Available online at www.sciencedirect.com
ScienceDirect
Operations Research Letters 36 (2008) 333–337
Operations Research Letters
www.elsevier.com/locate/orl

Profit-based latency problems on the line

Sofie Coene*, Frits C.R. Spieksma

Katholieke Universiteit Leuven, Operations Research Group, Naamsestraat 69, B-3000 Leuven, Belgium

Received 3 September 2007; accepted 21 November 2007
Available online 28 November 2007

Abstract

We consider a latency problem with a profit p_i for each client. When serving a client, a revenue of $p_i - t$ is collected. The goal is to find routes for the servers such that total collected revenue is maximized. We study the complexity of different variants of this problem on the line.
© 2007 Elsevier B.V. All rights reserved.

Keywords: Minimum latency; Traveling repairman; Dynamic programming; Complexity

1. Introduction

Consider the following problem. Given are a set of n clients located in some metric space and profits p_i associated with each client i, $1 \leq i \leq n$. In addition, a single server is given, positioned at the origin at time $t = 0$. The server travels at unit speed. If the server serves client i at time t, the revenue collected by the server equals $p_i - t$. The goal is to select clients and to find a route for the server serving the selected clients, such that total collected revenue is maximal. We refer to this problem as the traveling repairman problem with profits, or TRPP for short. In this paper we restrict ourselves to the line as a metric space. Notice that in the TRPP: (i) not every client needs to be served, and (ii) the revenue collected at a client depends on the time needed to reach that client.

In this paper we show:

– how a dynamic program solves the TRPP on the line in polynomial time, thereby generalizing a classical result from Afrati et al. [1],
– how this result can be generalized to the problem with multiple identical servers (referred to as MTRPP on the line),
– that the problem with multiple non-identical servers and release dates for each client, is NP-hard.

In the proof of the latter result we settle the complexity of an open problem mentioned in de Paepe et al. [4].

* Corresponding author.
E-mail address: sofie.coene@econ.kuleuven.be (S. Coene).

The paper is organized as follows. In Section 2 we describe the literature for the traveling repairman problem and motivate the TRPP on the line. In Section 3 we describe the dynamic programming algorithm. We settle the complexity of different variants of MTRPP in Section 4. In Section 5 we state some open problems.

2. Literature and motivation

The TRPP is a generalization of the well-known traveling repairman problem (TRP), also known as the minimum latency problem (MLT). In this problem, no profits are given and the goal is to serve all clients with minimal total latency. Afrati et al. [1] give an $O(n^2)$ dynamic program for the TRP on the line which was later improved to $O(n)$ by García et al. [7]. In [12] it is proven that the TRP on the line with release dates is (weakly) NP-hard. Minieka [9] shows that the problem on a tree network is polynomial for trees with unit weights and develops a polynomial time algorithm for the problem on weighted trees when the number of leaves is bounded. The TRP on weighted trees is proven to be strongly NP-hard by Sitters [11]. The problem on the line with multiple identical servers is solvable in $O(n^4)$, see Wu [13] and Averbakh and Berman [2]. We refer to Wu [14] et al. and the references contained therein for papers dealing with exact algorithms for the TRP (i.e. the problem with an arbitrary metric space).

In de Paepe et al. [4] a framework is described dealing with the computational complexity of dial-a-ride problems (which include latency problems, and in particular latency problems on the line). However, as far as we are aware there is no work on latency problems with profits. This is in contrast with the situation for the traveling salesman problem (TSP), see Feillet et al. [5] for a survey on the TSP with profits.

2.1. Motivation

As indicated above, a variety of latency problems arise in many different settings. However, a common characteristic is the focus on minimizing total waiting time of the clients. Here we consider a profit-oriented objective. In situations where a server receives some revenue by performing a service, this objective can be more appropriate. Of course, one needs to balance in some way the waiting times and the profits. This can be done in various ways (bounding the total waiting time by a constant from above, bounding the total revenue realized by a constant factor from below). Here, however, we propose to combine profit and latency in a single objective (as described in the introduction). By doing so, it is clear that a latency problem with a profit objective is at least as hard as the corresponding "ordinary" latency problem.

