Operations Research

Charlemagne's Challenge: The Periodic Latency Problem

Sofie Coene, Frits C. R. Spieksma, Gerhard J. Woeginger,

Please scroll down for article—it is on subsequent pages

INFORMS is the largest professional society in the world for professionals in the fields of operations research, management science, and analytics.
For more information on INFORMS, its publications, membership, or meetings visit http://www.informs.org

Charlemagne's Challenge:
The Periodic Latency Problem

Sofie Coene, Frits C. R. Spieksma
Operations Research Group, Katholieke Universiteit Leuven, B-3000 Leuven, Belgium
{sofie.coene@econ.kuleuven.be, frits.spieksma@econ.kuleuven.be}

Gerhard J. Woeginger
Department of Mathematics, Eindhoven University of Technology, 5600 MB Eindhoven, The Netherlands, gwoegi@win.tue.nl

Latency problems are characterized by their focus on minimizing the waiting time for all clients. We study *periodic* latency problems, a nontrivial extension of standard latency problems. In a periodic latency problem each client has to be visited regularly: there is a server traveling at unit speed, and there is a set of n clients with given positions. The server must visit the clients over and over again, subject to the constraint that successive visits to client i are at most q_i time units away from each other.

We investigate two main problems. In problem PLPP the goal is to find a repeatable route for the server visiting as many clients as possible without violating their q_is. In problem PLP the goal is to minimize the number of servers needed to serve all clients. Depending on the topology of the underlying network, we derive polynomial-time algorithms or hardness results for these two problems. Our results draw sharp separation lines between easy and hard cases.

Prologue

During his reign in the years 768–814, Charlemagne traveled constantly through his empire in Western Europe. Counts had been appointed to govern different pieces of Charlemagne's empire (called counties). On his travels, Charlemagne visited his counts regularly. One reason for these visits was to ensure the loyalty of his counts. Indeed, when a count was not visited for a certain period, the count would no longer obey Charlemagne, and declare independence, thereby rising against the emperor. Clearly, this would force Charlemagne to act and start an expensive war against the rebelling count. Charlemagne's challenge was to find a visiting sequence of his counts so that the time elapsed between two consecutive visits to a count would not exceed the "loyalty period" of that count.

1. Introduction

In the periodic latency problem, we are given a set of customers that need to be visited periodically. There is a distance between each pair of customers, and for each customer there is a maximum allowed time period between two consecutive visits. A single server is used for visiting customers, and if the server visits a customer periodically, a given profit is incurred. The goal is to visit a subset of the customers periodically in order to maximize total collected profit. We call this problem the periodic latency problem

with profits (PLPP). We also consider the problem where multiple servers are available and all customers need to be served. We assume here that a customer must be served by a single server; see §5 for the relevance of this assumption. The goal is then to minimize the number of servers required to serve all customers periodically. We call this problem the periodic latency problem (PLP). Further, we also deal with the periodic latency problem with profits and multiple servers (MPLPP), where again the goal is to find routes for the servers such that total collected profit is maximized.

Why "Latency?" Latency problems are characterized by their focus on total waiting time as an objective function; see, e.g., de Paepe et al. (2004) and the references contained therein. Latency problems differ from problems where travel time of the server is the objective. Notice that the problems that we study here share the same fundamental property with latency problems: any period in time matters to all customers. That is why we refer to the problems described here as periodic latency problems; indeed, time matters for the customers, whereas the distance traveled by the server is of no interest.

Problem Statement and Notation. Let us now describe the PLPP in greater detail. We are given a set of customers $N = \{1, 2, \ldots, n\}$ with their positions $x_1, x_2, \ldots, x_n$ in some metric space; for each pair of customers $i, j \in N$, there is a distance d_{ij}. We are also given a

server that travels at unit speed (and always at full speed). There is a number q_i associated with every customer i that indicates the *periodicity* of customer i, $i \in N$. More precisely, q_i is the maximal amount of time that is allowed to pass between two consecutive visits to customer i, $i \in N$. Each customer $i \in N$ also has an associated profit p_i. A customer i is called *served* when the time elapsed between each two consecutive visits does not exceed q_i, $i \in N$. The goal is to find a travel plan for the server that maximizes the total profit of the customers served. This travel plan can be represented by a list of customers (of infinite length) that prescribes the sequence in which the served customers are visited. Thus, in a feasible solution, (i) each served customer is visited an infinite number of times, and (ii) the time elapsed between two consecutive visits to customer i does not exceed q_i, $i \in N$. Notice that it is not required to visit all customers. We assume that all data are integral.

Clearly, referring back to Charlemagne's challenge, a count is a customer, Charlemagne is the server, and the loyalty periods are represented by the q_is. If the profit of a customer represents the area of the county, Charlemagne's challenge is to maximize the size of his empire without having to fight internal wars.

Motivation. We see the PLPP and the PLP as basic problems with applications in diverse areas. We describe here three different fields where these periodic latency problems occur. Recently, Studer (2008) described "rounding," a management process that can help to improve management and leadership skills. Studer believes that managers should make regular rounds to check on their employees. In that way, managers find out what matters to employees, and potential problems can be dealt with before they occur. This "rounding" model is based on the rounds doctors and nurses make to check on their patients in a hospital. Dimov et al. (2007) explore a method called "minirounds" that appears to improve physician-patient communication and satisfaction at a hospital. Minirounds are defined as follows: "A series of short patient encounters [each lasting about a minute] during which the physician asks patients about any changes in their condition and provide a concise daily update" (Dimov et al. 2007, p. 48). It is clear that efficiently organizing these minirounds is an instance of PLP. Notice that the latter application suggests a specific topology of the customers: Karuno et al. (1997) mention the tree network as being relevant for representing the corridor structure in hospitals and offices. We extensively study this topology in this work.

Another field where periodic latency problems occur is maintenance, more precisely, preventive periodic maintenance. Machines located at given positions (say different plants) need to be inspected regularly. Obviously, when a machine is viewed as a client, and when the periodicity of each machine is given, the PLP arises. Although there is a considerable amount of literature on preventive periodic maintenance (see, e.g., Dekker et al. 1997 for an overview),

many contributions are stochastic, and we are not aware of deterministic situations where distances between machines are taken into account (see Anily et al. 1998 for a related problem).

