A Zero-Inventory Production-Distribution Problem with Time-Window Constraints

R. Armstrong, L. Lei and S. Gao
Rutgers Business School, Rutgers University

In this paper, we study a sub-problem of integrated production and distribution for a perishable product that cannot be inventoried. The manufacturing facility has a limited production rate, and the delivery truck has a non-negligible traveling time between locations. The order in which customers may receive deliveries is fixed. Each customer requests a delivery quantity and a time window for receiving the delivery. Since the production facility and the shipping trucks are limited resources, not all the customers may receive the delivery within their specified time windows. The problem is then to choose a subset of customers from the given sequence to receive the deliveries to maximize the amount of demand satisfied without violating the product lifetime, the production capacity, and the delivery time window constraints. The general version of the problem is NP-hard even with relaxed time windows and instantaneous truck traveling time. We analyze the properties of this problem and show the conditions under which optimality is achieved. A greedy heuristic is proposed based on this analysis. Empirical observations on the error gaps achieved by this heuristic are reported.

(Key words: integrated production and distribution, time window constraints, zero-inventory, heuristics, empirical studies)

1. Introduction

Many business applications involve make-to-order manufacturing and distribution operations, where customers accept deliveries only within specified time windows (Ghiani, et al. [1]). Scheduling the operations to meet such time window constraints becomes challenging when product inventory is not allowed. The manufacturing practice that motivated our study on this type of scheduling problem is the one encountered at a company that produces various adhesive chemicals. One of the products, an adhesive material used for making plywood panels, has a life span of 7 days, after that the strength of adhesiveness degrades quickly. During the house construction season, business customers place orders with the company and specify the time windows within which they expect to receive the delivery. Due to the nature of the product, no inventory is maintained and the production and distribution operation schedules must be highly coordinated. When the production facility has a limited production rate and when the transportation time is not instantaneous, any inefficiency in this integrated schedule may either cause the product to expire before it reaches the customers.
(in which case the production resource is wasted and additional expenses will be incurred to provide an environmental safe disposal of expired products), or the delivery may fail to meet the customer’s receiving time windows (in which case the contract with the customer is cancelled or a special arrangement must be made). There are many other examples of such short shelf-life (perishable) products. Radioactive material in biomedical research and clinic applications typically have a short shelf-life (e.g., 8 days for Iodine 131 and only 64 hours for Yttrium 90 (from [www.perkinelmer.com]). Bakery products, with a shelf life of greater than seven days may suffer loss or significant reduction in marketability. A 36-hour delay in shipping farmland products would eliminate as high as 40% of their shelf-life (from [www.fda.gov]).

The resulting production and distribution problem (PDP) for short shelf-life products is concerned with when and how much to produce and which truck route to follow so that a specified objective function (e.g., the total cost, the fleet size, the total time span, or the total amount of demand, etc) is optimized while all the constraints are satisfied. When the production and transportation capacities are constrained and the customer receiving time windows are fixed, it is sometimes necessary to give up the orders by certain customers. Customers whose orders cannot be satisfied must be notified immediately so that they may seek another supplier or work out internally to allow a change in the delivery time window. Optimally solving such an integrated scheduling problem subject to the given constraints is usually difficult due to its inherent combinatorial nature.

In this paper, we consider a subproblem of this integrated production and distribution problem. We are interested in coordinating the operations of a production line and a single delivery truck with a given customer sequence which specifies the relative orders that customers may receive the deliveries. The given customer sequence is represented as $S = \langle 1, 2, 3, ..., n-1, n \rangle$, where $n$ denotes the total number of customers to be evaluated for receiving deliveries. By a given customer sequence, it means that for all $i, j \in S$, $i < j$, if both customers $i$ and $j$ receive the deliveries in a distribution schedule, then $i$ should receive the delivery before $j$ does. Real life situations with a fixed customer sequence occur, for examples, when the customer orders are handled in a first come first serve basis or when the customers are geographically located along an interstate highway. Customer $i$, $i \in S$, has demand $d_i$ and a specified receiving time window $[a_i, b_i]$ within which a delivery can be realized. That is, if customer $i$ receives a delivery,
then whenever the truck arrives at \(i\) before \(a_i\), a waiting time is incurred, and whenever the truck arrives after \(b_i\), the delivery is rejected and the demand is lost. The plant that produces the product has a limited production rate \(r > 0\), so that it takes \(d_i / r\) time units to produce for customer \(i\). No partial order fulfillment is allowed. Items once produced must be delivered to the respective customer within \(B\) time units or the product will expire (i.e., the life span constraint). We assume the truck has enough capacity to carry all the supplies to the \(n\) customers in set \(S\). Therefore, the truck capacity is not a concern.

