International Journal of Pure and Applied Mathematics
Volume 62 No. 4 2010, 419-432

FLEET QUICKEST ROUTING ON GRIDS: A POLYNOMIAL ALGORITHM

Giovanni Andreatta[1], Luigi De Giovanni[2 §], Giorgio Salmaso[3]
[1,2,3]Department of Pure and Applied Mathematics
University of Padova
63, Via Trieste, Padova, 35121, ITALY
[1]e-mail: giovanni.andreatta@unipd.it
[2]e-mail: luigi@math.unipd.it
[3]e-mail: giorgio.salmaso@studenti.unipd.it

Abstract: Determining the shortest path for a vehicle moving on a network is easy. If more vehicles share the same network resources and we want vehicles to reach their destinations as soon as possible, moving on shortest paths might not be the optimal solution. In fact, conflicts may arise, requiring some vehicles to stay idle at some point of the shortest path: choosing an appropriate schedule on different paths may be quicker. The problem of finding the best set of conflict-free paths and schedule arises in many contexts: the coordination of automated guided vehicles, the management of airport groundside traffic, etc. After defining the general problem and proving its NP-hardness, we give a polynomial optimal dispatching algorithm for grids.

AMS Subject Classification: 90B06, 90C35, 05C85
Key Words: fleet routing, quickest paths, conflict management, exact dispatching algorithms

1. Introduction

We consider a routing network where a fleet of vehicles have to move from their origin to their destination. Network nodes correspond to vehicle origins and destinations, route crossing points or points where vehicles are allowed to stop and stay idle. Arcs correspond to directed or undirected routes between

Received: May 25, 2010

§Correspondence author

couples of nodes. For each arc/node a capacity is given, corresponding to the maximum number of vehicles that can move/stay idle on it. For each arc a fixed non negative traversal time is also given, which does not depend on the time one enters the arc nor on the number of vehicles moving on it. We want to determine a route for each vehicle in order to let it reach the desired destination *as quickly as possible*, e.g. minimizing the sum over all the arrival times. The problem arises in many contexts: the routing of automated guided vehicles in container terminals (see [6]), the coordination of ground service vehicles in airport aprons (see [11]), aircraft taxiing operations (see [9]), etc. Our research has been stimulated by the preparatory work for the project "Integrated Airport Apron Safety Fleet Management - AAS", funded by the European Commission in the Seventh Framework Programme (grant agreement 213061), aiming at improving the efficiency and the control of airport groundside movements by advanced vehicle and staff management (see [1]).

A possible solution to the proposed problem is to easily compute shortest paths on the graph underlying the routing network and to route vehicles on those shortest paths. In this case, however, conflicts may arise: several vehicles may need the same node or arc at the same time and the capacity constraints might be violated. To avoid conflicts, vehicles may stay idle at some nodes of their shortest path for some time, waiting for the next arcs to become available. It follows that moving on a set of shortest paths may not be the optimal choice. In this work we consider the problem of coordinating, by a single centralized dispatcher, a set of vehicles on a routing network. The dispatcher has to determine the schedule of the vehicles on the network, that is, *what* resources to use (the routing of each vehicle) and *when* to use them.

The problem has been the object of several studies, proposing different approaches. For example, [7] considers a simulation method to determine the schedule of aircraft moving on airport aprons; [10] proposes a metaheuristic approach to the same problem, using genetic algorithms; [3] uses queueing models to analyze the traffic of airport groundside vehicles; [9] and [11] use Mixed Integer Linear Programming to model and solve the problem of coordinating aircraft taxiing operations and airport groundside traffic as a flow problem on time-expanded networks; [6] proposes a heuristic algorithm for routing automated guided vehicles in container terminals, based on dynamically solving a set of shortest path problems with time windows. The problem is also related to network flows over time and, in particular to the Quickest Multicommodity Flow Problem: for each commodity, a given amount of flow has to be routed as quickly as possible on a capacitated network with fixed traversal time on arcs and the possibility of changing flow values over time. As for the case of fleet

routing, the time needed for the transmission plays an essential role, so that the feasibility of flows depends not only on the selected routes, but also on the time each commodity reaches the arcs on the routes themselves. [8] considers the *fractional* multicommodity flow over time problem, showing it is NP-hard (in the strong sense, if storage at intermediate nodes is not allowed) and proposing polynomial algorithms for special classes of networks. Concerning *integral* multicommodity flows over time, which is more related to the object of our study, the problem is strongly NP-hard, as from [4] and [8].

