INFORMS Journal on Computing
Vol. 20, No. 1, Winter 2008, pp. 21–33
ISSN 1091-9856 | EISSN 1526-5528 | 08 | 2001 | 0021

DOI 10.1287/ijoc.1060.0208

The Integrated Production and Transportation Scheduling Problem for a Product with a Short Lifespan

H. Neil Geismar
Department of Information and Operations Management, Mays Business School, Texas A&M University, College Station, Texas 77843, ngeismar@mays.tamu.edu

Gilbert Laporte
HEC Montréal, Montréal, Quebec H3T 2A7, Canada, gilbert@crt.umontreal.ca

Lei Lei
Management Science and Information Systems Department, Rutgers Business School, Newark, New Jersey 07102, llei@andromeda.rutgers.edu

Chelliah Sriskandarajah
School of Management, University of Texas at Dallas, Richardson, Texas 75083, chelliah@utdallas.edu

The integrated production and transportation scheduling problem (PTSP) with capacity constraints is common in many industries. An optimal solution to PTSP requires one to simultaneously solve the production scheduling and the transportation routing problems, which requires excessive computational time, even for relatively small problems. In this study, we consider a variation of PTSP that involves a short shelf life product; hence, there is no inventory of the product in process. Once a lot of the product is produced, it must be transported with nonnegligible transportation time directly to various customer sites within its limited lifespan. The objective is to determine the minimum time required to complete producing and delivering the product to meet the demand of a given set of customers over a wide geographic region. This problem is NP-hard in the strong sense. We analyze the properties of this problem, develop lower bounds on the optimal solution, and propose a two-phase heuristic based on the analysis. The first phase uses either a genetic or a memetic algorithm to select a locally optimal permutation of the given set of customers; the second phase partitions the customer sequence and then uses the Gilmore-Gomory algorithm to order the subsequences of customers to form the integrated schedule. Empirical observations on the performance of this heuristic are reported.

Key words: supply chain; production and distribution; logistics; genetic algorithms; memetic algorithms
History: Accepted by Michel Gendreau, Area Editor for Heuristic Search and Learning; received September 2003; revised June 2005, June 2006; accepted October 2006. Published online in *Articles in Advance* October 5, 2007.

1. Introduction

The zero-inventory production and transportation problem is common in many industries in which a time-sensitive product cannot be inventoried due to its short shelf life. The delivery of the product must be made within a strictly limited time after its production. Therefore, the production and distribution operations must be highly integrated. When the production facility has a limited production rate and the transportation time is not instantaneous, any inefficiency in the integrated schedule may either cause the product to expire before it reaches the customers (in which case the production resources are wasted and additional expenses will be incurred to provide an environmentally safe disposal of the expired product), or cause the delivery to fail to meet a customer's delivery deadline (in which case the contract with the customer is canceled or special compensation must be made).

Our study was motivated by a practical scheduling problem encountered at a leading manufacturer of various industrial chemicals. One of the products, an adhesive material used by its customers for making plywood panels, has a lifespan of seven days. There are two types of business users for such short shelf-life products. A *one-time* user requests delivery and usually specifies a time window for the delivery. A *periodic user*, on the other hand, prefers that only a portion of the total order quantity arrive at regular intervals. We are interested in the scheduling problem related to serving the demand from customers who are periodic users. For each week in the manufacturer's planning horizon, each customer is assigned a particular day on which the shipment will arrive.

This specification of the delivery dates is important in practice because the product cannot be inventoried and the customers usually plan for their own production runs based on the expected arrival time of this product, which is used as an input. Consequently, the customers also request that the manufacturer quote and confirm the delivery schedules. If the specified delivery dates cannot be met, an alternative delivery arrangement must be negotiated between the customer and the plant. Because periodic users usually prefer to reserve a portion of the lifespan of such products to allow for internal scheduling flexibility, the remaining lifespan available for the coordination of production and distribution can sometimes be quite small.

In practice, management faces many operational issues with which optimization techniques could help. For example:

- Will one truck be enough for a day's deliveries? What is the minimum number of trucks needed for a day?
- Can the driver make the deliveries within the regular working hours of a day, or will overtime be needed?
- Can additional customers be added to a day's delivery schedule?
- What is the maximum number of customers that can be served with the given fleet size?
- Can a mandated delivery time be met with the given production capacity and the fleet?
- What is the maximum profit a company can make with the given capacity?
- What is the minimum total tardiness for the given set of customers?

As one can see, a major subproblem behind these operational issues is to determine the minimum time needed (i.e., minimum makespan) to produce and deliver to a given set of customers, subject to a given production rate, a given product lifespan, and a single capacitated truck. This problem will be the focus of this study. Solving this problem requires determining when and how much should be produced at the plant, when the trucks should leave the plant, which customers should be served on which trips, and what travel route a truck should follow. Due to the sequencing and routing issues involved, optimally solving this integrated scheduling problem is difficult. Our goal is to develop a heuristic that solves this fundamental production and transportation scheduling problem (PTSP) for a system with one plant and one truck.

Naturally, this study is dependent on previous work concerning the vehicle routing problem (VRP): Christofides (1985), Laporte and Semet (2002), Gendreau et al. (2002), Cordeau et al. (2002, 2004, 2005), and Cordeau and Laporte (2004).

Although the classic production-transportation problem has received significant attention (Palekar et al. 1990, Hochbaum and Hong 1996, Lotfi and Chen 1991, Diaby et al. 1992, Tuy et al. 1996, Sarmiento and Nagi 1999), most previous work has focused on optimizing the plant production capacity levels with a given transportation cost, rather than considering the transportation times and capacity. These studies also assume that the product can be carried in inventory without a time limit and that transportation is instantaneous. There are also several papers (Arbib et al. 1999 and Nahmias 1982) on the production scheduling problem with perishable products. The transportation and routing issues, however, are not considered in these studies. To our knowledge, only two papers have explicitly considered the integrated scheduling issue for perishable products.

Chen and Vairaktarakis (2005) studied the problem of integrated production and distribution for customer orders that must be directly delivered to customer sites as the orders are completed (i.e., no inventory). An unlimited number of transporters is available for the order shipments. Each has a fixed capacity and noninstantaneous transportation time. They analyze solution approaches for machine scheduling and transporter routing to minimize the transportation cost, plus either the maximum delivery time or the average delivery time. Eight variations of the problem are considered: single versus multiple-parallel machine processing lines, single versus multiple customers, and minimizing average delivery time versus maximum delivery time. For each variation, they either propose an efficient solution procedure or prove the variation to be intractable and develop a heuristic algorithm together with a performance analysis. Their empirical results show that the proposed heuristics are capable of producing near-optimal solutions for a wide range of randomly generated test cases.

Chen and Pundoor (2004) address the integrated production and distribution problem for products whose time sensitivity arises from their short selling season and short lead times, e.g., fashion items. The model consists of multiple overseas plants and a single domestic distribution center. They considered four different performance measures based on delivery lead time and total production and delivery costs. Decision variables include the assignment of orders to plants, the scheduling of the assigned orders at each plant, and the schedule for shipping completed orders. For each problem, they provide an efficient exact algorithm or prove intractability and present a fast heuristic that generates a near-optimal solution.

The integrated scheduling problem that we study extends from these two recent papers, sharing features such as nonnegligible transportation time and

delivery consolidation. The most notable feature that differentiates our problem from previous work is the combination of the product's limited lifespan and the truck routing decisions, which leads to the possibility that the product may expire before it reaches a customer. The problem is further complicated by limited transportation capacity, fixed production rate, and the lack of correlation between the size of a customer's demand and its location. On the other hand, these complications also make the resulting integrated scheduling problem challenging and interesting. The particular variation that we consider involves a single production facility, a single delivery truck, and a given set of customers at different locations over a geographic region. We do not consider machine scheduling within the plant, but rather which customer orders are assigned to each production run, the size of each run, and the start time of each run, given a constant production rate.