Since both the manufacturing and transportation resources are limited and the time windows could be tight, there may be a subset of customers in set \(S\) that will not be served. The resulting problem is to choose a subset of customers, \(\sigma\), where \(\sigma \subset S\), to receive the deliveries so that the total amount of feasibly satisfied demand is maximized. We denote this problem as \(P\). Note that for any given subset \(\sigma\), the production and distribution schedule is known and fixed. Therefore, any feasible subset \(\sigma\) that satisfies the production capacity, the product life span, and the delivery time window constraints, defines a feasible solution/schedule for \(P\).

**Example 1.** Suppose that the given sequence of customer orders is \(S=\langle A, B, C, D, E\rangle\), \(r=10\), and the customer-specific data are listed in the table below:

<table>
<thead>
<tr>
<th>Customer j</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>(d_k)</td>
<td>30</td>
<td>40</td>
<td>60</td>
<td>30</td>
<td>40</td>
</tr>
<tr>
<td>(a_k)</td>
<td>19</td>
<td>26</td>
<td>31</td>
<td>32</td>
<td>38</td>
</tr>
<tr>
<td>(b_k)</td>
<td>24</td>
<td>32</td>
<td>36</td>
<td>35</td>
<td>39</td>
</tr>
</tbody>
</table>

Let \(\sigma=S\), by which the delivery truck leaves the plant as soon as the order for customer E is completed. Since the product life span starts as soon as a customer order is completed, the product for customer B would expire before the truck arrives at the customer. In addition, the truck will arrive at customers D and E later than the latest delivery time. Therefore, \(\sigma\) is not a feasible schedule (see Figure 1a).
One of the feasible solution/schedule to this example problem is achieved by deleting customer E, or let \( \sigma' = S/\{E\} = \langle A, B, C, D \rangle \), which satisfies all the constraints and results in a total amount of unsatisfied demand of 4 units. Note that with the new schedule \( \sigma' \), there will be a waiting time at customers C (see Figure 1b). The optimal solution to this example problem is given by \( \sigma^* = S/\{D\} = \langle A, B, C, E \rangle \), which results in the minimum amount of unsatisfied demand of 3 units.

Identifying the optimal subset of customers to be deleted becomes difficult when \( n \) is large. In section 2, we show the general version of \( P \) is NP-hard, even though it becomes trivial when \( B = b_i = \infty, \forall i \in S \).

In the literature, most published results on PDP can be classified into the two categories, those assuming negligible transportation cost but a non-negligible transportation time and those assuming an instantaneous transportation time but a
non-negligible cost. An early work on PDP with non-negligible transportation time was by Potts [2]. He considered the case with a single machine but sufficient number of vehicles so that once a job is completed it is transported to the customer without the need of consolidating the deliveries (i.e., one job per direct shipment trip). The objective was to sequence the jobs on that single machine so that the makespan (including the delivery time to the last customer) is minimized. A heuristic and worst case performance bound were discussed. Heuristics and approximation algorithms on extensions of this classical model were given by Hall and Shmoys [3], Woeginger [4], and Zdrzalka [5]. Lee and Chen [6] showed that most of these variations are NP-hard, but a few cases are (pseudo) polynomial time solvable. Another important subarea in PDP assumes a negligible shipping time but considers the transportation cost. A classical model of this (Pinedo [7]) is the single machine job sequencing problem with an objective to optimize a machine scheduling performance measure plus the transportation cost as a function of the number of delivery batches subject to a batch size. Complexity analysis and algorithms for different variations of this classical model were studied by Cheng and Gordon [8], Cheng, Gordon and Kovalyov [9], Cheng, Kovalyov and Lin [10], Herrmann and Lee [11], Yuan [12], and Wang and Cheng [13]. A more recent paper related to this area is given by Hall and Potts [14]. They considered a three-stage supply chain process with a supplier, manufacturer and customers. The PDP problems posted from a supplier’s view, the manufacturer’s view, and entire chain’s view were discussed. Both special cases with polynomial algorithms and general case complexity analysis were presented. A comprehensive review on the related work can be found in Chen [15].