In this paper we state the Fleet Quickest Routing (FQR) problem and present some preliminary results, extending the ones in [2]. After showing that the problem is NP-hard, we focus on the FQR problem on grid networks: we present a polynomial dispatching algorithm to solve the problem to optimality and we outline some directions for future research.

2. Problem Statement and Complexity

We consider a fleet of vehicles V to be moved on a routing network $\mathcal{N} = (N, A)$, where N is the set of nodes and $A = A_D \cup A_U$ is the set of directed (A_D) and undirected (A_U) arcs connecting pairs of nodes. Each vehicle $v \in V$ is available at an initial node $s_v \in N$ at time $t_v^\alpha \geq 0$, and has to reach a destination $d_v \in N$. For each arc $(i, j) \in A$, a traversal time $\tau_{ij} \geq 0$ is given, together with a capacity u_{ij}, meaning that at most u_{ij} vehicles at a time are allowed to move on (i, j). Analogously, node capacity u_i corresponds to the maximum number of vehicles that can stay simultaneously idle on $i \in N$. Stopping on arcs is not allowed.

Given a vehicle $v \in V$, a *vehicle schedule* $\mathcal{S}_v$ is an ordered sequence

$$\mathcal{S}_v = \langle (t_0, s_v, j_0), (t_1, i_1, j_1) ... (t_{K_v}, i_{K_v}, d_v) \rangle,$$

where $t_k \geq 0$ and $(i_k, j_k) \in A$, $\forall\ k = 0..K_v$. A triplet $(t, i, j) \in \mathcal{S}_v$ means that v starts moving on arc $(i, j) \in A$ at time t and reaches j at time $t + \tau_{ij}$. $\mathcal{S}_v$ must define a chain on the routing network from s_v to d_v, respecting arc traversal time, that is: $t_0 \geq t_v^\alpha$; $i_{k+1} = j_k$ and $t_{k+1} \geq t_k + \tau_{i_k j_k}$, $\forall\ k = 0...K_v - 1$.

In order to take node and arc capacities into account we remark that chains may contain cycles, that is, vehicles may traverse the same arc or stay on the same node more than once. Given a schedule $\mathcal{S}_v$, let T_i^v and T_{ij}^v be the set of *all* the interval times spent by vehicle v on node i and to move from i to j:

$$T_i^v = \beta_i^v \cup \gamma_i^v \bigcup_{k=0..K_v-1 : j_k = i} [t_k + \tau_{i_k i}, t_{k+1}],$$

where $\beta_i^v = [t_v^\alpha, t_0]$ if $i = s_v$, $\emptyset$ otherwise (β_i^v adds the first time interval spent by v on its origin node), and $\gamma_i^v = \{t_{K_v} + \tau_{i_{K_v} d_v}\}$ if $i = d_v$, $\emptyset$ otherwise (γ_i^v adds the time in which v finally reaches its destination).

$$T_{ij}^v = \bigcup_{(t,i,j) \in \mathcal{S}_v} (t, t + \tau_{ij}).$$

Let $I_i^v(t)$ and $I_{ij}^v(t)$ be the corresponding indicator functions:

$$I_i^v(t) = \begin{cases} 1 & \text{if } t \in T_i^v, \\ 0 & \text{else,} \end{cases} \qquad I_{ij}^v(t) = \begin{cases} 1 & \text{if } t \in T_{ij}^v, \\ 0 & \text{else.} \end{cases}$$

A *fleet schedule* is a set of vehicle schedules $\mathcal{S}_v$, one for each $v \in V$, satisfying node and arc capacities, that is:

$$\sum_{v \in V} I_i^v(t) \leq u_i, \qquad \forall\, t \geq 0, i \in N, \tag{1}$$

$$\sum_{v \in V} I_{ij}^v(t) \leq u_{ij}, \qquad \forall\, t \geq 0, (i,j) \in A_D, \tag{2}$$

$$\sum_{v \in V} \left(I_{ij}^v(t) + I_{ji}^v(t) \right) \leq u_{ij}, \qquad \forall\, t \geq 0, (i,j) \in A_U. \tag{3}$$

