Production, Manufacturing and Logistics

Order sequencing on a unidirectional cyclical picking line

Jason Matthews, Stephan Visagie *

Department of Logistics, University of Stellenbosch, South Africa

ARTICLE INFO

Article history:
Received 14 October 2011
Accepted 10 May 2013
Available online 24 May 2013

Keywords:
Distribution
Order-picking
Sequencing
Combinatorial optimization
Unidirectional carousel

ABSTRACT

A real life order-picking configuration that requires multiple pickers to cyclically move around fixed locations in a single direction is considered. This configuration is not the same, but shows similarities to, unidirectional carousel systems described in literature. The problem of minimising the pickers’ travel distance to pick all orders on this system is a variant of the clustered travelling salesman problem. An integer programming (IP) formulation of this problem cannot be solved in a realistic time frame for real life instances of the problem. A relaxation of this IP formulation is proposed that can be used to determine a lower bound on an optimal solution. It is shown that the solution obtained from this relaxation can always be transformed to a feasible solution for the IP formulation that is, at most, within one pick cycle of the lower bound. The computational results and performance of the proposed methods as well as adapted order sequencing approaches for bidirectional carousel systems from literature are compared to one another by means of real life historical data instances obtained from a retail distribution centre. © 2013 Elsevier B.V. All rights reserved.

1. Background and introduction

Order-picking is the process of retrieving products from storage (or buffer areas) in response to a specific customer request (De Koster et al., 2007). It is usually the most resource intensive operation, normally engaging the majority of the overall workforce and therefore accounting for more than 60% of the overall costs in a distribution centre (DC) (Van den Berg and Zijm, 1999). The order-picking system in a DC owned by PEP is considered in this paper (PEP Stores, 2012).

PEP operates approximately 1500 retail stores in South Africa. It predominantly sells apparel and footwear, but also other products ranging from cellular phones to home décor. The customer orders processed by PEP’s DCs are destined for its retail branches. One of the factors that has influenced the current picking system is the size of the products which, in turn, has an impact on the size and number of storage locations for any given product. Moreover, a large number of branches of various sizes and with different customer profiles create a set of non-uniform orders which must be picked by the DC on a regular basis. Fashion trends and the seasonal nature of clothing products create an ever changing product profile at the DC.

A key managerial characteristic of the operations in PEP’s DC is the central planning of its branch level inventory. PEP does not allow branch managers to request stock. A central planning department decides how much of that available stock at the DC to branches. PEP’s planning process therefore focuses on individual stock keeping units (SKUs) rather than individual branches. Once a SKU arrives in the DC, the central planning department decides how much of that available SKU should go to each branch. Instructions containing SKU details and the amounts which have to be dispatched to each branch are then issued to the DC. This prompts the DC to select a subset of these SKUs on a FIFO basis, to be picked in a single run or wave. All the branch requirements for all the relevant SKUs are picked during a specific wave.

The DC uses a picking line to process a wave. Fig. 1 illustrates the picking line in the DC containing m locations. For each wave, a single SKU is assigned to a single location on the picking line. Pickers move in a clockwise direction around a conveyor belt picking the required stock for each branch. Before a wave of picking starts the amounts of each SKU that must be picked for all the branches are known in advance. Each location can store up to 5 pallets of stock. Enough stock is therefore stored at each location before picking commences. Restocks and stock outs rarely occur in practice. However, should more than five pallet loads of a SKU be required to fulfil all the branch requirements, additional stock will be stored in an adjacent location. In this case the warehouse management system will see the two locations as a single, merged location.

A voice recognition system (VRS) is used to direct the pickers around the picking line. The pickers wear headsets and speak to a computer that is trained to their voices. The VRS directs a picker to complete all the picks for a single branch (which constitutes an order). Upon completion the picker will start picking for a new branch (or order). An order may be viewed as the total SKU
requirements of a branch, and is wave specific. When picking an order, the picker will always be sent to the closest required SKU in a clockwise direction. The VRS will ensure that a picker finishes all the picks for an order before being assigned a new order – i.e. that all pickers will sequentially pick orders. While picking, pickers place the stock directly into cartons which are destined for the branches. The moment a picker starts a new carton, a barcode label is stuck onto the carton and scanned with a handheld device. Multiple cartons may be used for a single order, but only the stock for a single order may be placed in the same carton. Packed cartons are placed on the conveying belt which transports them to the dispatch area.

At the start of a wave all the orders for that wave are known and are all picked. Once all the orders in a wave have been completed, there may be left over stock if the total stock assigned to branches is not equal to a multiple of full pallets or if picking inaccuracies have occurred. The left-over stock is taken back to the storage racks and is stored for future use. The picking line is then filled with a new set of SKUs for a new wave of picking to start. Depending on the size of a wave it may take up to 2 days to complete a single wave.

PEP currently sequentially assigns each order in a fixed list to the next available picker. Each order in the list is sequentially assigned to the next available picker. This is done regardless of the previous orders that were picked or the current position of the picker requesting a new order. The stochastic nature of the picking process implies each order is assigned to a random picker, therefore each picker picks a random sequence of orders.

In this study, the problem of determining a sequence of orders to minimise the distance that pickers travel during a wave is considered. Although there are typically multiple pickers in any picking line, pickers may pass each other as there is enough space between the SKUs and the conveyor to allow this. The problem of sequencing multiple orders can then be transformed back into a single picker problem.

We make the following assumptions while modelling PEP’s order picking system.