The general PDP that requires an integrated simultaneous optimization of production sequencing and transporter routing is even harder to solve. Very few related studies in this regard are available in the literature. Among those, Hurter and Van Buer [16], and Van Buer, Woodruff and Olson [17], proposed heuristics for coordinating the two type of schedules (production and distribution) for a problem encountered in newspaper industry. Geismar, Lei and Sriskadarajah [18] studied an integrated optimization problem with a perishable product and the truck routing issues. The objective is to minimize the total time needed to produce and to deliver to a given set of customers (i.e., makespan) without the time window constraints. They proved that the problem in general is NP-hard in strong
sense, and proposed and evaluated a set of greedy heuristics against the lower bounds on the optimal solutions.

In section 2, we analyze the properties of $P$ and the conditions of optimality, upon which a greedy search algorithm is proposed in section 3. The empirical observations on the computational performance of this algorithm under randomly generated test cases are reported in section 4, and finally in section 5 we discuss future extensions from this study.

2. Properties of $P$

We shall start our analysis with a formal definition of $P$. Let

$Y_{i,j} = 1$: If the truck visits customer $j$ immediately after visiting customer $i$;

$w_j$: Waiting time at customer $j$ if $Y_{ij} = 1$, for some $i < j$, and if the truck arrives at customer $j$ before $a_j$. Let indices 0 and $n+1$ denote the plant, and let $d_{n+1} = 0$;

Then, $P$ can be stated as the following mixed integer program.

$$\text{Max. } \sum_{i=0}^{n-1} \sum_{j=i+1}^{n} Y_{i,j} \cdot d_j \quad \text{// Total amount of satisfied demand} \quad (1)$$

Subject to:

Time window constraints for delivery times

$$\frac{1}{r} \sum_{i=0}^{n-1} \sum_{j=i+1}^{n} Y_{i,j} \cdot d_j + \sum_{i=0}^{k-1} \sum_{j=i+1}^{k} Y_{i,j} \cdot \tau_{i,j} + \sum_{i=0}^{k-1} w_i \geq a_k \cdot \sum_{i=0}^{k-1} Y_{i,k} \quad k = 1, 2, \ldots, n \quad (2)$$

$$\frac{1}{r} \sum_{i=0}^{n-1} \sum_{j=i+1}^{n} Y_{i,j} \cdot d_j + \sum_{i=0}^{k-1} \sum_{j=i+1}^{k} Y_{i,j} \cdot \tau_{i,j} + \sum_{i=0}^{k-1} w_i \leq b_k + M \cdot (1 - \sum_{i=0}^{k-1} Y_{i,k}) \quad k = 1, 2, \ldots, n \quad (3)$$

Product life-span constraint

$$\frac{1}{r} \sum_{i=k}^{n-1} \sum_{j=i+1}^{n} Y_{i,j} \cdot d_j + \sum_{i=0}^{k-1} \sum_{j=i+1}^{k} Y_{i,j} \cdot \tau_{i,j} + \sum_{i=0}^{k-1} w_i \leq B, \quad k = 1, 2, \ldots, n \quad (4)$$
Flow balance constraints

\[\sum_{h=0}^{i-1} Y_{h,j} = \sum_{j=i+1}^{n+1} Y_{i,j} \quad i = 1, 2, \ldots, n \quad (5)\]

\[\sum_{h=1}^{n} Y_{0,h} = 1, \quad \sum_{h=1}^{n} Y_{h,n+1} = 1 \quad (6)\]

\[Y_{i,j} = 0 \text{ or } 1, \quad w_i \geq 0, \quad \forall i, j, k \in S\]

While the relative order in which customers may be visited, \( S \), is fixed, problem \( P \) is non-trivial. To see this, consider a special case in which we completely relax the time windows, allow an instantaneous traveling time, and let \( d_1 > \sum_{j=2,3,\ldots,n} d_j \) so that customer 1 always receives the delivery at the optimal solution. After the order for customer 1 is completed, it expires in \( B \) time units. Now, let \( \sum_{i=1,\ldots,j} Y_{i,j} = X_j \), where \( X_j = 1 \) if customer \( j \) is selected to receive the delivery and \( X_j = 0 \) otherwise, \( \forall j \in S/\{1\} \). Then, \( P \) becomes:

\[
\text{Max.} \sum_{j \in S/\{1\}} d_j X_j,
\]

Subject to:

\[\sum_{j \in S/\{1\}} d_j X_j \leq r \cdot B, \quad X_j \in \{0,1\}, \quad \forall j \in S/\{1\},\]

or a binary knapsack problem.