A third field motivating PLP and PLPP concerns real-time task scheduling. In fact, periodic scheduling problems were already introduced in 1973 by Liu and Layland (1973). They study the problem of scheduling periodic tasks with hard deadlines that coincide with the task periods. At that time, computers were used more and more to monitor and control industrial processes, and the efficient scheduling of these control and monitor functions became important. Since then computers have evolved; more recent papers such as Bar-Noy et al. (2004) and Patil and Garg (2006) study periodic scheduling problems in the field of wireless devices. Here, in order to reduce power consumption, the goal is to find a periodic schedule such that wireless devices only need to be "awake" when they are being served. In this case a client is served regularly after a predefined amount of time (the period of the client), and there is no travel time; each client, however, requires a service time.

Related Problems. Many routing and scheduling problems require a periodic solution. In the periodic TSP (Paletta 2002), a set of customers, each with a certain frequency, is given. Each customer needs to be visited according to its frequency within a given planning period T. A solution then consists of a set of routes, one route for every day in the planning period T. This setting can be generalized to the periodic VRP, where more than one vehicle is available and several routes can be performed each day of the planning horizon T (Mourgaya and Vanderbeck 2006). These problems, however, do not belong to the class of latency problems.

Other applications of periodic scheduling problems can be found in scheduling of robotic cells; see, e.g., Crama et al. (2000); or in digital-signal processing; see Verhaegh et al. (2001). A signal-processing algorithm consists of elementary operations that need to be carried out repeatedly on successive samples of a given digital signal. Korst et al. (1997) study the problem of nonpreemptively scheduling periodic tasks on a minimum number of identical processors. They mention an application where a number of continuous data streams must be read from a minimal number of identical disk units. Any situation where there is a repetitive execution of operations with strict timing requirements is relevant (Verhaegh et al. 2001). In these settings, precedence constraints might also be present.

A nonperiodic latency problem with profits is discussed in Coene and Spieksma (2008). There, a profit p_i is associated with every customer i, and these profits go down linearly with time while the customer is waiting to be served. The goal is to select customers and to find a route for the server visiting these customers such that the collected profit is maximal. A similar problem with time windows is described by Frederickson and Wittman (2007). Each service request is assigned to a time window, and the goal is

to find a tour that visits the maximum number of locations during their time windows. Each request that the repairman completes yields a given profit. These problems, however, are not periodic.

A problem that is probably closest to our setting is the problem described in Campbell and Hardin (2005). In their problem, the number of servers needed to serve all customers is minimized, under the assumption that each customer $i \in N$ needs to be visited *precisely* every q_i time units. They show that a solution exists that is periodic (see §3) with length $T = lcm(q_1, q_2, \ldots, q_n)$.

We give an overview of our results in §2. We elaborate on the issue of periodicity in §3. In §4 we deal with the single-server problem, i.e., PLPP. In §5 we deal with PLP. The results of §§4 and 5 can be generalized to a setting with an arbitrary given number of servers, where the goal is to maximize the number of customers visited by these servers (the MPLPP); this is shown in §6. Finally, §7 discusses possible extensions.

2. The Results

Figure 1 summarizes our results concerning the complexity of the PLPP for different settings of p_i and q_i, in different metric spaces. From Theorem 4 and Corollary 2 it follows that in a general case with arbitrary profits and frequencies (d), the PLPP on the line and the PLPP on the circle are solvable in polynomial time. Hence, the same results hold for the more restricted topologies in (a), (b), and (c). Figure 1(a) shows results for the PLPP with unitary profits and a common frequency Q; this result is due to Theorem 8 and Corollary 3. Figure 1(b) represents the results for the PLPP with unitary profits and arbitrary frequencies; this follows from Theorem 6. Similarly, Figure 1(c) holds for PLPP with arbitrary profits and a common frequency Q, see Theorem 5.
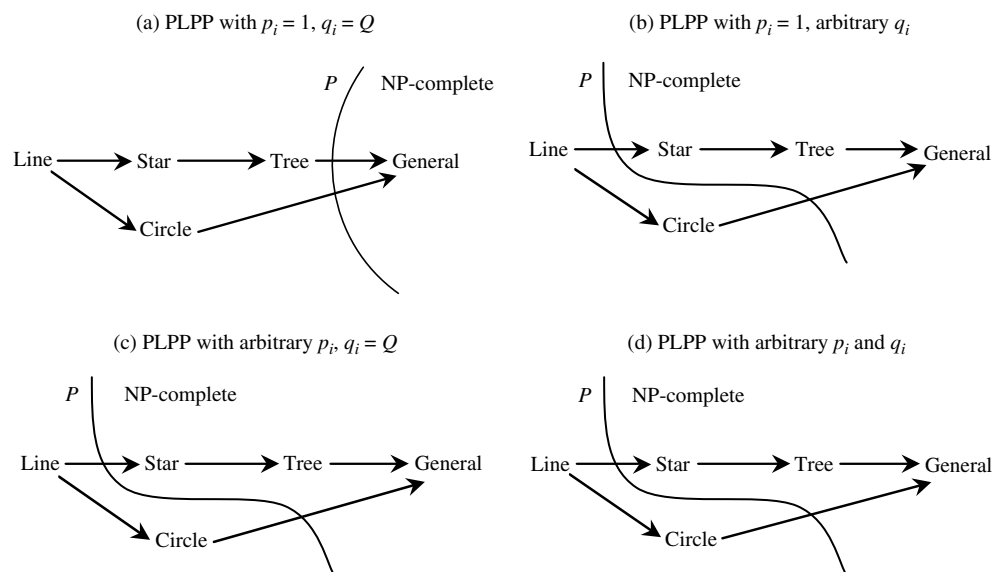
In the PLP, the goal is to minimize the number of servers needed to visit all customers. Profits are not applicable in this case. Results are represented in Figure 2 and follow directly from Theorem 9, Corollary 4, and Theorem 10.