For example, consider the case of Figure 1, where numbers on arcs represent the traversal times, arcs are all undirected and all capacities are equal to one. A possible fleet schedule is

$$\mathcal{S}'_a = \langle (0,4,3), (1,3,2), (2,2,1), (4,1,3), (5,3,5) \rangle,$$

$$\mathcal{S}'_b = \langle (0,5,2), (2,2,3), (3,3,1) \rangle.$$

The schedule is not feasible as node conflicts arise: $I_2^a(2) + I_2^b(2) = 2$ and $I_1^a(4) + I_1^b(4) = 2$. If vehicle b stays idle on node 5 for half time unit, we have an alternative schedule

$$\mathcal{S}''_a = \langle (0,4,3), (1,3,2), (2,2,1), (4,1,3), (5,3,5) \rangle,$$

$$\mathcal{S}''_b = \langle (0.5,5,2), (2.5,2,3), (3.5,3,1) \rangle,$$

where node conflicts are avoided, but new conflicts on the undirected arc $(1,3)$ arise: $I_{13}^a(t) + I_{31}^b(t) = 2, \forall\, t \in (4, 4.5)$. Introducing a further stop of two time units at node 2 for b, we finally obtain the following feasible fleet schedule:

$$\mathcal{S}_a = \langle (0,4,3), (1,3,2), (2,2,1), (4,1,3), (5,3,5) \rangle,$$

$$\mathcal{S}_b = \langle (0.5,5,2), (4.5,2,3), (5.5,3,1) \rangle.$$

A schedule can be evaluated by considering the arrival time of each vehicle,

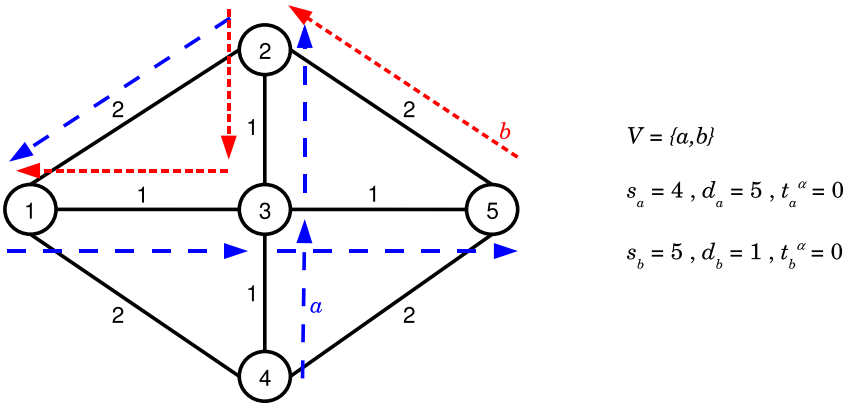

Figure 1: Sample fleet schedule

that is, the time $\Omega(\mathcal{S}_v) = t_{K_v} + \tau_{i_{K_v} d_v}$ at which a vehicle v finally reaches its destination d_v. In the example above, $\Omega(\mathcal{S}_a) = 6$ and $\Omega(\mathcal{S}_b) = 6.5$.

Given a routing network $\mathcal{N}$ and a fleet of vehicles V, the *Fleet Quickest Routing* (FQR) problem is to determine a fleet schedule minimizing the sum over all the vehicle arrival times:

$$\begin{array}{lll} \text{(FQR)} & \min & \displaystyle\sum_{v \in V} \Omega(\mathcal{S}_v) \\ & \text{s.t.} & \mathcal{S}_v \text{ is a vehicle schedule} \quad \forall\, v \in V \\ & \text{and} & (1),\ (2),\ (3) \text{ above.} \end{array}$$

To study the complexity of FQR we introduce its recognition version R-FQR: given an instance of FQR and a time threshold H, is there any feasible schedule such that the sum over all the vehicle arrival times is less than or equal to H?

The NP-completeness of R-FQR, and thus the NP-hardness of FQR, can be obtained by reduction from the $k-$Disjoint Path (kDP) problem, which is NP-complete [5]. kDP can be defined as follows (see [12, vol. C]): given a directed graph $G = (N, A)$ and a set of k origin-destination pairs, determine if a set of k node-disjoint paths from each origin to each destination does exist.