We restrict the analysis in this paper to a linear profit function (i.e. $p_i - t$) in the objective function. We use this particular function because it arises in the pricing problem of an integer programming formulation modeling the latency problem with multiple servers. Indeed, consider a situation with K non-identical servers (meaning that their speed may differ) whose job is to service a set of n clients on the line. We now describe a set-partitioning formulation for this problem. We define c_{rk} as the total latency of a feasible route r ($r = 1, \ldots, R$; with R the number of feasible routes) served by server k ($k = 1, \ldots, K$). Then, variable x_{rk} is equal to 1 if route r is served by server k and 0 otherwise.

$$\text{Minimize} \quad \sum_{r=1}^{R}\sum_{k=1}^{K} C_{rk}x_{rk} \tag{1}$$

$$\text{subject to} \quad \sum_{r:i\in r}\sum_{k=1}^{K} x_{rk} = 1 \quad \text{for } i = 1, \ldots, n; \tag{2}$$

$$\sum_{r=1}^{R} x_{rk} \le 1 \quad \text{for } k = 1, \ldots, K; \tag{3}$$

$$x_{rk} \in \{0, 1\}.$$

The objective is to minimize total latency (1), there is a constraint for each client stating that it must be served exactly once (2) (see also [6]), and every server can be used at most once (3). When solving the linear programming relaxation of this integer program, the dual variables u_i ($i = 1, \ldots, n$) corresponding to constraints (2) act as the profits in the resulting pricing problem. Indeed, verifying for a fixed server k whether a violated dual constraint exists, amounts to minimizing $C_{rk} - \sum_{i:i\in r} u_i$. This corresponds to the objective function in the TRPP. In this paper we consider the line-metric. Practical examples where this metric is relevant are "shoreline"-problems [10]. These problems arise when scheduling and routing cargo-ships to visit a number of ports which are usually located along a shoreline. Friese and Rambau [6] describe a setting with multiple elevators who need to serve a set of requests.

Summarizing, the TRPP is interesting because: (i) it is a first attempt to combine a latency objective with profits, (ii) latency problems on the line are relevant and their complexity is often unresolved [4], (iii) the TRPP occurs as the pricing problem of an integer programming formulation modeling the latency problem with multiple servers, as is explained above.

3. A polynomial algorithm for the TRPP

We represent an instance of the TRPP on the line as depicted in Fig. 1. The server starts in the origin $x_0 = y_0 = 0$. In this section, we refer to the clients left of the origin as $x_1, x_2, \ldots, x_r$, and to the clients right of the origin as $y_1, y_2, \ldots, y_q$. To each client a profit p_{x_i} resp. p_{y_j} (with $i = 0, \ldots, r$; $j = 0, \ldots, q$) is associated. Notice that we sometimes identify a client with its position on the line. We make a distinction between "served" clients and "visited" clients. A client is served when a server has performed a service at that client and has collected profit there; when the server passes a client without performing the service, this client has been visited but not served. The goal is to select clients and to find a route for the server such that total profits of the clients served minus the latency of the corresponding route is maximal. Clearly, the TRPP on the line generalizes the traveling repairman problem on the line: in case each of the profits is huge, it is optimal to serve all clients, and the problem reduces to finding a route with minimal latency. In general however, not every client needs to be served in the TRPP; observe however that, for those clients that are served, it is optimal to visit the client the first time the server passes by. The optimal route thus has a spiral shape as illustrated in Fig. 1 (see [4]).

Given an instance of TRPP it is not yet clear which clients, and in particular how many clients, need to be selected. We deal with this issue by proposing a procedure that keeps track of the number of clients to be visited.

We now define the ingredients of our dynamic program (DP). A state in our DP is denoted by $[x_i, y_j, l]$ which corresponds to the situation where the server is positioned in x_i (as its leftmost visited client), where the rightmost visited client is y_j, and where l clients are to be served outside that interval ($0 \le i \le r$, $0 \le j \le q$, $1 \le l \le r + q$). Similarly, in the state $[y_j, x_i, l]$ the server is at position y_j (as its rightmost visited client), the leftmost visited client is x_i, and l clients will be served outside this interval.

Definition 1. $\forall i, j, l$: $P[x_i, y_j, l]$ equals the maximal value of the difference between

(i) the profits of the clients served in $[x_i, y_j]$, and
(ii) the latency costs incurred for these clients served in $[x_i, y_j]$ taking into account that l clients will be served outside this interval.

We refer to $P[x_i, y_j, l]$ as the revenue.