Both problems PLPP and PLP do not have service times for the customers. Although these times may be relevant for certain applications, allowing for customer-dependent service times is a nontrivial extension of our problems. Because, in case of service times, it no longer holds that a customer is always visited when the server is at the customer's location, the problem becomes much more difficult. We will indicate in the different sections what the impact of service times on the problem's complexity is.
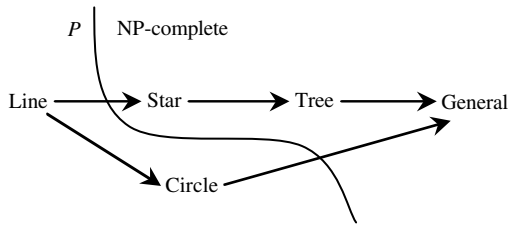
We see as our main contribution a complete classification of both PLPP and PLP for the following topologies: line, circle, star, tree, general; and we consider arbitrary profits p_i versus $p_i = 1$, and arbitrary periodicities q_i versus $q_i = Q$ for all $i \in N$. We achieve this classification by either describing a polynomial-time algorithm or giving an NP-hardness result. Although some of these results are standard, others are not. In particular, the $O(n^2)$ algorithm for PLPP on the line and the $O(n^4)$ algorithm for PLP are not straightforward.

3. Periodicity

In this section, periodicity of solutions is being analyzed. The analysis in this section is restricted to the PLPP. However, everything also holds for the PLP because a solution to the PLP consists of s single-server solutions, where s denotes the number of servers used.

A solution to the PLPP can be represented by an infinite sequence of customers: customer i appearing on the pth position of the sequence indicates that the pth customer visited is customer i. Let L be a sequence representing a

Figure 1. Complexity results PLPP.

(a) PLPP with $p_i = 1$, $q_i = Q$

(b) PLPP with $p_i = 1$, arbitrary q_i

(c) PLPP with arbitrary p_i, $q_i = Q$

(d) PLPP with arbitrary p_i and q_i

Figure 2. Complexity results PLP.

solution to the PLPP and S the subset of customers visited. Notice that for $|S| = 1$, periodicity is trivial.

We call any sequence of customers of finite length a *subsequence*.

DEFINITION 1. A subsequence is called a *k-cycle* if
 • the subsequence starts and ends with some customer $i \in N$ not appearing elsewhere in the subsequence, and
 • each served customer $j \neq i$ appears with a frequency of at least 1, and at most k, in the subsequence.

DEFINITION 2. We say that a solution to PLPP is *periodic* if, when viewed after some position p, the sequence representing this solution is a concatenation of identical subsequences.

Clearly, from the viewpoint of implementation and optimization, periodic solutions are preferable over nonperiodic ones. Fortunately, we lose nothing by restricting the search for a solution to the class of periodic solutions:

THEOREM 1. *If there exists a feasible solution for an instance of PLPP that visits customer-set $S \subseteq N$, then there exists a periodic feasible solution visiting S.*

PROOF. We will show how to modify a given feasible solution into a periodic feasible solution. Let $Q = \max_{i \in S} q_i$. All customers are positioned at distinct locations, so $d_{ij} \geqslant 1$ (indeed, we can replace two customers at the same location by the one with the more stringent q_i). Consider any feasible solution to PLPP, represented by a sequence L. Let us assume that the number of served customers is greater than 1: $|S| \geqslant 2$. The proof relies on the following observation.

OBSERVATION 1. L consists of infinitely many k-cycles, for some k.

To argue that this observation holds, consider any position p in the sequence L and the customer i visited at p. Then, at position $p + 1$, a customer $j \neq i$ is visited and at position $p + 2$, either again customer i is served or a "new" customer l is served. At a certain position $q > p$, $|S| - 1$ different customers will have been served since position p. Thus, one customer, say customer l, will then necessarily be visited at a position $q' > q$ and also at a position $p' < p$; the total number of time units elapsed between these two visits can be no more than Q. It follows that between two consecutive visits to this customer l, all other customers are

visited at least once. Thus, we have found a k-cycle containing position p. Because this holds for any position p, the observation is valid.

Now, because for a specific customer at most n^Q different k-cycles exist, the total number of different k-cycles that can appear in L cannot exceed n^{Q+1}. The observation above then implies that a same k-cycle will appear more than once. By repeating a subsequence that starts with this k-cycle, plus everything that followed this k-cycle up to its next appearance, we modify L into a feasible solution that is periodic. □

4. The Periodic Latency Problem with Profits (PLPP)

In this section, we prove the results summarized in Figure 1. Section 4.1 deals with the PLPP on the line or on a circle; §4.2 deals with the PLPP on a star, §4.3 deals with the PLPP on a tree; and finally, §4.4 deals with the PLPP on a arbitrary topology.

4.1. PLPP on the Line and the Circle

Consider first an instance of the PLPP with arbitrary profits p_i and arbitrary periodicities q_i where all customers are positioned on the line. We assume that $x_1 < x_2 < \cdots < x_n$, and we denote the distance between x_i and x_j as d_{ij} for $i, j \in N$. We first show that for this special topology, we can restrict ourselves to solutions where the server simply oscillates between two customers. In other words, there is a solution that is a 2-cycle (see Definition 1).

THEOREM 2. *If there exists a feasible solution for a set of customers $S \subseteq N$, then there is a 2-cycle serving customer set S, with $|S| \geqslant 2$.*

PROOF. Consider any pair of locations $x_i < x_j$, $i, j \in S$. Because $i, j \in S$, there exist two moments in time $t_i < t_j$ such that (i) the server is at x_i at time t_i, (ii) the server is at x_j at time t_j, and (iii) neither x_i nor x_j is visited at any time t with $t_i < t < t_j$. Because $i \in S$, it must be true that $q_i \geqslant t_j - t_i + d_{ji} \geqslant d_{ij} + d_{ji}$. The first inequality is true because being able to serve both customers i and j implies that the periodicity of customer i cannot be smaller than the time the server needs to travel from x_i to x_j and back (recall that we assume that the server travels at unit speed). Thus, the time needed to travel from x_i to x_j is at least equal to the distance d_{ij}, which gives us the second inequality. Similarly, we can argue that $q_j \geqslant d_{ji} + d_{ij}$. Thus, for all customers i and j in S, it holds that

$$q_i \geqslant d_{ij} + d_{ji} \qquad (1)$$

and

$$q_j \geqslant d_{ji} + d_{ij}. \qquad (2)$$

We now exhibit a 2-cycle that is able to serve the customers in S. Indeed, consider a server that travels from the

leftmost customer in S to the rightmost customer in S, and back, and repeating this pattern. Each customer $i \in S$ is served as long as

$$q_i \geqslant \max_{j \in S}(d_{ij} + d_{ji}). \qquad (3)$$

This, however, is implied by (1) and (2), and hence a 2-cycle serves the customers in S. $\square$

Using Theorem 2, it is not hard to argue that we can answer the question, can we visit all given customers in $O(n)$ time? We refer to this problem as decision-PLPP.

