Robust Line Planning in case of Multiple Pools and Disruptions

Apostolos Bessas[1,3], Spyros Kontogiannis[1,2], and Christos Zaroliagis[1,3]

[1] R.A. Computer Technology Institute, N. Kazantzaki Str., Patras University Campus, 26500 Patras, Greece
[2] Computer Science Department, University of Ioannina, 45110 Ioannina, Greece
[3] Department of Computer Engineering and Informatics, University of Patras, 26500 Patras, Greece
Email: `mpessas@ceid.upatras.gr`, `kontog@cs.uoi.gr`, `zaro@ceid.upatras.gr`

Abstract. We consider the line planning problem in public transportation, under a robustness perspective. We present a mechanism for robust line planning in the case of multiple line pools, when the line operators have a different utility function per pool. We conduct an experimental study of our mechanism on both synthetic and real-world data that shows fast convergence to the optimum. We also explore a wide range of scenarios, varying from an arbitrary initial state (to be solved) to small disruptions in a previously optimal solution (to be recovered). Our experiments with the latter scenario show that our mechanism can be used as an online recovery scheme causing the system to re-converge to its optimum extremely fast.

1 Introduction

Line planning is an important phase in the hierarchical planning process of every railway (or public transportation) network[1]. The goal is to determine the routes (or lines) of trains that will serve the customers along with the frequency each train will serve a particular route. Typically, the final set of lines is chosen by a (predefined) set of candidate lines, called the *line pool*. In certain cases, there may be *multiple line pools* representing the availability of the network infrastructure at different time slots or zones. This is due to variations in customer traffic (e.g., rush-hour pool, late evening pool), maintenance (some part of the network at a specific time zone may be unavailable), dependencies between lines (e.g., the choice of a high-speed line may affect the choice of lines for other trains), etc.

The line planning problem has been extensively studied under cost-oriented or customer-oriented approaches (see e.g., [3, 4, 7, 9]). Recently, robustness issues have been started to be investigated. In the *robust line planning* problem, the task is to provide a set of lines along with their frequencies, which are robust to fluctuations of input parameters; typical fluctuations include, for instance,

[1] For the sake of convenience, we concentrate in this work on railway networks, but the methods and ideas developed can be applied to any public transportation network.

disruptions to daily operations (e.g., delays), or varying customer demands. In [8], a game-theoretic approach to robust line planning was presented that delivers lines and frequencies that are robust to delays.

A different perspective of robust line planning was investigated in [1]. This perspective stems form recent regulations in the European Union that introduce competition and free railway markets. Under these rules the following scenario emerges: there is a (usually state) authority that manages the railway network infrastructure, referred to as the *Network Operator* (NOP), and a (potentially) large number of *Line Operators* (LOPs) operating as commercial organizations which want to offer services to their customers using the given railway network. These LOPs act as competing agents for the exploitation of the shared infrastructure and are unwilling to disclose their utility functions that demonstrate their true incentives. The network operator wishes to set up a fair cost sharing scheme for the usage of the shared resources and to ensure the maximum possible level of satisfaction of the competing agents (by maximizing their aggregate utility functions). The former implies a resource pricing scheme that is robust against changes in the demands of the LOPs, while the latter establishes a notion of a socially optimal solution, which could also be considered as a fair solution, in the sense that the average level of satisfaction is maximized. In other words, the NOP wishes to establish an incentive-compatible mechanism that provides *robustness* to the system in the sense that it tolerates the agents' unknown incentives and elasticity of demand requests and it eventually stabilizes the system at an equilibrium point that is as close as possible to the social optimum.

The first such mechanism, for robust line planning in the aforementioned scenario, was presented in [1]. In that paper, the following mechanism was investigated (motivated by the pioneering work of Kelly et al. [5, 6] in communication networks): the LOPs offer bids, which they (dynamically) update for buying frequencies. The NOP announces an (anonymous) resource pricing scheme, which indirectly implies an allocation of frequencies to the LOPs, given their own bids. For the case of a single pool of lines, a distributed, dynamic, LOP bidding and (resource) price updating scheme was presented, whose equilibrium point is the unknown social optimum – assuming strict concavity and monotonicity of the private (unknown) utility functions. This development was complemented by an experimental study on a discrete variant of the distributed, dynamic scheme on both synthetic and real-world data showing that the mechanism converges really fast to the social optimum. The approach to the single pool was extended to derive an analogous mechanism for the case of multiple line pools, where it was assumed that (i) the NOP can periodically exploit a whole set of (disjointly operating) line pools and he decides on how to divide the whole infrastructure among the different pools so that the resource capacity constraints are preserved; (ii) each LOP may be interested in different lines from different pools; and (iii) each LOP has a single utility function which depends on the aggregate frequency that she gets from all the pools in which she is involved.