\( P \) has some interesting properties. To identify these properties, let \( \sigma = \langle [1],[2],\ldots,[m] \rangle >, m \leq n, \ 1 \leq i \leq n, \ 1 \leq i \leq m, \) and \( \sigma \subset S \), be a subset of \( S \), which preserves the order of \( S \) and defines a production and transportation schedule. For any given schedule \( \sigma \) and customer \([j], [j] \in \sigma \), let’s calculate the following:

**Earliest delivery time at** \([j]\): \( T_{[j]} = \max \{a_{[j]}, T_{[j-1]} + \tau_{[j-1],[j]}\} \quad (7)\)

**Waiting time at** \([j]\): \( w_{[j]} = \max \{0, a_{[j]} - T_{[j-1]} - \tau_{[j-1],[j]}\} \quad (8)\)

**Tardiness at** \([j]\): \( \Delta_{[j]} = \max \{0, T_{[j]} - b_{[j]}\} \quad (9)\)

**Life-span lateness of** \([j]\): \( l_{[j]} = \max \{0, T_{[j]} - \left[\sum_{\forall [i] \in \sigma, i \leq j} d_{[i]}\right]/r + B\} \quad (10)\)

where in (10), quantity \( \sum_{\forall [i] \in \sigma, i \leq j} d_{[i]} / r \) defines the time point at which the life
span of the order for customer \([j]\) starts. Let \(T_{[0]} = T_0 = (\sum_{i \in \sigma} d_{[i]})/r\) stands for the time that the truck leaves the plant (i.e., node 0). If \(\sigma\) is a feasible schedule, then it must satisfy constraints (2)-(6), and ensure \(\Delta_i = 0, l_i = 0, \forall i \in \sigma\). Let

\[
G(\sigma) = \sum_{i \in \sigma} d_i
\]

be the objective value of a feasible \(\sigma\). In addition, we assume that the triangle inequality holds for the traveling times, or \(\tau_{i,k} + \tau_{k,j} \geq \tau_{i,j}, \forall i,j,k \in S \cup \{0\}\).

**Definition 1.** (The slack of node \(j\)) For any given schedule \(\sigma\) and node \(j, j \in \sigma\), the slack of node \(j\) is defined as \(s_j = T_j - a_j\).

By definition 1, slack \(s_j\) denotes the maximum reduction in the delivery time, \(T_j\), at node \(j\) (i.e., customer \(j\)) achieved by deleting some other node(s) to make the truck deliver to \(j\) earlier, which however cannot be earlier than \(a_j\).

Now, let \(Z\) be an arbitrary tardy node in \(\sigma\) with \(\Delta_Z > 0\), and node \(j\) be a predecessor of \(Z\) with a slack \(s_j \geq 0\) (see Figure 2), and consider the following result.

![Figure 2](image_url)

**Proposition 1:** For any given schedule \(\sigma\) and any pair of nodes in \(\sigma\), \(j\) and \(Z\), with \(j < Z\) and \(\Delta_Z > 0\). If \(s_j < \Delta_Z\) then deleting node \(i\), where \(i < j\) and \(i \in \sigma\), does not change the infeasibility of \(Z\).

**Proof.** Let \(T'_{j}\) and \(\Delta'_{Z}\) be the new delivery time to node \(j\), and the new tardiness of node \(Z\), after node \(i\) is deleted. Then, we have \(T'_{j} \leq T_j\) by triangle inequality, and \(T_Z - T'_{Z} \leq T_j - T'_{j}\) by (7) and the fact that \(j < Z\). Since

\[
\Delta'_Z = \Delta_Z - (T_Z - T'_{Z}) \geq \Delta_Z - (T_j - T'_{j}) \geq \Delta_Z - s_j
\]

and \(\Delta_Z - s_j > 0\), we have \(\Delta'_Z > 0\), and the infeasibility of node \(Z\) remains after \(i\) is deleted.

\[\Diamond\]
Proposition 1 indicates that it is inefficient to delete node \( i \) to exchange for the feasibility of node \( Z \) if there exists a node \( j \) between nodes \( i \) and \( Z \) that \( 0 \leq s_j < \Delta_Z \). The results of Proposition 1 also lead to the claim by Proposition 2 below (with proof skipped).

**Proposition 2.** For any given schedule \( \sigma \) and node \( k \in \sigma \), as other nodes are deleted from \( \sigma \), the slack and the delivery time of node \( k \) monotonically decrease.