COROLLARY 1. *Decision-PLPP on the line can be answered in $O(n)$.*

PROOF. Given positions $x_1, \ldots, x_n$, we verify whether (3) holds. Given the line topology, the maximum in (3) can only be attained for $j = 1$ or $j = n$. It follows that we need to verify $O(n)$ inequalities. $\square$

Notice that in the presence of service times, Theorem 2 no longer holds. In fact, a straightforward argument shows that the problem is difficult.

THEOREM 3. *Decision-PLPP on the line with service times is NP-complete.*

PROOF. We reduce from the partitioning problem. Consider an instance of partition with a set $X = \{x_0, \ldots, x_n\}$ with associated values $a(x_i)$ such that $\sum a(x_i) = 2k$. Can this set be partitioned into two sets X_1 and X_2 such that $\sum_{x \in X_1} a(x) = \sum_{x \in X_2} a(x) = k$ and each element occurs exactly once?

Then, we build the following instance of PLPP on the line with service times. There is a set of $n + 1$ customers positioned at the origin, service times $s_i := a_i$ and periodicities $q_i := 2k$ for all $i = 1, \ldots, n$. Finally, $q_0 = k$ and $s_0 = 0$. The question is then whether all clients can be served periodically.

It then holds that a solution to PLPP on the line visiting all customers exists if a solution to partition exists, and vice versa. $\square$

THEOREM 4. *PLPP on the line with arbitrary p_i and arbitrary q_i can be solved in $O(n^2)$.*

PROOF. We only need to search for a best 2-cycle (Theorem 2). Clearly, considering each possible combination of leftmost and rightmost customer, and then checking whether all intermediate customers can be served, yields an immediate $O(n^3)$ algorithm. We now proceed to describe an $O(n^2)$ algorithm.

LEMMA 1. *Consider a server traveling on the interval $[x_i, x_i + L]$; this server serves a customer j if and only if*
  (i) $x_i \leqslant x_j$;
  (ii) $x_j \leqslant x_i + L$;
  (iii) $2L + 2x_i - 2x_j \leqslant q_j$;
  (iv) $2x_j - 2x_i \leqslant q_j$.

PROOF. Conditions (i) and (ii) state that point x_j must be contained in the interval $[x_i, x_i + L]$. Conditions (iii) and (iv) follow from the following observation. The time needed for the server to travel from x_j to the rightmost point of the interval and back to x_j (i.e., twice the distance between x_j and $x_i + L$), and the time needed to travel to the leftmost point of the interval and back (i.e., twice the distance between x_i and x_j), respectively, may not be larger than the periodicity q_j. $\square$

According to Lemma 1, a customer j is thus served by a server traveling in $[x_i, x_i + L]$ if and only if $x_i \leqslant x_j$ and $x_i \geqslant x_j - \frac{1}{2}q_i$ and if L lies in the "*activity interval*"

$$A_j := [x_j - x_i; x_j + \tfrac{1}{2}q_j - x_i].$$

The set of served customers $S_i(L)$ is then the following:

$$S_i(L) := \{j: x_i \leqslant x_j, x_i \geqslant x_j - \tfrac{1}{2}q_j, \text{ and } L \in A_j\}.$$

Our algorithm consists of a preprocessing step and a main algorithm computing the best value for L—i.e., L for which $\sum_{j \in S(L)} p_j$ is maximized—and this is done for every i.

(*Preprocessing*) For every customer i, let $l(i) = x_i$ and $r(i) = x_i + \frac{1}{2}q_i$. Sort all values of $l(i)$ and $r(i)$ in a global nondecreasing list T.

(*Main algorithm*) For each customer i, the interval $[x_i, x_i + L]$ maximizing the resulting profit of the served customers can be computed as follows.

(i) For all customers in T, select the customers j not violating conditions $x_i \leqslant x_j$ and $x_i \geqslant x_j - \frac{1}{2}q_j$ and add their $l(j)$ and $r(j)$ values to T'. Notice that T' is sorted.

(ii) The first entry in T' will be $l(i) = x_i$. This corresponds to the choice of $L = 0$, and the corresponding profit P equals p_i. Set $P_{\max} = p_i$.

(iii) Work through T' and determine the profits for the corresponding values of L.

   (a) If the next element in T' is a lower bound $l(j)$ of an activity interval A_j, set $P := P + p_j$.

   (b) If the next element in T' is an upper bound $r(j)$ of an activity interval A_j, set $P := P - p_j$.

   (c) If $P > P_{\max}$, set $P_{\max} := P$.

Select the highest $P_{\max}$ over all customers i.

The preprocessing step takes $O(n \log n)$, the main algorithm has $O(n)$ iterations, and each iteration takes $O(n)$, yielding a total complexity of $O(n^2)$. $\square$

Notice that Theorem 4 is also valid for a server whose speed differs between traveling to the right and traveling to the left.

Consider now an instance of PLPP where all customers are positioned on a circle; we can simply extend Theorem 4 to this case:

COROLLARY 2. *PLPP on the circle with arbitrary p_i and arbitrary q_i can be solved in $O(n^2)$.*

PROOF. When considering the circle, two different situations can occur. Either the server travels through the whole circle and the solution to PLPP is a 1-cycle, or the server covers only part of the cycle, which is then equivalent to serving customers on the line, and the solution to PLPP is a 2-cycle (Theorem 2). Thus, in order to find the optimal solution, we have to compare the 1-cycle with the best 2-cycle. The best 2-cycle can be found using the algorithm for the line. However, on the circle it holds that the leftmost customer served is also the leftmost position (i.e., position 0) on the corresponding line. It follows that in the main algorithm, for each i, the elements in the sorted list need to be shifted such that i is the leftmost point on the line. This can be done in linear time. The main algorithm then can be executed as before, yielding an $O(n^2)$ algorithm. □

4.2. PLPP on a Star

Let us consider a star graph (called a star). The customers are positioned in the end nodes and, in addition to a profit p_i and a periodicity q_i, there is a distance d_i given that denotes the distance between the position of customer i and the center of the star, $i \in N$.