We begin with notations and definitions (§2). Section 3 establishes lower bounds for the problem's optimal solution. We then examine the structure of the PTSP and describe an algorithm for scheduling production and delivery for a given ordering of customers (§4). Two evolutionary algorithms for improving solutions are described in §5. Results from test runs of the heuristic are presented in §6. Section 7 concludes the paper and makes recommendations for future study.

2. Notation and Definitions

We consider a plant P with a constant production rate $r > 0$ for a single product that has a limited lifespan. We implement this expiration time by requiring that all items of a production lot be delivered within B time units after the lot has completed production. There is a set of customers, $N = \{1, 2, \ldots, n\}$; each has a geographic location and a demand, q_i, $i \in N$, to be satisfied. Our objective is to minimize the makespan H, the time required to manufacture and deliver goods to satisfy all demand. This objective was chosen because it requires close cooperation by the production and the transportation functions. Furthermore, it is an important subproblem to many other problems faced in this system. Production and distribution can occur continuously throughout the period, so this problem can be envisioned as planning a day's operations for a product whose lifespan is less than one day.

A truck with capacity Q, $\max_{i \in N} q_i \le Q \le \sum_{i=1}^{n} q_i$, is used to deliver the product to customers. Since the truck has a limited capacity, it will need to return to the plant several times during the planning period H. That is, the route that the truck uses in the planning period can be decomposed into multiple trips. Each trip starts at P (location 0), travels to a sequence of customer locations, and finally ends at P. We assume that each customer is visited only once, that the truck's speed is constant, that the time required for the truck to travel from the plant to customer k ($\tau_{0,k}$) and that the time to travel from customer k to customer j ($\tau_{k,j}$) are known, $1 \le j, k \le n$, and symmetric ($\tau_{k,j} = \tau_{j,k}$), $0 \le j, k \le n$, that $\tau_{0,k} \le B$, $1 \le k \le n$, that the truck loading and unloading times are negligible (otherwise they may be included in the travel time), and that all demand must be satisfied.

For a trip that serves customers $i+1, i+2, \ldots, j-1, j$, the quantity loaded onto the truck when the truck leaves plant P for this trip is $\sum_{h=i+1}^{j} q_h \le Q$. The selection of customers visited on each trip is also constrained by the travel times and the product's lifespan: If customer j is the last to be served on a trip, then the truck must reach customer j before the product expires. Hence, the problem is to determine which customers are served on which trip, the order in which these customers are visited, and the sequence of truck trips that satisfies all demand and minimizes H, subject to the customers' demands, the customers' locations, the plant's production rate, the truck capacity, and the product's lifespan.

We now motivate our heuristic approach by proving the intractability of PTSP.

Theorem 1. *PTSP is NP-hard in the strong sense.*

Proof. We show that TSP is a special case of PTSP. Consider the following instance of PTSP: $N = \{1, 2, \ldots, n\}$ is a set of customers; each has a geographic location and a demand, q_i, $i \in N$, to be satisfied by one truck with capacity Q, and the lifespan of the product is B, where $Q = \sum_{i=1}^{n} q_i$ and $B \ge 2\sum_{i=1}^{n} \tau_{0,i}$. The production is instantaneous, that is, r is very large so that any set of customers' demands is produced in zero time. Assume that the customers are located in a plane and that the distance matrix satisfies the triangle inequality, that is, $\tau_{i,j} + \tau_{j,k} \ge \tau_{i,k}$ for all i, j, k. Also assume that any two customer locations i, j, and the plant, P, do not fall in a straight line, that is, $\tau_{i,0} + \tau_{0,j} > \tau_{i,j}$, for all i, j.

It is sufficient to show that the optimal solution to this special case of PTSP is given by one tour. Note that B is large enough so that the product will not expire in any single tour. Suppose that an optimal solution to this instance is given by more than one tour. Consider merging any two tours in this solution to form one tour as follows:

Two tours: $\langle P, i_1, i_2, \ldots, i_t, P\rangle$, $\langle P, j_1, j_2, \ldots, j_s, P\rangle$.

Merged single tour: $\langle P, i_1, i_2, \ldots, i_t, j_1, j_2, \ldots, j_s, P\rangle$

Since two customer locations i_t, j_1 and the plant P do not fall in a straight line, we have $\tau_{i_t,0} + \tau_{0,j_1} > \tau_{i_t,j_1}$, so the merged tour's travel distance is less than the

total distance of the two original tours. This contradicts the optimality of the solution given above. We can continue this process of successively merging two tours into one tour until we obtain a single tour in which all customers are served. Each such merge will improve the objective function H (here the time to travel the total distance). Thus, a single-tour solution is optimal for this special case. Hence, the above special case of PTSP is equivalent to solving the Euclidian TSP, which is NP-hard in the strong sense (Papadimitrious 1977). □

Given this intractability, our approach will be to find solutions to PTSP for a fixed permutation of customers σ (§4), then to propose heuristics (§5) to improve these solutions by finding better permutations. Before that, we develop lower bounds that will be used to evaluate the heuristics.

3. Lower Bounds on the Makespan

We know of no previously published lower bounds explicitly for this combined production and transportation problem. Thus, we begin with an elementary lower bound based on the processing time required for the plant to satisfy all demand and then present two lower bounds from the VRP literature. Which lower bound is more effective depends on the data of a particular instance.

Theorem 2. *Given a PTSP with n customers, each customer's demand q_i, $i = 1, \dots, n$, a matrix of travel times $\tau_{i,j}$, $0 \le i, j \le n$, where 0 represents the plant, and a production rate r, a lower bound on the makespan H is* $\mathrm{LB1} = (1/r)\sum_{i=1}^{n} q_i + 2\min_{1\le i\le n}\{\tau_{0,i}\}$.

Proof. The time required to produce goods to satisfy all demand is $(1/r)\sum_{i=1}^{n} q_i$. There must be one final trip after the last lot is produced. The second term represents the minimum time of this trip. □

The first VRP lower bound is based on Labbé et al. (1991) and Bourjolly and Rebetez (2005). Define $F = \{i\colon q_i > Q/3\}$; no more than two elements of F can be served on any trip. To determine the most efficient grouping of the elements of F into trips, we solve a nonbipartite weighted matching problem on the following graph: $V = F$, $E = \{(a,b)\colon a \in F,\ b \in F,\ q_a + q_b \le Q,\ \tau_{a,b} + \min\{\tau_{0,a}, \tau_{0,b}\} \le B\} \cup \{(a,a)\colon a \in F\}$. The weight of edge (a,b) is $\tau_{0,a} + \tau_{a,b} + \tau_{b,0}$. Note that E also contains loops (a,a) with weight $2\tau_{0,a}$, $\forall a \in F$. Loop (a,a) will be in the matching if $q_a \ge 2Q/3$. Additionally, if $|F|$ is odd, then the matching must contain at least one loop. The matching also must have loops if $|\{i\colon q_i > Q/2\}| > |F|/2$. It is easy to see that a minimal-weight matching over (V,E) provides an optimal vehicle routing schedule over F, and the number of trips is $f \ge \lceil |F|/2 \rceil$. Let M_F be the total weight of this minimal matching.