1. A picker must complete an entire order before starting another. A picker packs stock directly into cartons and therefore order batching (processing several orders simultaneously) is not permissible. Each carton is closed and shipped when the order is completed or the carton is full.
2. The time taken to physically pick a SKU is constant over all the orders.
3. A picker walks at a constant speed.
4. An order may start at any location regardless of whether the order requires the SKU at that location and the order will finish at the last location where a SKU is picked.
5. The time required to switch between orders is constant between all pickers and orders.
6. A picker may not physically pick the first SKU of a new order from the same location as the last pick of the previous order. If the next order requires that same SKU the picker must complete an entire cycle to pick at that location again. This assumption is made based on PEP’s requirements for improved picking accuracy.

If one views the SKUs as moving relative to a static picker, then Assumptions 1–5 define a problem which shows many similarities to the unidirectional carousel systems found in literature. A carousel is an automated warehousing system that is used for the order-picking of small to medium sized products. A carousel system holds different SKUs, which may or may not hold multiple locations within the carousel. The system is serviced by a picker, either human or robotic, who remains at a fixed location. To pick a SKU, the bidirectional carousel is automatically rotated in either direction, such that the required SKU is in front of the picker. A unidirectional carousel can only rotate in a single direction (Bartholdi and Hackman, 2007).

The cyclical set up of the SKUs in the picking line, the automation of pick sequencing by the VRS and the assumption of single picker service resembles a unidirectional carousel system. The main difference between this system and the unidirectional carousel systems in literature is the presence of wave picking. Wave picking implies that all the SKUs in the picking line are picked at least once and all the orders are known. For carousel systems, however, not all SKUs will be picked and new orders may be added to the pending set during a picking operation. In literature, optimisation procedures to reduce overall pick time for carousel systems are based on expected SKU mixes in orders based on historical data (Litvak and Vlasiou, 2010). These expected SKU mixes in orders are often used to determine the optimum locations for products, which is not considered here.

The problem of sequencing a set of orders, which consists of a number of picks, into a sequence minimising the distance or rotations by a bidirectional carousel has been investigated in literature. One of the initial problems investigated is the sequencing of the set of items associated with a single order, otherwise known as a pick strategy, in bidirectional carousels. Bartholdi and Platzman (1986) showed that an optimal sequence can be found in linear time. The Nearest Item (NI) heuristic, where the nearest item is always presented to the picker, was investigated and upper bounds presented for the distance to pick all the SKUs for a single order. Litvak et al. (2001) developed a tight upper bound for the NI heuristic. A heuristic known as the $m$-step method, which forces the carousel to change direction after at most $m$ items have been picked, was shown to outperform the NI heuristic by Litvak and Adam (2002). For PEP’s picking line system the pick strategy is simply to pick the closest SKU in a clockwise direction.

Bartholdi and Platzman (1986) considered the problem of sequencing multiple orders on a bidirectional carousel system. A hierarchical heuristic was developed and proved to be within 1 cycle of an optimal solution. The basis of this heuristic was to force all order to be picked on their shortest spanning interval (SSI). An SSI is a path with the shortest possible distance to pick an order. These

---

1 Minor changes are required in the calculation of predetermined parameters for all models presented here if this assumption is removed. The assumption is included to compare results to historical data.
orders are then linked with the smallest possible rotations by means of shortest matching techniques. A Nearest Order (NO) heuristic, that always selects a nearest unlinked SSI to be picked next, was also tested and upper bounds presented. Van den Berg (1996) also solved this problem of linking SSIs by transforming the problem to a rural postman problem. An effective polynomial time algorithm was developed to generate a solution to this type of rural postman problem. It is shown that this solution requires at most 1.5 cycles more than an optimal solution to the original problem.

Both these aspects of order sequencing and SKU locations within carousel systems are discussed for various configurations by Hassini (2009), who also looks at main operations and applications. Litvak and Vlasiou (2010) mentioned different configurations of carousel systems including a unidirectional system, but points out that bidirectional carousel systems have received the majority of attention in literature, because they are the most common and more efficient.

Based on the fact that there are no methods in literature for unidirectional carousels, the methods for bidirectional carousels are adapted in this paper. A tight lower bound as well as a feasible solution within 1 cycle of a lower bound is then formulated and presented. Several heuristics adapted from literature are further compared and it is shown that PEP’s current approach may be improved. This research has prompted PEP to schedule the implementation of the order sequencing optimisation into their warehouse management system.

In Section 2 several approaches to bidirectional carousels as well as a travelling salesman problem (TSP) variation from literature are adapted for use in the picking line considered here. A cyclical approach is introduced in Section 3, which is shown to produce a solution within at most 1 cycle of a lower bound. All approaches are tested using historical data and compared to historical results from PEP in Section 4. The paper is concluded in Section 5.

2. Adaptations of algorithms in literature

In the previous section several heuristics for order sequencing in a bidirectional carousel were mentioned. In this section two of the most applicable approaches have been adapted for use in PEP’s unidirectional order sequencing problem considered here. In addition, a clustered travelling salesman formulation is introduced together with a known heuristic to solve this formulation.

2.1. SSI linking approach (SSIL)

Following the hierarchical heuristic introduced by Bartholdi and Platzman (1986), the order sequencing problem can be solved by using the assumption that each order should be picked along its SSI. For the picking line problem considered here, an SSI of an order \( k \) is a path around the picking line in a clockwise direction, past the smallest number of locations to pass all the SKUs required by that order. Each SSI has a starting and an ending position. One SSI per order is then selected and all the starting positions are linked to ending positions to create a feasible sequence of orders. The linking of starting and ending positions is accomplished by using a standard TSP formulation where the distance matrix represents the distances between ending and starting positions of orders. This approach is called the SSI linking approach (SSIL).

For bidirectional carousels picking can start from either end of the SSI as the carousel can rotate in both directions and the algorithm will determine from which end to start. For the unidirectional case an SSI can only start from one end. However, multiple SSIs may exist. If multiple SSIs exist, the SSI which results in the shortest linking cost is selected for that order.

