming, linear time complexity, maximum gap problem

1. Introduction

A fundamental problem in warehouse management is the determination of a distance-minimal picker route. For a rectilinear one-block warehouse with m aisles, the algorithm by Ratliff and Rosenthal (1983) determines such an optimal picker route by solving a dynamic program that has $\mathcal{O}(m)$ stages, a constant number of states per stage, and likewise a constant number of transitions between consecutive stages. As a result, their dynamic-programming algorithm runs in linear time $\mathcal{O}(m)$ (as stated by Ratliff and Rosenthal). Other sources like (Roodbergen and de Koster, 2001b; de Koster *et al.*, 2007) say that the 'running time [is] linear in the number of aisles and the number of pick locations'. Certainly, the complexity depends on whether one considers the construction of the state space of the dynamic program including the computation of cost coefficients. We did however not find a reference providing an algorithm that constructs and solves the picker routing problem in linear time $\mathcal{O}(m+n)$, where n is the total number of picking requests (common are also the synonyms picking locations or picking positions).

A straightforward approach is to sort the set of picking positions lexicographically by aisle and distance from the bottom cross-aisle, which results in a superlinear $\mathcal{O}(n \log(n))$ construction algorithm for the state space. In this note, we show that a better overall linear time complexity can be achieved for a given unsorted sequence of picking positions. The complicating part is the computation of the maximum gap between picking positions in each aisle that has at least two items to be picked. This *maximum gap problem* (MGP) can be formulated as follows: Given an unsorted sequence of at least two different numbers, determine the maximum difference between two consecutive elements in the sorted sequence. For the MGP, we identified (Gonzalez, 1975) as an (unavailable) scientific reference and (Preparata and Shamos, 1985, p. 261) citing Gonzalez' work, providing a pseudo code, and discussing the algorithm as an example for the usefulness of the *pigeonhole principle* (a.k.a. *Dedekind's box argument* (Dedekind used the German word *Schubfachprinzip*)

*Corresponding author.
Email addresses: khessler@uni-mainz.de (Katrin Heßler), irnich@uni-mainz.de (Stefan Irnich)

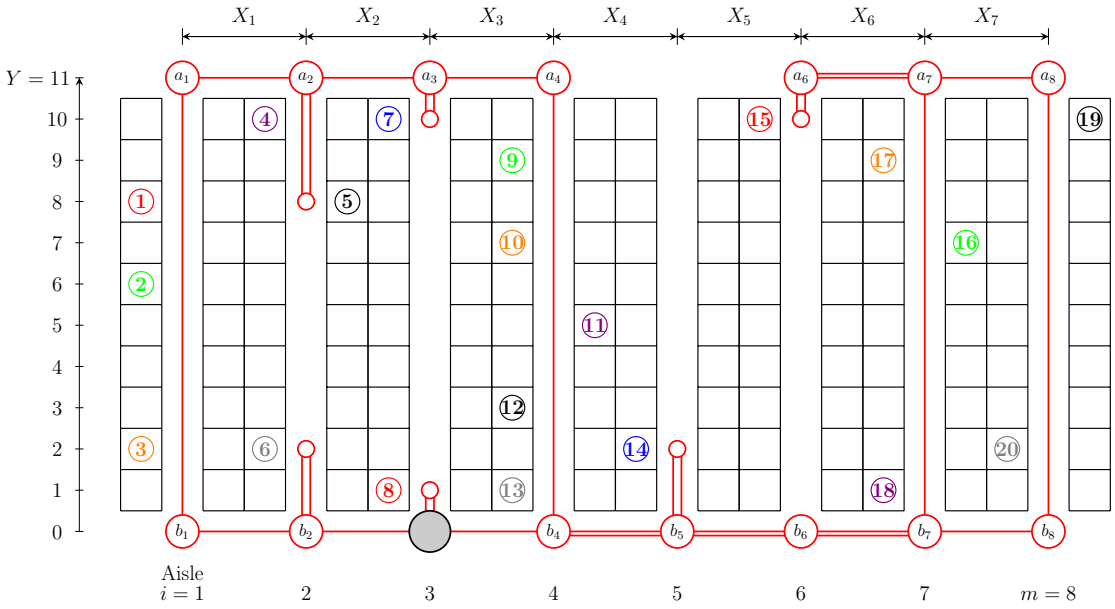


Figure 1: Warehouse with $m = 8$ aisles and $n = 20$ picking positions and optimal picker tour. Note that retrieving items from the left-hand or right-hand side in the same aisle does not make a difference for the length of the picking tour; the side is chosen arbitrarily here.

or *the chest of drawers argument*). We confirm that MGP can be solved in linear time in the length of the sequence, but exemplify that the pigeonhole principle is not the true reason for the algorithms' correctness. Finally, we use the MGP once per aisle resulting in an algorithm for the picker routing problem running in linear time in the sum of the number of aisles and the number of picking requests.

