Available online at www.sciencedirect.com
SCIENCE DIRECT®
European Journal of Operational Research 165 (2005) 668–684
EUROPEAN JOURNAL OF OPERATIONAL RESEARCH
www.elsevier.com/locate/dsw

Discrete Optimization

# Vehicle minimization for periodic deliveries

Ann Melissa Campbell [a], Jill R. Hardin [b,*]

[a] *Department of Management Sciences, Tippie College of Business, 108 John Pappajohn Business Building, Suite W244, University of Iowa, Iowa City, IA 52242, USA*
[b] *Department of Statistical Sciences and Operations Research, Virginia Commonwealth University, 1001 West Main Street, P.O. Box 843083, Richmond, VA 23284-3083, USA*

Received 3 March 2003; accepted 4 March 2004
Available online 11 May 2004

**Abstract**

We consider the problem of minimizing the number of vehicles required to make strictly periodic, single destination deliveries to a set of customers, under the initial assumption that each delivery requires the use of a vehicle for a full day. This problem is motivated by inventory routing problems in which customers consume goods at a steady rate. We evaluate the complexity of the problem and discuss its general properties, including problem decomposition. We also present an algorithm that we show is optimal for special cases. Finally, we extend these results to a general version of the problem.


*Keywords:* Scheduling; Periodic; Complexity theory; Routing; Packing

## 1. Introduction

With the rising use of vendor managed inventory systems, there is an increasing focus on finding efficient ways to manage these systems. Giving the vendor the ability to decide the timing and quantity of deliveries for customers clearly gives the vendor greater flexibility and room for increased efficiency. How to take full advantage of this increased flexibility, though, is still under intense investigation because of the size and complexity of the resulting problems. Solution methodologies must determine times and quantities for the deliveries in conjunction with finding the most efficient routes for making these deliveries, where the routing portion alone is known to be NP-complete [6]. Studies of the resulting problem, known as the inventory routing problem (IRP), look at the problem from both stochastic and deterministic perspectives and decompose the problem in a variety of ways. Literature reviews are included in [2].

In studying the IRP, the savings resulting from conversion to a vendor managed system have been particularly significant when a majority of customers has the capacity to receive full truckload deliveries. This situation makes the routing

* Corresponding author.
*E-mail addresses:* ann-campbell@uiowa.edu (A.M. Campbell), jrhardin@vcu.edu (J.R. Hardin).

0377-2217/$ - see front matter 
doi:10.1016/j.ejor.2003.09.036

portion rather trivial since it is cheaper to postpone deliveries so that these single stop routes can be used. Gallego and Simchi-Levi [5] evaluated the long run effectiveness of strict direct shipping (separate loads to each customer) and concluded that direct shipping is at least 94% effective over all inventory routing strategies whenever minimal customer capacity is at least 71% of truck capacity. The effectiveness, as expected, deteriorates as customer capacity gets smaller. This creates an interesting set of questions: if all routes are restricted to be single stop, is the resulting problem still NP-hard? What algorithms, exact or heuristic, would be good to use for solving this new problem? These questions motivate our study, but the answers are applicable to other situations with periodic demands such as in other areas of truck [7] and ship routing [3] as well as machine scheduling applications [16,20].

This paper examines the structure of the problem, evaluates its complexity in detail, and presents an algorithm that is optimal for special cases, including where all periods are evenly divisible. We also evaluate the worst case performance and extend the algorithm and our analysis to a version of the problem with variable execution times.

The structure of the paper is as follows. In Section 2, we formally define the problem and necessary notation. Section 3 briefly describes the relevant literature, and Section 4 addresses complexity issues. Problem decomposition is discussed in Section 5, an algorithm and a discussion of its performance are in Section 6, and the generalized version of the problem and algorithm are introduced in Section 7. Finally, conclusions and future research questions related to this work are described in Section 8.

## 2. Problem definition

If a customer consumes a product at a steady rate, the customer will reach zero inventory at evenly spaced intervals in time known as *periods*. If a vendor follows a zero inventory direct shipping policy for all customers, then the long run average cost of any schedule should require the same amount of mileage since it is dictated by the set of customers and their periods. Different schedules serving the same set of customers may vary, though, in the number of vehicles required, so the requirements for this resource become key in the formal problem definition.

We consider the problem of minimizing the number of vehicles required to make periodic single destination deliveries to a set of customers $N = \{1, \ldots, n\}$ where customer $i$ requires a delivery precisely every $p_i$ days. The initial assumption is that each delivery requires the use of a full vehicle for an entire day. We wish to determine delivery start days $s_i \in \{1, \ldots, p_i\}$, $\forall i$, in order to minimize the total number of vehicles used on any day of the schedule, where a schedule is created by delivering to customer $i$ on days $\{s_i + qp_i, q = 0, 1, 2, \ldots\}$, $\forall i$. We will refer to this problem as vehicle minimization for periodic deliveries (VMPD). In the paper, we will often refer to $p_i$ and $s_i$ which represent the period and start day for a customer $i$.

## 3. Literature review

Consumption rates for customers have been simplified to periods in the routing literature, creating what is known as the *periodic routing problem* (PRP). Initial formulations and algorithms are proposed in [4,9,23]. Even though algorithms have been proposed for variations of the periodic routing problem with the same objective of minimizing fleet size, as in [7], the routing component clearly has a big influence on the algorithmic design, and the periodicity of the deliveries in these variations of the PRP is also enforced differently. In [7], the periods are defined by the restriction that deliveries to customer $i$ must be at least $K_i$ days apart and no more than $U_i$ days apart. This alters the problem structure dramatically.

The closest work appears to be in machine scheduling, where the issue is how to schedule periodic jobs so as to minimize the number of machines required. In most variations, the periodicity of jobs is not as strictly defined as in the VMPD, such as in [18] where deliveries have "periodic deadlines" and variable, integral execution times. In [12], periodicity is again imposed

through deadlines, and the authors are able to establish certain versions of the problems as being polynomially solvable and others as NP-hard.

Jan Korst's Ph.D. thesis [16] considers the same restrictive form of periodicity that we do in addition to a less restrictive version. Korst considers a periodic scheduling problem very similar to the one we propose but with variable, integral execution times rather than restricting all execution times to one. In the periodic scheduling problem, the question is how to find a schedule of start dates as well as processor assignments that minimizes the number of processors required. He proves the problem to be NP-hard in the strong sense, both when customers are restricted to be repeatedly served by the same vehicle and when they can be served by any vehicle. The proof is based on a reduction from 3-partition and depends on the variability in processing times, so the complexity of the problems we consider is not directly implied by this result. Korst provides a test for determining when two jobs will overlap that will be employed later in this paper.