2.2. Nearest end heuristic (NE)

Using Bartholdi and Platzman (1986) NI heuristic as pick strategy, and NO heuristic for order sequencing a Nearest End (NE) heuristic is introduced. In the case of bidirectional carousels, the NI heuristic is used for a single order and always picks the nearest item in the order. The NO heuristic relies on predetermined SSIs for each order and will always pick the nearest SSI to the current position. The nearest neighbour philosophy was adapted for the picking line problem by simultaneously considering the order sequence and the item sequence within each order. This was achieved by sequentially selecting the order with the nearest ending position from the current position. This heuristic is a tour construction heuristic, as it gradually builds up a tour, and is described in Algorithm 1.

Algorithm 1. Nearest end heuristic

<table>
<thead>
<tr>
<th>Data:</th>
<th>A set of orders and SKU locations.</th>
</tr>
</thead>
<tbody>
<tr>
<td>for The number of orders do</td>
<td>Search for the order which has the closest ending position if picking starts from the current position; Add this order to the sequence; Update the current position to be the ending position of the newly added order;</td>
</tr>
<tr>
<td>end</td>
<td></td>
</tr>
</tbody>
</table>

2.3. A travelling salesman approach

Consider a digraph \( G \) where SKU \( p \) in order \( k \) is represented by a node \( \nu_{pk} \). Furthermore, let the set of SKUs associated with a single order \( k \) be a cluster of nodes \( \nu_k \). Note that if 2 orders require the same SKU there will be two nodes in \( G \), one for each order. Let the distance between nodes \( \nu_{pk} \) and \( \nu_{qk} \) be the number of locations passed in a clockwise direction between the locations of the two SKUs. Every SKU in each order must be visited consecutively to complete all the orders. This problem may therefore be seen as a clustered travelling salesman problem (CTSP) where a minimum cost closed tour must be found between all nodes (or picks) such that the nodes within each cluster (or order) are visited consecutively (Fischetti et al., 2007). Once the tour enters a cluster all the nodes in that cluster must be visited – corresponding to the action of picking all the SKUs required by the order associated with the cluster. If all the nodes in a cluster are visited the tour must continue to the next cluster (i.e. order) and again all nodes (SKUs) in that cluster must be visited before moving to the next cluster. Because distances between all nodes are the clockwise walking distances between the SKUs which the nodes represent, the length of the closed tour will be the distance that needs to be travelled to pick all the orders.

The CTSP is a variation of the TSP and is therefore NP hard. In addition, the typical size of real life practical problems has in excess of 1200 clusters containing an average of 25 nodes each. An exact approach to the CTSP is not a viable option due to the limited time available to generate a solution in the DC. Guttmann-Beck et al., 2000 proposed several heuristics to solve CTSPs and provides worst case bounds if the CTSP satisfies the triangular inequality. Guttmann-Beck et al. (2000) discussed four main variations; (I) the case where a start and an end node in a cluster is given, (II) two nodes are given from which the starting and ending nodes of the cluster must be selected, (III) only a starting node is given, and (IV) no starting or ending node is given. The fourth case best
describes the problem of sequencing orders on a picking line as no starting or ending locations for an order is given. For variation (IV) the heuristic proposed by Guttmann-Beck et al. (2000) considers two options and selects the best one:

(IV)a. Determine the shortest path for each cluster. Use the two end nodes of each shortest path to solve the problem as the variation where the start and end nodes are given.

(IV)b. Within each cluster select the two nodes which are furthest apart. Follow the solution procedures for the variation where starting and ending nodes are selected from 2 possibilities (i.e. variation (II)).

For the problem considered here, variation (IV)a is precisely the SSIL approach. Therefore only variation (IV)b will be considered, which will be referred to as the TSP approach (TSPA). The two possible positions that are considered for each order, are the starting position and the ending positions of an SSI. These two positions represent the two positions which are furthest apart if the order is started from the ending position of an SSI which results in the longest span for that order. The orders must then be linked up in the shortest number of locations by selecting a suitable starting position for each order.

3. A cyclical approach

For all the adapted methods from literature the objective was to minimise the total number of locations traversed to pick all the orders. When sequencing the orders for a picking line the resulting sequence may, however, consist of a number of interlinking sub-tours. A subtour is defined as a subsequence of the complete order sequence where the starting location of the first order in the subsequence is the same as the ending location of the last order in the subsequence. The length of each subtour is equal to a number of complete cycles around the conveyor belt and thus a multiple of the number of complete cycles. When sequencing the orders for a picking line the resulting sequence is the same as the ending location of the last order in the subsequence. The length of each subtour is equal to a number of complete cycles. The subsequence is the same as the ending location of the last order in the subsequence where the starting location of the first order in the subsequence may, however, consist of a number of interlinking subtours. When sequencing the orders for a picking line the resulting sequence is the same as the ending location of the last order in the subsequence.

A span for order \( k \) may be represented by \( S_k = (i, e_k) \), where \( i \) is the starting location and \( e_k \) the ending location of order \( k \) starting at location \( i \). Furthermore, the idea of a cut is introduced.

Definition 1. The span of an order is the smallest set of locations passed to complete the order, given a starting location.

A span for order \( k \) may start at any location \( i \), but for a given starting position \( i \), the ending position \( e_k \) is fixed and will always correspond to a location containing a SKU required by order \( k \). Thus a starting position for an order has a unique span associated with it, because an order must be completed once it is started. A span for order \( k \) may be represented by \( S_k = (i, e_k) \), where \( i \) is the starting location and \( e_k \) the ending location of order \( k \) starting at location \( i \). Furthermore, the idea of a cut is introduced.

