ANZIAM J. 57(2015), 166–174
doi:10.1017/S1446181115000140

EFFICIENT ALGORITHMS FOR TRAVELLING SALESMAN PROBLEMS ARISING IN WAREHOUSE ORDER PICKING

H. CHARKHGARD[✉1] and M. SAVELSBERGH[2]

(Received 12 November, 2013; accepted 6 May, 2015; first published online 22 September 2015)

Abstract

We investigate two routing problems that arise when order pickers traverse an aisle in a warehouse. The routing problems can be viewed as Euclidean travelling salesman problems with points on two parallel lines. We show that if the order picker traverses only a section of the aisle and then returns, then an optimal solution can be found in linear time, and if the order picker traverses the entire aisle, then an optimal solution can be found in quadratic time. Moreover, we show how to approximate the routing cost in linear time by computing a minimum spanning tree for the points on the parallel lines.

2010 *Mathematics subject classification*: primary 90B06; secondary 11Y16.

Keywords and phrases: order batching, order picking, picker routing, travelling salesman problem, minimum spanning tree problem.

1. Introduction

Studies have estimated that 20% of logistic costs are related to warehousing (see for example [3]), and that up to 65% of total warehouse operating costs are because of order picking, that is, the process of retrieving articles from a storage area in a warehouse to satisfy customers' demands [5, 11]. As a consequence, optimizing order picking may significantly reduce warehousing and logistic costs.

A well-established technique for optimizing order picking is *order batching* [4], that is, grouping orders and picking them in a group in a single picking tour. Thus, the *order batching problem* (OBP) seeks to cluster orders into groups so as to minimize the total order processing time, which includes *travel time*, *search time*, *pick time* and *setup time*. Travel time refers to the time required for an order picker to travel between locations in the order picking tour, search time refers to the time required to identify

[1]School of Mathematical and Physical Sciences, University of Newcastle, NSW 2308, Australia; e-mail: Hadi.Charkhgard@uon.edu.au.
[2]H. Milton Stewart School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA 30332-0205, USA; e-mail: Martin.Savelsbergh@isye.gatech.edu.

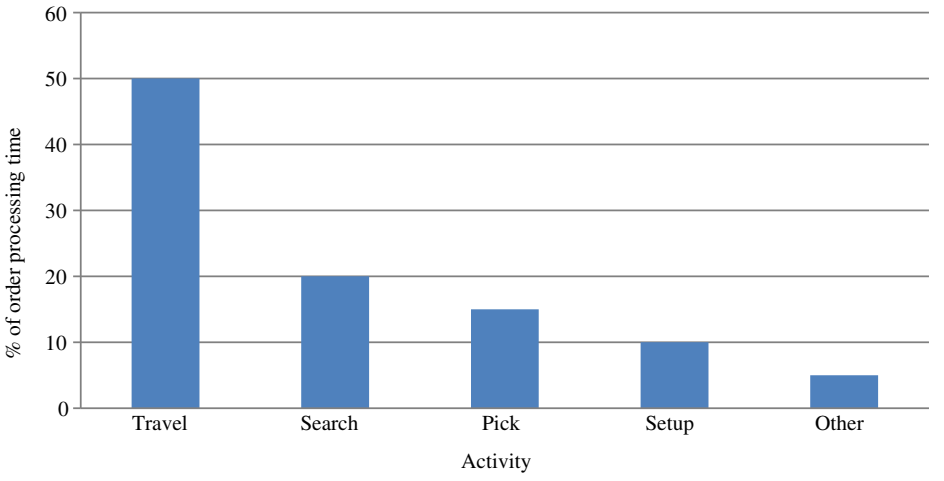

FIGURE 1. Typical distribution of order processing time [11].

the articles to be picked, pick time refers to the time required to transfer the required number of articles from their storage locations to the cart or vehicle and setup time refers to the time required for administrative and setup tasks at the beginning and the end of each picking tour [2]. A typical distribution of order processing time can be found in Figure 1. Because of the significance of travel time, the objective function in the OBP is usually the minimization of the travel time (or distance) across all picking tours.