The aforementioned theoretical framework demonstrated the potential of converging to the social optimum via a mechanism that exploits the selfishness of LOPs. A significant issue is the speed or rate of convergence of this mechanism. Since there was no theoretical treatment of this issue, its lack was covered in [1] for the single pool case via a complementary experimental study. Despite, however, the significance of the convergence rate issue, the mechanism for the multiple pool case was *not* experimentally evaluated in [1].

For the case of multiple line pools, it is often more realistic to assume that each LOP has a different utility function per pool, since different pools are expected to provide different profits (e.g., intercity versus regional lines, or rush-hour versus late-evening lines). Moreover, it seems more natural to assume that each LOP has a different utility function per pool that depends on the frequency she gets for that pool, rather than a single utility function that depends on the total frequency she gets across all pools.

In this work, we continue this line of research by further investigating the multiple pool case. In particular, we make the following contributions: (1) Contrary to the approach in [1], we consider the case where each LOP has a different utility function for each line pool she is interested in, and show how the approach in [1] can be extended in order to provide a mechanism for this case, too. (2) We conduct an experimental study on a discrete variant of the new mechanism on both synthetic and real-world data demonstrating its fast convergence to the social optimum. (3) We conduct an additional experimental study, on both synthetic and real-world data, to investigate the robustness of the system in the case of disruptions that affect the available capacity, which may be reduced (due to temporary unavailability of part of the network), or increased (by allowing usage of additional infrastructure during certain busy periods). In this case, we show that the NOP can re-converge (recover) the system to the social optimum pretty fast, starting from a previous optimal solution.

Due to space limitations, the reader is referred to the full version [2] for the missing details and proofs.

2 Multiple Line Pools: Different Utilities per Pool

The exposition in this section follows that in [1]. In the *line planning* problem, the NOP provides the public transportation infrastructure in the form of a directed graph $G = (V, L)$, where V is the node set representing train stations and important railway junctions, and L is the edge set representing direct connections (of railway tracks) between elements of V. Each edge $\ell \in L$ is associated with a capacity $c_\ell > 0$, which limits the number of trains that can use this edge in the period examined. A line p is a path in G. We assume that there is set K of line pools, where each pool corresponds to a different period of the day and represents a different set of possible routes. We envision the line pools to be implemented in disjoint time intervals (e.g., via some sort of time division multiplexing), and also to concern different characteristics of the involved lines (e.g., high-speed pool, regular-speed pool, local-trains pool, rush-hour pool, night-shift pool, etc.). The capacity of each resource (edge) refers to its usage (number of trains) over the whole time period we consider (e.g., a day), and if a particular pool consumes (say) 50% of the whole infrastructure, then this implies that for all the lines in this pool, each resource may exploit at most half of its capacity. It is up to the NOP to determine how to split a whole operational period of the railway infrastructure among the different pools, so that (for the whole period) the resource capacity constraints are not violated.

There is also a set P of LOPs, who choose their lines from K. We assume that each LOP $p \in P$ is interested only in one line in per pool (we can always enforce this assumption by considering a LOP interested in more than one routes as different LOPs distinguished by the specific route). Each line pool and the preferences of LOPs to lines in it are represented by a *routing matrix* $\boldsymbol{R}(k) \in \{0,1\}^{|L|\times|P|}, k \in K$. Each row $\boldsymbol{R}_{\ell,\star}(k)$ corresponds to a different edge $\ell \in L$, and each column $\boldsymbol{R}_{\star,p}(k)$ corresponds to a different LOP $p \in P$, showing which edges comprise her line in pool k.

Each LOP $p \in P$ acquires a *frequency* of trains that she wishes to route over her paths in $\boldsymbol{R}_{\star,p}(k), k \in K$, such that no edge capacity constraint is violated by the aggregate frequency running through it by all LOPs and pools. A utility function $U_{p,k} \colon \mathbb{R}_{\geq 0} \mapsto \mathbb{R}_{\geq 0}$ determines the *level of satisfaction* of LOP $p \in P$ in pool $k \in K$ for being given an end-to-end frequency $x_{p,k} > 0$. Having different utility functions per pool instead of a single utility function across all pools, is more generic and hence more realistic, since a LOP p can indeed have different valuations for different periods of a day (rush-hour pool vs night-shift pool) and/or different types of trains (high-speed pool vs local-trains pool). These utility functions are assumed to be strictly increasing, strictly concave, non-negative real functions of the end-to-end frequency $x_{p,k}$ allocated to LOP $p \in P$ in pool $k \in K$. The aggregate satisfaction level U_p of LOP $p \in P$ across all pools is given by the sum of the individual gains she has in each pool, $U_p(\boldsymbol{x}_p) = U_p(x_{p,1}, \ldots, x_{p,k}) = \sum_{k\in K} U_{p,k}(x_{p,k})$, where $\boldsymbol{x}_p = (x_{p,k})_{k\in K}$ is the vector of frequencies that p gets for all the pools. The utility functions are *private* to the LOP; she is not willing to share them for competitiveness reasons, not even with the NOP. This has a few implications on the necessary approach to handle the problem.