Definition 2. The cut of a location is the number of spans passing that location.

The cut for each location forms a lower bound for the number of cycles needed to pick a set of spans as it represents a minimum number of times a location must be passed to pick all of the spans.

A lower bound (in terms of cycles) for the problem is determined by assigning only starting positions to orders while minimising a largest (or maximal) cut. No pairing of specific orders is done and only after the problem is solved are the starting positions paired up with ending positions of other orders, forcing all orders to follow directly after a unique preceding order with the objective of minimising the maximal cut(s). The removal of these specific order pairings reduces the number of variables by factor \( n \). The following model achieves this objective. Let

\[
X_{ik} = \begin{cases} 
1 & \text{if order } k \text{ starts at location } i \\
0 & \text{otherwise}, 
\end{cases}
\]

and

\[
C \text{ be the maximal cut.}
\]

The following parameters are set in the model. Let

\[
n \text{ be the total number of orders,}
\]

\[
m \text{ be the total number of locations,}
\]

\[
d_{ij} = \begin{cases} 
1 & \text{if order } k \text{ starting at location } i \text{ passes location } j \\
0 & \text{otherwise}
\end{cases}
\]

and

\[
e_{ij} = \begin{cases} 
1 & \text{if order } k \text{ starting at location } i \text{ is completed at location } j \\
0 & \text{otherwise}
\end{cases}
\]

Both \( d_{ij} \) and \( e_{ij} \) are derived based on the definition of a span. In terms of these symbols the objective is to minimise \( C \)

\[
\text{subject to }
\]

\[
\sum_{i=1}^{m} x_{ik} = 1 \quad k = 1, \ldots, n, 
\]

\[
\sum_{j=1}^{m} \sum_{k=1}^{n} d_{ij} x_{ik} \leq C 
\]

\[
\sum_{j=1}^{m} \sum_{k=1}^{n} x_{ij} e_{kj} = 0 \quad j = 1, \ldots, m - 1, 
\]

\[
\sum_{k=1}^{n} x_{ik} - \sum_{j=1}^{m} \sum_{k=1}^{n} x_{ik} e_{km} = 0, 
\]

\[
x_{ik} \in \{0, 1\} \quad i = 1, \ldots, m, \quad k = 1, \ldots, n. 
\]

The objective function (1) minimises the maximal cut. Constraint set (2) ensures that each order is allocated a starting position. Constraint set (3) calculates the size of the maximal cut. Pairing constraint set (4) and constraint (5) ensures that the number of orders starting at a location is equal to the number of orders ending at that location. Therefore all starting and ending positions can be paired with each other. For example, if the starting position \( i \) assigned to order \( k \) results in order \( k \) having an ending position \( j \), there must be an order \( l \) which is assigned a starting position \( j + 1 \). These constraints do not necessarily break subtours or pair up orders but are additional feasibility constraints as a feasible solution requires each order to directly follow after another one.

The parameter \( d_{ij} \) is generated by assigning a value of 1 to all the locations in the span of order \( k \) starting at location \( i \). Thus the method of generating \( d_{ij} \) ensures that Assumption 6 is met.

As an example, consider a carousel with the following SKU configuration 1-2-3-4-5-6 and order \( k \) which requires the following SKUs: 1, 2, 3, 6. A schematic representation of this layout is given in Fig. 2. If order \( k \) starts at location 1 then \( d_{13} = d_{14} = d_{15} = d_{16} = 0, e_{15} = 1 \) and \( e_{16} = 0 \) where \( j = 5 \) must be generated, or if order \( k \) starts at location 5 then \( d_{54} = d_{55} = d_{56} = 0, e_{55} = 1 \) and \( e_{56} = 0 \) where \( j = 5 \) must be generated. In general all locations are considered as starting positions when generating \( d_{jk} \), but only starting locations 1 and 5 are shown here for illustrative purposes.

If Assumption 6 is not included the formulation could be adjusted by replacing pairing constraint set (4) and constraint (5) with

\[
\sum_{k=1}^{n} x_{ik} - \sum_{j=1}^{m} \sum_{k=1}^{n} x_{ik} e_{ij} = 0 \quad j = 1, \ldots, m, 
\]
If order \( k \) starts at location 1 then \( d_{12} = d_{13} = d_{14} = 1, d_{15} = 1 \) if \( j \neq 5 \) must be generated, or if order \( k \) order starts at location 5 then \( d_{52} = d_{53} = d_{54} = 1, d_{55} = 0 \) if \( j \neq 2 \) must be generated.

A novel algorithm, with the general framework supplied in Algorithm 2, uses a solution to formulation (1)–(6) to connect all the starting and ending positions, resulting in a set \( T \) of closed subtours.

All the spans generated by formulation (1)–(6) are assigned to a closed subtour in which each span follows directly on its predecessor. In addition, each location in the subtour is passed an equal number of times as all subtours are closed. Therefore the number of cycles required to complete an individual subtour is precisely the maximal cut associated with the set of spans in that subtour. The number of cycles required to complete all subtours in \( T \) is therefore \( C \).

Lemma 1 states that Algorithm 2 will find a set \( T \) of closed subtours containing all orders such that the size of \( T, |T| \), is at most the number of locations.