It is easy to see that the PLPP where all $p_i = 1$ and $q_i = Q$ for all $i \in N$ is solvable in polynomial time. Indeed, by selecting customers with the smallest d_i until the tour length exceeds Q, an optimal solution is found. In fact, a more general result is shown in §4.3. Notice that in this case adding service times does not alter complexity; they can just be added to the d_i. In §5 we show that adding a second server already makes the problem NP-hard.

When profits p_i are arbitrary, and $q_i = Q$, PLPP on a star can be shown to be equivalent to the knapsack problem.

THEOREM 5. *PLPP on a star with arbitrary profits p_i, and with all $q_i = Q$, is NP-hard.*

PROOF. We reduce from knapsack, which is a weakly NP-hard problem (Garey and Johnson 1979). Consider an instance of the knapsack problem with n items where each item i has a certain value w_i and a requirement a_i. The knapsack has size B, and the question is whether we can fit a subset of the items with a total value of W in this knapsack.

Now construct an instance of PLPP on a star as follows. There are n customers. Each customer $i \in N$ has an associated weight $p_i := w_i$, a periodicity $Q := B$, and a distance to the center of the star $d_i := \frac{1}{2} a_i$. Does there exist a subset of the customers with total profit equal to W such that each customer is visited at least once within each time period Q?

In case the instance of knapsack is a yes-instance, we can copy that solution to the instance of PLPP: selected items correspond to selected customers. Starting in the center, it is clear that visiting the selected customers in any order and repeating that pattern gives a feasible solution. When the instance of PLPP admits a yes, there is a set of customers that we can apparently serve. A served customer i implies a travel time of $2d_i = a_i$. Feasibility of the PLPP

instance ensures that the corresponding set of items fits in the knapsack. □

When we consider PLPP on a star with $p_i = 1$, and arbitrary periodicities q_i, the problem becomes NP-hard.

THEOREM 6. *PLPP on a star with all $p_i = 1$, and arbitrary periodicities q_i, is NP-hard.*

PROOF. We show that 3-partition can be reduced to PLPP on a star with $p_i = 1$ and arbitrary periodicities q_i. This proof is based on the proof of Korst et al. (1997) for NP-hardness of scheduling periodic tasks with slack (PSSP).

An instance of 3-partition consists of a set A with $3m$ items and a positive integer B representing the size of the m bins. Each item $a_i \in A$ has an associated size z_i for which it holds that $B/4 < z_i < B/2$ and $\sum_{a_i \in A} z_i = mB$. Can A be packed into m bins, each containing three items?

We construct an instance of our special case of PLPP on a star such that the items can be packed in m bins if the customers in the corresponding PLPP, each with their respective periodicity, can be served by a single server. We are given a set of $n := 3m + 1$ customers, each with periodicity $q_i := m(B + 2)$ and distance $d_i := \frac{1}{2} z_i$ to the center of the star, for $i = 1, \ldots, 3m$. Further, customer $3m + 1$ has $q_{3m+1} := B + 2$ and $d_{3m+1} := 1$. The question is whether there is a solution serving all $3m + 1$ customers.

If 3-partition has a solution, it is clear how to copy that solution to the PLPP instance, and get a solution serving all $3m + 1$ customers. If the PLPP instance has a solution in which all $3m + 1$ customers are served, then between two consecutive visits to customer $3m + 1$ there are exactly B time units left that can be used to visit other customers. Each of the $3m$ customers left must be visited at least once in time period $m(B + 2)$. In that time period there are m available time slots of B time units. Thus, the $3m$ customers can be assigned to the m different time slots if and only if the corresponding items can be packed in m bins. □

The following holds when periods are arbitrary and distances unitary:

THEOREM 7. *PLPP on a star with unitary distances, unitary profits, and arbitrary periodicities is solvable in polynomial time.*

PROOF. Consider an optimal solution visiting a maximal subset S of customers. Set T is the set of unvisited customers, $N = S + T$. It holds that for any two customers $i \in T$ and $j \in S$ with $q_i > q_j$, i can be moved to subset T and j to subset S. A new optimal solution is obtained. Thus, an optimal solution can easily be found by selecting customers in decreasing order of q until a feasible solution is no longer obtained. □

4.3. PLPP on a Tree

We argue here that PLPP on a tree with $p_i = 1$ and $q_i = Q$ is nothing but an orienteering problem (OP). Coene et al. (2008) describe how to modify an algorithm from Johnson

and Niemi (1983) in order to solve the orienteering problem restricted to a tree. An instance of OP on a tree consists of a set of vertices N where each vertex $i \in N$ has an associated profit p_i and each edge between two vertices i and j in the tree has a cost c_{ij}. A maximum on the cost C is given. The goal is to find a route visiting a subset of the vertices with cost no more than C and collecting a maximal amount of profit. An instance of PLPP with $p_i = 1$ and $q_i = Q$ is then an orienteering problem with $p_i = 1$ ($\forall i \in N$), $c_{ij} := d_{ij}$, and with $C := Q$. The algorithm in Coene et al. (2008) solves OP on a tree in $O(nP_{tot})$, with P_{tot} the sum of all given profits; because $p_i = 1$, $\forall i \in N$, total running time for PLPP is only $O(n^2)$.

COROLLARY 3. *PLPP on a tree with $p_i = 1$ and $q_i = Q$ is solvable in $O(n^2)$.*

4.4. PLPP on an Arbitrary Topology

THEOREM 8. *PLPP with $p_i = 1$, and $q_i = Q$, is NP-hard.*

PROOF. We prove NP-completeness of this variant of PLPP by a reduction from the Hamiltonian cycle. An instance of the Hamiltonian cycle problem is specified as follows: given a graph $G = (V, E)$, does there exist a Hamiltonian cycle in G?