The remaining customers in $F^c = \{i\colon q_i \le Q/3\}$ may be served in one of these f trips or in additional trips. A lower bound for the amount of additional trips needed is $g = \lceil \sum_{i=1}^{n}(q_i/Q) - f \rceil$. Each of these trips will require at minimum time $2\tau_{0,j}$, for some $j \in F^c$. Since the trips can hold at least three elements of F^c, $|F^c| \ge 3g - 2$. Let $l_1, l_2, \dots, l_{3(g-1)}$ be the $3(g-1)$ elements of F^c that are closest to the plant, where $\tau_{0,l_1} \le \tau_{0,l_2} \le \cdots \le \tau_{0,l_{3(g-1)}} \le \tau_{0,l_{|F^c|}}$. The minimum distance traveled in these g additional trips is then $2\tau_{0,l_3} + 2\tau_{0,l_6} + \cdots + 2\tau_{0,l_{3(g-1)}} + 2\tau_{0,l_{|F^c|}}$. Therefore, because no deliveries are made until production for at least one customer has been completed, which requires at least $\min_{1\le i\le n}\{q_i/r\}$, a lower bound for H is

$$\mathrm{LB2} = M_F + 2\sum_{i=1}^{g-1} \tau_{0,l_{3i}} + 2\tau_{0,l_{|F^c|}} + \min_{1\le i\le n}\left\{\frac{q_i}{r}\right\}.$$

Hadjiconstantinou et al. (1995) derive a lower bound for the VRP by formulating it as a binary integer program with n constraints and one variable for each possible truck route. They address the dual problem and handle the large number of constraints by dividing the routes into sets of q-paths: a *q-path* is a truck route that carries q items. For each customer i, the shortest round trip carrying q items is found, for $q = q_i, \dots, Q$. If the length of this minimal trip is denoted $\psi(q,i)$, then each customer's optimal dual variable value is

$$u_i^* = q_i \min\left\{\frac{\psi(q,i)}{q}\colon q_i \le q \le Q\right\}, \quad \text{and}$$

$$\mathrm{LB3} = \sum_{i=1}^{n} u_i^* + \min_{1\le i\le n}\left\{\frac{q_i}{r}\right\}.$$

The lower bound used in §6 to evaluate the heuristics described in §5 is $\mathrm{LB} = \max\{\mathrm{LB1}, \mathrm{LB2}, \mathrm{LB3}\}$.

4. Scheduling a Given Permutation of Customers

We now describe our algorithm for finding an effective production and distribution schedule for a given permutation of customers. We begin with a scheme for finding a set of feasible trips (§4.1), then present an algorithm for minimizing the makespan for serving these trips (§4.2). Algorithm Shortest Path (§4.3) combines these elements with the Gilmore-Gomory algorithm (1964) for no-wait flow shops.

4.1. Finding a Set of Feasible Trips

We follow Beasley's (1983) route first–cluster second method to divide the customers into a set of feasible trips. For any permutation σ of the n customers, a set of trips that serves all customers with minimal travel time can be found by finding a shortest path on a directed graph. The graph G_σ is defined as follows:

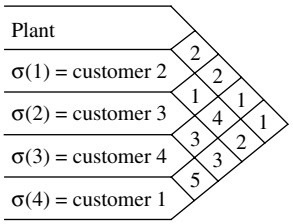

Figure 1 Travel Times for Example 1

Definition 1. Given σ and $V(G_\sigma) = \{0, \sigma(1), \sigma(2), \ldots, \sigma(n)\}$, for each pair of nodes, $\sigma(i), \sigma(j) \in V(G_\sigma)$, where $0 \le i < j \le n$ and $\sigma(0) = 0$, add arc $\sigma(i) \to \sigma(j)$ to $E(G_\sigma)$ if and only if the total demand of customers $\sigma(i+1), \sigma(i+2), \ldots, \sigma(j)$, is no more than the truck capacity Q, i.e., $\sum_{h=i+1}^{j} q_{\sigma(h)} \le Q$, and all deliveries on such a trip can be made before the product expires, i.e., $\tau_{0,\sigma(i+1)} + \sum_{h=i+1}^{j-1} \tau_{\sigma(h),\sigma(h+1)} \le B$. Note that arc $\sigma(i) \to \sigma(j)$ in G_σ corresponds to the trip in which the truck travels the route $0 \mapsto \sigma(i+1) \mapsto \sigma(i+2) \mapsto \cdots \mapsto \sigma(j) \mapsto 0$. We use the notation $(0, \sigma(i+1)) \mapsto \sigma(j)$ to signify the trip *generated* by arc $\sigma(i) \to \sigma(j)$.

Clearly, there is a one-to-one correspondence between the paths from 0 to $\sigma(n)$ on graph G_σ and the sequences of feasible truck trips that visit all customers in order σ and satisfy the truck capacity constraint and the product lifespan constraint.

Example 1. $n = 4$, $Q = 10$, $B = 5$, $\sigma = (2, 3, 4, 1)$. Customer demands: $q_1 = 2$, $q_2 = 4$, $q_3 = 3$, and $q_4 = 5$. The symmetric travel times are in Figure 1.

We first construct Table 1 to determine which arcs will be in G_σ, which is shown in Figure 2. A description of the paths from 0 to $\sigma(4)$ in G_σ is in Table 2.

4.2. Minimizing the Makespan for a Given Set of Trips

For a sequence σ, let Θ_σ be the set of all feasible paths from node 0 to node $\sigma(n)$ on G_σ. For each path $s_\sigma \in \Theta_\sigma$, we develop a formula for H_{s_σ}, the time required to satisfy all demand by coordinated production and travel for the sequence of trips generated by that path, and then a linear program that minimizes H_{s_σ}.

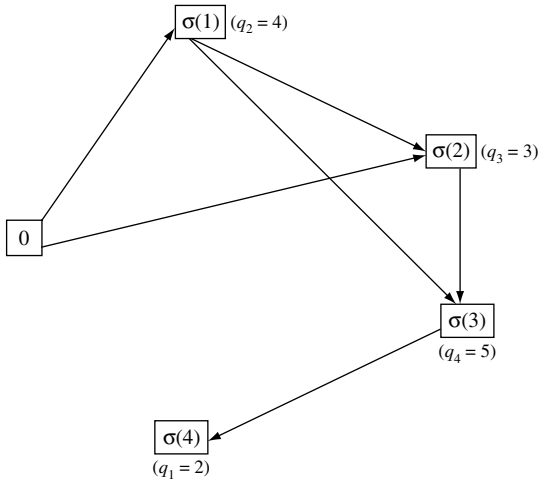

Figure 2 Graph $G(\sigma)$ for Example 1, $\sigma = (2, 3, 4, 1)$

If the number of arcs in path $s_\sigma \in \Theta_\sigma$ is m, then s_σ generates a sequence of m trips. If $\sigma(i_j)$ is the last customer visited in trip j, $j = 1, \ldots, m$, and $i_0 = 0$ ($i_m = n$ by implication), then the sequence of trips generated by s_σ is

$$\langle (0, \sigma(i_0+1)) \mapsto \sigma(i_1), (0, \sigma(i_1+1)) \mapsto \sigma(i_2), \ldots, (0, \sigma(i_{m-1}+1)) \mapsto \sigma(i_m) \rangle.$$

While the truck is delivering goods on trip j, the plant is producing goods for trip $j+1$, $j = 1, \ldots, m-1$. The time required for the truck to make trip j is

$$T(\sigma(i_{j-1}+1), \sigma(i_j)) = \tau_{0,\sigma(i_{j-1}+1)} + \sum_{k=i_{j-1}+1}^{i_j-1} \tau_{\sigma(k),\sigma(k+1)} + \tau_{\sigma(i_j),0}.$$

The time required for the plant to produce goods for trip $j+1$ is $(1/r)\sum_{k=i_j+1}^{i_{j+1}} q_{\sigma(k)}$.

Therefore, if $T(\sigma(i_{j-1}+1), \sigma(i_j)) \le (1/r)\sum_{k=i_j+1}^{i_{j+1}} q_{\sigma(k)}$, then the truck will be ready to depart for trip

Table 1 Potential Arcs for Example 1

Potential arcs	Generated truck trips	Demand	Delivery time
$0 \to \sigma(1)$	$0 \mapsto \sigma(1) \mapsto 0$	4	2
$0 \to \sigma(2)$	$0 \mapsto \sigma(1) \mapsto \sigma(2) \mapsto 0$	7	3
$0 \to \sigma(3)$	$0 \mapsto \sigma(1) \mapsto \sigma(2) \mapsto \sigma(3) \mapsto 0$	12*	6*
$\sigma(1) \to \sigma(2)$	$0 \mapsto \sigma(2) \mapsto 0$	3	2
$\sigma(1) \to \sigma(3)$	$0 \mapsto \sigma(2) \mapsto \sigma(3) \mapsto 0$	8	5
$\sigma(1) \to \sigma(4)$	$0 \mapsto \sigma(2) \mapsto \sigma(3) \mapsto \sigma(4) \mapsto 0$	10	10*
$\sigma(2) \to \sigma(3)$	$0 \mapsto \sigma(3) \mapsto 0$	5	1
$\sigma(2) \to \sigma(4)$	$0 \mapsto \sigma(3) \mapsto \sigma(4) \mapsto 0$	7	6*
$\sigma(3) \to \sigma(4)$	$0 \mapsto \sigma(4) \mapsto 0$	2	1

Note. Asterisks indicate values that disqualify the potential arc.