Other significant results emerge from papers by Park and Yun [20] and Gaudioso et al. [8]. Both address a periodic scheduling problem similar to the one described above, but with one key difference. The work content $w_i$ associated with each activity $i$ cannot be carried from one period to another. In Korst's variation, a job with work content $w_i$ occupies the assigned processor for $w_i$ days, where in [8,20] a job occupies $w_i$ processors on a day. Note that both problems are identical when all $w_i$ values equal one, as in the problem we study. We discovered these papers quite recently and found that they contain some of the same results we had already derived for the VMPD. Both introduce a large integer program to solve the resulting load minimization problem, but Park and Yun also describe a method for decomposing it into smaller subproblems. They divide the set of customers into sets $N_1, N_2, \ldots, N_d$, where for any $i \in N_q$ and any $j \in N_r$, $p_i$ and $p_j$ are relatively prime. They then use integer programming to solve each resulting subproblem, minimizing the number of vehicles required to service each subset of customers $N_q$. With $K_q$ representing the minimum number of vehicles required for customer set $N_q$, Park and Yun show that the minimum number of vehicles required to service the customers in $N = N_1 \cup N_2 \cup \cdots \cup N_d$ is exactly $K_1 + K_2 + \cdots + K_d$ since the decomposition does not increase the number of vehicles required. We take advantage of this result in later sections. Gaudioso et al. [8] also present lower bounds on the minimum number of processors required to schedule all jobs, and they develop an implicit enumeration scheme to solve the problem. This scheme provides the basis of the heuristic procedure developed for the problem.

Other related literature includes [15,19], both of which add a set-up time between tasks with specific properties, and papers such as [17] that consider alternate objectives such as minimization of the cycle time (the length of the schedule that is repeated indefinitely). Kats and Levner [14] address how to sequence the movements of robots in order to minimize the number of robots required to complete a series of tasks over a fixed time horizon. In this case, the periodicity comes from the sequence of each robot's movements rather than from the jobs being processed.

## 4. Complexity

### 4.1. Complexity of VMPD

In investigating the complexity of the VMPD, we begin by trying to determine whether or not VMPD is in NP. Toward this end, we present the following observation.

**Observation 1.** *To obtain a solution for the problem over an infinite time horizon, it is necessary only to find a solution over a time horizon of length L, where* $L = \mathrm{lcm}(p_1, p_2, \ldots, p_n)$.

The validity of this observation is easily established. Any customer $i$ receiving a delivery on day $x$ will also receive a delivery on day $x + L = x + qp_i$, where $q = L/p_i$. Note that, by definition of $L$, $q$ is a positive integer. Thus any schedule of periodic deliveries repeats itself every $L$ days. Unfortunately, finding a schedule over a shorter time horizon is insufficient for solving the VMPD.

**Theorem 1** [21]. *If $a$ and $b$ are positive integers, then* $\operatorname{lcm}(a,b) = ab/\gcd(a,b)$.

**Theorem 2** [21]. *The number of bit operations needed to find the greatest common divisor of two positive integers $a$ and $b$ with $a > b$ is* $O((\log_2 a)^3)$.

Combining these two results, it is clear that $L$ can be computed in polynomial time. This does not imply, however, that VMPD itself is in NP. The magnitude of $L$ can be exponential in the binary representation of the problem, so that any solution method that requires examining each day over such a horizon cannot be polynomial. As we will soon see, VMPD is in fact not in NP, as verification of any solution is NP-hard.

The decision version of VMPD (DVMPD) is defined as follows: *Given customer set* $N = \{1,\ldots,n\}$ *with periods* $p_1,\ldots,p_n$, *respectively, and a positive integer* $k \leqslant n$, *does there exist a start day assignment* $s_i \in \{1,\ldots,p_i\}$, $i = 1,\ldots,n$, *so that at most $k$ vehicles are required on any day of the schedule*?

Towards establishing the complexity of DVMPD, we present the *simultaneous congruences problem* (SCP).

**Definition 1** (*SCP*). Given a set $T$ of $n$ ordered pairs of positive integers $(a_1,b_1),(a_2,b_2),\ldots,(a_n,b_n)$ determine the cardinality of the largest subset $T' \subseteq T$ for which there is a positive integer $x$ with the property that $x \equiv a_i \pmod{b_i}$ for all $(a_i,b_i) \in T'$.

**Theorem 3** [1]. *The simultaneous congruences problem is strongly NP-hard.*

**Observation 2.** *Verifying a solution to DVMPD is equivalent to solving the simultaneous congruences problem.*

**Proof.** Let $(a_i,b_i) = (s_i,p_i)$, $i = 1,\ldots,n$. For any day $x$ in the schedule, $x \equiv s_i \pmod{p_i}$ corresponds to customer $i$ receiving a delivery on day $x$. Finding the largest cardinality subset $T' \subseteq T$ as specified by SCP gives the maximum number of deliveries occurring on any day of the schedule. Comparing $|T'|$ to $k$ verifies the solution to *DVMPD*. □

**Theorem 4.** *DVMPD is strongly NP-hard.*

**Proof.** Verification of a solution to the problem is strongly NP-hard, thus the problem itself is strongly NP-hard. □

Given that DVMPD is strongly NP-hard, then we can say that VMPD is strongly NP-hard. Yet even in cases when a general problem can be shown NP-hard, we are often able to identify special cases which are in $P$. We now consider the complexity of some special cases of VMPD.

### 4.2. Scheduling with a single vehicle

Suppose that we simply wish to determine if deliveries to all customers in $N$ can be made with a single vehicle. That is, we wish to determine start days for all customers so that no two customers require deliveries on the same day.

**Theorem 5** [16]. *If deliveries for customers $i$ and $j$ are scheduled to begin on days $s_i$ and $s_j$ respectively, then $i$ and $j$ will require simultaneous deliveries if and only if* $s_i \equiv s_j \pmod{g_{ij}}$, *where* $g_{ij} = \gcd(p_i,p_j)$.

Given a set of start days $s_i\ \forall_i$, we can use this result to determine in $O(n^2)$ time if all customer deliveries can be made with a single vehicle. We need simply use the condition given in Theorem 5 to determine if any two customers require simultaneous deliveries. It would seem possible to use this straightforward condition of interaction to determine in polynomial time if a set of customers can be scheduled on the same vehicle. Unfortunately, the following result makes this unlikely.

**Theorem 6** ($k$-PD). *Given customer set $N$ and a positive integer $k \leqslant n$, we wish to determine if there is a subset of $k$ customers such that all $k$ can be serviced with the same vehicle. This problem is NP-complete.*

**Proof.** We show that $k$-PD is NP-complete via a reduction from a version of Independent Set, which is well known to be NP-complete [6].

Independent set ($k$-IS) can be stated as follows [6]:

> Given a graph $G = (V, E)$ and a positive integer $k$, does $G$ contain a subset $V' \subseteq V$ of cardinality at least $k$ so that if $u, v \in V'$, then $(u, v) \notin E$? If such a subset exists it is called a *k-independent set*.

Given an instance of $k$-IS (a graph $G$ and an integer $k > 0$), we assume without loss of generality that $V = \{v_1, \ldots, v_n\}$ and that G is a connected simple graph (no loops and no parallel arcs). If $G$ is not connected, we can find an independent set by examining each component separately. Furthermore, we may assume that $G$ is not complete; otherwise, $k$-IS is easy. Create an instance of VMPD as follows:

- Let $\overline{G}$ be the complement of $G$. That is, $\overline{G} = (V, \overline{E})$, where $\overline{E} = \{(i, j) \in V \times V, i \neq j\} \setminus E$. Let $\overline{E} = \{e_1, e_2, \ldots, e_{\bar{m}}\}$.
- Let $\pi_1, \pi_2, \ldots, \pi_{n+\bar{m}}$ denote the first $n + \bar{m}$ primes.
- For each $v_i \in V$, create customer $i$ for VMPD with period

$$p_i = \pi_i \prod_{e_l \in \overline{E} \text{ s.t. } \exists j, e_l = (v_i, v_j)} \pi_{n+l}.$$

  Including $\pi_i$ in the product ensures that customer $i$ has a well defined period, even if it does not have any incident edges in $\overline{G}$.