**Example 2.** Consider the time windows and traveling time between the locations of four consecutive customers, \( i-1 \), \( i \), \( i+1 \) and \( i+2 \) in a given schedule \( \sigma \) (see Figure 3). Suppose the delivery time to node \( i-1 \) is \( T_{i-1} = a_{i-1} = 80 \). Then, before node \( i \) is deleted, we have
\[
\begin{align*}
  w_i &= 8, \quad T_i = 98, \quad s_i = 0, \quad \Delta_i = 0 \\
  w_{i+1} &= 0, \quad T_{i+1} = 113, \quad s_{i+1} = 13, \quad \Delta_{i+1} = 0 \\
  w_{i+2} &= 0, \quad T_{i+2} = 125, \quad s_{i+2} = 9, \quad \Delta_{i+2} = 4
\end{align*}
\]
After node \( i \) is deleted, we have
\[
\begin{align*}
  w'_{i+1} &= 2, \quad T'_{i+1} = 100, \quad w'_{i+2} = 4, \quad T'_{i+2} = 116, \quad \text{and} \quad s'_{i+1} = s'_{i+2} = 0.
\end{align*}
\]

**Figure 3. An illustrative example of Proposition 2**

Now, let \( X = \langle X_1, X_2, \ldots, X_n \rangle \in \{0,1\}^n \) be a binary vector. Let \( \Delta(X) = \sum_{1 \leq i \leq n} \Delta_i \cdot X_i \), \( L(X) = \sum_{1 \leq i \leq n} l_i \cdot X_i \), and \( G(X) = \sum_{1 \leq i \leq n} d_i \cdot X_i \), where \( \Delta_i \) and \( l_i \) are defined by (9) and (10), respectively.

**Proposition 3 (Condition of optimality).** If \( X^* = \langle X_1^*, X_2^*, \ldots, X_n^* \rangle \) is an optimal solution to \( P \), then for every binary vector \( X = \langle X_1, X_2, \ldots, X_n \rangle \in \{0,1\}^n \), we have either \( \Delta(X) > 0 \), \( L(X) > 0 \), or \( G(X) \leq G(X^*) \).
An optimal solution to $P$, $X^*$, defines a subset of $S$

$$\sigma^* = [1, 2, \ldots, m]$$

so that $[i] \in \sigma^*$ if and only if $X_i^* = 1$. The property of optimality stated in Proposition 3 implies that exchanging any element of $\sigma^*$ for any element in $S/\sigma^*$ either violates the feasibility or does not improve the value of $G(\sigma^*)$. This condition together with the properties of $P$ stated in Propositions 1 and 2 lay a foundation for our greedy search algorithm to be discussed in the next section.

### 3. A greedy Heuristic for $P$

To start, let $\sigma^*$ be the optimal schedule that solves $P$, $\phi^* = S/\sigma^*$ be the subset of customers/nodes deleted under schedule $\sigma^*$, and $G(\sigma^*)$ be the maximum amount of feasibly satisfied demand achieved by $\sigma^*$.

The heuristic that we propose for solving $P$ searches for $\phi^*$, instead of $\sigma^*$, via an iterative search process. Each iteration starts with a given schedule $\sigma$ and initially $\sigma = S$. Let $G(\sigma)$ be the total amount of feasibly satisfied demand by $\sigma$, then $G(\sigma) = \sum_{j \in \sigma} d_j$ if schedule $\sigma$ is feasible and $G(\sigma) < \sum_{j \in \sigma} d_j$ if $\sigma$ contains at least one infeasible delivery. Since some previously infeasible nodes may become feasible as we delete an infeasible node from $\sigma$, the delivery times to nodes during the calculation for the value of $G(\sigma)$ are updated whenever a node is deleted (starting from the first infeasible node in a given $\sigma$).

During each iteration, our proposed heuristic, called *Algorithm H*, searches for the pair of nodes, $(i^*, k^*)$, where $1 \leq i^* \leq k^* \leq n$, $(i^*, k^*) \subset \sigma$, and $i^* = k^*$ is allowed, to delete from $\sigma$ so that the value of $G$ can be improved by the most. That is

$$G(\sigma / \{i^*, k^*\}) = \text{Max.}\{G(\sigma / \{i, k\}) \mid \forall i, k \in \sigma\} \quad (11)$$

After nodes $(i^*, k^*)$ are permanently deleted from $\sigma$, the reduced schedule $\sigma' = \sigma / \{i^*, k^*\}$ becomes the given schedule for the next iteration. The search continues until $\sigma$ becomes feasible and the deliveries to the remaining nodes in $\sigma$ satisfy (2)-(6).
As we discussed in section 2, when \( n \) is large, an exhaustive enumeration of all possible pairs of \((i, k)\) in each iteration for identifying \((i^*, k^*)\) could be inefficient. To handle this issue, the following result together with Proposition 1 can be applied to speed up the search for \((i^*, k^*)\) in each iteration without sacrificing the solution quality.