The *picker routing problem* (PRP) seeks to minimize the distance travelled by a (single) picker, given a set of pick locations that has to be visited. It is a special case of the *travelling salesman problem* (TSP) due to the typical rectangular layout of the storage area in a warehouse. Ratliff and Rosenthal [9] have shown that the PRP can be solved in polynomial time. However, their algorithm is too time consuming to be incorporated as a subroutine in the algorithms for the OBP. Furthermore, the resulting order picking tours are not necessarily intuitive and may, therefore, increase the number of picker errors. As a consequence, researchers have focused more on restricted routing strategies, that is, routing strategies that produce pick tours with a specific structure. Figure 2 displays the *S-shape*, the *return*, the *largest gap* and the *combined* strategies [6, 8, 10].

In the *S-shape* routing strategy, an order picker enters an aisle and traverses the aisle if there exists at least one article that has to be picked from that aisle, then goes to the next aisle. The order picker returns to his starting point after traversing the last aisle which has to be visited. In the *return* strategy, an order picker enters an aisle and returns after visiting the most distant pick location. In the *largest gap* strategy, an order picker traverses the first and last aisles from which articles have to be picked entirely, whereas the other aisles are traversed partially, in and out, from both ends, in such way that the distance that is *not* traversed is maximum. In the *combined* routing

https://doi.org/10.1017/S1446181115000140 Published online by Cambridge University Press