We claim that $G$ has an independent set of size $k$ if and only if there are at least $k$ customers in $k$-PD such that all $k$ can be serviced with the same vehicle.

($\Rightarrow$) Suppose $G$ has an independent set $V'$. Schedule the customers associated with vertices in $V'$ to begin deliveries on days $1, 2, \ldots, |V'|$. We claim that deliveries for these customers require exactly one vehicle over the planning horizon. First, observe that if $v_i, v_j \in V'$, then $(v_i, v_j) \notin E \Rightarrow (v_i, v_j) \in \overline{E}$. Suppose that $(v_i, v_j) = e_l \in \overline{E}$. By definition, $\pi_{n+l} | p_i$ and $\pi_{n+l} | p_j$. Since $G$ is simple, $\gcd(p_i, p_j) = \pi_{n+l}$. By Theorem 5 we know that deliveries for customers $i$ and $j$ overlap if and only if $s_i \equiv s_j \pmod{g_{ij}}$, that is, if $s_i \equiv s_j \pmod{\pi_{n+l}}$. Note, however, that $s_i, s_j \in \{1, \ldots, |V'|\}$. Since $|V'| \leqslant n < \pi_{n+l}$, $1 \leqslant |s_i - s_j| < \pi_{n+l} \Rightarrow \pi_{n+l} \nmid (s_i - s_j)$. Thus, no two customers associated with vertices in $V'$ overlap.

($\Leftarrow$) Suppose that we can find a subset $U$ of $k$ customers which can all be serviced by the same vehicle. Let $s_i$, $i = 1, \ldots, k$, define the feasible schedule. We will show that if customers $i$ and $j$ are in $U$, then $(v_i, v_j) \notin E$, so $v_i$ and $v_j$ may simultaneously belong to an independent set. To accomplish this, we will prove the contrapositive; that is, we will show that if $(v_i, v_j) \in E$ then $i$ and $j$ cannot be scheduled on the same vehicle. Consider an edge $(v_i, v_j) \in E$. Note that $p_i$ is the product of prime numbers associated with $v_i$ and the edges incident to $v_i$ in $\overline{G}$. Thus $\gcd(p_i, p_j) > 1$ if and only if the same prime number divides each of $p_i$ and $p_j$, which will only occur if both are incident to the same edge in $\overline{E}$. That is, $\gcd(p_i, p_j) > 1$ if and only if $(v_i, v_j) \in \overline{E}$. Since $(v_i, v_j) \in E$, this does not hold, and $\gcd(p_i, p_j) = 1$. Because deliveries for two customers will necessarily collide if their periods are relatively prime, when $(v_i, v_j) \in E$, $i$ and $j$ cannot be scheduled on the same vehicle. We have shown, then, that if two customers can be scheduled on the same vehicle, then the edge between the corresponding vertices cannot appear in $G$. Thus $U$ defines a $k$-independent set in $G$.

We have shown that the "yes" instances of $k$-PD correspond exactly to the "yes" instances of $k$-IS. We have still to establish that the reduction we have provided is indeed polynomial. The prime number theorem [11] implies that the magnitude of $\pi_q$ is $O(q \log q)$. Thus the magnitude of the largest prime used in the above reduction is at most $O([n + \bar{m}] \log[n + \bar{m}])$. Furthermore, each period is the product of at most $n$ prime numbers in the set $\{\pi_1, \ldots, \pi_{n+\bar{m}}\}$, thus each number is of magnitude at most $O(([n + \bar{m}] \log[n + \bar{m}])^n)$, and the largest $p_i$ can be represented by $O(n \log[n + \bar{m}])$ bits. The other steps in the reduction are clearly polynomial. □

### 4.3. Constrained VMPD

In discussing the problem of minimizing the number of vehicles required to complete a schedule, we have not presented any constraints, until now, on the relationship between customers and vehicles such as requiring that the same vehicle be

used for every repeat visit to a customer. That is, we may use vehicle $A$ to make a delivery on day $s_i$ and vehicle $B$ to make a delivery on day $s_i + qp_i$ for any positive integer $q$. Now we add the restriction that customer $i$'s delivery must always be made by the same vehicle and refer to this new problem as the constrained VMPD (CVMPD). Such a situation may arise, for instance, if a customer becomes familiar with a certain driver, or if particular locations are difficult to find and having the same driver make all deliveries to that location is more reasonable. Since it changes the structure of the problem, the complexity of this version must be addressed separately.

**Theorem 7.** *The decision version of CVMPD is NP-complete.*

**Proof.** Korst [16] showed that the problem is NP-hard when deliveries may require an arbitrary length of time. We will show that the special case when deliveries require one full day is still strongly NP-hard using a reduction from the vertex coloring problem. The decision version of vertex coloring can be stated as follows:

> Given a graph $G = (V, E)$ and a positive integer $k$, is there a partition of $V$ into at most $k$ sets $W_1, W_2, \ldots, W_k$ so that if $u, v \in W_i$, then $(u, v) \notin E$? If such a partition exists it is called a $k$-coloring.

Given an instance of vertex coloring (a graph $G$ and an integer $k > 0$), we assume without loss of generality that $V = \{v_1, \ldots, v_n\}$ and that $G$ is a connected simple graph (no loops and no parallel arcs). If $G$ is not connected, the graph may be colored by examining each component separately. Loops make vertex coloring infeasible. Parallel arcs do not affect the coloring at all. Furthermore, we may assume that $G$ is not complete; if it is, the coloring problem is easy. Create an instance of VMPD exactly as in the proof of Theorem 6. By the argument given in that proof, this is a polynomial transformation.

We claim that $G$ has a $k$-coloring if and only if periodic deliveries for customers $c_1, \ldots, c_n$ can be scheduled so that at most $k$ vehicles are required on any day.

($\Rightarrow$) Suppose $G$ has a $k$-coloring. Let $W_1, \ldots, W_k$ represent the corresponding partition of $V$. Note that $W_i$ is an independent set of $G$ for each $i$. Thus, by precisely the same argument as that presented in the proof of Theorem 6, the customers associated with the vertices in $W_i$ can be serviced using a single truck. Therefore we can schedule all customers using at most $k$ vehicles on any day.

($\Leftarrow$) Suppose that we can choose start day $s_i$ for customer $i$, $i = 1, \ldots, n$, so that at most $k$ vehicles are needed on any day. Again, by the same argument presented in the proof of Theorem 6, if customers $i$ and $j$ are scheduled on the same vehicle, then $(v_i, v_j) \notin E$. Thus $v_i$ and $v_j$ may belong to the same set in a partition induced by a coloring. Let $C_r$, $r \in \{1, \ldots, k\}$, denote the set of customers scheduled on the $r$th vehicle. Define a partition of the vertices as $W_r = \{v_i : i \in C_r\}$, $r \in \{1, \ldots, k\}$. Then the preceding argument establishes that this is a $k$-coloring for $G$. This completes the proof. □

## 5. Problem size reduction

As discussed in the literature review, Park and Yun [20] address the solution of this problem by decomposing the set of customers into subsets $N_1, N_2, \ldots, N_d$, where for any $i \in N_q$ and any $j \in N_r$, $p_i$ and $p_j$ are relatively prime. They then use integer programming to solve each subproblem, where subproblem $q$ consists of minimizing the number of vehicles required to service the customers in $N_q$. They show that the minimum number of vehicles required to service all of the customers is exactly the sum of the number of vehicles required to serve each subset.