The NOP, on the other hand, wishes to allocate to each LOP a frequency vector $\hat{\boldsymbol{x}}_p = \sum_{k\in K} \hat{x}_{p,k}$ such that the cumulative satisfaction of all the LOPs is maximized, while respecting all the edge capacity constraint. To achieve this, the NOP divides the whole railway infrastructure to the pools, using variables $f_k, k \in K$ that determine the proportion of the total capacity of the edges that is assigned to pool k. Hence, the NOP wishes to solve the following strictly convex optimization problem:

$$\begin{aligned} \max \quad & \sum_{p\in P} U_p(\boldsymbol{x}_p) = \sum_{p\in P}\sum_{k\in K} U_{p,k}\left(x_{p,k}\right) \\ s.t. \quad & \sum_{p\in P} R_{\ell,p}(k)\cdot x_{p,k} \leq c_\ell \cdot f_k,\ \forall(\ell,k) \in L\times K \\ & \sum_{k\in K} f_k \leq 1 \quad ; \quad \boldsymbol{x},\boldsymbol{f} \geq 0 \end{aligned} \tag{MSC-II}$$

Clearly, the NOP cannot solve this problem directly for (at least) two reasons: (i) the utility functions are unknown to him; (ii) the scale of the problem can be too large (as it is typical with railway networks) so that it can be solved efficiently via a centralized computation. The latter is particularly important when the whole system is already at some equilibrium state and then suddenly a (small, relative to the size of the whole problem) perturbation in the problem parameters occurs. Rather than having a whole new re-computation of the new optimal solution from scratch, it is particularly desirable that a dynamical scheme allows convergence to the new optimal solution, starting from this warm start (of the previously optimal solution). All the above reasons dictate searching for a different solution approach, that has to be as decentralized as possible.

We adopt the approach in [1] to design a mechanism that will be run by the NOP in order to solve the above problem. In particular, rather than having the NOP directly deciding for the frequencies of all the LOPs in each pool, we first let each LOP make her own bid for frequency in each pool. Then, the NOP considers the solution of a convex program which is similar, but not identical to (MSC-II) using a set of (strictly increasing, strictly concave) *pseudo-utilities.* Our goal is to exploit the rational (competitive) behavior of the LOPs, in order to assure that eventually the optimal solution reached for this new program is identical to that of (MSC-II), as required.

In particular, each LOP $p \in P$ announces (non-negative) bids $w_{p,k} \geq 0$ (one per pool), which she is committed to spend for acquiring frequencies in the pools. Then, the NOP replaces the unknown utility functions with the pseudo-utilities $w_{p,k} \log(x_{p,k})$ in order to determine a frequency vector that maximizes the aggregate level of pseudo-satisfaction. Observe that these used pseudo-utilities are also strictly increasing, strictly concave functions of the LOPs' frequencies. This means that NOP wishes to solve the following (strictly convex) optimization problem that is completely known to him:

$$\begin{array}{lll}
\max & \displaystyle\sum_{p\in P}\sum_{k\in K} w_{p,k}\log(x_{p,k}) & \\
s.t. & \displaystyle\sum_{p\in P} R_{\ell,p}(k)\cdot x_{p,k} \leq c_\ell \cdot f_k,\ \forall(\ell,k)\in L\times K & \text{(MNET-II)} \\
& \displaystyle\sum_{k\in K} f_k \leq 1 \quad ; \quad \boldsymbol{x},\boldsymbol{f} \geq \mathbf{0} &
\end{array}$$

This problem can of course be solved in polynomial time, given the bid vector of the LOPs $\boldsymbol{w} = (w_{p,k})_{(p,k)\in P\times K}$, and let $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{f}})$ be its optimal solution. From the KKT-conditions of this program it follows that at optimality the NOP must assign frequency $\bar{x}_{p,k} = \frac{w_{p,k}}{\bar{\mu}_{p,k}}$, where $\bar{\mu}_{p,k}$ is the aggregation of Lagrange dual values $\bar{\Lambda}_{p,k}$ along the path requested by p in pool k, and is interpreted as the (path) per-unit price $\bar{\mu}_{p,k}$ for acquiring frequency $\bar{x}_{p,k}$ at a total cost of $w_{p,k}$. Now, $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{f}})$ is the optimal solution for any bid vector declared by the LOPs, and in particular it also holds for the true bid vector that the LOPs would really wish to afford. Also from the KKT-conditions of (MSC-II) and (MNET-II), we can easily observe that they would be identical iff $U'_{p,k}(\bar{x}_{p,k}) = \frac{w_{p,k}}{\bar{x}_{p,k}}$. Our next step is to somehow assure that this is indeed the case. To this direction, we exploit the rational behavior of the LOPs: Each LOP wishes to maximize her own aggregate level of satisfaction, therefore, she would declare a bid vector that would actually achieve this.