Property 1. *R-FQR is NP-complete.*

Proof. Let us consider the kDP problem on graph $G = (N, A)$ and a set of origin-destination pairs $(a_v, b_v), v = 1...k$, and build the following instance of R-

FQR. The routing network $\mathcal{N}$ coincides with the graph G, with $u_i = 1, \forall\ i \in N$, $\tau_{ij} = 0, u_{ij} = \infty, \forall\ (i,j) \in A$. The set V contains a vehicle v for each origin-destination pair: $s_v = a_v$, $d_v = b_v$ and $t_v^\alpha = 0$. Finally, set the threshold $H = 0$. We now show that the obtained R-FQR instance is a *yes-instance* if and only if the kDP distance is a *yes-instance*. By solving R-FQR we obtain a vehicle schedule $\mathcal{S}_v$ for each $v \in V$. If the objective function is zero, then all the triplets in $\mathcal{S}_v$ are of the type $(0, i, j)$ and each node is used by one vehicle at most once, otherwise node capacity (1) would be violated. This means that the paths implied by $\mathcal{S}_v$ are node-disjoint. If G contains the required set of node-disjoint paths, the same paths provide a solution to R-FQR where no vehicle needs to stop at any node, so that the objective function value is zero. □

3. Problem Statement in Grid Networks

In this work we will consider the special case of the *FQR problem on grid networks*. A grid network is defined by a graph $G = (N, A)$ where N is the set of nodes and A is the set of undirected arcs, such that the network is a planar grid with m rows and n columns (see Figure 2). Columns correspond to vertical lanes. Each row corresponds to a horizontal lane and is also called *level*. Note that, for each node $i \in N$, horizontal and vertical coordinates $x(i)$ and $y(i)$ can be defined, corresponding to the horizontal and vertical lane of the node itself: node i can be identified by the pair $\langle x(i), y(i) \rangle$. Traversing a horizontal or vertical arc takes one time unit, that is $\tau_{ij} = 1, \forall\ (i,j) \in A$. Nodes and arcs have unit capacity ($u_i = u_{ij} = 1, \forall i \in N, (i,j) \in A$). All vehicles of the fleet V are available at time 0 ($t_v^\alpha = 0, \forall\ v \in V$) and are initially placed at the same level, corresponding to the bottom of the grid ($y(s_v) = 1, \forall\ v \in V$). The destination of all vehicles lays on the highest level ($y(d_v) = m, \forall\ v \in V$). No two vehicles have the same origin, nor the same destination. As for the general FQR, we want to find a feasible schedule for the fleet V minimizing the sum of all the vehicle arrival times.

In the following, we will concentrate on this simplified FQR problem and we will derive a polynomial algorithm to solve it to optimality.

4. A Polynomial Dispatching Algorithm for Grid Networks

It is easy to observe that the length of a shortest path between any two nodes i and j on the grid network corresponds to their *Manhattan distance*: $|x(i) -$

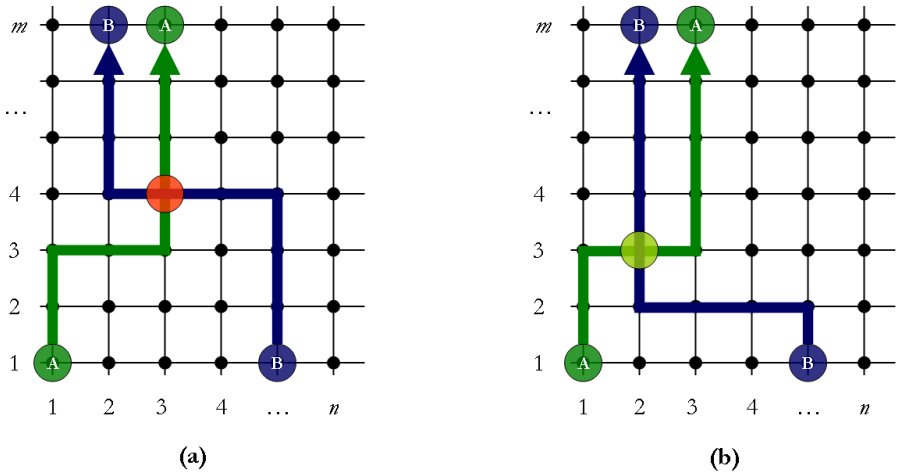

Figure 2: A grid network with alternative schedules

$x(j)| + |y(i) - y(j)|$. Every path with length equal to the Manhattan distance is called *Manhattan path*.