Let \( \sigma \) be any given candidate schedule containing nodes \( i', i, v \) and \( z \), where \( i' < i < v < z \) and \( z \) is a tardy node/customer.

**Proposition 4.** For any given candidate schedule \( \sigma \) and nodes \( i' < i < v < z \) in \( \sigma \) with \( s_v < \Delta_z \), if \( |\phi^*| \leq 2 \) and \( z \in \sigma^* \), then \( \phi^* \neq \{i', i\} \).

**Proof.** (The proof is similar to that for Proposition 1 and thus skipped).

Proposition 4 implies the following. If \( |\phi^*| \leq 2 \), \( z \) is a tardy node (see Figure 2 in section 2) and \( v \) is the nearest predecessor of \( z \) with \( s_v < \Delta_z \), then the optimal solution \((i^*, k^*)\) can only be one of the following pairs:

1. \( \{(i, k)|1 \leq i \leq v, k = v\} \)  
2. \( \{(i, k)|1 \leq i < v, v < k < z\} \)  
3. \( \{(i, k)|v \leq i \leq k \leq n\} \)

Let \( \sigma, |\sigma| \leq |S| \), be the candidate schedule to be evaluated, \( \phi(\sigma) \) be the set of nodes that are deleted from \( \sigma \) during the search process by \( H \), and \( G(\sigma) \) be the total amount of feasibly satisfied demand with \( G(\sigma) \leq \sum_{j \in \sigma} d_j \). Then, the search algorithm \( H \) can be outlined as follows.

---

**Algorithm H:**  
{Initially, let \( \sigma = S \), \( \phi(\sigma) = \emptyset \})

{Input: \( \sigma \), \( \phi(\sigma) \), and \( G(\sigma) \), and let \( G^{low} = G(\sigma) \})

**Step 1.** Given current schedule \( \sigma \), evaluate \( \phi(\sigma) \). If \( \phi(\sigma) = \emptyset \), terminate the search and output \( \sigma \) with \( G^H = G(\sigma) \). Otherwise, for the first \( Z, Z \in \phi(\sigma) \), applying (12a)-(12c) to choose the pair of nodes, \((i^*, k^*)\), to permanently delete from the given \( \sigma \).

//Applying (12a)-(12c) to choose \((i^*, k^*)\) excludes those pairs of nodes that are inferior//

**Step 2.** If \( G(\sigma / \{i^*, k^*\}) > G^{low} \), then permanently eliminate \((i^*, k^*)\) from \( \sigma \), and let
σ \Leftarrow σ / \{i^*, k^*\} and G^{low} = G(σ / \{i^*, k^*\}). If G(σ / \{i^*, k^*\}) = G^{low}, then eliminate tardy node Z from σ, and let σ \Leftarrow σ / \{Z\} without changing the value of G^{low}. Return to Step 1;

Note that each iteration permanently deletes at least one node, the search by algorithm H terminates therefore in no more than n iterations. Since within each iteration, the number of pairs of nodes being evaluated is bounded from above by \(O(n^2)\), the computational complexity of Algorithm H is thus \(O(n^3)\). Also note that during the search process, the inequalities

\[ G(S) \leq G(σ_1) \leq \ldots \leq G(σ_i) \leq G(σ_{i+1}) \leq \ldots \leq G^H \]

hold, where \(σ_i\) and \(σ_{i+1}\) are schedules to be evaluated in two consecutive iterations during the search process. In addition, we have the following result (proof skipped).

**Proposition 5.** (Conditional Optimality of Algorithm H) If \(φ* \leq 2\), then \(σ_1 = σ^*\) and algorithm H finds the optimal solution to \(P\) after the first iteration.

However, as we can see, if \(φ* > 2\), then the optimality of the solution by Algorithm H is no longer guaranteed. In that case, the algorithm terminates with a greedy feasible solution. Example 3 below illustrates the use of Algorithm H to search for a feasible solution to \(P\).