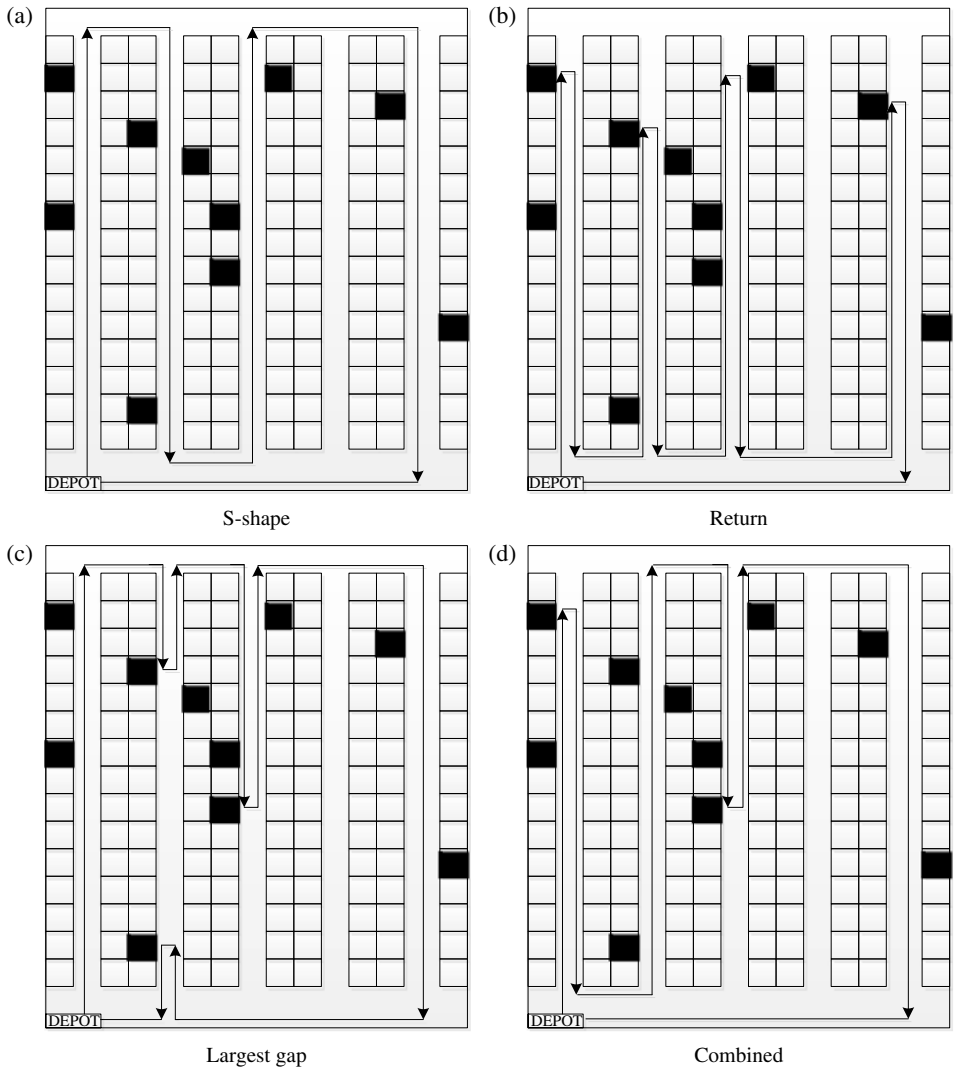

FIGURE 2. Restricted routing strategies.

strategy, each aisle is either traversed entirely or entered and left from the same end [7], which usually generates a near-optimal solution [1].

All strategies involve the solutions of two fundamental aisle routing problems (see Figure 3):

- optimally pick all required items while traversing the entire aisle (referred to as the *passing strategy*); and
- optimally pick all required items and return to the end of the aisle where entered (referred to as the *returning strategy*).

https://doi.org/10.1017/S1446181115000140 Published online by Cambridge University Press

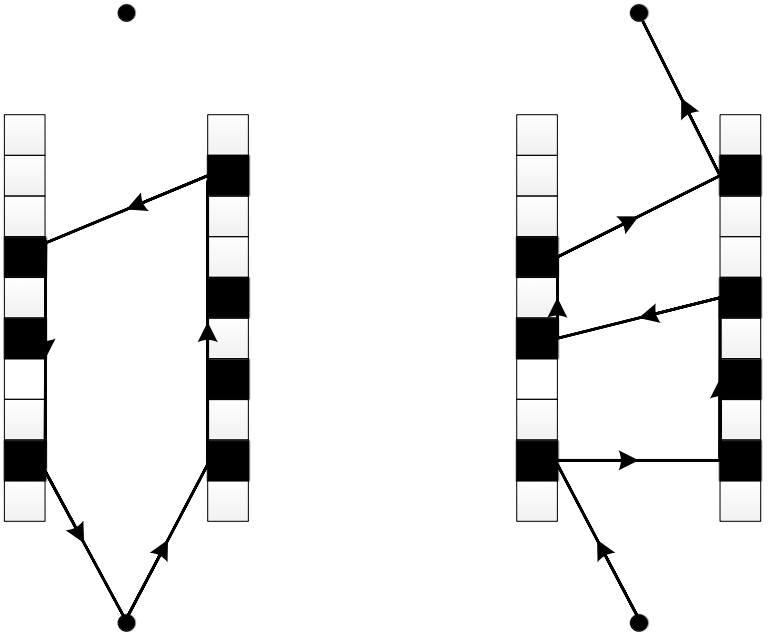

FIGURE 3. Two aisle routing problems.

In most of the algorithms for the OBP, the travel time incurred while crossing from one side of the aisle to the other side of the aisle is ignored for efficiency reasons. In this paper, we show that these two *aisle routing problems*, which are special cases of the TSP, can be solved efficiently: for the passing strategy in $O(n^2)$ time and for the returning strategy in $O(n)$ time, where n is the number of pick locations in an aisle. Because $O(n^2)$ time may be computationally prohibitive when solving large instances of the OBP, we show that an approximate cost for the passing strategy, derived from the minimum spanning tree for the pick locations, can be computed in $O(n)$ time. The details are provided in the next section.

2. Efficient algorithms for aisle routing problems

We first introduce the concepts and notation that will facilitate the presentation and discussion of the proposed algorithms. We denote the starting point for the order picker with s and the end point for the order picker, in case of the passing strategy, with t. Let n_r and n_l be the numbers of pick locations to be visited on the right- and the left-hand sides of the aisle, respectively. Furthermore, let the ordered set of pick locations on the right-hand side of the aisle be $\{r_1, r_2, \ldots, r_{n_r}\}$, with r_1 closest to s, and the ordered set of pick locations on the left-hand side of the aisle be $\{l_1, l_2, \ldots, l_{n_l}\}$, with l_1 closest to s. We denote the (Euclidean) distance between points i and j by $d(i, j)$. Finally, we

https://doi.org/10.1017/S1446181115000140 Published online by Cambridge University Press

assume that there exists *at least one pick location* on each side of the aisle (otherwise, the aisle routing problems are trivial).