Table 2 Example 1 with $n = 4$, $Q = 10$

Θ_σ: Paths from 0 to $\sigma(4)$	Generated truck trips
$0 \to \sigma(1) \to \sigma(2) \to \sigma(3) \to \sigma(4)$	$0 \mapsto \sigma(1) \mapsto 0$
	$0 \mapsto \sigma(2) \mapsto 0$
	$0 \mapsto \sigma(3) \mapsto 0$
	$0 \mapsto \sigma(4) \mapsto 0$
$0 \to \sigma(1) \to \sigma(3) \to \sigma(4)$	$0 \mapsto \sigma(1) \mapsto 0$
	$0 \mapsto \sigma(2) \mapsto \sigma(3) \mapsto 0$
	$0 \mapsto \sigma(4) \mapsto 0$
$0 \to \sigma(2) \to \sigma(3) \to \sigma(4)$	$0 \mapsto \sigma(1) \mapsto \sigma(2) \mapsto 0$
	$0 \mapsto \sigma(3) \mapsto 0$
	$0 \mapsto \sigma(4) \mapsto 0$

$j+1$ when the plant has completed producing the needed goods. Conversely, if $T(\sigma(i_{j-1}+1), \sigma(i_j)) > (1/r)\sum_{k=i_j+1}^{i_{j+1}} q_{\sigma(k)}$, then the plant could delay its start of processing for this trip so that the product's limited lifespan will not start before the truck is available. Additionally, the truck is idle while Trip 1 is being processed, and the plant is idle while trip m is being delivered. We model this by adding a null trip that has total demand and travel time each equal to zero. These observations lead to the following formula for H_{s_σ}:

$$H_{s_\sigma} = \max\left\{0, \frac{1}{r}\sum_{k=1}^{i_1} q_{\sigma(k)}\right\} + \sum_{j=1}^{m-1} \max\left\{\frac{1}{r}\sum_{k=i_j+1}^{i_{j+1}} q_{\sigma(k)}, T(\sigma(i_{j-1}+1), \sigma(i_j))\right\} + \max\{0, T(\sigma(i_{m-1}+1), \sigma(n))\}. \quad (1)$$

From (1) we obtain an upper bound for the optimal value of H: $H^* \le \min_\sigma \min_{s_\sigma \in \Theta_\sigma}\{H_{s_\sigma}\}$.

Equation (1) has similarities to the makespan expression for the two-machine no-wait flow shop scheduling problem. This leads to using the Gilmore-Gomory (1964) algorithm (described in the Online Supplement to this paper, available at http://joc.pubs.informs.org/ecompanion.html) as part of our Algorithm Shortest Path. If there were no delay between the completion of a lot's processing and the beginning of the trip that delivers that lot, then the Gilmore-Gomory algorithm would generate an optimal ordering of these trips. However, mandating that there be no such delay is overly restrictive. It is only necessary that all items in the lot be delivered to the appropriate customers before the product expires.

Example 2. Consider the three lots in Table 3. Figure 3 displays a Gantt chart for the schedule created for this example by the formulation in (1). Because we are considering this problem as a two-machine no-wait flowshop, we use the notation M_p for the plant and M_t for the transporter in all Gantt charts. The total time for this schedule is 33.

Note that since lot Y has a short delivery time, if its production starts earlier, it can still be delivered before the product expires. This would allow lot Z to begin production earlier. Such an adjustment could

Table 3 **Example 2: Product Lifespan = 10**

Trip	Production time	Travel time
X	6	10
Y	5	5
Z	10	7

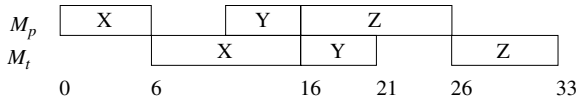

Figure 3 **Gantt Chart for Table 3**

lead to the more efficient schedule in Figure 4, whose total time is 28.

We now present a linear-programming formulation that can provide such a feasible solution. Start with a permutation of customers σ and a path $s_\sigma \in \Theta_\sigma$ that generates a sequence of trips defined by $\{i_0 = 0, i_1, i_2, \ldots, i_m = n\}$. Let u_j be the start time of production for the lot serving customers $i_{j-1}+1, \ldots, i_j$, for $j = 1, \ldots, m$. Let v_j be the start time of the trip that delivers to customers $i_{j-1}+1, \ldots, i_j$, for $j = 1, \ldots, m$. Note that the planning horizon concludes when the last trip returns to the plant:

$$H_{s_\sigma} = v_m + T(\sigma(i_{m-1}+1), n).$$

LP_Compress:

$$\min H_{s_\sigma} = v_m + T(\sigma(i_{m-1}+1), n),$$

such that

$$u_{j+1} - u_j \ge \frac{1}{r}\sum_{k=i_{j-1}+1}^{i_j} q_{\sigma(k)}, \quad j = 1, \ldots, m-1 \quad (2)$$

$$v_{j+1} - v_j \ge T(\sigma(i_{j-1}+1), \sigma(i_j)), \quad j = 1, \ldots, m-1 \quad (3)$$

$$v_j - u_j \ge \frac{1}{r}\sum_{k=i_{j-1}+1}^{i_j} q_{\sigma(k)}, \quad j = 1, \ldots, m \quad (4)$$

$$v_j - u_j \le B + \frac{1}{r}\sum_{k=i_{j-1}+1}^{i_j} q_{\sigma(k)} - \tau_{0,\sigma(i_{j-1}+1)} - \sum_{k=i_{j-1}+1}^{i_j} \tau_{\sigma(k),\sigma(k+1)}, \quad j = 1, \ldots, m \quad (5)$$

$$u_j \ge 0, \quad j = 1, \ldots, m$$

$$v_j \ge 0, \quad j = 1, \ldots, m.$$

- Constraints (2) state that processing on lot $j+1$ cannot begin until processing on lot j has been completed.
- Constraints (3) state that the trip to deliver lot $j+1$ cannot begin until the truck has returned from delivering lot j.

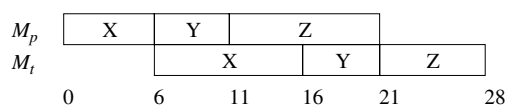

Figure 4 **Gantt Chart for Improved Schedule for Table 3**

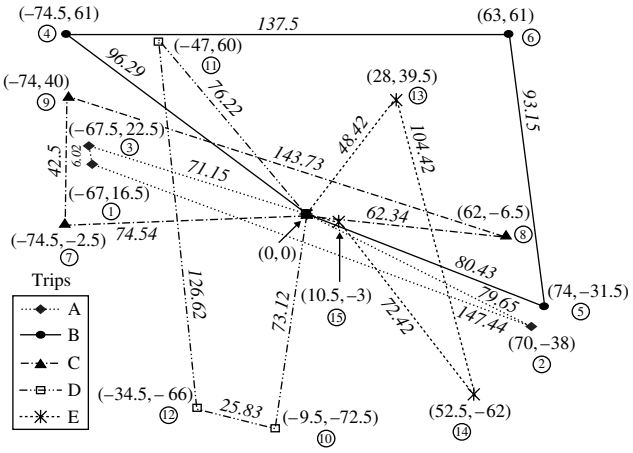

Figure 5 Tours of Table 4

Notes. Italicized numbers represent distances. Circled numbers are customer labels.

- Constraints (4) state that the trip to deliver lot j cannot begin until the processing of lot j has been completed.
- Constraints (5) state that all items in lot j must be delivered before they expire.