**Example 3.** Let the given customer sequence be \(S = <1, 2, 3, 4, 5, 6, 7, 8, 9, 10>\), \(B = 630\), and \(r = 10\), with following set of customer demand and time windows:

<table>
<thead>
<tr>
<th>Customer j</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>(d_k)</td>
<td>90</td>
<td>29</td>
<td>53</td>
<td>27</td>
<td>36</td>
<td>89</td>
<td>45</td>
<td>18</td>
<td>84</td>
<td>38</td>
</tr>
<tr>
<td>(a_k)</td>
<td>128</td>
<td>182</td>
<td>274</td>
<td>298</td>
<td>388</td>
<td>480</td>
<td>534</td>
<td>538</td>
<td>636</td>
<td>726</td>
</tr>
<tr>
<td>(b_k)</td>
<td>144</td>
<td>239</td>
<td>288</td>
<td>385</td>
<td>447</td>
<td>487</td>
<td>560</td>
<td>603</td>
<td>657</td>
<td>756</td>
</tr>
</tbody>
</table>

Iteration 1: Given the initial schedule \(σ = S = <1, 2, 3, 4, 5, 6, 7, 8, 9, 10>\), we find \(i^* = 4, j^* = 8\) and \(G(σ / \{4, 8\}) = \max\{G(σ / \{i, j\}) \mid \forall i, j \in σ\} = 406\). We permanently delete these two nodes, which gives \(σ = <1, 2, 3, 5, 6, 7, 9, 10>\) for the next iteration.
Iteration 2: Given schedule $\sigma =<1,2,3,5,6,7,9,10>$, we find $i^* = j^* = 10$ and $G(\sigma/\{10\}) = \max \{G(\sigma/\{i,j\}) | \forall i,j \in \sigma\} = 406$ with $\Delta_k = 0, l_k = 0, \forall k \in \sigma/\{10\}$. Permanently delete node 10 from $\sigma$ and stop the search with schedule $\sigma = <1,2,3,5,6,7,9>$, which is also the optimal solution to $P$ in this example.

To observe the empirical performance of Algorithm H, we randomly generated 490 test cases where the number of customers, $n$, ranged from 10 to 80. The customer demands were sampled from Uniform (100, 1500). The production rate, $r$, ranged from 10 to 50, while the product life span, $B$, ranged from 500 to 800. The truck traveling times were generated based upon Euclidean norm distances. Each customer time window was defined by a beginning time, $a_i$, and an ending time, $b_i$, $1 \leq i \leq n$, where the values of $a_i$ were randomly sampled from a Uniform distribution with a mean equal to the total truck travel time from plant to customer $i$ along the given sequence of $n$ customers, $S$. The widths of customer time windows were randomly sampled from Uniform $[0.05T, 0.2T)$, where parameter $T$ stands for the total travel time needed to visit all the $n$ customers. Given the randomly generated width for each customer window, $\omega_i$, the value of $b_i$, $1 \leq i \leq n$, was then set to be $a_i + \omega_i$.

<table>
<thead>
<tr>
<th>$n$</th>
<th>5% Unfulfillment Rate</th>
<th>15% Unfulfillment Rate</th>
<th>20% Unfulfillment Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CPLEX Time</td>
<td>Heuristic Time</td>
<td>Average Gap</td>
</tr>
<tr>
<td>10</td>
<td>0.18</td>
<td>1.42</td>
<td>0.00%</td>
</tr>
<tr>
<td>20</td>
<td>20.64</td>
<td>3.33</td>
<td>0.00%</td>
</tr>
<tr>
<td>30</td>
<td>109.1</td>
<td>5.15</td>
<td>0.17%</td>
</tr>
<tr>
<td>40</td>
<td>1506</td>
<td>9.47</td>
<td>0.19%</td>
</tr>
<tr>
<td>50</td>
<td>3227</td>
<td>13.40</td>
<td>-0.17%</td>
</tr>
<tr>
<td>60</td>
<td>3600</td>
<td>20.64</td>
<td>-1.26%</td>
</tr>
<tr>
<td>70</td>
<td>3600</td>
<td>39.66</td>
<td>-2.03%</td>
</tr>
<tr>
<td>80</td>
<td>3600</td>
<td>45.98</td>
<td>-2.42%</td>
</tr>
</tbody>
</table>

Table 3. Empirical Observations on the Performance of Algorithm H.