Remark 2. If we are able to route all vehicles on Manhattan paths *without stopping*, we obtain an optimal solution to the FQR problem on grid networks.

In fact, routes would have a number of vertical and horizontal movements as small as possible and vehicles would never stay idle at any nodes, so that all of the arrival times are minimized. Note that the solution would be optimal also if the minimization of the maximum over all the vehicles arrival times is taken as objective function.

Of course, conflicts may arise. For example, if we consider the routes in Figure 2a, both vehicles A and B would arrive at the node $\langle 3,4\rangle$ at the same time $t = 5$: $I^A_{\langle 3,4\rangle}(5) + I^B_{\langle 3,4\rangle}(5) = 2 > u_{\langle 3,4\rangle} = 1$. The conflict may be resolved, for example, by stopping vehicle A at node $\langle 3,3\rangle$ for one time unit, but, in so doing, we will loose optimality. Alternatively, we may route A and B according to the Figure 2b, where A and B cross node $\langle 2,3\rangle$ at time 3 and 5 respectively, so that vehicles do not need to stop and routing optimality is preserved. The question is thus the following: is it possible to choose Manhattan paths and to avoid any node or arc conflicts?

We observe that, in the example above, every Manhattan path for B contains more horizontal steps than A and, in the conflict-free solution (Figure 2b),

the level occupied by B at time t is always less than or equal to that of A, for all $t \geq 0$. Generalizing this observation, the algorithm we propose would give priority to the horizontal movement of the vehicles with a *high enough* number of remaining horizontal steps. Let us introduce the following notation. Given a vehicle $v \in V$ and a time index t:

— $H_0(v)$ is the initial horizontal distance, that is the number of horizontal steps in *any* Manhattan path from s_v to d_v: $H_0(v) = |x(d_v) - x(s_v)|$;

— $p_t(v) \in N$ is the position (node) of vehicle v at time t;

— $H_t(v)$ is the current horizontal distance: $H_t(v) = |x(p_t(v)) - x(d_v)|$;

— $L_t(v)$ is the current level of vehicle v: $L_t(v) = y(p_t(v))$;

— M_t is the maximum level reached by any vehicle at time t: $M_t = \max_{v \in V}\{L_t(v)\}$

Finally, we arbitrarily fix an orientation on horizontal lanes in such a way that alternate one-way horizontal movements are allowed (for example, odd lanes allow moving from left to right and even lanes from right to left). We introduce the following definitions:

— $concordant_t(v)$ (resp. $discordant_t(v)$) is *true* if $H_t(v) > 0$ (horizontal steps remaining) and v is on a horizontal lane allowing (resp. not allowing) the horizontal step towards the desired destination, *false* otherwise.

— K_t is the maximum current horizontal distance $H_t(v)$ of a discordant vehicle on the current maximum level M_t (it plays the role of the threshold horizontal distance in the proposed algorithm).

The Dispatching Algorithm (DA) for grid networks is presented in Figure 3. It determines a fleet schedule using function `push` (see Figure 4) to add triplets implying one-arc horizontal (*direction* `= HOR`) or vertical (*direction* `= VER`) steps (no triplet is added for vehicles that have already reached their destination). At iteration t (corresponding to time t), DA moves horizontally all concordant vehicles, except those on the highest occupied level whose distance is less than the threshold K_t set at step 5. In fact, this threshold allows us to *give priority* to the vehicles with *high enough* horizontal steps remaining, as observed before. All other vehicles are pushed vertically: the horizontal movement is not possible (discordant vehicles) or would cause a non-Manhattan path (vehicles aligned with their destinations).