While solving the integer program modelled in [20] is certain to yield the optimal solution to the problem, the integer programs can be quite large in size and correspondingly time consuming to solve even with this decomposition property. Consider the possibility that if only one customer out of possibly hundreds has a period equal to the least common multiple of the periods of the other customers, then no decomposition is possible since the above condition cannot be satisfied. One of the

difficulties in solving the IRP is the wide variety of consumption rates involved, with some customers requiring multiple trips per day and others with months between deliveries, so such scenarios are not far-fetched. Consider an example where all customers have periods from the set $\{2, 3, 5, 6, 8, 9, 10\}$. This problem also cannot be decomposed at all, even without one period being the *lcm* of the others. Thus, while the decomposition into relatively prime sets is useful, it is not always applicable.

Even if the customer set can be decomposed in this way, the resulting integer programs can still be quite large because both the number of variables and the number of constraints are proportional to the least common multiple of the periods of the customers in the subsets. A general instance of VMPD, however, may include many customers with the same delivery period. It would be helpful if we could use this information to reduce the size of the integer programs. More specifically, consider the following policy. If there are $M \geqslant p$ customers with the same period $p$, assign $K = \lfloor M/p \rfloor$ vehicles to service $Kp$ of those customers. It is a simple matter to verify that those $K$ vehicles can handle the deliveries for all $Kp$ customers, and that they are completely occupied with those deliveries and can handle no others. It would be useful if we could make such an assignment and eliminate those customers from the integer programs in which they appear. As it turns out, we cannot.

**Theorem 8.** *The preprocessing described above may eliminate optimal solutions.*

**Proof.** Consider the customer set $N = \{2_a, 3_a, 6_a, 6_b, 6_c, 6_d, 6_e, 6_f, 6_g\}$, where customer $2_a$ has period 2, customer $3_a$ has period 3, and customers $6_a$, $6_b$, $6_c$, $6_d$, $6_e$, $6_f$, $6_g$ have period 6. To find a solution for this instance of VMPD, we need only consider a schedule over 6 days, since $\text{lcm}(2, 3, 6) = 6$. If we preprocess, we will assign one vehicle to completely handle deliveries to customers $6_a$, $6_b$, $6_c$, $6_d$, $6_e$, $6_f$. Since the numbers 2 and 3 are relatively prime, two additional vehicles will be required to handle deliveries to the remaining customers. We are left, then, with a schedule like the following, requiring three vehicles:

| Day | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| Vehicle 1 | $6_a$ | $6_b$ | $6_c$ | $6_d$ | $6_e$ | $6_f$ |
| Vehicle 2 | $2_a$ | $6_g$ | $2_a$ | $3_a$ | $2_a$ | |
| Vehicle 3 | $3_a$ | | | | | |

The customers in $N$, however, can be scheduled with no more than two vehicles, as evidenced by the following schedule:

| Day | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| Vehicle 1 | $2_a$ | $6_a$ | $2_a$ | $6_b$ | $2_a$ | $6_c$ |
| Vehicle 2 | $3_a$ | $6_d$ | $6_e$ | $3_a$ | $6_f$ | $6_g$ |

□

Thus the size of the integer programs proposed by Park and Yun cannot be reduced by such preprocessing.

Based on the ideas behind the decomposition scheme, we have made the following observations.

**Observation 3.** *If there are k customers whose periods are pairwise relatively prime, then the optimal schedule will require at least k vehicles.*

**Observation 4.** *If $p_1, p_2, \ldots, p_n$ are pairwise relatively prime, then the optimal schedule will require exactly n vehicles.*

These observations follow directly from results in [8,20].

**Observation 5.** *If among the set of customers $N$, there are $k$ distinct periods represented that are pairwise relatively prime and $n_i$ customers of each period $f_i \in \{f_1, \ldots, f_k\}$ then the optimal schedule will require at least $\sum_{i=1}^{k} \lceil \frac{n_i}{f_i} \rceil$ vehicles.*

**Observation 6.** *Suppose that customers in N have periods from the set $\{f_1, \ldots, f_k\}$ where the $f_i$ are*

*pairwise relatively prime. Suppose further that there are $n_i$ customers with period $f_i$, where $n_1 + \cdots + n_k = |N|$. A delivery schedule for $N$ that minimizes the number of vehicles will then require precisely $\lceil \frac{n_1}{f_1} \rceil + \cdots + \lceil \frac{n_k}{f_k} \rceil$ vehicles.*

**Proof.** The intuition for these observations is based on the same set of ideas, so we will present the proof only for the last one. Using the method of Park and Yun, we can partition the customers into $k$ sets, $N_1, \ldots, N_k$ by letting $N_i = \{j \in N : p_j = f_r\}$. Since the periods of any two customers from different sets are relatively prime, we can then schedule each of the $N_i$ separately using $\lceil \frac{n_i}{f_i} \rceil$ vehicles. The optimal schedule will then require $\lceil \frac{n_1}{f_1} \rceil + \cdots + \lceil \frac{n_k}{f_k} \rceil$ vehicles. □

## 6. Greedy algorithm

Since the VMPD and many of its variants have been shown to be NP-hard, we know there can be no polynomial algorithms that can solve the problem exactly. Thus, we were interested in developing either polynomial or pseudo-polynomial algorithms, with run times still much less than integer programs, that can yield good solutions to the problem. We will present a greedy algorithm that can be used as a heuristic for any instance of the problem, but we show that it can give optimal solutions for several special cases. Gaudioso et al. [8] present a branching heuristic with several possible branching rules, and we note that the algorithm we present corresponds to one choice of these rules. We go further by supplying an analysis of the algorithm and its worst case performance. We present the algorithm here for continuity and clarity.

Consider the greedy method, `Greedy Scheduling`, given in Algorithm 1.

**Algorithm 1.** `Greedy Scheduling`

```
1:  for i = 1 to n do
2:    minnum = large
3:    for startday = 1 to p_i do
4:      maxfound = 0
5:      for checkday = startday to L, step
            by p_i do
6:        if vehicles on checkday > maxfound
            then
7:          maxfound = vehicles on checkday
8:        end if
9:      end for
10:     if maxfound < minnum then
11:       minnum = maxfound
12:       minnumday = startday
13:     end if
14:   end for
15:   s_i = minnumday
16: end for
```

`Greedy Scheduling` finds, for each customer $i$, the earliest start day for which customer $i$ will conflict with the fewest other previously scheduled customers on any of its delivery days.