THEOREM 3. *The linear program* LP_Compress *can be solved in* $O(n^2)$ *time.*

PROOF. Note that LP_Compress is a system of $4m-2$ difference constraints. If constraints (2), (3), and (4) are multiplied by -1, then the system has the form $\mathcal{A}x \le b$. The resulting system has an associated *constraint graph* with $2m$ nodes (one for each variable) and $4m-2$ arcs (one for each constraint): for each inequality of the form $x - y \le z$, there is an arc (y, x) with length z. If this graph has no negative cycles, then finding a shortest path in this graph is equivalent to solving LP_Compress (Ahuja et al. 1993).

We now show that there are no negative cycles in the constraint graph. The graph has the following arcs:

$$(u_{j+1}, u_j), \quad \text{with length } -\frac{1}{r}\sum_{k=i_{j-1}+1}^{i_j} q_{\sigma(k)}, \quad j = 1, \ldots, m-1$$

$$(v_{j+1}, v_j), \quad \text{with length } -T(\sigma(i_{j-1}+1), \sigma(i_j)), \quad j = 1, \ldots, m-1$$

$$(v_j, u_j), \quad \text{with length } -\frac{1}{r}\sum_{k=i_{j-1}+1}^{i_j} q_{\sigma(k)}, \quad j = 1, \ldots, m$$

$$(u_j, v_j), \quad \text{with length } B + \frac{1}{r}\sum_{k=i_{j-1}+1}^{i_j} q_{\sigma(k)} - \tau_{0,\sigma(i_{j-1}+1)} - \sum_{k=i_{j-1}+1}^{i_j} \tau_{\sigma(k),\sigma(k+1)}, \quad j = 1, \ldots, m$$

Therefore, the only cycles are $v_j \to u_j \to v_j$, $j = 1, \ldots, m$. The length of each cycle is

$$B + \frac{1}{r}\sum_{k=i_{j-1}+1}^{i_j} q_{\sigma(k)} - \tau_{0,\sigma(i_{j-1}+1)} - \sum_{k=i_{j-1}+1}^{i_j} \tau_{\sigma(k),\sigma(k+1)} - \frac{1}{r}\sum_{k=i_{j-1}+1}^{i_j} q_{\sigma(k)} = B - \tau_{0,\sigma(i_{j-1}+1)} - \sum_{k=i_{j-1}+1}^{i_j} \tau_{\sigma(k),\sigma(k+1)} \ge 0.$$

The nonnegativity is guaranteed by Definition 1. Therefore, LP_Compress can be solved as a shortest-path problem by a label-correcting algorithm in $O(m^2) \le O(n^2)$ time. □

Table 4 Trips for Example 3

Trip	Route	Production time	Travel time
A	$0 \mapsto 2 \mapsto 1 \mapsto 3 \mapsto 0$	311	304.26
B	$0 \mapsto 4 \mapsto 6 \mapsto 5 \mapsto 0$	341	407.36
C	$0 \mapsto 7 \mapsto 9 \mapsto 8 \mapsto 0$	348	323.11
D	$0 \mapsto 10 \mapsto 12 \mapsto 11 \mapsto 0$	279	301.79
E	$0 \mapsto 13 \mapsto 14 \mapsto 15 \mapsto 0$	328	236.18

4.3. Algorithm Shortest Path

Algorithm Shortest Path uses the elements of the previous subsections to convert a permutation σ of n customers into trips that minimize the truck's total travel time. It assigns customers to trips by finding a shortest path from node 0 to node $\sigma(n)$ on the graph of feasible trips (§4.1). It then attempts to reduce each trip's travel time by using 2-OPT on each pair of customers within each trip. These trips are ordered using the Gilmore-Gomory (1964) algorithm for two-machine no-wait flowshops and then scheduled by LP_Compress (§4.2).

ALGORITHM SHORTEST PATH
Input: A permutation σ of n customers
Create graph G_σ with $V(G_\sigma) = \{0, \sigma(1), \sigma(2), \ldots, \sigma(n)\}$ in accordance with Definition 1

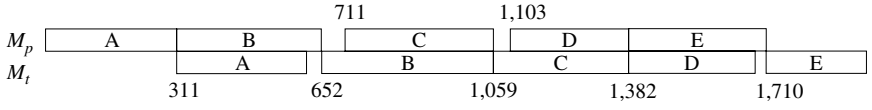

Figure 6 Gantt Chart for Table 4, $H = 1{,}946.65$

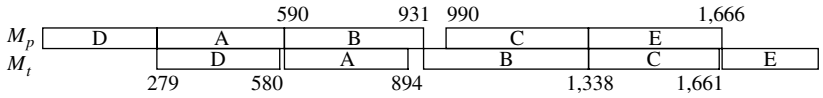

Figure 7 **Gantt Chart for Table 4 After Application of the Gilmore-Gomory Algorithm,** $H = 1{,}902.54$

Figure 8 **Gantt Chart for Table 4 After Application of LP_Compress,** $H = 1{,}897.18$

To each arc (k, l), assign length $T(k+1, l)$
Find a shortest path from node 0 to node $\sigma(n)$ in G_σ
For each arc $(\sigma(i_{j-1}), \sigma(i_j))$ in the shortest path
  Define trip $(0, \sigma(i_{j-1}+1)) \mapsto \sigma(i_j)$
  For each pair of customers on this trip
    Exchange them using 2-OPT
    If the new trip is feasible and requires less
        travel time than the original trip then
      Keep this exchange
    Else
      Undo this exchange
    End If
  End For
Use Gilmore-Gomory to order the trips
Use LP_Compress to schedule the trips

The complexity of creating graph G_σ is $O(n \log n)$, finding the shortest path can be done in $O(n)$ time via the reaching algorithm (Ahuja et al. 1993), the 2-OPT step is $O(n \log n)$, and the Gilmore-Gomory (1964) algorithm is $O(m \log m)$. LP_Compress can be solved in $O(n^2)$ time. Therefore, the complexity of Algorithm Shortest Path is $O(n^2)$.

Example 3. We illustrate Algorithm Shortest Path by an example with fifteen customers. Each customer's demand and location in the x-y plane is in Table S1 in the Online Supplement. Also in the Online Supplement is Table S2, which lists the Euclidean distance between each pair of customers for this example. First we consider $\sigma = (1, 2, \ldots, 15)$. If $Q = 600$, $B = 600$, and $r = 1$, then the shortest path s_σ on graph G_σ yields five trips with three customers each. After the customers of each trip are ordered by the 2-OPT heuristic, the trips, their production times, and their delivery times are as presented in Table 4. A map of these tours is in Figure 5.

Figure 6 displays a Gantt chart for this example if the trips are scheduled in the given order. The total time for this schedule is $H = 1{,}946.65$. The Gilmore-Gomory algorithm reorders the trips to D, A, B, C, E, which yields a makespan of $H = 1{,}902.54$. A Gantt chart for this schedule is in Figure 7. LP_Compress reduces the makespan to $H = 1{,}897.18$ by starting the production for trip C and for trip E, and the delivery for trip E, 5.36 time units earlier, resulting in the schedule of Figure 8.

To see how the initial permutation can effect the final result, now suppose that $\sigma = (4, 11, 13, 6, 5, 8, 9, 3, 1, 7, 15, 2, 14, 10, 12)$ and we have the trips listed in Table 5. 2-OPT cannot improve these trips. A map of these tours is in Figure 9.

If the trips are executed in order (A, B, C, D, E), then the makespan is $H = 2{,}024.65$. This schedule is shown in Figure 10. Gilmore-Gomory (1964) reorders the trips to (A, E, C, B, D), in which case the makespan is $H = 1{,}650.34$. This is a 13% improvement over the schedule generated from $\sigma = (1, 2, \ldots, 15)$. This better schedule is shown in Figure 11. LP_Compress cannot improve this schedule.