Table 3 shows the average deviation of the heuristic solutions from the optimal solutions obtained by CPLEX Solver (on a 1.2GHz Dell Latitude). The empirical observations are listed against the number of customers and the average un-fulfillment ratio at the levels of 5%, 15% and 20%, respectively, where the average un-fulfillment ratio stands for $|\phi^*|/|S|$ averaged from ten test cases generated for each observation reported here. The required search time by CPLEX
Solver increases quickly as the number of customers goes beyond 40. Therefore, we set a one-hour CPU time limit so that whenever the solution time by CPLEX exceeds the limit, the best feasible solution obtained so far is used as a surrogate. Since algorithm \( H \) guarantees to find the optimal solution for the case with \( |\varphi^*| \leq 2 \) and usually finds a close-to-optimal solution if the ratio of \( |\varphi^*| / |S| \) is low, the deviations from the optimal solution (i.e., the error gaps) with respect to the average ratio \( |\varphi^*| / |S| = 5\% \) were minimal. In a number of cases, we noticed that the solutions by algorithm \( H \) outperformed the surrogates of optimal solutions. The magnitude of error gaps, however, increases as the average unfulfillment ratio, \( |\varphi^*| / |S| \), and the number of customers, \( n \), increase. When this ratio increases to 20\% and \( n \) goes beyond 60, the deviation from optimal solutions (or from the surrogate of optimal solutions) exceeds 12\%. This is due to the myopic nature, as all the greedy heuristics, of algorithm \( H \) which focuses only on the nodes that make the most immediate improvement of the objective value. When the number of nodes that needs to be deleted at the optimal solution is large, the chance that Algorithm \( H \) fails to identify the right nodes increases. A graphical plot of the results reported in Table 3 is given in Figure 5 below.

![Graphical Plot](image)

**Figure 5.** Empirical Observations on the Performance of Algorithm \( H \).

Tables 4a and 4b present the deviations of the heuristic solution from the CPLEX solution, against relative standard deviations of the demand, and relative widths of customer receiving time windows, respectively. The results (Table 4a) show that the heterogeneity in customer demand has a negative impact on the
Table 4a. Average error gaps against relative standard deviation of demand.

Table 4b. Average error gaps against relative mean width of time windows.

Figure 6a. Performance against relative Standard deviations of demand.

Figure 6b. Performance against Relative window widths.

performance of algorithm $H$. This is because the heterogeneity of customer demands, so the time to produce customer orders, together with the heterogeneity of customer time windows and truck traveling times between customers make it difficult for the greedy heuristic to identify the right subset of customers to delete
from the schedule. On the other hand, the results in Table 4b show that as the width of customer receiving time windows becomes more relaxed (i.e., increases), the performance of the heuristic improves. As the ratio of mean window width to the total travel time increases to the level of 45%, the average deviation from the optimal CPLEX solution reduces to be within 2.1% of the optimal CPLEX solution, even with $n=40$ customers and a relatively high level of average unfulfillment rate $|\varphi^s|/|S|=20\%$. This implies that, in reality, negotiating with customers to increase the width of their receiving time windows has potential to improve the operation performance.

4. Concluding remarks

We studied the single-truck production and distribution problem with a short shelf-life product, a given sequence of customers, a limited production capacity, and delivery time window constraints. The problem is to choose a subset of customers to receive the deliveries so that the total amount of satisfied demand is maximized while all the constraints are satisfied. We showed that the general version of this problem is NP-hard even with relaxed time windows and instantaneous truck traveling time, and analyzed the problem properties and the conditions under which the optimality of a solution is achieved. A greedy heuristic is proposed based on this analysis. Empirical observations on the error gaps achieved by this heuristic are reported.

The work we presented in this paper can be extended in several directions. When the delivery truck has a limited capacity, there is a customer partition issue by which the given customer sequence must be partitioned into several disjoint trips so that the total demand handled by each trip does not exceed the truck capacity. The problem is then how to partition to maximize the total amount of satisfied demand. The problem $P$ considered in this paper is a sub-problem of the zero-inventory PDP. A more general problem would allow the truck route to be optimized. Furthermore, we have only considered maximizing the total demand to be satisfied. There are many other objectives of interests from both a theoretical and a practical point of view. Examples of such objectives could include minimizing the fleet size, the deviation of actual delivery time from the customer receiving time windows, and the maximum time interval between the production completion time and the delivery time at a customer site subject to a fixed fleet size.
References