The remainder of this note is structured as follows: Section 2 defines the picker routing problem and sketches the dynamic programming algorithm of Ratliff and Rosenthal. The presentation of the linear time algorithm for the MGP follows in Section 3, before conclusions and an outlook are provided in Section 4.

2. The Picker Routing Problem and the Dynamic Programming Algorithm of Ratliff and Rosenthal

The basic picker routing problem of Ratliff and Rosenthal can be formalized as follows (see Figure 1 for visualization): We are given a rectilinear one-block warehouse with m aisles (running vertically). Let $I = \{1, 2, \ldots, m\}$ denote the set of aisles, numbered from left to right. The total dimension of the warehouse is X times Y, where the horizontal dimension X is given by the sum $X_1, X_2, \ldots, X_{m-1}$ of distances $X_i > 0$ between aisle i and aisle $i+1$. The aisles are connected by top and bottom cross-aisles running horizontally at the vertical positions Y and 0, respectively. A picking request is a pair $p = (i, y)$ for which $i \in I$ determines the aisle and $y \in [0, Y]$ the picking position in this aisle (e.g., the first picking request in Figure 1 is $p = (1, 8)$). We assume that all n picking requests are given as a not necessarily sorted sequence $\mathcal{P} = ((i_j, y_j))_{j \in J}$, where $J = \{1, 2, \ldots, n\}$ denotes the index set of $\mathcal{P}$. Moreover, it is assumed that the picker starts and ends her/his route at the so called depot located at an intersection between one aisle and cross-aisle (located at position $(3, 0)$ in Figure 1). Therefore, we further assume that $\mathcal{P}$ includes the depot as an artificial picking request of the form $(i, 0)$ or (i, Y) depending on whether the depot is located on the top or bottom. An instance of the picker routing problem is given by $((X_i)_{i=1}^{m-1}, Y, \mathcal{P} = (i_k, y_k)_{k=1}^{n})$.

The distance between two picking positions (i_k, y_k) and (i_ℓ, y_ℓ) is

$$d_{k\ell} = \begin{cases} |y_k - y_\ell|, & \text{if } i_k = i_\ell \\ \sum_{i=i_k}^{i_\ell - 1} X_i + \min\{2Y - y_k - y_\ell, y_k + y_\ell\}, & \text{if } i_k < i_\ell \\ \sum_{i=i_\ell}^{i_k - 1} X_i + \min\{2Y - y_k - y_\ell, y_k + y_\ell\}, & \text{if } i_k > i_\ell \end{cases},$$

where the first case is for identical aisles and the second (third) case for different aisles when (i_k, y_k) is stored left (right) to (i_ℓ, y_ℓ). The minimum term in the latter cases reflects that the shortest path between the two picking positions is either by using the top or bottom cross-aisles. For these distances $(d_{k\ell})_{k,\ell \in J}$, an optimal TSP tour is the solution to the picker routing problem. Ratliff and Rosenthal have shown that this type of TSP can be better solved by exploiting the warehouse geometric structure.

The dynamic-programming algorithm of Ratliff and Rosenthal alternates between the decision how the aisle $i \in I$ is traversed and the decision how the cross-aisle between aisles i and $i+1$ is traversed. Since the latter decisions are simpler to explain, we start with them. For cross-aisles connecting aisles i and $i+1$, the only potentially optimal decisions are

$$\{11, 02, 20, 22, 00\}$$

where the first (second) number indicates how often the top (bottom) cross-aisle is traversed. Note that 00 is only possible if all aisles left to i or right to $i+1$ are empty. In Figure 1, the cross-aisle traversal 11 is used to connect aisles 1–4, and 7 with 8, traversal 02 to connect aisles 4–6, and traversal 22 to connect aisle 6 with 7.