Before discussing the algorithms for solving the aisle routing problem, we introduce a notion and an important property of optimal solutions.

Definition 2.1. A tour $\{1, \ldots, n\}$ is said to have the no-crossing property if, for every pair of links $\{i, i+1\}$ and $\{j, j+1\}$ with $j > i+1$, the intersection of the closed segments $(i, i+1)$ and $(j, j+1)$ is empty.

Lemma 2.2. *There exists an optimal tour that has the no-crossing property.*

Proof. Suppose that no such optimal tour exists, and let k be the point in the intersection of $(i, i+1)$ and $(j, j+1)$. Because of the triangle inequality, we have $d(i, j) \leq d(i, k) + d(k, j)$ and $d(i+1, j+1) \leq d(i+1, k) + d(k, j+1)$. But that implies that $d(i, j) + d(i+1, j+1) \leq d(i, k) + d(k, j) + d(i+1, k) + d(k, j+1) = d(i, i+1) + d(j, j+1)$, which is a contradiction. □

We start our analysis by considering the returning strategy.

Lemma 2.3. *An optimal tour for the returning strategy is to go to the first pick location on the right-hand side of the aisle, visit all pick locations on the right-hand side of the aisle, cross over to the most distant pick location on the left-hand side of the aisle, visit all pick locations on the left-hand side of the aisle and return to the starting point.*

Proof. Since it is the only tour with no-crossing property, it is an optimal tour. □

Corollary 2.4. *An optimal solution for the returning strategy can be computed in linear time.*

Next, we consider the passing strategy. We start with the following observation.

Proposition 2.5. *If $1 \leq i < j \leq n_r$, then the order picker must visit r_i before r_j in an optimal tour. Similarly, if $1 \leq i < j \leq n_l$, then the order picker must visit l_i before l_j in an optimal tour.*

Proof. Suppose that this is not true. Then there must exist at least two pick locations on one side of aisle, that is, r_i and r_j or l_i and l_j, which have not been visited in order. But this implies that the tour does not satisfy the no-crossing property, which is a contradiction. □

Proposition 2.5 allows us to develop an efficient dynamic programming algorithm for finding a path from s to t with the minimum cost. Let $f_L(i, j)$ be the minimum cost of visiting $\{l_1, l_2, \ldots, l_i\}$ and $\{r_1, r_2, \ldots, r_j\}$ and ending on the left, that is, at l_i, and $f_R(i, j)$ be the minimum cost of visiting $\{l_1, l_2, \ldots, l_i\}$ and $\{r_1, r_2, \ldots, r_j\}$ and ending on the right, that is, at r_j. We have $f_L(0, 0) = 0$, $f_R(0, 0) = 0$, $f_L(1, 0) = d(s, l_1)$ and $f_R(0, 1) = d(s, r_1)$. The values of the remaining costs can be calculated using the following recursive equations:

$$f_L(i, j) = \min\{f_L(i-1, j) + d(l_{i-1}, l_i), f_R(i-1, j) + d(r_j, l_i)\}$$

https://doi.org/10.1017/S1446181115000140 Published online by Cambridge University Press

and

$$f_R(i, j) = \min\{f_R(i, j-1) + d(r_{j-1}, r_j), f_L(i, j-1) + d(l_i, r_j)\}.$$

The optimal value for the passing strategy is

$$\min\{f_L(n_l, n_r) + d(n_l, t), f_R(n_l, n_r) + d(n_r, t)\}.$$

Finally, given that the number of pick locations is n, all values of $f_L(i, j)$ and $f_R(i, j)$ can be computed in $O(n^2)$. Thus, we have the following observation.

Proposition 2.6. *An optimal solution for the passing strategy can be computed in $O(n^2)$ time.*

Even though an optimal solution for the passing strategy can be found in $O(n^2)$ time, this may be too expensive computationally as a subroutine in an algorithm for the OBP. Consequently, we next consider computing an approximate cost for the passing strategy.