Table 5 **Trips for a Different Ordering of Customers**

Trip	Route	Production time	Travel time
A	$0 \mapsto 4 \mapsto 11 \mapsto 13 \mapsto 6 \mapsto 0$	380	330.33
B	$0 \mapsto 5 \mapsto 8 \mapsto 0$	240	170.50
C	$0 \mapsto 9 \mapsto 3 \mapsto 1 \mapsto 7 \mapsto 0$	412	203.78
D	$0 \mapsto 15 \mapsto 0$	149	21.84
E	$0 \mapsto 2 \mapsto 14 \mapsto 10 \mapsto 12 \mapsto 0$	426	272.54

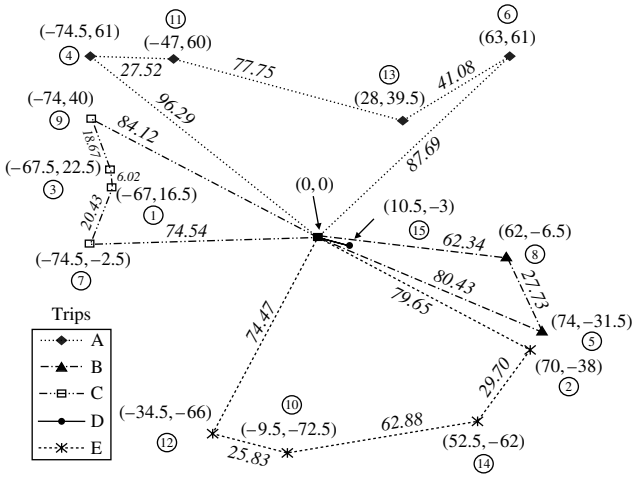

Figure 9 **Tours of Table 5**

Notes. Italicized numbers represent distances. Circled numbers are customer labels.

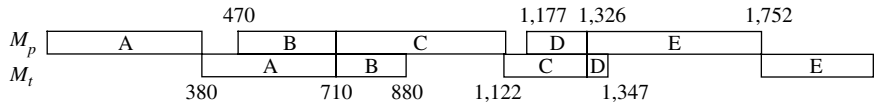

Figure 10 **Gantt Chart for Table 5,** $H = 2{,}024.65$

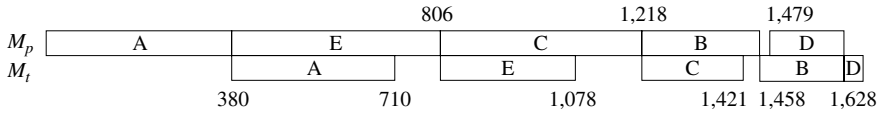

Figure 11 **Gantt Chart for Table 5 After Application of the Gilmore-Gomory Algorithm,** $H = 1{,}650.34$

We can also use these data to demonstrate the benefit of integration between production and transportation. If the two functions were independent, then production would most likely be scheduled contiguously to minimize its makespan and to reduce setups. For $Q = 600$, $B = 300$, and $r = 3$, such an arrangement would yield the trips shown in Table 6 and the schedule shown in Figure 12, in which the product cannot be delivered before it expires. If $B \geq 1{,}176.29$, then this schedule would be feasible and reasonably efficient with $H = 1{,}722.85$. A feasible and optimal schedule for this set of trips is presented in Figure 13: $H = 1{,}648.18$.

LP_Compress finds the optimum makespan for the sequence of trips generated by s_σ, so Algorithm Shortest Path should find an efficient schedule for a given permutation of customers. However, to find the best possible path and schedule, the LP must be solved for all sequences s_σ arising from each permutation σ, which would require an exponential number of LPs to be solved. Hence, we will next use evolutionary algorithms to find efficient permutations σ.

5. Evolutionary Algorithms

The two phases of our heuristic correspond to the route first–cluster second structure (Beasley 1983). Permutations in which customers are served are generated by two different evolutionary algorithms. We use each as Phase 1 to compare their effectiveness. The second phase is Algorithm Shortest Path, in which the permutation of customers is divided into trips, the order in which the customers are visited is optimized within each trip, the trips themselves are reordered for efficiency, then production and delivery are scheduled. Hence, Algorithm Shortest Path is used to evaluate and guide the processing of each of the algorithms used in the first phase.

Table 6 **Trips for** $Q = 600$, $B = 300$, **and** $r = 3$

Trip	Route	Production time	Travel time
A	$0 \mapsto 2 \mapsto 1 \mapsto 3 \mapsto 0$	103.67	304.26
B	$0 \mapsto 4 \mapsto 0$	29.00	192.57
C	$0 \mapsto 6 \mapsto 5 \mapsto 0$	84.67	261.27
D	$0 \mapsto 7 \mapsto 9 \mapsto 8 \mapsto 0$	116	323.11
E	$0 \mapsto 10 \mapsto 12 \mapsto 11 \mapsto 0$	93	301.79
F	$0 \mapsto 13 \mapsto 14 \mapsto 15 \mapsto 0$	109.30	236.18

The theory of evolutionary algorithms (Goldberg 1989) is based upon the theory of evolution and natural selection. The process begins with a population of individual solutions (permutations of the n customers with no trip delimiters, in this implementation). The better solutions (those that lead to schedules with shorter makespans) are considered to be more fit. Hence, in a process similar to biological reproduction, characteristics of two fit solutions are combined in a process called *crossover*. The intent is that the beneficial qualities of the "parents" will be carried forward to form a new solution that may be better than either parent. Additionally, some solutions *mutate*. After crossover and mutation, the population is ordered by fitness, and those less fit are deleted: they die off. The amount that die off is sufficient to restore the population to its original size. This completes the first generation. The algorithm repeats for a fixed number of generations.

The first algorithm considered for Phase 1 is a traditional genetic algorithm (GA). The second is a memetic algorithm (MA) based on Prins (2004).

5.1. Genetic Algorithm

Processing in the genetic algorithm begins by randomly generating an initial population of permutations. Permutations are assigned to couples through a process described below. The number of couples is half of the population size, and it is possible for an individual to be assigned to more than one couple. For each couple, a random number is compared to a fixed probability to determine whether that couple generates offspring (new solutions) via crossover. Following crossover, a small portion of this enlarged population is randomly chosen for mutation. Afterward, all members of the population are evaluated for fitness, and the population is pruned back to its original size.

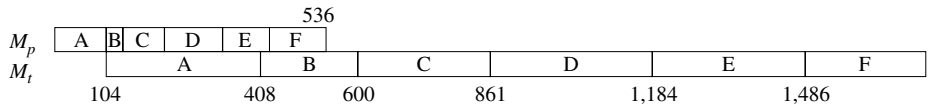

Figure 12 **Gantt Chart for Table 6 if Production and Transportation Are Scheduled Independently,** $H = 1{,}722.85$

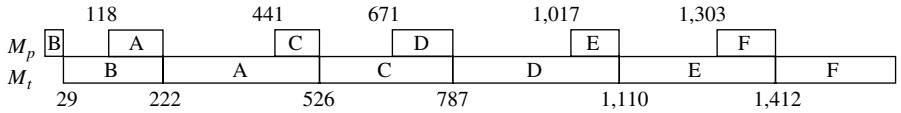

Figure 13 **Gantt Chart for Table 6 if Production and Transportation Cooperate,** $H = 1{,}648.18$

The two individuals that form a couple are selected by the binary tournament method (Cervone and Michalski 2000). Two permutations are randomly chosen, and the more fit becomes the first potential parent. This process is repeated to find the second potential parent. These two permutations form two new individuals via *order crossover*. In order crossover, two random schedule positions a_1 and a_2 are chosen (assume $a_1 < a_2$). Those customers scheduled in positions a_1 through a_2 in Parent 1 are placed into the same locations in Child 1. Child 1 is completed by filling the remaining positions with the customers not copied from Parent 1 in the order in which they appear in Parent 2, starting at position $a_2 + 1$, moving through n, and then proceeding circularly back to positions 1 through $a_1 - 1$. Child 2 is formed by repeating the process with Parents 1 and 2 exchanged (see Figure 14). Only the more fit child is added to the population.

This algorithm uses the pairwise exchange mutation operator, in which two randomly-chosen customers switch positions. A pairwise exchange mutation is illustrated in Figure 15.