Inside an aisle, the only potentially optimal traversals are

$$\{1pass, 2pass, top, bottom, void, gap\}$$

where $1pass$ ($2pass$) means that the picker completely traverses the aisle once (twice), *top* (*bottom*) that the picker enters the aisle from the top (bottom), moves to the lowest (highest) picking position in this aisle, U-turns, and exits the aisle from the same side as entered. The decision *void* can be made if the aisle does not contain picking requests and means that the picker does not enter the aisle at all. The last decision *gap* is the one that is in our focus now. If an aisle contains two or more picking positions, it is possible to enter and leave the aisle from both ends. As all picking positions must be served either way, this leaves a gap between two consecutive picking positions of the aisle. We can see in Figure 1 that the distance-minimal picker tour uses *gap* in aisles $i = 2$ and $i = 3$.

All traversal decisions have an associated cost, and computing this cost is trivial for all decision except *gap*. To simplify the presentation, we define the sequence of relevant positions in aisle $i \in I$ as

$$\mathcal{A}_i = \{y_k : (i_k, y_k) \in \mathcal{P} \text{ and } i_k = i\}.$$

Then, for this aisle i, the cost of the aisle traversal *gap* is

$$c(i, gap) = 2Y - 2\max\{y_\ell - y_k : y_k < y_\ell \in \mathcal{A}_i \text{ and there is no } y_q \in \mathcal{A}_i \text{ with } y_k < y_q < y_\ell\}.$$

The maximum term is the MGP for the sequence $\mathcal{A}_i$ as stated in the introduction.

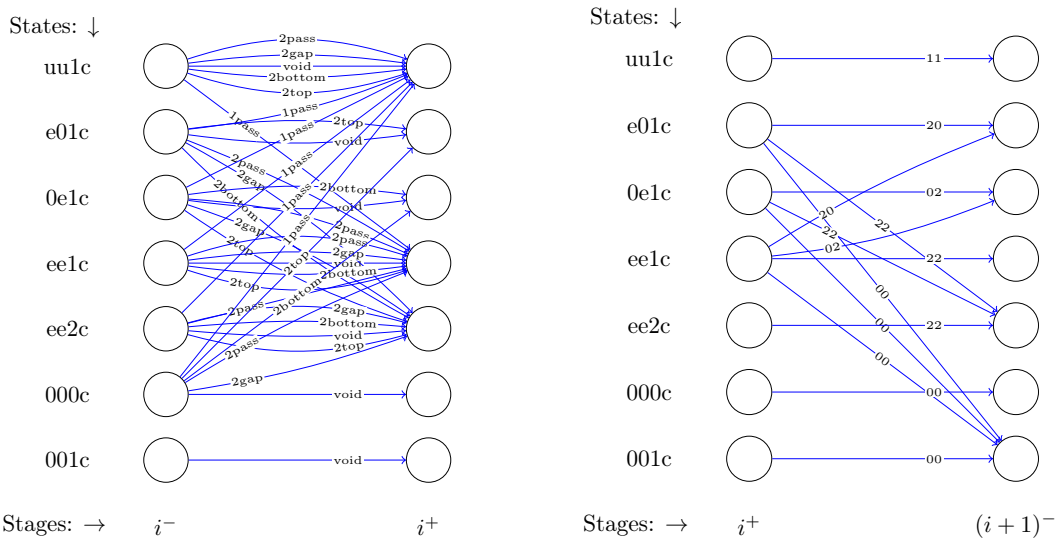
Example 1. *For the sequence $\mathcal{A}_i = (y_1, \ldots, y_7) = (18, 2, 11, 5, 14, 6, 20)$ with $s_i = 7$ elements, sorting produces the sequence*

$$(2, 5, 6, 11, 14, 18, 20)$$

from which we can immediately read the largest gap $5 = 11 - 6$ as the largest difference between consecutive elements. However, sorting is not linear in the number of elements. We seek for a linear-time procedure that allows the determination of the largest gap.

For all other decisions, the cost is simple to compute. Let the minimum and maximum elements in each aisle $i \in I$ be

$$\underline{y}_i = \min \mathcal{A}_i \qquad \text{and} \qquad \bar{y}_i = \max \mathcal{A}_i,$$



(a) Traversals within an aisle i. Note that *void* is only valid in empty aisles.

(b) Traversals of the cross-aisles.

Figure 2: Possible transitions between states.

defining $\underline{y}_i = \infty$ and $\bar{y}_i = -\infty$ for empty sets. Then, $c(i, 11) = c(i, 20) = c(i, 02) = 2X_i$, $c(i, 22) = 4X_i$, $c(i, 00) = 0$, $c(i, 1pass) = Y$, $c(i, 2pass) = 2Y$, $c(i, top) = 2(Y - \underline{y}_i)$, $c(i, bottom) = 2\bar{y}_i$, and $c(i, void) = 0$.