Now consider the following instance of PLPP. A node in V corresponds to a customer, and we set $n := |V|$. For each pair of customers i, j that is connected via an edge in E, we set $d_{ij} := 1$, else we set $d_{ij} := 2$. The periodicity is $q_i := |V|$ for each $i \in N$. Now, does there exist a solution to PLPP with value n?

If a Hamiltonian cycle exists in G, there exists a tour in the PLPP instance with length n visiting all the locations. This solution is periodic, and, vice versa, if a solution serving all customers in the PLPP instance exists, G must contain a Hamiltonian cycle. □

Notice that, by performing a similar reduction from the Euclidean TSP, it follows that in case of Euclidean distances the problem also is NP-hard.

5. The Periodic Latency Problem (PLP)

In this section we study the PLP and we prove the results summarized by Figure 2. In the PLP our goal is to minimize the number of servers needed to visit all customers. We assume that every customer must be assigned to and served by a single server; this is a crucial assumption, as can be seen from the example in Figure 3. Observe that when each customer must be served by one server, three servers are needed to serve all three customers in the example. If, however, that assumption is dropped, the three customers from the example can be served using only two servers, each server alternately serving the middle customer.

Determining for an arbitrary topology whether a single server suffices to serve all customers is NP-hard; this follows directly from Theorem 8. In this section we show that

Figure 3. PLP.
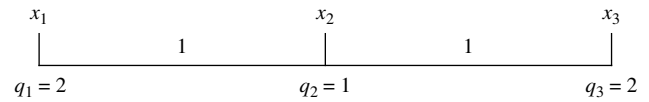
a dynamic programming approach solves the case of the line in polynomial time, whereas PLP on a star remains NP-hard.

5.1. PLP on the Line

We develop a dynamic programming algorithm for the PLP on the line. Given are n customers with their positions $x_1 < x_2 < \cdots < x_n$ on the line, and a set of n identical servers. The goal is to minimize the number of servers necessary to visit all customers periodically, i.e., without violating any q_i.

There exists an optimal schedule to an instance of PLP on the line of the following shape:

• Every server s commutes between the left and the right endpoint of its interval I_s.

• The intervals I_s and I_t of any two different servers s and t are either disjoint, or one of them contains the other one.

• The left endpoint of interval I_s is only assigned to server s.

We now describe an informal argument establishing the second property described above. Suppose that every optimal solution contains two intervals I_s and I_t that are neither disjoint nor contained. Then, a client i at position x_i exists, contained in both intervals, which is served by one of the servers, say server t, and cannot be served by server s. Define $x_{s,l}$ and $x_{s,r}$ as the leftmost and rightmost point, respectively, of I_s and analogue $x_{t,l}$ and $x_{t,r}$ for I_t. We can assume then, without loss of generality, that $x_{s,l} \leqslant x_{t,l} \leqslant x_i \leqslant x_{s,r} \leqslant x_{t,r}$. Because x_i is served by server t and cannot be served by s, it must hold that $d_{x_{s,l}, x_i} \geqslant d_{x_{t,r}, x_i}$. As a result, all customers positioned on the interval $[x_i, x_{s,r}]$ and served by server s can also be served by server t, yielding two disjoint intervals I_s and I_t.

We say that in some schedule a server *covers* customer i if the time elapsed between two consecutive visits of this server to the customers is at most q_i. The following lemma clearly holds:

LEMMA 2. *Let $a \leqslant b \leqslant c \leqslant d$ be four request points. Assume that server s commutes on the outer interval $[a, d]$, and that server t commutes on the inner interval $[b, c]$. Then every point in interval $[b, c]$ that is covered by the outer server s will also be covered by the inner server t.*

Indeed, the frequency with which any point x_i within $[b, c]$ is visited by the server commuting in $[b, c]$ is higher than for the server commuting in $[a, d]$.

A Dynamic Programming Algorithm. Consider an arbitrary instance defined by customer's positions $x_1 < x_2 < \cdots < x_n$ and their periodicities q_i. Add to this

instance two clients x_0 and x_{n+1} with $x_0 < x_1$ and $x_n < x_{n+1}$ with $q_0 = q_{n+1} = LARGE$. There is one server that serves x_0 as well as x_{n+1} and no other clients. We now define state $F[i; j]$ as follows.

DEFINITION 3. Let i and j be integers with $0 \leqslant i, j \leqslant n+1$.

Then state $F[i; j]$ denotes the smallest number of servers that can serve all customers $k = i, i+1, \ldots, j$, with customers i and j being served by the same server; in case customers i and j cannot be served by a single server, we set $F[i; j] = \infty$.

Notice that in this definition we explicitly allow the situation $j < i$, which yields an empty interval and an empty set of requests. We will now compute all the values $F[i; j]$ step by step and in increasing order of $j - i$ (the number of customers minus one in the underlying interval).

All cases with $j < i$ have empty intervals, and hence need $F[i; j] = 0$ servers. In the remaining cases we have $j \geqslant i$. For a customer i' positioned in $[x_i, x_j]$, it holds that either (i) i' can be served by server s serving customers i and j, or (ii) i' cannot be served by s. Let S be the number of servers necessary to visit the customers within $[x_i, x_j]$ not visited by s, then $F[i; j] = 1 + S$.

We compute S, and thus $F[i; j]$, by building a graph called G_{ij}. This graph has a node v_k for each customer k, $i < k < j$, that cannot be served by the server traveling between i and j. Let K be the set containing all these customers. There is also a terminal node t in G_{ij}. If K is empty, then $F[i; j] = 1$ and $S = 0$. Otherwise, let $succ(k)$ be the smallest b in K such that $b > k$; if $k = \max\{K\}$, $succ(k) = t$. There is an arc in G_{ij} between every pair of nodes v_r and $v_{succ(s)}$ with $s \geqslant r + 1$. If r and s cannot be served by the same server, then arc $(v_r, v_{succ(s)})$ has weight ∞; if r and s can be served by the same server, then arc $(v_r, v_{succ(s)})$ has weight $F[r; s]$.