**Example 1.** Let $N = \{1, 2, 3, 4, 5\}$, with $p_1 = 2$, $p_2 = 3$, $p_3 = 4$, $p_4 = 6$, and $p_5 = 12$. We must consider a schedule over $L = 12$ days. `Greedy Scheduling` first schedules customer 1. Since it will conflict with no other customers scheduled so far, we set $s_j = 1$ and we have the following schedule:

| Day | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Vehicle 1 | 1 | | 1 | | 1 | | 1 | | 1 | | 1 | |

We will assume from here forward that customers in $N$ are numbered such that $p_1 \leqslant p_2 \leqslant \cdots \leqslant p_n$, and we let $L = \mathrm{lcm}(p_1, \ldots, p_n)$. Next, we consider customer 2. Regardless of whether deliveries to customer 2 begin on day 1, 2, or 3, deliveries will conflict with deliveries to customer

1 at some point over the twelve day horizon. Thus, we set $s_2 = 1$, resulting in the following schedule:

| Day | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Vehicle 1 | 1 | | 1 | 2 | 1 | | 1 | | 1 | 2 | 1 | |
| Vehicle 2 | 2 | | | | | | 2 | | | | | |

Now we focus on customer 3, with $p_3 = 4$. Choosing start day 1 or 3 results in conflict with both customers 1 and 2 on the same day; choosing start day 2 or 4 results in conflict only with customer 2. So we set $s_3 = 2$ and we have the following schedule:

| Day | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Vehicle 1 | 1 | 3 | 1 | 2 | 1 | 3 | 1 | | 1 | 2 | 1 | |
| Vehicle 2 | 2 | | | | | | 2 | | | 3 | | |

`Greedy Scheduling` next considers customer 4, with a period of 6. Choosing start day 1 or 4 results in conflict with two other customers on some day of the schedule; choosing any other start day results in conflict with at most one customer on any day of the schedule. Thus, we set $s_4 = 2$ and our schedule becomes:

| Day | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Vehicle 1 | 1 | 3 | 1 | 2 | 1 | 3 | 1 | 4 | 1 | 2 | 1 | |
| Vehicle 2 | 2 | 4 | | | | | 2 | | | 3 | | |

Finally, `Greedy Scheduling` considers customer 5, with a period of 12. Customer 5 can be scheduled for delivery on day 12 which results in no conflict with any other customer. Scheduling customer 5 on any other day of the horizon results in conflict with at least one other customer. We set $s_5 = 12$, and obtain our final delivery schedule:

| Day | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Vehicle 1 | 1 | 3 | 1 | 2 | 1 | 3 | 1 | 4 | 1 | 2 | 1 | 5 |
| Vehicle 2 | 2 | 4 | | | | | 2 | | | 3 | | |

### 6.1. Relatively prime periods

**Theorem 9.** *Suppose that customers in $N$ have periods from the set $\{f_i, \ldots, f_k\}$, $f_i \leqslant f_{i+1}$, where the $f_i$ are pairwise relatively prime. Suppose further that there are $n_i$ customers with period $f_i$, where $n_1 + \cdots + n_k = |N|$. Then* `Greedy Scheduling` *provides an optimal schedule.*

**Proof.** We present a proof by induction. We first note that `Greedy Scheduling` will always choose to schedule the first customer to begin deliveries on day 1. For each subsequent customer, then, it chooses the earliest start day which results in the fewest conflicts with other customers on any day of the schedule.

Consider the $n_1$ customers of period $f_1$. If $n_1 \leqslant f_1$, then `Greedy Scheduling` will assign start days $1, 2, \ldots, n_1$ to these customers; otherwise, it will assign start day $c$ to $\lceil \frac{n_1}{f_1} \rceil$ customers, $c = 1, \ldots, n_1 - \lfloor \frac{n_1}{f_1} \rfloor$, and start day $c$ to $\lfloor \frac{n_1}{f_1} \rfloor$ customers, $c = n_1 - \lfloor \frac{n_1}{f_1} \rfloor + 1, \ldots, f_1$. As any schedule for these $n_1$ customers will require at least $\lceil \frac{n_1}{f_1} \rceil$ vehicles, `Greedy Scheduling` creates an optimal schedule. This establishes the basis for the inductive proof.

Suppose that the schedule created by `Greedy Scheduling` for the first $n_1 + \cdots + n_{r-1}$ customers is optimal, and consider scheduling the first customer $j$ of period $f_r$. Since $f_r$ is relatively prime

to each of $f_1, \ldots, f_{r-1}$, the Chinese remainder theorem [10] implies that, for any start day $s \in \{1, \ldots, f_r\}$, there is a day $x \in \{1, \ldots, L\}$ on which a delivery to customer $j$ occurs simultaneously with at least $\lceil \frac{n_i}{f_i} \rceil$ customers of period $f_i$, $i = 1, \ldots, r-1$. Therefore `Greedy Scheduling` will assign start day 1 to the first customer of period $f_r$. It then continues assigning start days to the remaining $n_r - 1$ customers of period $f_r$ in a circular fashion, similar to that described for the $n_i$ customers of period $f_1$.

Note, then, that each time a new period $f_i$ is encountered, `Greedy Scheduling` will be forced to schedule those $n_i$ customers in a manner identical to the way it would schedule them if no other customers were present. In essence, `Greedy Scheduling` assigns start days to customers of frequency $f_i$ independently of all other customers. By Observation 6, `Greedy Scheduling` finds an optimal schedule. □

### 6.2. Evenly divisible periods

**Definition 2.** We say that the periods $p_1 \leqslant p_2 \leqslant \cdots \leqslant p_n$ are *evenly divisible* if $p_i | p_{i+1}$, $i = 1, \ldots, n-1$.

**Theorem 10.** *If* $p_i | p_{i+1}$, $i = 1, \ldots, n-1$, *then* `Greedy Scheduling` *produces an optimal schedule.*

**Proof.** First, observe that, for any $k \leqslant n$, $\mathrm{lcm}(p_1, \ldots, p_k) = p_k$, since $p_k$ is itself a multiple of each $p_i$, $i < k$. We will present a proof by induction.

As has been stated previously, `Greedy Scheduling` easily finds an optimal schedule for one customer. Assume that `Greedy Scheduling` has produced an optimal solution after scheduling the first $k-1$ customers and is ready to schedule customer $k$.

We observe that `Greedy Scheduling` will assign $s_k \in \{1, \ldots, p_{k-1}\}$. Since $\mathrm{lcm}(p_1, \ldots, p_{k-1}) = p_{k-1}$, the schedule for the first $k-1$ customers will repeat every $p_{k-1}$ days. The number of deliveries scheduled on day $x$ will be the same as the number of deliveries scheduled on day $x + qp_{k-1}$ for all positive integers $q$. Because $p_k$ is a multiple of $p_{k-1}$, if $x$, where $p_{k-1} < x \leqslant p_k$, is a candidate for start day of customer $k$, then so is day $x - p_{k-1}$. `Greedy Scheduling` always chooses the earliest start day using the minimum number of vehicles if there is a tie, so the algorithm will assign $s_k \in \{1, \ldots, p_{k-1}\}$.

Suppose that a start day can be found for customer $k$ that does not require the addition of a vehicle. Clearly, since the schedule is optimal for the first $k-1$ customers and no additional vehicle is required, the schedule for the first $k$ customers is optimal. Now suppose that for each $s \in \{1, \ldots, p_{k-1}\}$, assigning $s_k = s$ would require an additional vehicle. We claim that, in this case, the schedule is "full"; that is, the same number of deliveries are scheduled for each day in the horizon.