Ratliff and Rosenthal finally define seven states

$$\{uu1c, e01c, 0e1c, ee1c, ee2c, 000c, 001c\}$$

that characterize the so-called *partial tour subgraphs* (PTSs). Two PTSs are defined for each aisle $i \in I$, both representing the respective picker tour from aisle 1 to aisle i, the first one before the traversal of aisle i has been decided and the second after this decision. For each PTS, the first two symbols are either u (=uneven, odd), e (=even), or 0 (=zero) describing the degree of the rightmost cross-aisle vertices at the top and bottom of the PTS (vertices a_i and b_i in Figure 1). The last two symbols indicate an empty PTS (0c), or a PTS that has one (1c) or two (2c) connected components, respectively. Figure 2 visualizes the possible transitions between the seven states. Figures 2a and 2b are the graphical representation of Tables I and II in (Ratliff and Rosenthal, 1983, p. 515f).

3. Solution of the Maximum Gap Problem

For this section, we omit the aisle index i and assume that $\mathcal{A} = (a_k)_{k=1}^s$ is an arbitrary sequence of s numbers (not necessarily integer numbers, real numbers are allowed). Clearly, the minimum $\underline{y} = \min \mathcal{A}$ and the maximum $\bar{y} = \max \mathcal{A}$ can be determined in time $\mathcal{O}(s)$. In case $\underline{y} = \bar{y}$, the solution of the MGP is defined as 0 (uninteresting for our purposes). We therefore assume $\underline{y} < \bar{y}$, i.e., $s \geq 2$ in the following.

The procedure we describe now is based on the work of Gonzalez (1975) as described in (Preparata and Shamos, 1985). At some places we slightly deviate from their description for the sake of clarity. We start defining the real number $\Delta = (\bar{y} - \underline{y})/(s-1)$. Δ is a tight lower bound for the maximum gap, and this value is assumed if all values $y_j \in \mathcal{A}$ are equidistantly distributed in $[\underline{y}, \bar{y}]$. We define $s-1$ (half open) intervals $I_p = (\underline{y} + (p-1)\Delta, \underline{y} + p\Delta]$ for $p \in \{1, 2, \ldots, s-1\}$. These intervals $I_1, \ldots, I_{s-1}$ partition the interval $(\underline{y}, \bar{y}]$. Therefore, each number $y_j \in \mathcal{A}$ not equal to $\underline{y}$ falls into exactly one interval $I_1, \ldots, I_{s-1}$. Accordingly, we define $s-1$ possibly empty buckets

$$B_p = I_p \cap \mathcal{A}$$

for $p \in \{1, 2, \ldots, s-1\}$. The minimum and maximum element of each bucket is denoted as

$$a_p = \min B_p \qquad \text{and} \qquad b_p = \max B_p,$$

assuming again that empty sets give $a_p = \infty$ and $b_p = -\infty$. Also the computation of all values a_p and b_p can be done within a single loop over $\mathcal{A}$, i.e., in linear time.

Since any two elements in the same bucket have an absolute difference strictly smaller than Δ, the maximum gap must occur between elements from different buckets (one might interpret this fact as an example of the pigeonhole principle). Therefore, the maximum gap is of the form $a_q - b_p$ for $p < q$ under the condition that all intermediate buckets are empty, i.e., $B_r = \varnothing$, equivalent to $a_r = \infty$ and $b_r = -\infty$, for all r with $p < r < q$. These differences can also be computed in linear time, proving that the MGP is solvable in linear time. A borderline case is related to the first bucket B_1 which, by definition, does not include the minimum element $\underline{y}$. In order to properly account for the case that the maximum gap is formed by $\underline{y}$ and the next largest element, we define $b_0 = \underline{y}$ and allow $0 \leq p < q \leq s-1$.

Example 2. *(continued from Example 1) For the sequence* $\mathcal{A} = (y_1, \ldots, y_7) = (18, 2, 11, 5, 14, 6, 20)$ *with* $s = 7$ *elements, we have the minimum* $\underline{y} = 2$ *and maximum* $\bar{y} = 20$*. Hence* $\Delta = (20-2)/(7-1) = 3$*. The* $s - 1 = 6$ *intervals are*

$$I_1 = (2, 5], \quad I_2 = (5, 8], \quad I_3 = (8, 11], \quad I_4 = (11, 14], \quad I_5 = (14, 17], \quad I_6 = (17, 20]$$

with the following buckets, minima, and maxima:

$$\begin{array}{rcccccc} & B_1 = \{5\}, & B_2 = \{6\}, & B_3 = \{11\}, & B_4 = \{14\}, & B_5 = \varnothing, & B_6 = \{18, 20\} \\ & a_1 = 5, & a_2 = 6, & a_3 = 11, & a_4 = 14, & a_5 = \infty, & a_6 = 18 \\ b_0 = 2, & b_1 = 5, & b_2 = 6, & b_3 = 11, & b_4 = 14, & b_5 = -\infty, & b_6 = 20 \end{array}$$