Let $SP(G_{ij})$ denote the value of a shortest path in G_{ij} between its first node, being the node corresponding the smallest customer in K, and t. We claim that $F[i; j] = 1 + SP(G_{ij})$. Indeed, the shortest-path value of G_{ij} equals the minimum number of servers needed to serve the customers within $[x_i, x_j]$ that cannot be served by the server traveling between i and j. Because we calculate the values for $F[i; j]$ in increasing order of $(j - i)$, the arc lengths $F[r; s]$ with $i < r < s < j$ are calculated in previous iterations. Notice that G_{ij} is a directed, acyclic graph, for which it is easy to compute the shortest path in $O(n^2)$ time.

The optimal solution is then given by $F[0; n + 1] - 1$. We have $O(n^2)$ states, each requiring $O(n^2)$ time, yielding total complexity equal to $O(n^4)$.

THEOREM 9. *PLP on the line can be solved in $O(n^4)$.*

5.2. PLP on a Circle

The algorithm for the problem on the line can be extended to the circle, adding a factor n to the running time of the line algorithm.

COROLLARY 4. *PLP on the circle can be solved in $O(n^5)$.*

PROOF. Previously, we discussed that there always exists an optimal solution where either (i) intervals served by two different servers do not overlap at all, or (ii) an interval of one server is completely contained by an interval served by an other server. It is not hard to see that the same holds for PLP on a circle.

A solution to PLP on a circle then either consists of a 1-cycle, possibly containing several 2-cycles; or consists of only 2-cycles. When removing all customers served by the 1-cycle in the first case, we are left with solving an instance of the second case. As intervals have no overlap unless they are completely contained, the problem reduces to solving n instances on the line, with for each instance a different leftmost node on the line. Solving all these line instances and selecting the best from these n solutions yields an optimal solution to the PLP on the circle. Thus, the optimal solution to an instance of PLP on the circle can be found in $O(n^5)$. □

5.3. PLP on a Tree

We prove that the PLP on a star with $q_i = Q$ is NP-hard; NP-hardness of the PLP on a tree with arbitrary q_i follows immediately from this result. In §4.2 it was shown that the periodic latency problem on a star graph with all customers having the same profit and requiring equal frequency is easy to solve; adding a second server, though, makes the problem much harder to solve.

THEOREM 10. *PLP on star is NP-hard, even if $q_i = Q$, for all $i \in N$.*

PROOF. We reduce from partition. An instance of the partition problem has a set $X = \{x_1, \ldots, x_n\}$ with $\sum_{i=1}^{n} s(x_i) = 2k$, where $s(x_i)$ denotes the size of x_i. Can the set X be partitioned into two sets X_1 and X_2 such that $\sum_{x \in X_1} s(x) = \sum_{x \in X_2} s(x) = k$ and each element occurs exactly once?

An instance of PLP on a star is constructed as follows. There is a star with n spokes, and a customer located at each spoke with distance $\frac{1}{2}x_i$ from the center of the spoke, for $i = 1, \ldots, n$. Each customer (spoke) has a periodicity $Q := k$, meaning that the maximal time that can pass between two consecutive visits to a customer equals k. There are two servers, traveling at unit speed, positioned in the center of the graph. Now, does there exist a route for each of the servers such that all customers can be served periodically?

If a solution to partition exists, the set of customers in X_1 can be assigned to one server and the customers in X_2 to the other server, and each server can visit these customers in Q time units. If a solution to PLP on the star exists, each customer is visited once within every Q time units. Because total travel time to visit all customers equals $2Q$, it must be the case that every server travels exactly Q time units. Customers visited by server 1 can be assigned to one

set and customers visited by server 2 to the second set, and a solution for partition is obtained. □

As a result, PLP on a tree is NP-hard.

6. The Case of Multiple Servers: MPLPP

The complexity results from the previous section can be extended to a periodic latency problem with profits and multiple servers, denoted by MPLPP. As before, we are given a set of n customers with their positions $x_1, x_2, \ldots, x_n$; and each customer i has an associated profit p_i and periodicity q_i. Further, S identical servers are given, with $S < n$. (The case $S = 1$ is dealt with in §4). The goal is then to find a repeatable route for each server collecting a maximal amount of profit without violating the q_is. We show that when all costumers are positioned on the line/circle, an alternative dynamic programming algorithm can be applied to solve the MPLPP in polynomial time. On a tree (and hence on an arbitrary graph), the problem is NP-hard. One can easily check that Lemma 2 still holds for MPLPP on the line. Then a state in the dynamic programming algorithm is defined as follows.

DEFINITION 4. Let i and j be integers with $1 \leqslant i, j \leqslant n$. Let i' and j' be integers such that either (i) $1 \leqslant i' \leqslant j' \leqslant n$ and $x_1 \leqslant x_{i'} \leqslant x_{j'} \leqslant x_n$ where x_i and x_j are contained in the interval $[x_{i'}, x_{j'}]$, or such that (ii) $i' = j' = 0$.

Then $P[i; j; i'; j'; s]$ denotes the maximum profit that can be collected serving customers in the interval $[x_i, x_j]$ using $s \leqslant S$ servers, excluding the customers that are already covered by an external server that commutes on the interval $[x_{i'}, x_{j'}]$. (In the case $i' = j' = 0$, there is no such external server.)

Now, we compute all the values of $P[i; j; i'; j'; s]$. All cases with $j < i$ have empty intervals, and hence yield a profit $P[i; j; i'; j'; s] = 0$. When $j \geqslant i$ we can distinct several cases. If the leftmost point i is covered by the external server or is not served at all, it holds that $P[i; j; i'; j'; s] = P[i + 1; j; i'; j'; s]$. Otherwise, i is served by a server from s, say s_1, traveling on the interval $[x_i, x_l]$ with $i \leqslant l \leqslant j$. Total profit for this state then consists of three parts: (i) $p(l) = \sum_{i \in S(s_1)} p_i$, the profit of the set of customers $S(s_1)$ covered by s_1; (ii) $\alpha(l, s') = P[i + 1, l, i, l, s']$, the profit of the servers traveling within the interval covered by s_1; and (iii) $\beta(l, s - s') = P[l + 1, j, i', j', s - s' - 1]$, the profit that can be collected in the remaining interval by the remaining servers. The optimal value is then:

$$P[i; j; i'; j'; s] = \max_{l, s'} \{p(l) + \alpha(l, s') + \beta(l, s - s') \mid i \leqslant l \leqslant j,$$

$$0 \leqslant s' \leqslant s\}.$$

Therefore, $P[i; j; i'; j'; s] = \max[P[i + 1; j; i'; j'; s],$ $\max_{l, s'}\{p(l) + \alpha(l, s') + \beta(l, s - s') \mid i \leqslant l \leqslant j, 0 \leqslant s' \leqslant s\}]$. The state $P[i; j; 0; 0; S]$ yielding maximal profit gives the optimal solution. There are $O(n^4 S)$ states to be computed, and each state requires $O(n^2 S)$ time. Overall running time is then $O(n^6 S^2)$.