The claim that requiring an additional vehicle implies a full schedule is based on the following argument. Suppose that the optimal schedule for the first $k-1$ customers requires $R$ vehicles. For each $s \in \{1, \ldots, p_{k-1}\}$, there is some $t \in \{s + qp_k, q = 1, \ldots, \frac{p_n}{p_k} - 1\}$ on which $R$ deliveries are already scheduled. We note, however, that the deliveries on days in $\{s + qp_k, q = 1, \ldots, \frac{p_n}{p_k} - 1\}$ are all to precisely the same set of customers, since $p_i | p_k$ $\forall i \leqslant k$. That is, if customer $j$ receives a delivery on day $s$, it will also receive a delivery on day $s + qp_k = s + (q\frac{p_k}{p_i})p_i$, for any $q \in \{1, \ldots, \frac{p_n}{p_k} - 1\}$. Since this is true for each start day in $\{1, \ldots, p_{k-1}\}$, the same number of deliveries must occur on each day of the schedule. Daily use of all vehicles is clearly the best that can be done, and any optimal schedule for customers $1, \ldots, k$ would require the use of an additional vehicle. Since all start days are equivalent, `Greedy Scheduling` will assign $s_k = 1$ to produce an optimal schedule. This completes the proof. □

Though it is not likely that all customers will naturally require delivery periods that are multiples of one another or the same as other periods, this result can still be very helpful in practice. Customers can be restricted to choose from a defined set of delivery periods, for example, or the vendor can influence a customer to modify the amount of inventory held to alter the time required between deliveries. This is not uncommon in vendor managed inventory relationships. Another

option is to use this result simply for estimating the number of vehicles required to serve a set of customers. For example, the periodicity of each delivery could temporarily be replaced by the closest of a series of divisible values and the algorithm invoked to create a quick estimate. The idea of using periods that are all evenly divisible into later periods is similar to the ideas behind the well-known inventory replenishment policies developed by Roundy. In [22], a period is the time between order for a set of retailers $(T_1, \ldots, T_n)$ or their warehouse $(T_o)$. The authors show that a policy where either $T_i/T_o$ or $T_o/T_i$ is integral for all $i$ is within 6% of optimality. Since the value $T_i$ represents the time between receiving an order and reaching a zero-inventory level, the parallels to our problem are obvious. This result can be improved to 98% when all periods are even further restricted to be powers of two.

**Theorem 11.** *If the customer set $N = \{1, \ldots, n\}$ can be partitioned into $N_1, \ldots, N_k$ so that the periods for customers within each $N_i$ are evenly divisible and so that if $v \in N_i$, $w \in N_j$, $i \neq j$, then $p_v$ and $p_w$ are relatively prime, then* `Greedy Scheduling` *finds an optimal solution.*

**Proof.** Straightforward application of Theorems 9 and 10, along with observations made by Park and Yun, establish this result. □

Thus even if all periods are not evenly divisible, but can be decomposed into pairwise relatively prime groups that are evenly divisible within the groups, we still can find an optimal solution without solving an integer program.

### 6.3. Performance of the greedy algorithm

In general, `Greedy Scheduling` is pseudopolynomial (it runs in $O(nL)$ time) since its execution requires examining each day of the $L$-day horizon at least once for each customer. If, however, $L$ is polynomial in $n$, then `Greedy Scheduling` is polynomial. Note also that in the case that $f_1, \ldots, f_k$ are pairwise relatively prime. Observation 6 implies that scheduling can be done in $O(n)$ time: when scheduling customers of period $f_i$, assign start days $\{1, \ldots, f_i\}$ in cyclical fashion. In the case of evenly divisible periods, `Greedy Scheduling` can also be modified to run in $O(n)$ time, based on arguments similar to those given in the proof of Theorem 10.

While pseudopolynomial algorithms are not as desirable as those that run in polynomial time, we note that, even in the worst case, this is still much faster than Park and Yun's integer programming approach. Those integer programs have $L$ constraints and more than $n$ variables, implying that their solution time is likely to be exponential in $n$. Unfortunately, in the general case, we cannot depend on `Greedy Scheduling` to provide optimal schedules.

**Theorem 12.** `Greedy Scheduling` *does not yield optimal schedules for general instances of VMPD.*

In spite of the fact that `Greedy Scheduling` does not always yield an optimal solution, it can still be used to obtain feasible schedules for VMPD which are of reasonable quality. Note, for instance, the example given in the proof of Theorem 8. `Greedy Scheduling` yields the optimal schedule for this example. It often yields much better schedules than would be produced if customers of each distinct period were scheduled separately, and never worse (given that each period is scheduled separately according to the steps of `Greedy Scheduling`). Even though `Greedy Scheduling` runs in pseudopolynomial time in the worst case, this is still an improvement over integer programming when a relatively quick, feasible, approximate solution is desired.

#### 6.3.1. Bound on greedy algorithm

We can make some statements, though, about the performance of `Greedy Scheduling` relative to optimality.

**Theorem 13.** *Let $\mathrm{OPT}_I$ be the optimal number of vehicles required to schedule customers in instance I of VMPD. Then* `Greedy Scheduling` *produces a schedule requiring at most $\mathrm{OPT}_I + D - 1$ vehicles where $D$ is the number of distinct periods in $p_1, \ldots, p_n$.*

**Proof.** Assume that customers in $N$ have periods from the set $\{f_1, \ldots, f_k\}$, $f_i \leqslant f_{i+1}$, and that there are $n_i$ customers with period $f_i$ with $n_1 + \cdots + n_k = |N|$. `Greedy Scheduling` can clearly find an optimal solution for the set of $n_1$ customers with period $f_1$. Note, however, that there may be many alternative optima; that is, there may be many different schedules, each requiring the minimum number of vehicles. The choice of such a schedule can significantly impact scheduling of subsequent customers. Schedule choice may result in the premature addition of a vehicle when customers of period $f_{i+1}$, for example, are considered. Even if such a premature increase occurs, up to $f_{i+1} - 1$ additional customers of period $f_{i+1}$ can be scheduled without another increase in the number of vehicles (due to the structure of `Greedy Scheduling`). In any case, there still may be multiple schedules requiring the same number of vehicles to serve the set of currently scheduled customers. As before, the choice of schedule impacts when additional vehicles are required as additional periods are considered. Using this basic argument, we can see that the largest difference in deliveries occurring on any two days of the schedule is at most $D - 1$ since a vehicle can only be added "prematurely" once for each distinct period. Thus the number of vehicles required by `Greedy Scheduling` will exceed the minimum possible by at most $D - 1$. □

*6.3.2. Bound on any algorithm*

Given the structure of the `Greedy Scheduling` algorithm, it is difficult to give a good upper bound on its performance that is unique strictly to `Greedy Scheduling` to use in bounding results. We can make some claims, though, on the performance of *any* algorithm that can be used to create feasible schedules for a set of customers with known periods.