DA solves the FQR problem on grid networks to optimality. In fact, we can show that vehicles are routed on one Manhattan path without stopping and that no conflicts are generated. All the types of potential conflicts arising on a

1. t := 0; for each $v \in V$, set $p_0(v)$:=s_v and $\mathcal{S}_v = \emptyset$;
2. Repeat
3. M_t := $\max_{v \in V}\{L_t(v)\}$
4. Δ := $\{v \in V : discordant_t(v)$ = true and $L_t(v) = M_t\}$
5. K_t := $\begin{cases} \max_{v \in \Delta}\{H_t(v)\} & \text{if } \Delta \neq \emptyset \\ 0 & \text{otherwise} \end{cases}$
6. for each $v \in V$
7. if $concordant_t(v)$ then
8. if $L_t(v) < M_t$ then push(t,v,HOR)
9. else if $H_t(v) \geq K_t$ then push(t,v,HOR)
10. else push(t,v,VER)
11. else push(t,v,VER)
12. t := $t + 1$
13. Until $p_t(v) = d_v$, $\forall\, v \in N$

Figure 3: The dispatching algorithm (DA) for grid networks

grid network are illustrated in Figure 5 and numbered from 1 to 12.

Lemma 3. *If a vehicle $v \in V$ is pushed vertically at step/time t then either $H_t(v) = H_0(v)$ or $H_t(v) = 0$.*

Proof. We show the assert by induction. The assert is trivially true for $t = 0$. By induction hypothesis, assume that each v' vertically pushed at time $t - 1$ either has not yet moved horizontally ($H_{t-1}(v') = H_0(v')$) or is aligned ($H_{t-1}(v') = 0$). According to the lemma hypothesis, v is pushed vertically and two cases holds, corresponding to DA steps 10 and 11. In the first case, v is concordant and $\exists\ w \in V : w$ is discordant, $L_t(w) = L_t(v) = M_t$ and $H_t(v) <$

```
push(t,v,direction) :
    if p_t(v) = d_v then p_{t+1}(v) = p_t(v); return.
    if direction = HOR then p_{t+1}(v) = ⟨x(p_t(v)) ± 1, y(p_t(v))⟩
    if direction = VER then p_{t+1}(v) = ⟨x(p_t(v)), y(p_t(v)) + 1⟩
    S_v := S_v ∪ (t, p_t(v), p_{t+1}(v))
    return.
```

Figure 4: The `push` function

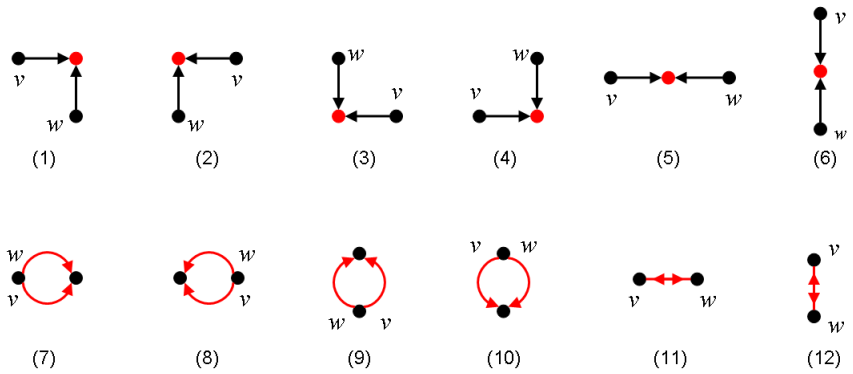

Figure 5: Potential conflicts on a grid network

$H_t(w)$. As w is discordant, it was not pushed horizontally at iteration $t-1$ (it would have been on the same discordant level at $t-1$). Thus, w was vertically pushed at step $t-1$, meaning that $H_t(w) = H_{t-1}(w)$ and either $H_{t-1}(w) = H_0(w)$ or $H_{t-1}(w) = 0$, as from the induction hypothesis. From $H_t(v) < H_t(w) = H_{t-1}(w)$ it follows that $H_t(w) = H_0(w)$, since $H_t(v)$ cannot be less than 0. Thus, w have not moved horizontally yet. DA never let any vehicle idle, so that both v and w have performed t moves. Also, they performed the same number of vertical moves (M_t) and, hence, the same number of horizontal

moves, namely 0. As a consequence, v was vertically pushed at step $t-1$, meaning that $H_t(v) = H_{t-1}(v)$ and, from induction hypothesis either $H_t(v) = H_0(v)$ or $H_t(v) = 0$, which prove the first case. In the second case (step 11), v is aligned ($H_t(v) = 0$) or discordant: at time $t-1$ it was necessarily pushed vertically ($H_t(v) = H_{t-1}(v)$) and, by the induction hypothesis we have the assert. □

Lemma 4. *If a conflict of type* 1 *or* 2 *between vehicles* v *and* w *arises at time* t*, then* $H_t(w) = 0$ *and* $H_t(v) > 0$.