THEOREM 11. *MPLPP on the line can be solved in* $O(n^6 S^2)$.

As in the previous section, the algorithm for MPLPP on the line can be extended by adding a factor n to the running time.

COROLLARY 5. *MPLPP on the circle can be solved in* $O(n^7 S^2)$.

NP-hardness of MPLPP on a tree follows from the fact that MPLPP on a star is *NP*-hard.

THEOREM 12. *MPLPP on a star is NP-hard, even if all* $q_i = Q$.

PROOF. It is easy to see that the proof of Theorem 10 is directly applicable to this problem. □

It follows that the problem is *NP*-hard on arbitrary graphs.

7. Conclusion

We were able to settle complexity of a number of variants of the PLPP and the PLP. Our results still hold when customers are weighted, because a customer will always be served when a server passes by. However, some interesting questions remain. For instance, how can we deal with more realistic instances including service times? We have shown that they make the problems much harder. Indeed, when service times are added for the customers, Theorem 2, which is crucial for the dynamic programming algorithm, no longer holds. This is even true when service times are constant over all customers. Deciding whether to serve a customer or not can have a large impact on the feasibility to serve other customers in the same interval. Further, it is also not clear what happens when servers have restricted capacity. Also, what if servers are not identical, meaning that they travel at different speeds or with different operating costs?

Epilogue

In the Treaty of Verdun of 843 the three grandsons of Charlemagne divided the empire into three kingdoms. Hence, 30 years after Charlemagne's death, one needed at least three servers to guarantee the loyalty of all the counts . . . .

Acknowledgments

References

Anily, S., C. A. Glass, R. Hassin. 1998. The scheduling of maintenance service. *Discrete Appl. Math.* **82**(1–3) 27–42.

Bar-Noy, A., V. Dreizin, B. Patt-Shamir. 2004. Efficient algorithms for periodic scheduling. *Comput. Networks* **45**(2) 155–173.

Campbell, A. M., J. R. Hardin. 2005. Vehicle minimization for periodic deliveries. *Eur. J. Oper. Res.* **165**(3) 668–684.

Coene, S., F. C. R. Spieksma. 2008. Profit-based latency problems on the line. *Oper. Res. Lett.* **36**(3) 333–337.

Coene, S., C. Filippi, F. C. R. Spieksma, E. Stevanato. 2008. The traveling salesman problem on trees: Balancing profits and costs. Submitted.

Crama, Y., V. Kats, J. van de Klundert, E. Levner. 2000. Cyclic scheduling in robotic flowshops. *Ann. Oper. Res.* **96**(1–4) 97–124.

Dekker, R., F. A. van der Duyn Schouten, R. E. Wildeman. 1997. A review of multi-component maintenance models with economic dependence. *Math. Methods Oper. Res.* **45**(3) 411–435.

de Paepe, W. E., J. K. Lenstra, J. Sgall, R. A. Sitters, L. Stougie. 2004. Computer-aided compexity classification of dial-a-ride problems. *INFORMS J. Comput.* **16**(2) 120–132.

Dimov, V., A. Kumar, R. Hebbar, W. Fares, P. Sharma. 2007. Mini-rounds improve physician-patient communication and satisfaction (abstract). *J. Hospital Medicine, SHM Annual Meeting* **2**(S2) 48.

Frederickson, G. N., B. Wittman. 2007. Approximation algorithms for the traveling repairman and speeding deliveryman problems with unit-time windows. *APPROX RANDOM 2007, Lecture Notes Comput. Sci.* (*LNCS*) **4627** 119–133.

Garey, M. R., D. S. Johnson. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* Freeman, San Francisco.

Johnson, D. S., K. A. Niemi. 1983. On knapsacks, partitions, and a new dynamic programming technique for trees. *Math. Oper. Res.* **8**(1) 1–14.

Karuno, Y., H. Nagamochi, T. Ibaraki. 1997. Vehicle scheduling on a tree with release and handling times. *Ann. Oper. Res.* **69**(1) 193–207.

Korst, J., E. Aarts, J. K. Lenstra. 1997. Scheduling periodic tasks with slack. *INFORMS J. Comput.* **9**(4) 351–362.

Liu, C. L., J. W. Layland. 1973. Scheduling algorithms for multiprogramming in a hard-real-time environment. *J. Assoc. Comput. Machinery* **20**(1) 46–61.

Mourgaya, M., F. Vanderbeck. 2006. Problème de tournées de véhicules multipériodiques: Classification et heuristique pour la planification tactique. *RAIRO Oper. Res.* **40**(2) 169–194.

Paletta, G. 2002. The period traveling salesman problem: A new heuristic algorithm. *Comput. Oper. Res.* **29**(10) 1343–1352.

Patil, S., V. K. Garg. 2006. Adaptive general perfectly periodic scheduling. *Inform. Processing Lett.* **98**(3) 107–114.

Studer, Q. 2008. *Results That Last: Hardwiring Behaviors That Will Take Your Company to the Top.* John Wiley & Sons, Inc., Hoboken, NJ.

Verhaegh, W. F. J., E. H. L. Aarts, P. C. N. van Gorp, P. E. R. Lippens. 2001. A two-stage solution approach to multidimensional periodic scheduling. *IEEE Trans. Comput.-Aided Design Integrated Circuits Systems* **20**(10) 1185–1199.