A lower bound $\sum_{i=1}^{n} 1/p_i$ on the number of vehicles required in any schedule is given in [8], but this lower bound does not take advantage of the existence of customers with relatively prime periods. The term $1/p_i$ reflects the amount of work required for a customer of a particular period if that work could be distributed evenly over the horizon. Given that we can decompose the customer set $N$ into $g$ relatively prime subsets $G_1, \ldots, G_g$, as defined by Park and Yun, we improve upon this lower bound and define the infimum of all optimal policies as follows:

$$\mathrm{INF}_{\mathrm{opt}} = \sum_{j=1,\ldots,g} \left\lceil \sum_{i \in G_j} \frac{1}{p_i} \right\rceil. \tag{1}$$

There may actually be very few such groupings but many periods that are relatively prime. In this case, we can possibly strengthen the above given that we have $c$ relatively prime periods, so that at least $c$ vehicles are required:

$$\mathrm{INF}_{\mathrm{opt}} = c + \max\left(0, \left\lceil \sum_{i=1 \in n} \frac{1}{p_i} - c \right\rceil\right). \tag{2}$$

For simplicity and ease of presentation, we will restrict our analysis to instances where the number of customers equals the number of distinct periods (one customer for each period). If there is only one customer of each period, the worst any algorithm can do is require $n$ vehicles. This is an upper bound on what could happen, yet the algorithm we proposed could often perform quite better than this. If we take the ratio of this upper bound with (2), for example, we find the following:

$$\frac{n}{c + \max\left(0, \left\lceil \sum_{i=1 \in n} \frac{1}{p_i} - c \right\rceil\right)}. \tag{3}$$

This simplifies to yield the following result.

**Theorem 14.** *Any algorithm for the VMPD will achieve at worst a factor* $\frac{n}{c}$ *result, where* $c$ *is the number of pairwise relatively prime periods.*

Thus, as $c$ approaches $n$, the effectiveness of any algorithm, including `Greedy Scheduling` improves.

## 7. Variable execution times

In the original literature on the vehicle routing problem, it is assumed that each route fully occupies a vehicle for the day. Later work examines the idea of multiple trips per day for a vehicle, as we now will do for the VMPD. In this generalization

of the VMPD, we can represent the fraction of a day that would be occupied by a delivery to customer $i$ by a value $e_i$, where $e_i \leqslant 1$, and 1 is the length of time available for each vehicle each day. We still make the assumption that a vehicle must return to the depot before visiting another customer, but the same vehicle may be used to make multiple deliveries in a given day. Because each vehicle can make multiple trips, this variation will be identified as the MTVMPD. The MTVMPD is essentially a scaled version of the full problem modeled in [20].

This problem is clearly NP-hard in the strong sense since if each customer has a period of 1, then the problem of how to minimize the number of vehicles required each day is the bin packing problem exactly. Based on the same argument, we can make the stronger statement:

**Theorem 15.** *MTVMPD is NP-hard in the strong sense even if all periods are identical.*

We can extend the greedy algorithm proposed earlier to handle varying execution times. The key difference in the problems, and also in the algorithms, is that a solution is no longer defined only by a start day. We also need to know which vehicle will be performing the delivery on each day that the customer receives a delivery over the lcm. The algorithm we propose handles this issue in a "first fit" fashion, similar to the structure of a well known bin packing heuristic. We again assume that the customers are sorted in order of nondecreasing period and are scheduled in this order. In scheduling each customer, the algorithm looks for the first start day where the customer can be visited on each repetition over the lcm without requiring an increase in the number of vehicles. On each day where a delivery occurs for a customer $i$, the delivery is assigned to the first vehicle that has time available to make the delivery (time available $\geqslant e_i$). If $i$ cannot be feasibly scheduled on any of the start days with the current number of vehicles, `Greedy Scheduling with First Fit` increases the number of vehicles by 1 and $i$ is assigned day 1 as its start day. From day 1 forward, customer $i$ is only assigned to the new vehicle if there is not sufficient time available on any of the existing vehicles. This procedure combines the virtues of the algorithm described earlier

**Algorithm 2.** `Greedy Scheduling with First Fit`

```
1:  minnum = 1
2:  for i = 1 to n do
3:    startday = 1
4:    done = 0
5:    while (startday ⩽ p_i) and (done = 0) do
6:      reset planned assignment values
7:      minreqd = minnum
8:      for checkday = startday to L, step by p_i do
9:        checkveh = 1
10:       done2 = 0
11:       while (checkveh ⩽ minnum) and (done2 = 0) do
12:         if time available on checkveh ⩾ e_i then
13:           planned assignment for i on checkday = checkveh
14:           done2 = 1
15:         end if
16:         checkveh + 1
17:       end while
18:       if done2 = 0 then
```

```
19:              minreqd = minnum + 1
20:            end if
21:          end for
22:          if minreqd = minnum then
23:            assign i to days and vehicles according to plan
24:            done = 1
25:          end if
26:        end while
27:        if done = 0 then
28:          minnum = minnum + 1
29:          startday for i = 1
30:          for checkday = 1 to L, step by p_i do
31:            checkveh = 1
32:            done2 = 0
33:            while (checkveh ⩽ minnum) and (done2 = 0) do
34:              if time available on checkveh ⩾ e_i then
35:                assignment for i on checkday = checkveh
36:                done2 = 1
37:              end if
38:              checkveh + 1
39:            end while
40:          end for
41:        end if
42:      end for
```

with the structure of bin packing algorithms that have been shown to perform well.

Due to the bin packing aspects of the problem, it is no longer true that the problem can be decomposed by the relatively prime groupings as before. All deliveries with relatively prime frequencies will still coincide, but these collisions will not necessarily require additional vehicles. If there are $n$ customers, for example, with pairwise relatively prime periods, the exact number of vehicles required will no longer be $n$ but

$$\left\lceil \sum_{i=1}^{n} e_i \right\rceil. \tag{4}$$

Adapting and bounding this algorithm for a general class of periods will be a focus of further study, but we can still extend some of our earlier statements for the case when all periods are evenly divisible.

### 7.1. Evenly divisible periods

The performance of this algorithm when periods are evenly divisible is very interesting. Since each period is the lcm of the previously scheduled periods, the first vehicle assignment becomes the assignment for all future visits to the same customer. This special structure combined with the results from using a "first fit" bin packing heuristic allows the following result:

**Theorem 16.** *If periods are evenly divisible, then the* `Greedy Scheduling with First Fit` *algorithm can be used to schedule customers such that the number of vehicles required is less than* 2 *times the optimal number of vehicles.*

**Proof.** The proof of this statement is based on arguments very similar to those that create the factor-two result established for the first fit heuristic developed for the bin packing problem [6], so

we will begin there. In an instance of the bin packing problem, there is a set of items $u_i$ each with size $s(u_i) \leqslant 1$. The objective is to find the minimum number of unit-sized bins needed such that each item can be inserted in a bin. The first fit algorithm [6] starts with a series of these unit-capacity bins, all of which are empty. The algorithm then places items $u_i$ into the bins, always placing the next indexed item into the lowest indexed bin for which there is sufficient available capacity. In other words, $u_i$ is inserted into the first bin in which it will "fit". It is clear that the following is true regarding the optimal number of bins ($\mathrm{OPT_{BP}}$):

$$\mathrm{OPT_{BP}} \geqslant \left\lceil \sum_{i=1}^{n} s(u_i) \right\rceil. \tag{5}$$

It is also true that the result of the first fit heuristic ($\mathrm{FF_{BP}}$) is such that

$$\mathrm{FF_{BP}} < \left\lceil 2 \sum_{i=1}^{n} s(u_i) \right\rceil. \tag{6}$$

The above is true since there is at most one non-empty bin in the first fit packing with assigned contents equal to .5 or less. Combined, the above two equations yield the following well known result

$$\mathrm{FF_{BP}} < 2\mathrm{OPT_{BP}}. \tag{7}$$

Improvements have been made to the bound for the first fit algorithm [13] so that

$$\mathrm{FF_{BP}} \leqslant \frac{17}{10}\mathrm{OPT_{BP}} + 2. \tag{8}$$

For the MTVMPD, we can also lower bound the optimal number of vehicles required ($\mathrm{OPT_{MTVMPD}}$).