Notice that the route followed by the server after having reached state $[x_i, y_j, l]$ does not depend on the route followed

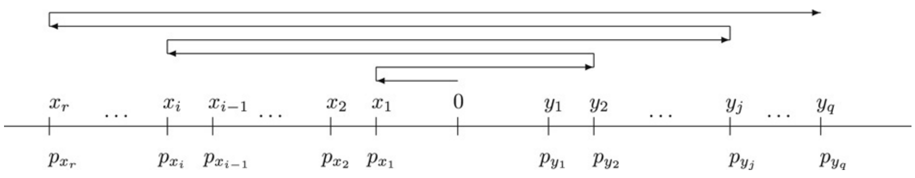

Fig. 1. The TRPP on the line.

within $[x_i, y_j, l]$. Thus, only the set of clients yielding a maximal value for $P[x_i, y_j, l]$ will be selected in an optimal solution. This observation is instrumental for the correctness of our DP algorithm, that we will now describe. First, we define $t[x_i, y_j]$ as the distance between clients x_i and y_j (for all i, j). In an initialization step we set the revenue in all states equal to $-\infty$; then we compute the revenue in a state as follows.

For $l = 1, \ldots, r + q$:

$P[x_0, y_0, l] = P[y_0, x_0, l] = 0.$

For $i = 0, \ldots, r;\ j = 0, \ldots, q;\ l = 1, \ldots, r + q$:

$$\begin{aligned} P[x_i, y_j, l] = \max\{&P[x_{i-1}, y_j, l+1] + p_{x_i} \\ &- (l+1) * t[x_{i-1}, x_i],\ P[x_{i-1}, y_j, l] \\ &- (l) * t[x_{i-1}, x_i],\ P[y_j, x_{i-1}, l+1] \\ &+ p_{x_i} - (l+1) * t[y_j, x_i], \\ &P[y_j, x_{i-1}, l] - (l) * t[y_j, x_i]\} \end{aligned} \tag{4}$$

$$\begin{aligned} P[y_j, x_i, l] = \max\{&P[y_{j-1}, x_i, l+1] + p_{y_j} \\ &- (l+1) * t[y_{j-1}, y_j], \\ &P[y_{j-1}, x_i, l] - (l) * t[y_{j-1}, y_j], \\ &P[x_i, y_{j-1}, l+1] + p_{y_j} \\ &- (l+1) * t[x_i, y_j], \\ &P[x_i, y_{j-1}, l] - (l) * t[x_i, y_j]\}. \end{aligned} \tag{5}$$

Observe that $P[x_0, y_j, l] = P[y_0, x_i, l] = -\infty$ for $i > 0$ and $j > 0$. Then the total revenue is:

$$\text{Total Revenue} = \max\{\max_{i,j}\{P[x_i, y_j, 0]\}, 0\}.$$

Theorem 1. *Algorithm DP is correct.*

Proof. We establish correctness of (4) (the arguments for (5) are similar) by using induction on i for a fixed j, thereby proving Theorem 1. Correctness of (4) is shown by arguing that it leads to values for $P[x_i, y_j, l]$ that satisfy Definition 1. In case $i = 0$, we already observed that $P[x_0, y_j, l] = -\infty$ (which is in agreement with Definition 1).

We will use induction and assume that $P[x_{i-1}, y_j, l+1]$ and $P[x_{i-1}, y_j, l]$ satisfy Definition 1. The question now is whether $P[x_i, y_j, l]$ computed using (4) satisfies Definition 1. Consider the revenue realized when the server is positioned in x_i, while y_j is the rightmost visited client. We distinguish two cases: x_i is served and x_i is not served. Consider first the case where x_i is served; as it is the last client visited, it is the last client served. Then, revenue can be broken down into three terms: (i) the revenue realized after serving a set of clients in $[x_{i-1}, y_j]$ when $l + 1$ clients will be served outside this interval, (ii) the travel time between the previous client visited and x_i, taking into account the $(l + 1)$ clients left to be served, (iii) the profit p_{x_i}. The previous client visited cannot lay outside $[x_i, y_j]$, in fact it is either x_{i-1} or y_j. Indeed, all clients within $[x_{i-1}, y_j]$ are met before x_i and it is optimal to visit them the first time the repairman passes by. Terms (ii) and (iii) are independent of the set of clients served and thus of the revenue realized in state $[x_{i-1}, y_j, l + 1]$. As a consequence, only the set of clients yielding the maximal revenue in $[x_{i-1}, y_j, l + 1]$ can lead to an optimal solution. It follows that the induction hypothesis tells us that the first term is accurately described by $P[x_{i-1}, y_j, l + 1]$ or by $P[y_j, x_{i-1}, l + 1]$. In addition, the second term equals $(l+1) * t[z, x_i]$ where z denotes the previous client visited (x_{i-1} or y_j), and finally the third term equals p_{x_i}. Now consider the case where x_i is not served. This means that only l clients are left to be served outside $[x_{i-1}, y_j]$ and no profit is realized in x_i. Revenue can be broken down into two terms: (i) the revenue realized after having served a set of clients in $[x_{i-1}, y_j]$ and l clients will be served outside this interval and (ii) the travel time between the previous client visited (x_{i-1} or y_j) and x_i, taking into account the l clients left to be served. Again, term (ii) is independent of the set of clients served and thus the revenue realized in state $[x_{i-1}, y_j, l]$. As a consequence, only the set of clients yielding the maximal revenue in $[x_{i-1}, y_j, l]$ can lead to an optimal solution. It follows that the induction hypothesis tells us that the first term is accurately described by $P[x_{i-1}, y_j, l]$ or by $P[y_j, x_{i-1}, l]$. In addition, the second term equals $(l) * t[z, x_i]$, where z denotes the previous client visited.