For genetic algorithms, De Jong (1975) suggests a high crossover probability, low mutation rate, and a moderate population size. We follow these guidelines in our computations. Each generation has 150 members. Experiments were run to determine good choices for the GA parameters. These indicated that each couple should have a 90% chance to form two new solutions via crossover, and that each member of the enlarged population should have a 10% chance of mutating. A generation is completed by keeping the 150 most fit individuals and deleting the rest. This process is continued through 50 generations.

5.2. Memetic Algorithm

A memetic algorithm bridges the gap between genetic algorithms and scatter search (Prins 2005). Scatter search is an evolutionary heuristic that joins solutions based on generalized path constructions in Euclidean space; its approach to finding new solutions is guided more by strategic design, rather than by randomization (Glover et al. 2003). In general, a memetic algorithm's order of processing is the same as that of GA (crossover, change random individuals, return to original population size). Furthermore, in our implementation, MA uses the same couple selection and crossover operators as GA. However, there are distinct differences in the implementation of MA. One of its major differences from GA is that the random mutation step is replaced by a local search that tries many different exchanges of elements within the individual in an attempt to improve its permutation. Hence, unlike a mutation, which acts as a diversification operator, the local search acts as an intensification operator (Prins 2005). Much like mutation, this is only done on a small portion of the population (10%).

The local search is applied to each of the $n(n+1)/2$ pairs of customers within a selected permutation. There are seven different moves that are tested on each pair, subject to feasibility. If a move reduces the permutation's makespan, the new order is kept and the search procedure terminates for this individual. For each pair (i, j) within a permutation

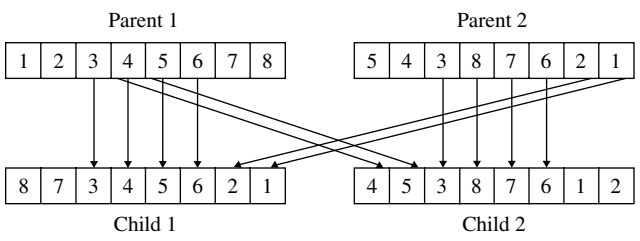

Figure 14 **An Example of Order Crossover:** $a_1 = 3$, $a_2 = 6$

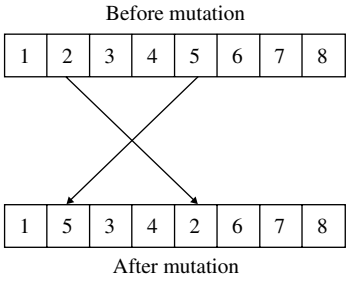

Figure 15 **An Example of Pairwise Exchange Mutation**

$\sigma = (0, \sigma(1), \ldots, \sigma(i), \ldots, \sigma(j), \ldots, \sigma(n))$, the following moves are tested. When customers are moved from one position to another, we assume that the appropriate sliding of customers in intermediate locations occurs:

1. move $\sigma(i)$ to position $j+1$;
2. move $\sigma(i)$ and $\sigma(i+1)$ to positions $j+1$ and $j+2$, respectively;
3. move $\sigma(i)$ and $\sigma(i+1)$ to positions $j+2$ and $j+1$, respectively;
4. swap $\sigma(i)$ and $\sigma(j)$;
5. swap $\sigma(i)$ and $\sigma(j)$, move $\sigma(i+1)$ to position $j+1$;
6. swap $\sigma(i)$ and $\sigma(j)$, swap $\sigma(i+1)$ and $\sigma(j+1)$;
7. move $\sigma(i+1)$ to position $j-1$, move $\sigma(j+1)$ to position $i+1$.

Population management is quite different, too. To ensure diversity, no permutation in the current population can lead to a schedule whose makespan is within a fixed value ($\Delta = 1$ in this study) of the makespan of the schedule generated by any other population member. This requires that the population be significantly smaller (30 individuals). Additionally, replacements in the population are made one member at a time: during each iteration, only one couple is formed, it has a fixed probability of generating a new solution through order crossover, and if their offspring is acceptable (its makespan is unique and in the elite half), it replaces an individual randomly chosen from the less-fit half of the population.

The number of iterations for the memetic algorithm was chosen specifically for each set of parameter values so that its processing time would approximate that of the corresponding run of the genetic algorithm.

6. Computational Results

We derived results for each heuristic over six different data sets: three have 40 customers each, and three have 50 customers each. The locations for each data set are randomly generated from a two-dimensional uniform distribution of equal length and width. The plant is in the center of each of these squares, which are 200×200, 300×300, and 400×400. There is one of each size for each customer count. The demands for each data set follow a discrete uniform distribution $U(100, 300)$ and are generated independently of the locations. Each algorithm was run for these six data sets over three production rates ($r = 1, 2, 3$), two different capacities ($Q = 300, 600$), and two different lifespans ($B = 300, 600$). Hence, we generated results for $6 \times 3 \times 2 \times 2 = 72$ different instances.

Computations were performed in BASIC under Windows XP Professional on a PowerSpec PC with an Intel Pentium 4 processor running at 3.20 GHz with 1.00 GB of memory. The average time for an instance of the GA heuristic over all six data sets was 2 minutes, 49 seconds. The average time for an instance of the MA heuristic was 2 minutes, 32 seconds.

Table 7 Comparison of the Genetic Phase 1 to the Memetic Phase 1

r	Q	B	GA	MA	GA < MA	GA/LB
1	300	300	10,040	10,295	0.067	1.006
2	300	300	9,157	9,570	0.050*	1.004
3	300	300	9,125	9,490	0.018*	1.002
1	300	600	10,041	10,371	0.046*	1.007
2	300	600	9,171	9,493	0.009*	1.007
3	300	600	9,153	9,497	0.018*	1.007
1	600	300	8,781	9,185	0.201	1.000
2	600	300	5,744	6,803	0.035*	1.198
3	600	300	5,421	6,684	0.015*	1.296
1	600	600	8,781	8,798	0.187	1.000
2	600	600	5,724	6,506	0.028*	1.203
3	600	600	5,403	6,344	0.000*	1.303

Notes. Data are averages over the six data sets. Column GA < MA lists p-values for the comparison.

We compare the results from using the two different evolutionary algorithms for Phase 1. Table 7 shows the makespan for each evolutionary algorithm and for each combination of production rate, capacity, and lifespan; these values are averages taken over the six data sets. GA (column 4) has a lower average makespan than MA (column 5) for each parameter combination. On average, GA is 6.30% better than MA. Paired t-tests revealed that globally GA is statistically better than MA (p-value $= 0.0005$). Furthermore, GA is statistically better than MA for $Q = 300$ ($p = 0.0006$) and for $Q = 600$ ($p = 0.0017$); for $B = 300$ ($p = 0.0029$) and for $B = 600$ ($p = 0.0016$); and for each production rate $r = 1, 2, 3$ ($p = 0.0263$, $p = 0.0040$, $p = 0.0020$, respectively). For specific combinations of these parameters, GA's superiority is statistically significant for each one, excepting both combinations with $r = 1$ and $Q = 600$, and the combination $r = 1$, $Q = 300$, $B = 300$. The p-values for these comparisons are in column 6 of Table 7; those that are statistically significant are noted by an asterisk. (Recall that the p-value indicates the probability that our test statistic would assume its observed value if MA = GA.) Our tests were run at the 5% significance level.

GA outperformed MA in part because it considers a far more diverse population of permutations: its population is five times larger at each generation. MA's restriction that no two individuals have the same makespan does create diversity, but its policy of discarding a unique permutation if it generates a makespan that equals that of a current population member can be counter-productive. The discarded permutation may have characteristics that, when combined with those of another population member, lead to a permutation that is superior to

both. Furthermore, verifying that each new permutation has a unique makespan consumes a significant amount of MA's allotted processing time.

Column 7 of Table 7 contains the ratios of GA's makespans to their respective lower bounds developed in §3. GA generated near-optimal solutions (less than 1% above the lower bound) for all instances in which $Q = 300$ or $r = 1$. The algorithms's performance compares less favorably to the lower bounds in the other four instances, largely because the transportation and routing component has more influence on the makespan, and this component requires solving the traveling-salesman problem.