**Lemma 1.** Algorithm 2 produces a set \( T \) of closed subtours such that \(|T| \leq m\).

**Proof.** It must be shown that all subtours in the set \( T \) will close and that \(|T| \leq m\). Assume that Algorithm 2 returns an open subtour. This indicates that the ending position of this open subtour has no corresponding starting position, which would indicate that the sets \( S \) and \( E \) (from Algorithm 2) do not fulfill pairing constraint set (4) and constraint (5). This contradicts the assumption that \( S \) and \( E \) were generated by means of formulation (1)–(6). Assume that \(|T| > m\). There is at most \( m \) possible starting and ending positions for subtours. According to the pigeon hole principle there must exist at least two subtours \( u \) and \( v \) in \( T \) which start at the same location \( i \). Assume that subtour \( u \) was generated before subtour \( v \). This implies that at some point in the algorithm subtour \( u \) ended at location \( i \), and closed, where it could have selected the first order in subtour \( v \) as this order was at that stage an unallocated order. This contradicts Algorithm 2 which closes a subtour only once all unallocated orders with a feasible starting position have been allocated to a subtour. \( \square \)

**Algorithm 2.** Subtour generation heuristic

**Input:** A set \( S \) of starting positions and a set \( E \) of ending positions for the spans obtained from formulation (1)–(6).

**Output:** A set \( T \) of subtours that links up all the orders

**while** All orders have not been allocated to a subtour do

Generate a new tabour \( t_i \), with the first available unallocated order;

Let the current ending position of \( t_i \) be location \( j \):

**while** \( t_i \), is not closed do

if An unallocated order exists which has a starting position corresponding to \( j + 1 \) mod \( m \);

then

Add this order to the end of the open subtour;

end else

Close the subtour by connecting the last order to the first order;

end

end

**Lemma 2.** A set \( T \) of closed subtours generated by means of Algorithm 2 forms a single closed tour of length at most \( C + 1 \) cycles.

**Proof.** Order the closed subtours in \( T \) by increasing the value of their starting positions. Connect these subtours by joining the ending and starting positions sequentially using this ordered set. This connection cost of linking the subtours is at most 1 cycle as all locations can be visited in at most 1 cycle. The number of cycles required to complete this single tour is the sum of the number of cycles required by each individual subtour (\( C \) cycles) and the number of cycles required to link all the subtours (1 cycle). Therefore, a set \( T \) of subtours generated by means of Algorithm 2 can be connected to form a single tour of length at most \( C + 1 \) cycles. \( \square \)

**Definition 3.** Adjacent maximal cuts are two maximal cuts such that no maximal cuts are associated with locations between them.

Let \( [i:j] \) represent a set of locations on a picking line starting from, and including, location \( i \) and moving in a clockwise direction to, and including, \( j \). Round brackets excludes the corresponding starting/ending point of the set, for example, \([i + 1:j]\) = \([i:j]\). Definition 3 implies that if cuts \( c_i \) and \( c_j \) are adjacent maximal cuts then at least one of the following two statements must hold: (a) all cuts \( c_w \) with \( w \in [i:j] \) are not maximal, and (b) all cuts \( c_w \) with \( w \in [j:i] \) are not maximal.

Let us consider the following relaxed version of formulation (1)–(6) with \( C \) representing the maximal cut. The objective is to minimise \( C \) subject to

\[
\sum_{i=1}^{m} x_{ik} = 1 \quad k = 1, \ldots, n, \tag{8}
\]

\[
\sum_{i=1}^{m} \sum_{k=1}^{n} d_{ik}x_{ik} \leq C' \quad j = 1, \ldots, m, \tag{9}
\]

\[
x_{ik} \in \{0, 1\} \quad i = 1, \ldots, m, \quad k = 1, \ldots, n. \tag{10}
\]

If all ending positions \( E \) and starting positions \( S \) for an optimal solution to formulation (8)–(11) are connected to generate subtours with length not greater than an optimal value of \( C \), then it is shown that constraint set (4) and constraint (5) do not increase the value of the maximal cut, \( C \).
To arrive at this result it must be shown that for any solution of formulation (8)–(11) there will be an equal number of starting and ending positions between two adjacent maximal cuts. This is achieved by means of Lemma 3.

**Lemma 3.** In a solution to formulation (8)–(11) the number of starting points between any two maximal cuts $c_i$ and $c_j$ is equal to the number of ending points between $c_i$ and $c_j$.

**Proof.** Consider a pair of adjacent maximal cuts $c_i$ and $c_j$ and without loss of generality let all the cuts $c_w$ with $w \in (i,j)$ be non-maximal.

In the case where there exists only 1 maximal cut, that is $i = j$, insert a dummy location next to location $i$ and force all orders picking from location $i$ in the final solution to also pick from the dummy location. A feasible solution is then generated such that a maximal cut is not adjacent to itself without increasing the maximal cut.

Consider all the spans passing at least one of the locations in $[i,j]$. All of these spans are divided into exactly one of 4 sets: (a) the set $A = \{(s_a, e_a) | s_a, e_a \in (i,j)\}$, representing those spans which pass all locations in $[i,j]$; (b) the set $B = \{(s_b, e_b) | s_b, e_b \in (i,j)\}$, representing those spans which pass $i$ and end before $j$; (c) the set $C = \{(s_c, e_c) | s_c, e_c \in (i,j)\}$, representing those spans which pass $j$ and have a starting position after $i$; and (d) the set $D = \{(s_d, e_d) | s_d, e_d \in (i,j)\}$, representing those spans that have both a starting and ending positions between $i$ and $j$.

To illustrate the definitions of these sets, a representative selection of different spans that fall into each set is shown in Fig. 3.

For the sets $A$ and $D$ there is an equal number of starting and ending positions between the maximal cuts. Therefore it must be shown that there is an equal number of starting positions for the spans in sets $B$ and $C$.

Both cuts $c_i$ and $c_j$ are maximal and therefore there are $C'$ spans passing $i$ and $C'$ spans passing $j$. There are $C' = |A| + |B|$ spans which pass location $i$. Similarly, there are $C' = |A| + |C|$ spans which pass location $j$. Therefore $|B| = |C|$, which proves the lemma. □

In Lemma 4 it is shown that the starting and ending positions between two adjacent maximal cuts can be connected without increasing the value of a maximal cut.