By taking the maximum of the four terms considered above, we see that $P[x_i, y_j, l]$ equals its definition. A similar argument holds for $P[y_j, x_i, l]$. □

To estimate the complexity of DP with $n = r + q$: we have at most n^3 possible states, and since every state has 4 elements in its maximization function, the time complexity of the algorithm is $O(n^3)$, and we can state our result.

Corollary 1. *DP solves TRPP on the line in polynomial time.*

Let us consider a number of directions in which Corollary 1 can be extended. First, notice that the distances $t[x_i, y_j]$ need not be symmetric. Thus, situations where the line models a river with clients located at upstream and downstream positions are solvable by DP. Next, we can extend Corollary 1 to other geometries. Consider n clients and one server positioned on a circle. Observe that in an optimal solution there is a circle segment between two clients not traversed by the server. Hence, by considering $O(n)$ TRPP instances on the line we can solve the problem on the circle.

We can also extend our result to the TRPP when the clients are positioned on the endpoints of a tree with width three and with positive real lengths on the edges. The observation here is that, in any optimal solution, the spokes at either end of the tree are visited in increasing order of their lengths [3]. Fig. 2 shows

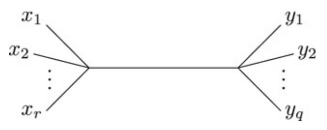

Fig. 2. A tree with width three.

an example of such a tree with r clients on the left side and q clients on the right side of the tree.

A state in DP $[x_i, y_j, l]$ then represents the situation in which x_i is the longest spoke visited at the left side and the current position of the server, y_j is the longest spoke visited at the right side, l is the number of clients left to be served. Similarly, we can define state $[y_j, x_i, l]$ with the current position of the server being y_j and x_i the longest spoke visited on the other side. DP can now be run using the states as defined here.

Thirdly, Corollary 1 can be extended to the TRPP on the line with common release dates. In this variant of the TRPP, a release date M is associated with every client, implying that a client cannot be served before $t = M$. Notice that, if M is large, this is equivalent to choosing a starting location for the server, since then the server can travel and choose the position where it will start at time $t = M$. Of course, in any optimal solution the server, starting at the origin at $t = 0$, will at time M either be in M or in $-M$ or at one of the clients in between these two points and wait there until the release date is reached. Thus, the server has at most n choices for its starting position at time $t = M$ and again DP is able to solve this problem.

A fourth extension is the TRPP on the line with constant repair times. Suppose that there is a repair time h at every client which is the same for all clients. This changes the revenue of a route only by a constant factor $\frac{1}{2}S(S+1)h$ where S is the total number of clients visited in the route. Thus, adding constant repair times has no influence on the computational complexity of the TRPP.

4. The complexity of MTRPP

In this section we discuss the TRPP with multiple servers (denoted by MTRPP). Given are the positions of n clients on the line and their associated profits p_i $(1 \le i \le n)$ and k servers, characterized by a speed s_j $(1 \le j \le k)$ and a starting location. The goal is to select clients and to find k routes serving each selected client, such that total collected revenue is maximized. Recall that the revenue collected at a client depends on the time needed to reach that client. Observe that, if all servers are in the origin at $t = 0$ and if all servers have equal speed, the problem becomes trivial: one server travels to the left and a second server travels to the right, and when they meet a client that contributes positively to the revenue, the client is visited. In case the starting locations of the k servers are arbitrary (but given), and the servers have identical speed, we claim the following:

Theorem 2. *MTRPP on the line with multiple identical servers is solvable in polynomial time.*

Proof. We only give a short sketch of the proof since it is similar to the proof of Wu [13] showing polynomial time solvability of the k-traveling repairmen problem on the line. First of all, note that the servers will never pass by one another because they all have the same speed. A solution to the MTRPP on the line with identical servers thus consists in a division of the line into k consecutive intervals and solving the TRPP on the line for each interval. Consider a line with n clients positioned at $x_1, x_2, \dots, x_n$ and k servers positioned at $z_1, z_2, \dots, z_k$. An interval on this line is denoted by $I(i, j)$, containing clients $(i, i+1, \dots, j)$ with $i \le j$. Then, define $P(i, j)$ as the maximal revenue of a set of routes visiting clients $(i, i+1, \dots, j)$ by the repairmen whose origins are within the interval $I(i, j)$. In a first phase compute $P(i, j)$ for each interval $I(i, j)$ containing exactly one origin. This can be done by computing revenue for each server k for its largest possible interval (i.e. the largest interval containing only server k) using DP. Then, maximal revenue for every smaller interval containing only server k is also known. Indeed, DP computes revenue for all combinations of i and j with l equal to zero. Total time complexity of this first phase is thus $O(n^4)$. In a second phase we select one interval for every server. For $1 < i < k$ and $z_i \le m < z_{i+1}$ we compute $P(1, m) = \max_{z_{i-1} \le j < z_i}\{P(1, j) + P(j+1, m)\}$, and total revenue $P(1, n) = \max_{z_{k-1} \le j < z_k}\{P(1, j) + P(j+1, n)\}$. Time complexity of the second phase is $O(n^2)$; total time complexity is thus dominated by the first phase and equal to $O(n^4)$. □

However, when the k servers have arbitrary speeds and clients have release dates $r_i \ge 0$, we claim that:

Theorem 3. *MTRPP on the line with non-identical servers and release dates is strongly NP-hard.*

We will prove NP-hardness for the MTRPP with non-identical servers and release dates by settling the complexity of an open problem mentioned in de Paepe et al. [4], i.e. $Q|s = t, r_j|line|\sum C_j$. This problem is a latency problem $(\sum C_j)$ on the line, with k non-identical servers (Q) and release dates (r_j); the phrase '$s = t$' refers to the fact that the clients only need to be visited (there is no transportation of clients). Notice that the clients do not have profits and each client needs to be served. The goal is to find routes for every server such that total latency is minimal. Recall that the profit variant of this problem, MTRPP with non-identical servers and release dates, is a generalization of this problem and thus at least as hard.

We transform numerical matching with target sums (NMTS) to $Q|s = t, r_j|line|\sum C_j$.

In an instance of NMTS we are given positive integers x_i $(1 \le i \le m)$, y_j $(1 \le j \le m)$ and b_l $(1 \le l \le m)$. The question is whether there exists a collection of m triples (i, j, l) such that: (i) $x_i + y_j = b_l$ for each triple, and (ii) each integer in the input occurs exactly once. This problem is proven to be NP-hard by Garey and Johnson [8].

Proof. We construct an instance of $Q|s = t, r_j|line|\sum C_j$ by specifying the speeds and the starting location of the servers, and the release dates and the location of the clients as follows. There are $k := m$ servers, the starting location of each server is the origin, and each server j has speed b_j (i.e. $s_j = b_j$, $j = 1, \dots, m$). There are $2m$ clients, m clients are located to the left of the origin at $-x_i$, $1 \le i \le m$ (the "left" clients), and

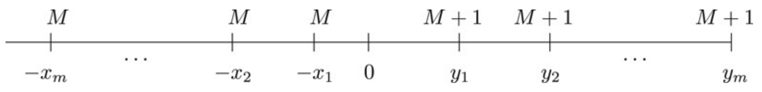

Fig. 3. MTRPP with release dates.

m clients are located to the right of the origin at y_j, $1 \le j \le m$ (the "right" clients), see Fig. 3. The release date of each left client equals $M \equiv \max_i x_i$ (i.e., $r_i = M$). The release date of each right client equals $M+1$ (i.e., $r_j = M+1$). The question is: does there exist a solution to $Q|s = t, r_j|line|\sum C_j$ such that total latency is equal to $mM + m(M+1)$? This completes the description of an instance of $Q|s = t, r_j|line|\sum C_j$.