Specifically, we propose to compute a minimum spanning tree (MST) on the pick locations and then connect s and t to their closest pick locations. This not only provides an approximate cost; it, in fact, provides a lower bound on the optimal value for the passing strategy.

To describe the algorithm, we assume that the pick locations, or points, are given by means of (x, y)-coordinates, labelled $1, \ldots, n$ in nondecreasing order of the y-coordinates, and in case of ties, the point closest to the previous point in the ordering is given preference. If l_1 and r_1 have same y-coordinates, arbitrarily label one of them as 1. A formal description of the algorithm for computing an MST is given in Algorithm 1.

Observation 2.7. For each point, there are at most four candidate points to which it can be connected by an edge in the spanning tree; see Figure 4.

Recall that for any cut $\delta(V)$ with $V \subseteq \{1, \ldots, n\}$, a MST will contain at least one of the minimum cost edges in the cut. In iteration i, the algorithm considers the cut defined by the connected component containing point i and adds a minimum cost edge in the cut to the partially constructed spanning tree. More specifically, in iteration i, the algorithm examines edges $(i, u(i))$, where $u(i)$ is the closest point *up* from i, and $(i, a(i))$, where $a(i)$ is the closest point *across and up* from i (i and $a(i)$ may have the same y-coordinate); see Figure 4.

The correctness of the algorithm follows from the following theorem.

Theorem 2.8. *At the start of iteration i for $i > 1$, the edge e identified in iteration $i - 1$ represents the minimum cost edge with end points in $\{1, \ldots, i\} \cup \{a(i)\}$, other than edge $(i, a(i))$, linking the two different connected components containing i and $a(i)$.*

Proof. It is easy to verify that this is true at the start of iteration 2. Next, assume that it is true at the start of iteration i and examine the situation at the start of iteration $i + 1$.

The only edge with both end points in $\{1, \ldots, i\} \cup \{a(i)\}$ that may provide a cheaper alternative to edge e (as known at the start of iteration i) is $(i, a(i))$. Therefore,

https://doi.org/10.1017/S1446181115000140 Published online by Cambridge University Press

Algorithm 1: Computing an MST when all the points are on two parallel lines

Input: Set of points $\{1, \ldots, n\}$
Input: Distance function $d(i, j)$ $\forall i, j \in \{1, \ldots, n\}$
Step 1. Order points
Step 2. Process points
$d = \infty$; e = undefined
for $i = 1, \ldots, n-1$ **do**
  $u(i)$ = up point
  $a(i)$ = across point
  Update shortest connection between component with i and component with $a(i)$
  if $d(i, a(i)) < d$ **then**
    $d = d(i, a(i))$; $e = (i, a(i))$
  Select minimum cost edge in cut defined by the points in the component with i
  if $d(i, u(i)) \leq d$ **then**
    Make $(i, u(i))$ permanent
  else
    Make e permanent
    $d = d(i, u(i))$; $e = (i, u(i))$

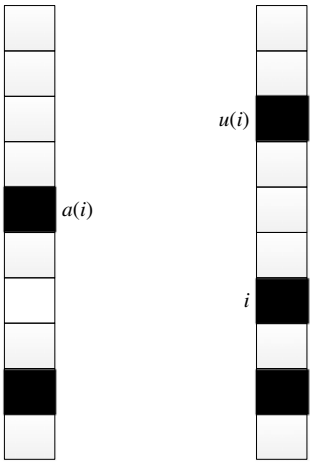

FIGURE 4. Four candidates for connecting to point i.

Algorithm 1 compares the cost of the two edges and performs an update of e, if necessary, before selecting the minimum cost edge in the cut defined by the points in the component containing i.

Now, we consider the following two cases.

Case 1: $i + 1 = a(i)$. This implies that $a(i + 1) = u(i)$. If we made $(i, u(i))$ permanent in iteration i, then $u(i)$ and i would now be part of the same component. Since e is

https://doi.org/10.1017/S1446181115000140 Published online by Cambridge University Press

the minimum cost edge with end points in $\{1, \ldots, i\} \cup \{a(i)\}$ linking the component containing i with the component containing $a(i)$, it is now the minimum cost edge with end points in $\{1, \ldots, i, i+1\} \cup \{a(i)\}$ linking the component containing $a(i+1) = u(i)$ with $a(i) = i + 1$, with the possible exception of $(i + 1, a(i + 1))$.