**Lemma 4.** A solution to formulation (8)–(11) can be transformed to a feasible solution for formulation (1)–(6) without increasing the objective function value resulting in $C' = C$.

**Proof.** Consider a pair of adjacent maximal cuts $c_i$ and $c_j$ and without loss of generality let all the cuts $c_w$ with $w \in (i,j)$ be non-maximal. The case where there exists only 1 maximal cut is handled as in Lemma 3.

Consider the set of starting positions $\hat{S} = \{s_\hat{a} | s_\hat{a} \in (i,j)\}$ and the set of ending positions $\hat{E} = \{e_\hat{a} | e_\hat{a} \in (i,j)\}$. Order both sets in increasing order. From Lemma 3 it follows that $|S| = |E| \geq |B| = |C|$. Consider any two corresponding ordered elements $e_\hat{a} \in \hat{E}$ and $s_\hat{a} \in \hat{S}$, where $e_\hat{a} = w$ and $w \in (i,j)$. It must be shown that $e_\hat{a} < s_\hat{a}$.

Consider the sets of orders $G = \{(s_g, e_g) | s_g, e_g \in (i,w)\}$ or $e_g \in (i,w)$). These orders fall into exactly one of 4 subsets, namely (a) the set $A = \{(s_a, e_a) | s_a, e_a \in (i,w)\}$ and $e_a \in (i,w)$; (b) the set $B = \{(s_b, e_a) | s_b, e_a \in (i,w)\}$ and $e_b \in (i,w)$; (c) the set $C = \{(s_c, e_c) | s_c \in (i,w)\}$ and $e_c \in (i,w)$; and (d) the set $D = \{(s_d, e_d) | s_d \in (i,w)\}$ and $e_d \in (i,j)\}$. Furthermore, let the set $B' = B \cup \hat{B}$. To illustrate the definitions of these sets, a representative selection of different spans that fall into each set is shown in Fig. 4.

Assume that $e_i \geq s_i$, then there are more starting positions $\{s_j | s_j \in (i,w)\}$ than ending positions $\{e_k | e_k \in (i,w)\}$. Therefore $|A| + |\hat{C}| + |\hat{D}| \geq |A| + |\hat{B}|$. Furthermore $|\hat{B}| = |B| + |B'|$, implying that $|\hat{C}| + |\hat{D}| \geq |B| - |B'|$ and $|C| + |D| \geq |B|$. There are $|\hat{C}| + |\hat{D}| + |B'|$ spans $(s_\hat{a}, e_\hat{a}) \in G$ passing location $w$. In addition there are $C' - |B'|$ spans $(s_i, e_j) \in G$ which pass all locations in $[i,j]$. Thus location $w$. Therefore, $c_{w} = |\hat{C}| + |\hat{D}| + |B'| = C'$ implying that $c_{w}$ is at least as large as a maximal cut which contradicts the assumption that $c_i$ and $c_j$ are adjacent maximal cuts. Therefore $e_i < s_i$ for any two corresponding elements $e_i \in \hat{E}$ and $s_i \in \hat{S}$.

Consider the matching where each element in $\hat{E}$ is matched with its corresponding element in $\hat{S}$. Let the span of the order associated with each starting position be extended (backwards) to the location following the ending position with which it is paired. This new set of spans now satisfies constraint set (4) and constraint (5). Consider all the paths that are created by connecting these new spans associated with each matching. Each path must begin with a span which passes location $i$. If a path starts with another span there would be an unmatched starting position in $S$. Similarly, each path should end with a span that passes location $j$. There are only $|B|$ such starting and ending positions and therefore $|B|$ such paths. Each path passes all locations in $[i,j]$. With the addition of the $C' - |B'|$ spans which pass all locations in $[i,j]$, each cut will have a value of $C'$. If this process is done for all adjacent cuts a feasible solution is found for the maximal cut formulation. Therefore an optimal solution for the relaxed maximal cut formulation can be transformed to a feasible solution for the maximal cut formulation. Furthermore, $C' = C$ because formulation (8)–(11) is a relaxation of formulation (1)–(6). □

4. Computational results

To evaluate the proposed methods, a representative set of historical data sets were obtained from PEP in consultation with management. For reporting purposes the data sets are categorised based on the number of orders. Large data sets have in excess of 1000 orders, medium data sets have between 500 and 1000 orders and small data sets have fewer than 500 orders.

The SSIL, TSPA, NE and the maximal cut approach were tested and compared to the historical results from PEP. All experiments were performed using an Intel (R) Core (TM)2 Duo 3 gigahertz with 3.7 gigabyte ram running Windows XP (Microsoft, 2011). The algorithms were coded in JAVA (Sun Microsystems, 2011) and all mixed integer formulations were solved by means of Lingo 11 (Lindo Systems, 2010). All approaches were solved within a reasonable time (i.e. less than 6 minutes) and are thus suitable for implementation at PEP. Therefore only the solution quality of each approach will be considered when comparing different approaches. The results for all the approaches are displayed in Table 1. The best solution for each data set is printed in bold. A graphical representation of the percentage improvements by the proposed methods in comparison to PEP’s approach is given in Fig. 5.