Proof. Under the hypothesis, $L_{t-1}(w) < L_{t-1}(v)$ (see Figure 5). As all vehicles start at the same level and never stay idle, there has been a time before $t-1$ when w moved horizontally while v continued moving vertically, that is $H_{t-1}(w) < H_0(w)$. At time $t-1$, w is moving vertically, so that, according to Lemma 3, $H_t(w) = H_{t-1}(w) = 0$. Also, v is moving horizontally meaning that $H_{t-1}(v) > 0$ and $H_t(v) \geq 0$. Actually, $H_t(v)$ cannot be 0, otherwise v and w would have the same destination, which contradicts our assumptions about distinct destination nodes. □

Property 5. *The Dispatching Algorithm does not generate conflicts.*

Proof. DA considers one-way horizontal lanes and conflicts of type 5 and 11 are excluded; DA never pushes vehicles down and conflicts of type 3, 4, 6, 10 and 12 are avoided; conflicts of type 7, 8 and 9 are excluded if node conflicts 1, 2, 3, 4, 5 and 6 are excluded (no two vehicles are on the same node simultaneously). Therefore, we have just to show that conflicts of type 1 and 2 never occur. Let us assume a conflict of type 1 or 2 occurs between vehicles v and w at time t and at level $l+1$. As DA never let any vehicle idle, it means that both v and w have performed t moves: l vertical moves and $t-l$ horizontal moves. From Lemma 4, $H_0(w) = t-l$ and $H_0(v) > t-l$, that is, $H_0(v) > H_0(w)$. As observed before, there should be a level d corresponding to a time d when w started moving horizontally (w was concordant) and v continued moving vertically. We have $H_d(w) = H_0(w)$ and $H_d(v) = H_0(v)$, that is, $H_d(v) > H_d(w)$. We have two cases: if v was discordant at level d, then $H_d(w) < H_d(v) \leq K_d$ and w could not move horizontally; if v was concordant at level d, then, if the DA pushes w horizontally, then v too should be pushed horizontally, as $H_d(v) > H_d(w) \geq K_d$. In any case, we have a contradiction, proving that the DA excludes conflicts of type 1 and 2. □

Remark 6. At each iteration, the DA pushes each vehicle on one of its Manhattan paths without stopping. Also, no conflicts are generated and, according to Remark 2, routings are optimal.

Of course, in order to avoid conflicts, DA may indefinitely push vehicles vertically, meaning that an indefinite number of rows may be needed to align vehicles. This circumstance is excluded by the following lemmas.

Lemma 7. *At each iteration t, no discordant vehicle is below the top level M_t.*

Proof. The assert is trivially true for $t = 0$. Let us assume, by induction hypothesis, that the assert is true at $t-1$, that is, all vehicles below the level M_{t-1} are concordant or aligned. At $t-1$, just concordant vehicles are pushed horizontally. We have two cases. If some vehicles are pushed vertically, then $M_t = M_{t-1} + 1$ and discordant vehicles, if any, lay on level M_t. If no vehicles are pushed vertically at time $t-1$, then $M_t = M_{t-1}$, and no discordant vehicles are below level M_t, from the induction hypothesis. □

Lemma 8. *For any iteration $t \geq 1$, $K_t < K_{t-1}$ or $K_t = 0$.*

Proof. If $K_t \neq 0$ then, by steps 3-5, the top level M_t contains a discordant vehicle $v : H_t(v) = K_t > 0$. Being v discordant at time t, it was necessarily pushed vertically at time $t-1$, so that $H_{t-1}(v) = H_t(v) = K_t > 0$ and v was concordant at step $t-1$. According to step 10, $L_{t-1}(v) = M_{t-1}$ and $H_{t-1}(v) < K_{t-1}$, that is, $K_t < K_{t-1}$. □

By Lemma 8, the maximum horizontal distance at the top level strictly decreases from one iteration to the other and becomes 0 (no discordant vehicle at the top level) after at most K_0 iterations, K_0 being the maximum initial horizontal distance. Then, by Lemma 7, all vehicles are concordant or aligned after at most K_0 iterations and, hence, at most K_0 horizontal lanes are necessary. We can thus formally prove the polynomial convergence of DA.