The values to compare are $a_1 - b_0 = 5-2 = 3$*,* $a_2 - b_1 = 6-5 = 1$*,* $a_3 - b_2 = 11-6 = 5$*,* $a_4 - b_3 = 14-11 = 3$*, and* $a_6 - b_4 = 18 - 14$ *(note that bucket* B_5 *is empty). The maximum of these values is gap* $= 5$ *resulting from the consecutive elements* 6 *and* 11 *of the sorted sequence, which is the solution to the MGP.*

Algorithm 1 provides a pseudo code of the procedure that we described verbally. This type of algorithm has also been described in some software programming forums (e.g. Chauhan, 2021) and websites (e.g., (Unknown), 2015). Note that Example 2 shows that the maximum gap can occur between two consecutive buckets even if some other bucket remains empty. We would like to stress that Algorithm 1 does *not* rely on the fact that sorting all $s-2$ numbers different from $\underline{y}$ and $\bar{y}$ into $s-1$ buckets must leave one bucket empty (again, the pigeonhole principle). Some online discussions and also (Preparata and Shamos, 1985) use such an argument.

4. Conclusion

We have shown in this note that the crucial step for proving the linearity of the dynamic-programming algorithm of Ratliff and Rosenthal is the computation of the cost coefficient of the traversal possibility *gap*, where a picker enters an aisle from the top as well as from the bottom leaving a middle segment of the aisle untraversed. Algorithm 2 summarizes our cost computation for all cost coefficients of the dynamic program. In particular, $c(i, gap)$ is computed from an unsorted sequence $\mathcal{A}_i$ of positions belonging to aisle i. In Step 10 of Algorithm 2, the procedure *MaximumGap* computes the solution to the associated maximum gap problem (MGP) in a time proportional to $|\mathcal{A}_i|$. Since $n = |\mathcal{A}_1| + |\mathcal{A}_2| + \cdots + |\mathcal{A}_{m-1}|$, it is shown that the cost coefficients of all transitions can be computed in $\mathcal{O}(m+n)$ time.

The algorithm of Ratliff and Rosenthal consists not only of the construction and resolution of the dynamic program, but of a third component, which is the construction of the final picker tour (like a step-by-step sequence of commands where the picker has to go). Ratliff and Rosenthal have already shown that also this last tour construction component is linear in the number of aisles (Ratliff and Rosenthal, 1983, Sect. 4). Hence, the entire algorithm can be implemented to run in $\mathcal{O}(m+n)$ time.

Algorithm 1: MaximumGap

Input : Sequence $\mathcal{A} = (y_j)_{j=1}^{s}$

$\underline{y} \leftarrow \infty$
$\bar{y} \leftarrow -\infty$
for $j = 1, 2, \ldots, s$ **do**
  $\underline{y} \leftarrow \min\{\underline{y}, y_j\}$, $\bar{y} \leftarrow \max\{\bar{y}, y_j\}$
If $\underline{y} \geq \bar{y}$ **return** 0
$\Delta \leftarrow (\bar{y} - \underline{y})/(s-1)$
for $p = 1, 2, \ldots, s-1$ **do**
  $a_p \leftarrow \infty$, $b_p \leftarrow -\infty$
for $j = 1, 2, \ldots, s : y_j \neq \underline{y}$ **do**
  $p \leftarrow \lceil (y_j - \underline{y})/\Delta \rceil$
  $a_p \leftarrow \min\{a_p, y_j\}$, $b_p \leftarrow \max\{b_p, y_j\}$
$gap \leftarrow 0$
$b_{prev} \leftarrow \underline{y}$
for $p = 1, 2, \ldots, s-1$ **do**
  if $a_p < \infty$ **then**
    $gap \leftarrow \max\{gap, a_p - b_{prev}\}$
    $b_{prev} \leftarrow b_p$
return gap