It is clear from the results in Table 1 that all the approaches outperform PEP’s approach. It is also clear from the results that the method used by PEP displays significantly worse results relative to the alternative approaches for the smaller data sets (i.e. data sets K–V). For all data sets the TSPA approach outperforms the SSIL approach. This is expected as the TSPA allows for orders to be picked on both its SSI and another span, allowing for more options. For the larger data sets the NE heuristic outperforms both the SSIL and the TSPA approach. However, as the number of orders decreases it appears that picking orders on SSIs becomes the better approach as the NE is outperformed by both the SSIL and the TSPA approach.
Fig. 3. A schematic representation of possible spans for each of the sets $\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}$ and $\mathcal{D}$, where locations $i$ and $j$ are adjacent maximal cuts.

Fig. 4. A schematic representation of possible spans for each of the sets $\hat{\mathcal{A}}$, $\hat{\mathcal{B}}$, $\hat{\mathcal{C}}$ and $\hat{\mathcal{D}}$, where locations $i$ and $j$ are adjacent maximal cuts.

Table 1  
The total number of cycles for the maximal cut approach in comparison to the SSIL approach, the NE approach, the TSPA approach and PEP's historical results.

<table>
<thead>
<tr>
<th>Data set</th>
<th>Size (orders, SKUs)</th>
<th>PEP historical</th>
<th>SSIL</th>
<th>TSPA</th>
<th>NE</th>
<th>Maximal cut approach</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>(1262, 49)</td>
<td>1262</td>
<td>1234</td>
<td>1234</td>
<td>1247</td>
<td>1232</td>
</tr>
<tr>
<td>B</td>
<td>(1264, 54)</td>
<td>1255</td>
<td>1253</td>
<td>1237</td>
<td>1232</td>
<td>1226</td>
</tr>
<tr>
<td>C</td>
<td>(1265, 51)</td>
<td>1254</td>
<td>1223</td>
<td>1214</td>
<td>1180</td>
<td>1161</td>
</tr>
<tr>
<td>D</td>
<td>(1263, 56)</td>
<td>1224</td>
<td>1206</td>
<td>1203</td>
<td>1108</td>
<td>1072</td>
</tr>
<tr>
<td>E</td>
<td>(1264, 51)</td>
<td>1234</td>
<td>1202</td>
<td>1174</td>
<td>1123</td>
<td>1069</td>
</tr>
<tr>
<td>F</td>
<td>(1258, 53)</td>
<td>1222</td>
<td>1195</td>
<td>1181</td>
<td>1062</td>
<td>1005</td>
</tr>
<tr>
<td>G</td>
<td>(1260, 56)</td>
<td>1227</td>
<td>1093</td>
<td>1081</td>
<td>1014</td>
<td>955</td>
</tr>
<tr>
<td>H</td>
<td>(1244, 54)</td>
<td>1242</td>
<td>1096</td>
<td>1068</td>
<td>1031</td>
<td>992</td>
</tr>
<tr>
<td>I</td>
<td>(1264, 56)</td>
<td>1202</td>
<td>1071</td>
<td>1022</td>
<td>993</td>
<td>947</td>
</tr>
<tr>
<td>J</td>
<td>(1258, 55)</td>
<td>1177</td>
<td>1195</td>
<td>1170</td>
<td>1104</td>
<td>1025</td>
</tr>
<tr>
<td>K</td>
<td>(943, 63)</td>
<td>640</td>
<td>293</td>
<td>265</td>
<td>295</td>
<td>259</td>
</tr>
<tr>
<td>L</td>
<td>(846, 56)</td>
<td>615</td>
<td>246</td>
<td>233</td>
<td>245</td>
<td>232</td>
</tr>
<tr>
<td>M</td>
<td>(728, 51)</td>
<td>457</td>
<td>183</td>
<td>156</td>
<td>206</td>
<td>152</td>
</tr>
<tr>
<td>N</td>
<td>(396, 63)</td>
<td>224</td>
<td>111</td>
<td>99</td>
<td>138</td>
<td>90</td>
</tr>
<tr>
<td>O</td>
<td>(733, 55)</td>
<td>461</td>
<td>135</td>
<td>126</td>
<td>157</td>
<td>125</td>
</tr>
<tr>
<td>P</td>
<td>(242, 64)</td>
<td>142</td>
<td>53</td>
<td>49</td>
<td>65</td>
<td>45</td>
</tr>
<tr>
<td>Q</td>
<td>(574, 48)</td>
<td>324</td>
<td>102</td>
<td>81</td>
<td>115</td>
<td>80</td>
</tr>
<tr>
<td>R</td>
<td>(90, 48)</td>
<td>40</td>
<td>8</td>
<td>7</td>
<td>9</td>
<td>7</td>
</tr>
<tr>
<td>S</td>
<td>(158, 55)</td>
<td>82</td>
<td>17</td>
<td>16</td>
<td>21</td>
<td>14</td>
</tr>
<tr>
<td>T</td>
<td>(82, 51)</td>
<td>36</td>
<td>11</td>
<td>10</td>
<td>11</td>
<td>8</td>
</tr>
<tr>
<td>U</td>
<td>(80, 56)</td>
<td>38</td>
<td>7</td>
<td>7</td>
<td>8</td>
<td>6</td>
</tr>
<tr>
<td>V</td>
<td>(89, 42)</td>
<td>40</td>
<td>11</td>
<td>10</td>
<td>13</td>
<td>9</td>
</tr>
</tbody>
</table>
for the smaller data sets. For all the data sets the maximal cut approach yields the best solution and thus dominates all the alternative approaches. The poor performance by PEP’s approach for the smaller data sets is due to orders with very few picks, which result in long walking distances between orders where no picking takes place. The range of lengths of spans for each order is large and smaller data sets is due to orders with very few picks, which result in long walking distances between orders where no picking takes place. The range of lengths of spans for each order is large and in long walking distances between orders where no picking takes place. The range of lengths of spans for each order is large and

The computational times, rounded to the closest second, for the SSIL approach, the NE approach, the TSPA approach and the maximal cut approach.