If we made e permanent in iteration i, then i and $a(i)$ would now be part of the same component. Because of the ordering of the points (and the fact that we have Euclidean distances), the minimum cost edge linking the component containing $u(i) = a(i + 1)$ with the component containing $a(i) = i + 1$ is either $(u(i), i) = e$ or $(i + 1, a(i + 1))$.

Case 2: $i + 1 = u(i)$. This implies that $a(i + 1) = a(i)$. The remainder of the argument is analogous to Case 1.

This completes the proof. □

Theorem 2.8 implies that at the start of iteration i, there do not exist paths from i to $u(i)$ and from i to $a(i)$ in the partially constructed spanning tree. Therefore, adding an edge that connects the component containing i with either $u(i)$ or with the component containing $a(i)$ does not create a cycle. Thus, in each iteration, Algorithm 1 selects a minimum cost edge in a cut, that is, the cut determined by the points in the component containing i, and never creates a cycle. Therefore, because Algorithm 1 selects $n - 1$ edges, upon completion it has constructed a minimum cost spanning tree. Since Algorithm 1 has $n - 1$ iterations and the work in each iteration takes constant time, it has time complexity $O(n)$.

3. Conclusion

We have investigated two routing problems arising in the context of warehouse order picking. They can be viewed as a Euclidean TSP with points on two parallel lines. We have developed a dynamic programming algorithm for its solution that runs in quadratic time. Furthermore, we have considered the MST problem with points on two parallel lines, and shown that it can be solved in linear time. The algorithms are intended to be used as core components in algorithms for the OBP.

References

[1] J. Bartholdi and S. T. Hackman, *Warehouse & distribution science* (2011); http://www.warehouse-science.com.

[2] E. P. Chew and L. C. Tang, "Travel time analysis for general item location assignment in a rectangular warehouse", *European J. Oper. Res.* **112** (1999) 582–597; doi:10.1016/S0377-2217(97)00416-5.

[3] R. de Koster, T. Le-Duc and K. J. Roodbergen, "Design and control of warehouse order picking: a literature review", *European J. Oper. Res.* **182** (2007) 481–501; doi:10.1016/j.ejor.2006.07.009.

[4] R. de Koster, K. Roodbergen and R. van Voorden, "Reduction of walking time in the distribution center of De Bijenkorf", in: *New trends in distribution logistics*, Volume 480 of *Lect. Notes Econom. Math. Syst.* (eds M. G. Speranza and P. Stähly), (Springer, Berlin, 1999) 215–234.

[5] E. Frazelle, *World-class warehousing and material handling* (McGraw-Hill, New York, 2002).

[6] R. W. Hall, "Distance approximations for routing manual pickers in a warehouse", *IIE Trans.* **25** (1993) 76–87; doi:10.1080/07408179308964306.

https://doi.org/10.1017/S1446181115000140 Published online by Cambridge University Press

[7] S. Henn, "Algorithms for on-line order batching in an order picking warehouse", *Comput. Oper. Res.* **39** (2012) 2549–2563; doi:10.1016/j.cor.2011.12.019.
[8] C. G. Petersen II, "An evaluation of order picking routing policies", *Int. J. Oper. Prod. Manage.* **17** (1997) 1098–1111; doi:10.1108/01443579710177860.
[9] H. D. Ratliff and A. S. Rosenthal, "Order-picking in a rectangular warehouse: a solvable case of the traveling salesman problem", *Oper. Res.* **31** (1983) 507–521; doi:10.1287/opre.31.3.507.
[10] K. J. Roodbergen, *Layout and routing methods for warehouses* (RSM Erasmus University, The Netherlands, 2001) hdl.handle.net/1765/861.
[11] J. A. Tompkins, J. A. White, Y. A. Bozer and J. M. A. Tanchoco, *Facilities planning* (John Wiley, Hoboken, NJ, 2003).