Algorithm 2: ComputeCostCoefficients

Input : Instance $((X_i)_{i=1}^{m-1}, Y, \mathcal{P} = (i_j, y_j)_{j=1}^{n})$

for $i = 1, 2, \ldots, m$ **do**
  Create the empty sequence $\mathcal{A}_i \leftarrow ()$
  $\underline{y}_i \leftarrow \infty$, $\bar{y}_i \leftarrow -\infty$
for $j = 1, 2, \ldots, n$ **do**
  Insert y_j into $\mathcal{A}_{i_j}$
  $\underline{y}_{i_j} \leftarrow \min\{\bar{y}_{i_j}, y_j\}$, $\bar{y}_{i_j} \leftarrow \max\{\bar{y}_{i_j}, y_j\}$
for $i = 1, 2, \ldots, m$ **do**
  $gap \leftarrow -\infty$
  if $|\mathcal{A}_i| \geq 2$ **then**
    $gap \leftarrow$ MaximumGap$(\mathcal{A}_i)$
  $c(i, 1pass) \leftarrow Y$
  $c(i, 2pass) \leftarrow 2Y$
  $c(i, top) \leftarrow 2(Y - \underline{y}_i)$
  $c(i, bottom) \leftarrow 2\bar{y}_i$
  $c(i, void) \leftarrow 0$
  $c(i, gap) \leftarrow 2Y - 2gap$
  if $i < m$ **then**
    $c(i, 11) \leftarrow c(i, 20) \leftarrow c(i, 02) \leftarrow 2X_i$
    $c(i, 22) \leftarrow 4X_i$
    $c(i, 00) \leftarrow 0$

Certainly, the difference between a linear and log-linear algorithm for picker routing is marginal if one wants to solve a single picker routing problem or a few problems of that kind. However, algorithms for the joint order batching and picker routing problem (Henn *et al.*, 2012) may solve a huge number of them, because the number of possible batches typically grows exponentially with the number of orders. In these integrated problems, a linear and log-linear algorithm can make a difference.

Finally, we would like to note that alternative warehouse layouts are often found in practice. Two-block and multi-block rectilinear warehouses (Roodbergen and de Koster, 2001b,a) can lead to shorter picker routes at the cost of a smaller storage space. Alternative layouts like fishbone, flying-V etc. try to further optimize the tradeoff between picker route length and storage space (Gue and Meller, 2009). Even if layouts are more sophisticated, extended dynamic-programming algorithms remain applicable and the fundamental traversal decisions per aisle remain as in the single-block case (see, e.g., Masae *et al.*, 2019): computing the maximum gap is the most complicated cost computation. Our linearity result is therefore directly generalizable to picker routing problems in other warehouse layouts.

References

Chauhan, A. (2021). Maximum adjacent difference in an array in its sorted form. Online forum. `https://www.geeksforgeeks.org/maximum-adjacent-difference-array-sorted-form/`.

de Koster, R., Le-Duc, T., and Roodbergen, K. J. (2007). Design and control of warehouse order picking: A literature review. *European Journal of Operational Research*, **182**(2), 481–501.

Gonzalez, T. (1975). Algorithms on sets and related problems. Technical report, Department of Computer Science, University of Oklahoma. (not available to us).

Gue, K. R. and Meller, R. D. (2009). Aisle configurations for unit-load warehouses. *IIE Transactions*, **41**(3), 171–182.

Henn, S., Koch, S., and Wäscher, G. (2012). Order batching in order picking warehouses: A survey of solution approaches. In *Warehousing in the Global Supply Chain*, pages 105–137. Springer, London.

Masae, M., Glock, C. H., and Vichitkunakorn, P. (2019). Optimal order picker routing in the chevron warehouse. *IISE Transactions*, **52**(6), 665–687.

Preparata, F. P. and Shamos, M. I. (1985). *Computational Geometry: An Introduction.* Texts and Monographs in Computer Science. Springer, New York, NY.

Ratliff, H. D. and Rosenthal, A. S. (1983). Order-picking in a rectangular warehouse: A solvable case of the traveling salesman problem. *Operations Research*, **31**(3), 507–521.

Roodbergen, K. J. and de Koster, R. (2001a). Routing methods for warehouses with multiple cross aisles. *International Journal of Production Research*, **39**(9), 1865–1883.

Roodbergen, K. J. and de Koster, R. (2001b). Routing order pickers in a warehouse with a middle aisle. *European Journal of Operational Research*, **133**(1), 32–43.

(Unknown) (2015). Algorithms and problem solving. Website. `http://www.zrzahid.com/tag/maximum%e2%80%a9-gap/`.