Table 3

<table>
<thead>
<tr>
<th>Data set</th>
<th>Size (orders, SKUs)</th>
<th>SSIL</th>
<th>TSPA</th>
<th>NE</th>
<th>Maximal cut</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>(1262,49,49)</td>
<td>1232</td>
<td>1234</td>
<td>1137</td>
<td>1089</td>
</tr>
<tr>
<td>B</td>
<td>(1264,54,54)</td>
<td>1226</td>
<td>1152</td>
<td>1101</td>
<td>996</td>
</tr>
<tr>
<td>C</td>
<td>(1265,51,51)</td>
<td>1161</td>
<td>1208</td>
<td>1201</td>
<td>1024</td>
</tr>
<tr>
<td>D</td>
<td>(1263,56,56)</td>
<td>1072</td>
<td>1131</td>
<td>1123</td>
<td>1017</td>
</tr>
<tr>
<td>E</td>
<td>(1264,51,51)</td>
<td>1069</td>
<td>1164</td>
<td>1078</td>
<td>1010</td>
</tr>
<tr>
<td>F</td>
<td>(1258,53,53)</td>
<td>1005</td>
<td>1190</td>
<td>1154</td>
<td>976</td>
</tr>
<tr>
<td>G</td>
<td>(1260,56,56)</td>
<td>955</td>
<td>1076</td>
<td>1052</td>
<td>946</td>
</tr>
<tr>
<td>H</td>
<td>(1244,54,54)</td>
<td>992</td>
<td>1069</td>
<td>1028</td>
<td>982</td>
</tr>
<tr>
<td>I</td>
<td>(1264,56,56)</td>
<td>947</td>
<td>1046</td>
<td>964</td>
<td>936</td>
</tr>
<tr>
<td>J</td>
<td>(1258,55,55)</td>
<td>1025</td>
<td>1193</td>
<td>1148</td>
<td>977</td>
</tr>
<tr>
<td>K</td>
<td>(943,63,63)</td>
<td>259</td>
<td>276</td>
<td>239</td>
<td>281</td>
</tr>
<tr>
<td>L</td>
<td>(846,56,56)</td>
<td>232</td>
<td>192</td>
<td>185</td>
<td>225</td>
</tr>
<tr>
<td>M</td>
<td>(728,51,51)</td>
<td>152</td>
<td>141</td>
<td>109</td>
<td>181</td>
</tr>
<tr>
<td>N</td>
<td>(396,63,63)</td>
<td>90</td>
<td>91</td>
<td>74</td>
<td>136</td>
</tr>
<tr>
<td>O</td>
<td>(733,55,55)</td>
<td>125</td>
<td>108</td>
<td>94</td>
<td>141</td>
</tr>
<tr>
<td>P</td>
<td>(242,64,64)</td>
<td>45</td>
<td>46</td>
<td>40</td>
<td>64</td>
</tr>
<tr>
<td>Q</td>
<td>(574,48,48)</td>
<td>80</td>
<td>66</td>
<td>45</td>
<td>97</td>
</tr>
<tr>
<td>R</td>
<td>(90,48,48)</td>
<td>7</td>
<td>2</td>
<td>2</td>
<td>8</td>
</tr>
<tr>
<td>S</td>
<td>(158,55,55)</td>
<td>14</td>
<td>8</td>
<td>6</td>
<td>20</td>
</tr>
<tr>
<td>T</td>
<td>(82,51,51)</td>
<td>8</td>
<td>4</td>
<td>4</td>
<td>11</td>
</tr>
<tr>
<td>U</td>
<td>(80,56,56)</td>
<td>6</td>
<td>3</td>
<td>3</td>
<td>8</td>
</tr>
<tr>
<td>V</td>
<td>(89,42,42)</td>
<td>9</td>
<td>5</td>
<td>3</td>
<td>11</td>
</tr>
</tbody>
</table>

is the fastest. All the computation times are, however, sufficiently fast to be used in practice for planning a picking line.

5. Conclusion

A problem that is a variant of a carousel order-picking system in which the carousel is unable to rotate in two directions (i.e. a unidirectional carousel) was identified in industry. The sequencing of the orders may be modelled as a clustered travelling salesman problem (CTSP), but the problem sizes from industry are too large to solve exactly within realistic times. Following the logic of methodologies for heuristics to solve the CTSP a relaxed formulation was introduced based on the concept of a maximal cut. This formulation finds good spans for orders and initially ignores subtours between orders, and is solvable in reasonable time. The solution obtained from this formulation is then transformed into a feasible solution for the exact formulation and it was proven that this transformation adds at most one cycle to the lower bound. The proposed method can thus always supply a solution within one cycle of a lower bound to the problem. Computational results are presented which illustrates the improved performance of the proposed solution method to adapted bidirectional carousel heuristics. The proposed method also outperforms historical results obtained from industry.

PEP has subsequently scheduled the implementation of the maximal cut methodology into their warehouse management system. The concept of assigning starting positions to each order before picking commences has been shown with simulation to be robust for use in a multiple-picker context. PEP has opted for the NE heuristic to determine the preferred starting positions. It is the easiest to implement in their current warehouse management system and it supplies comparatively good results relative to the SSIL and TSPA.

Following the solution to the order sequencing problem a natural extension is to determine optimal storage locations for the SKUs in the picking line. The concept of a maximal cut may be used to generate solution techniques and lower bounds for this SKU location problem while minimising picking time. A further problem is the scheduling of SKUs into available picking lines. The SKU scheduling problem is influenced by multiple business objectives such as clustering of SKUs from similar departments, SKU
prioritisation over the entire business, work balance in the DC as well as picking line efficiency.

References