We now establish correspondence between the two questions. Clearly, if there exists a numerical matching with target sums, we can direct each server j to the appropriate left location, where it waits till the release date M, serves the client at that location, travels to the appropriate right location (as given by the matching), arrives at time $M+1$ (due to the existence of a matching) and serves the right client. Total latency equals $mM + m(M+1)$.

If there exists a solution with total latency $mM + m(M+1)$, it follows that each client must be served at its release date. This implies that the m servers have to be present at $t = M$ at the locations of the m left clients, and at $t = M+1$ the servers need to be present at the m right clients. Hence, each server j travels from a unique left client at $-x_i$ to a unique right client at y_j in one time unit. It follows that NMTS has a solution. □

Corollary 2. *$Q|s = t, r_j|line|\sum C_j$ is strongly NP-hard.*

Notice that Yu et al. [15] show that NMTS remains hard even when $x_i = i$ ($i = 1, \ldots, m$) and $y_j = j$ ($j = 1, \ldots, m$). It follows that the TRPP problem remains hard when the clients are one distance-unit apart.

5. Open problems

The complexity of the following latency problems with profits is open at this moment:

(i) MTRPP with non-identical servers. (Notice also that the complexity of the problem without profits is still open, see [4].)
(ii) The weighted TRPP. (DP does not work here. In order to be able to calculate total latency incurred in a certain state, it is crucial to know how many clients will be served outside this state and the weights of these clients. We do not know which clients will be served outside this state and thus we do not know which weights to take into account for the computation of latency costs.)
(iii) It is unclear whether the $O(n)$ algorithm for the TRP in García et al. [7] could be used for the TRPP. Potentially, this could give a speedup of the current $O(n^3)$ bound of DP.

Acknowledgement

We are grateful to a referee whose remarks led to improved complexity bounds for our DP.

References

[1] F. Afrati, S. Cosmadakis, C.H. Papadimitriou, G. Papageorgiou, N. Papakostantinou, The complexity of the traveling repairman problem, Informatique Théorique et Applications 20 (1986) 79–87.
[2] I. Averbakh, O. Berman, Routing and location-routing p-deliverymen problems on a path, Transportation Science 28 (1994) 162–166.
[3] A. Blum, P. Chalasani, D. Coppersmith, B. Pulleyblank, P. Raghavan, M. Sudan, The minimum latency problem, in: Proceedings of the Twenty-sixth Annual ACM Symposium on the theory of Computing, STOC, 1994, pp. 163–171.
[4] W.E. de Paepe, J.K. Lenstra, J. Sgall, R.A. Sitters, L. Stougie, Computer-aided complexity classification of dial-a-ride problems, INFORMS Journal on Computing 16 (2004) 120–132.
[5] D. Feillet, P. Dejax, M. Gendreau, Traveling salesman problems with profits, Transportation Science 39 (2005) 188–205.
[6] P. Friese, J. Rambau, Online-optimization of multi-elevator transport systems with reoptimization algorithms based on set-partitioning models, Discrete Applied Mathematics 154 (2006) 1908–1931.
[7] A. García, P. Jodrá, J. Tejel, A note on the traveling repairman problem, Networks 40 (2002) 27–31.
[8] M.R. Garey, D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-completeness, Freeman, San Francisco, 1979.
[9] E. Minieka, The delivery man problem on a tree network, Annals of Operations Research 18 (1989) 261–266.
[10] H.N. Psaraftis, M.M. Solomon, T.L. Magnanti, T. Kim, Routing and scheduling on a shoreline with release times, Management Science 36 (1990) 212–223.
[11] R. Sitters, The minimum latency problem is NP-hard for weighted trees, in: Proceedings of the Ninth International Conference on Integer Programming and Combinatorial Optimization, IPCO, in: LNCS, vol. 2337, 2002, pp. 230–239.
[12] R. Sitters, Complexity and Approximation in Routing and Scheduling, Ph.D. Thesis, Technical University Eindhoven, 2004.
[13] B.Y. Wu, Polynomial time algorithms for some minimum latency problems, Information Processing Letters 75 (2000) 225–229.
[14] B.Y. Wu, Z. Huang, F. Zhan, Exact algorithms for the minimum latency problem, Information Processing Letters 92 (2004) 303–309.
[15] W. Yu, H. Hoogeveen, J.K. Lenstra, Minimizing makespan in a two-machine flow shop with delays and unit-time operations is NP-hard, Journal of Scheduling 7 (2004) 333–348.