Property 9. *Given an $m \times n$ grid (with $m \geq K_0$), DA converges in polynomial time $O(n^2)$ to an optimal solution of the FQR problem on grid networks.*

Proof. By Remark 6, Lemmas 7 and 8 and under the technical hypothesis $m \geq K_0$, DA converges to an optimal solution of FQR. Concerning the computational complexity, the number of iterations is bounded by $m + n$, the length of the longest Manhattan path on a $m \times n$ grid. At each iteration, $|V|$ push operations are performed in constant time and the algorithm converges in $O(|V|(m+n))$. The number of vehicles is bounded by n, as no two vehicles can start at the same node. The number m of necessary horizontal lanes is K_0 which is also bounded by n. □

5. Conclusions

The Fleet Quickest Routing (FQR) problem has been considered. It has relevant applications, for example, to the routing of automated guided vehicles or to the dispatching of aircraft and other equipment on airport groundsides. In this work we have considered the FQR problem on grid networks with some assumptions on the initial fleet arrangement. For this special case, we have proposed a conflict-free Dispatching Algorithm (DA) able to solve the problem to optimality in polynomial time. The algorithm is very efficient and could be implemented by a centralized real-time dispatcher. The efficiency and simplicity of the DA may suggest to organize the routing network as a grid, if possible, for example when a new network for automated guided vehicles has to be settled. The remaining assumptions, in particular the one concerning the horizontal alignment of vehicle starting positions, may be too restrictive and we are interested in extending the results to more general FQR problems. For example, the extension to grid networks with vehicle destinations on arbitrary levels of distinct vertical lanes is trivial, as the same DA works. Non-trivial are the extensions to more general cases, including grids with arbitrary vehicle origins and destinations (useful also to model non-synchronized starts), non-complete grids (modeling possible non-transit zones within the routing area) etc. Ideally, we would like to consider the FQR problem on arbitrary networks. This is the object of ongoing and future research.

References

[1] AAS Team, *Integrated Airport Apron Safety Fleet Management - AAS - Project*, http://www.aas-project.eu/, 2009.

[2] G. Andreatta, L. De Giovanni, G. Salmaso, Quickest paths on congested networks: some special cases, In: *Proceedings of International Network Optimization Conference - INOC 2009*, Pisa, April 26-29 (2009).

[3] K. Andersson, F. Carr, E. Feron, W.D. Hall, Analysis and modeling of ground operations at hub airports, In: *Proceedings of Air Traffic Management R&D Seminar*, Napoli, June 13-16 (2000).

[4] T. Erlebach, K, Jansen, Call scheduling in trees, rings and meshes, In: *Proceedings of the 30-th Hawaii International Conference on System Science*, IEEE Computer Society Press (1997), 221-222.

[5] S. Even, A. Itai, A. Shamir, On the complexity of timetable and multicommodity flow problems, *SIAM Journal on Computing*, **5** (1976), 691-703.

[6] E. Gawrilow, E. Köhler, R.H. Möhring, B. Stenzel, Dynamic routing of automated guided vehicles in real time, In: *Mathematics: Key Technology for the Future. Joint Projects between Universities and Industry 2004-2007* (Ed-s: W. Jäger, H.J. Krebs), Springer, London (2008), 165-178.

[7] J.B. Gotteland, N. Durand, J.M. Alliot, E. Page, Aircraft ground traffic optimization, In: *Proceedings of 4-th International Air Traffic Management R&D Seminar*, Santa Fe, December 3-7 (2001).

[8] A. Hall, S. Hippler, M. Skutella, Multicommodity flows over time: Efficient algorithms and complexity, *Theoretical Computer Science*, **379** (2007), 387-404.

[9] A.G. Marin, Airport management: Taxi planning, *Annals of Operations Research*, **143** (2006), 191-202.

[10] B. Pesic, N. Durand, J.M. Alliot, Aircraft ground traffic optimization using genetic algorithms, In: *Proceedings of GECCO*, San Francisco, July 7-11 (2001).

[11] P.C. Roling, H.G. Visser, Optimal airport surface traffic planning using mixed integer linear programming, *International Journal of Aerospace Engineering*, doi:10.1155/2008/732828 (2008).

[12] A. Schrijver, *Combinatorial Optimization*, Springer (2003).