To understand the contributions of the evolutionary algorithms, for each case we compared its result to the makespan of the best schedule in its respective original population. For GA, the average improvement was 17.57%. For MA, it was 12.02%.

7. Conclusions and Recommendations for Future Study

We have analyzed the PTSP for a product with a short lifespan. The case studied includes a single plant, a single limited-capacity truck, and a general number of customers whose locations and demands are random and not correlated. The goal is to group customers into trips, to design routes that serve all customers within each trip, to order the trips, and to schedule production for these trips, all in a way that minimizes the total time required for production and delivery while satisfying the capacity, transportation, and lifespan constraints. Solving this problem is an essential first step toward solving many other operational problems in a production and delivery system for a product with a short lifespan.

Because PTSP is NP-hard in the strong sense, we developed a two-phase heuristic that uses evolutionary algorithms. We compared two different evolutionary algorithms, a genetic algorithm and a memetic algorithm, for the first phase. The second phase divides the permutation output by Phase 1 into trips, reorders the customers within each trip, reorders the trips themselves, and uses a linear program to schedule the reordered trips most efficiently. We found good and, in most instances, near-optimal solutions for 40- and 50-customer data sets. Optimality was determined by comparing results to analytically derived lower bounds. The algorithm's relative difference from these lower bounds are quite good, but indicate that the algorithm is less effective for instances in which the truck's routing has more influence over the makespan. This is expected, given that routing requires solving the traveling-salesman problem. We also found that the genetic algorithm outperformed the memetic algorithm for all instances. This superiority was statistically significant globally, for each parameter value considered individually, and for 75% of the instances tested.

Future research on the PTSP for a product with a short lifespan should involve larger implementations: more trucks of differing capacities and multiple plants. A customer-related enhancement that bears further study is delivery time windows. Obviously, such a constraint will greatly reduce the number of feasible solutions. For some implementations, the dual problem should be considered: Given a fixed planning horizon, how can the percentage of demand satisfied be maximized? The solution to this problem may use a (possibly nonlinear) penalty function whose arguments are the sizes or the percentages of the shortages in each customer delivery. How variations in the penalty function influence whether all customers should have a portion of their demands satisfied, or some customers should have demands completely satisfied while others are not served, also should be considered.

Acknowledgments

This research was supported in part by the University of Texas at Dallas and the Natural Science and Engineering Research Council of Canada (Grant OGP0039682). An associate editor and three anonymous referees provided helpful comments that improved the presentation of the results. François Bellavance provided assistance in analyzing the statistical results.

References

Ahuja, R. K., T. L. Magnanti, J. B. Orlin. 1993. *Network Flows: Theory, Algorithms, and Applications*. Prentice-Hall, Upper Saddle River, NJ.

Arbib, C., D. Pacciarelli, S. Smriglio. 1999. A three-dimensional matching model for perishable production scheduling. *Discrete Appl. Math.* **92** 1–15.

Beasley, J. E. 1983. Route first–cluster second methods for vehicle routing. *Omega* **11** 403–408.

Bourjolly, J.-M., V. Rebetez. 2005. An analysis of lower bound procedures for the bin packing problem. *Comput. Oper. Res.* **32** 395–406.

Cervone, G., R. S. Michalski. 2000. Design and experiments with the LEM2 implementation of the learnable evolution model. Reports of the machine learning and inference laboratory, MLI 00-2, George Mason University, Fairfax, VA.

Chen, Z. L., G. Pundoor. 2004. Order assignment and scheduling in a supply chain. *Oper. Res.* **54** 555–574.

Chen, Z. L., G. Vairaktarakis. 2005. Integrated scheduling of production and distribution operations. *Management Sci.* **51** 614–628.

Christofides, N. 1985. Vehicle routing. E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, D. B. Shmoys, eds. *The Traveling Salesman Problem*. John Wiley & Sons, New York, 431–448.

Cordeau, J.-F., G. Laporte. 2004. Tabu search heuristics for the vehicle routing problem. C. Rego, B. Alidaee, eds. *Metaheuristic Optimization via Memory and Evolution: Tabu Search and Scatter Search*. Springer, New York, 145–163.

Cordeau, J.-F., G. Laporte, A. Mercier. 2004. A unified tabu search heuristic for vehicle routing problems with time windows. *J. Oper. Res. Soc.* **52** 928–936.

Cordeau, J.-F., G. Desaulniers, J. Desrosiers, M. M. Solomon, F. Soumis. 2002. VRP with time windows. P. Toth, D. Vigo, eds. *The Vehicle Routing Problem. SIAM Monographs on Discrete Mathematics and Applications*. SIAM, Philadelphia, 157–193.

Cordeau, J.-F., M. Gendreau, A. Hertz, G. Laporte, J.-S. Sormany. 2005. New heuristics for the vehicle routing problem. A. Langevin, D. Riopel, eds. *Logistics Systems Design and Optimization*. Springer, New York, 279–297.

De Jong, K. A. 1975. An analysis of the behavior of a class of genetic adaptive systems. Doctoral dissertation, University of Michigan. *Dissertation Abstr. Internat.* **36** 5140B. (University Microfilms 76-9381).

Diaby, M., H. C. Bahl, M. H. Karwan, S. Zionts. 1992. A Lagrangean relaxation approach for very-large-scale capacitated lot-sizing. *Management Sci.* **38** 1329–1340.

Gendreau, M., G. Laporte, J.-Y. Potvin. 2002. Metaheuristics for the VRP. P. Toth, D. Vigo, eds. *The Vehicle Routing Problem. SIAM Monographs on Discrete Mathematics and Applications*. SIAM, Philadelphia, 129–154.

Gilmore, P., R. Gomory. 1964. Sequencing a one-state variable machine: A solvable case of the traveling salesman problem. *Oper. Res.* **12** 675–679.

Glover, F., M. Laguna, R. Martí. 2003. Scatter search. A. Ghosh, S. Tsutsui, eds. *Advances in Evolutionary Computation: Theory and Applications*. Springer-Verlag, New York, 519–537.

Goldberg, D. E. 1989. *Genetic Algorithm in Search, Optimization and Machine Learning*. Addison-Wesley, Reading, MA.

Hadjiconstantinou, E., N. Christofides, A. Mingozzi. 1995. A new exact algorithm for the vehicle routing problem based on q-paths and k-shortest paths relaxations. *Ann. Oper. Res.* **61** 21–43.

Hochbaum, D., S. P. Hong. 1996. On the complexity of production and transportation problems. *SIAM J. Optim.* **6** 250–264.

Labbé, M., G. Laporte, H. Mercure. 1991. Capacitated vehicle routing on trees. *Oper. Res.* **39** 616–622.

Laporte, G., F. Semet. 2002. Classical heuristics for the capacitated VRP. P. Toth, D. Vigo, eds. *The Vehicle Routing Problem*. Wiley, Chichester, UK, 337–360.

Lotfi, V., W. H. Chen. 1991. An optimal algorithm for the multi-item capacitated production planning problem. *Eur. J. Oper. Res.* **52** 179–193.

Nahmias, S. 1982. Perishable inventory theory: A review. *Oper. Res.* **30** 680–708.

Palekar, U. S., M. H. Karwan, S. Zionts. 1990. A branch-and-bound method for the fixed-charge transportation problem. *Management Sci.* **36** 1092–1105.

Papadimitrious, C. H. 1977. The Euclidean traveling salesman problem is NP-complete. *Theoret. Comput. Sci.* **4** 237–244.

Prins, C. 2004. A simple and effective evolutionary algorithm for the vehicle routing problem. *Comput. Oper. Res.* **31** 1985–2002.

Prins, C. 2005. Private communication. Institute Charles Delaunay, University of Technology of Troyes, Troyes Cedex, France.

Sarmiento, A., M. R. Nagi. 1999. A review of integrated analysis of production-distribution systems. *IIE Trans.* **31** 1061–1074.

Tuy, H., S. Ghannadan, A. Migdalas, P. Varbrand. 1996. A strongly polynomial algorithm for a concave production-transportation problem with a fixed number of nonlinear variables. *Math. Programming* **72** 229–258.