**Lemma 1**

$$\mathrm{OPT_{MTVMPD}} \geqslant \left\lceil \sum_{i=1}^{n} \frac{e_i}{p_i} \right\rceil. \tag{9}$$

**Proof.** This result is based on the fact that number of deliveries to a customer $i$ over $L$ equals

$$\frac{L}{p_i}. \tag{10}$$

Each delivery requires $e_i$ amount of time each time it is executed, for a total of

$$\frac{e_i L}{p_i} \tag{11}$$

over the horizon. If we sum this amount over all customers and divide this total amount of work evenly over $L$, we obtain

$$\sum_{i=1}^{n} \frac{e_i L}{p_i L}. \tag{12}$$

Simplifying this expression completes the proof of the lemma. □

The `Greedy Scheduling with First Fit` algorithm works from smallest period to largest. This preserves many of the properties established earlier about the greedy algorithm when applied to evenly divisible periods. For example, we still only have to make each decision based on the customers assigned to the $p_i$ start days possible for customer $i$ since $p_i$ is the least common multiple of the periods of all of the prior customers. If a customer is assigned to a particular start day, each future occurrence will be on a day requiring the same number of vehicles and using the same vehicle.

For each customer, the algorithm evaluates each start day from the first to the $p_i$th to find the first day on which there is a vehicle (bin) in which there is available time (capacity) to include the delivery in question. If no vehicles on any of the possible start days have sufficient availability, a new vehicle is enlisted and first used on day 1 to make the delivery in question. Once a new vehicle is enlisted, it becomes available on the other days. In this way, no new vehicle is used on a day unless all but at most one other vehicle has at least 50% of its time scheduled. In other words, we have the same quality as with first fit as applied to bin packing: there is "at most one non-empty bin with assigned contents .5 or less". The result of using `Greedy Scheduling with First Fit` ($\mathrm{GFF_{MTVMPD}}$) is then:

$$\mathrm{GFF}_{\mathrm{MTVMPD}} < \left\lceil 2\sum_{i=1}^{n} \frac{e_i}{p_i} \right\rceil \tag{13}$$

which is the desired result. It is possible that the approximation factor may be improved to the level in (8), but it will require further study. □

## 8. Conclusions and future directions

In addition to the future research directions indicated within the paper, we are also interested in studying a variation that includes another dimension to the problem: the size of each customer's delivery. Each customer would still require a full truck load delivery, but each customer would require a visit by a vehicle with minimum capacity $c_i$ where $c_i$ is one of a set of options. The objective could be modified to charge based on the vehicles used or restrictions could be placed on the number of each size of vehicle available for use. This is much closer to the problems that occur in practice.

The VMPD is an interesting problem with a variety of applications in the changing world of logistics and scheduling. It is deceptively easy to describe, but NP-hard to solve. In this paper, we have presented classes of instances that can be solved exactly and theoretical results concerning the properties of the problem and its variants. These results should be helpful for decomposing problems as well as approximating solutions to practical problems. We have also described an extension, the MTVMPD, and modified the proposed algorithm and selected results accordingly.

## Acknowledgements

This research was partially funded by the National Science Foundation under Award Number DMI 02-37726 (Campbell). The authors wish to thank Martin Savelsbergh for his valuable input regarding this research.

## References

[1] S.K. Baruah, R.R. Howell, L.E. Rosier, On preemptive scheduling of periodic, real-time tasks on one processor, in: Mathematical Foundations of Computer Science, Springer-Verlag, Berlin, 1990, pp. 173–179.
[2] A. Campbell, L. Clarke, A. Kleywegt, M. Savelsbergh, Inventory routing, in: T. Crainic, G. Laporte (Eds.), Fleet Management and Logistics, Kluwer Academic Publishers, 1998.
[3] M. Christiansen, B. Nygreen, A method for solving ship routing problems with inventory constraints, Annals of Operations Research 81 (1998) 81.
[4] N. Christofides, J. Beasley, The period routing problem, Networks 14 (2) (1984) 237–256.
[5] G. Gallego, D. Simchi-Levi, On the effectiveness of direct shipping strategy for the one-warehouse multi-retailer R-systems, Management Science 36 (2) (1990) 240–243.
[6] M. Garey, D. Johnson, Computers and Intractability, W.H. Freeman and Company, 1979.
[7] M. Gaudioso, G. Paletta, A heuristic for the periodic vehicle routing problem, Transportation Science 26 (2) (1992) 86–92.
[8] M. Gaudioso, G. Paletta, S. Sanna, Management of periodic demands in distribution systems, European Journal of Operational Research 20 (1985) 234–238.
[9] B. Golden, A. Assad, R. Dahl, Analysis of a large scale vehicle routing problem with an inventory component, Large Scale Systems 7 (2–3) (1984) 181–190.
[10] R. Graham, D. Knuth, O. Patashnik, Concrete Mathematics, second ed., Addison-Wesley, 1994.
[11] T.W. Hungerford, Abstract Algebra: An Introduction, Saunders College Publishing, 1990.
[12] K. Jeffay, D. Stanat, C. Martel, On non-preemptive scheduling of periodic and sporadic tasks, in: Proceedings of the Twelfth IEEE Real-Time Systems Symposium, December 1991.
[13] D. Johnson, A. Demers, J. Ullman, M. Garey, R. Graham, Worst-case performance bounds for simple one-dimensional packing algorithms, SIAM Journal of Computing 3 (4) (1974) 299–325.
[14] V. Kats, E. Levner, Minimizing the number of robots to meet a given cyclic schedule, Annals of Operations Research 69 (1997) 209–226.
[15] V. Kats, E. Levner, Minimizing the number of vehicles in periodic scheduling: The non-Euclidean case, European Journal of Operational Research 107 (2) (1998) 371–377.
[16] J. Korst, Periodic multiprocessor scheduling, Ph.D. thesis, Technical University of Eindhoven, December 1992.
[17] T. Lee, M. Posner, Performance measures and schedules in periodic job shops, Operations Research 45 (1) (1997) 72–91.
[18] J.Y.-T. Leung, J. Whitehead, On the complexity of fixed-priority scheduling of periodic, real-time tasks, Performance Evaluation 2 (4) (1982) 237–250.

[19] J.B. Orlin, Minimizing the number of vehicles to meet a fixed periodic schedule: An application of periodic posets, Operations Research 30 (4) (1982) 760–776.
[20] K.S. Park, D.K. Yun, Optimal scheduling of periodic activities, Operations Research 33 (3) (1985) 690–695.
[21] K. Rosen, Elementary Number Theory and its Applications, Addison-Wesley, 1984.
[22] R. Roundy, 98%-effective integer-ratio lot-sizing for one-warehouse multi-retailer systems, Management Science 31 (11) (1985) 1416–1430.
[23] C.C. Tan, J.E. Beasley, A Heuristic algorithm for the period vehicle routing problem, OMEGA International Journal of Management 12 (5) (1984) 497–504.