In what follows, we assume that the LOPs are *price takers* meaning that each of them considers the prices announced by the NOP as *constants*, with no hope of affecting them by their own bid vector. This property is important in the following analysis, and is realistic when there exist many LOPs, each controlling only negligible fractions of the total flow (or bidding process) in the system. The following theorem (whose proof can be found in [2]) guarantees the existence of a mechanism for this problem.

Theorem 1. *Given a transportation network $G = (V, L)$, a set of line pools K and a set P of selfish, price-taking LOPs, each having a private utility function for each pool with parameter the frequency that is allocated to her in the particular pool, there is a mechanism (a pair of a frequency allocation mechanism and a resource pricing scheme) that computes in polynomial time the optimal solution of the sum of the utility functions of the players, while respecting the capacities of the edges.*

This polynomially tractable mechanism, based on the solvability of (MNET-II), is totally centralized and rather inconvenient for a dynamically changing (over time), large-scale railway system. The following lemma (whose proof can be found in [2]) is crucial in deriving a dynamic system for solving (MSC-II).

Lemma 1. *For any (fixed) vector $\boldsymbol{f}$ of capacity proportions that completely divides the railway infrastructure among the pools, the optimal value of* (MSC-II) *exclusively depends on the optimal vector $\bar{\boldsymbol{\Lambda}}$ of the per-unit-of-frequency prices of the resources.*

The above lemma suggests the following mechanism.

1. For every line pool $k \in K$, solve an instance of the single-pool case, using the decentralized mechanism in [1], obtaining the optimal solution $(\boldsymbol{x}_{\star,k}, \boldsymbol{\Lambda}_{\star,k})$.
2. The NOP calculates the cost of each pool and sets the variable $\zeta(t)$ to the average pool cost: $\zeta(t) = \frac{1}{|K|}\sum_{k\in K} \boldsymbol{c}^T \cdot \boldsymbol{\Lambda}_{\star,k}(t)$. Then, he updates the capacity proportion vector $\boldsymbol{f}$ and assigns a larger percentage of the total capacity to the most "expensive" line pools, so that their cost decreases. This update is described by the following differential equations:
$$\forall k \in K,\ \dot{f}_k(t) = \max\{0, \boldsymbol{c}^T \cdot \boldsymbol{\Lambda}_{\star,k}(t) - \zeta(t)\}. \qquad (1)$$
Note that, at the end, the vector $\boldsymbol{f}$ must be normalized, such that $\sum_{k\in K} f_k = 1$ (the proportion vector must completely divide the infrastructure at all times). This is done by dividing each $f_k(t)$ by $\sum_{k\in K} f_k(t)$.

Roughly speaking, the convergence of the above mechanism for a specific capacity proportion vector $\boldsymbol{f}$ is guaranteed by the convergence of the single-pool algorithm. When the $|K|$ single-pool instances are solved, the NOP updates the vector $\boldsymbol{f}$, so that the expensive pools get cheaper. The goal is that all pools should have the same cost. When this happens, we know for the optimal solution of both (MNET-II) and (MSC-II) $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{f}})$ and the accompanying Lagrange multipliers, $(\bar{\boldsymbol{\Lambda}}, \bar{\zeta})$, that:

- $U'_{p,k}(\bar{x}_{p.k}) = \frac{\bar{w}_{p,k}}{\bar{x}_{p,k}}$, due to the fact that each LOP computes its bid $\bar{w}_{p,k}$ by solving the convex optimization problem $\{\max \sum_{k \in K}(U_{p,k}(\bar{x}_{p,k}) - w_{p,k}); w_{p,k} \geq 0, \forall k \in K\}$.
- All the remaining KKT conditions, which are identical for the KKT systems of (MSC-II) and (MNET-II), are satisfied in the limit, due to the proper choice of NOP's updating scheme for the vector $\boldsymbol{f}$ allocating the infrastructure's capacity to the pools. More details can be found in [2].

Hence, $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{f}})$ is the optimal solution of both (MSC-II) and (MNET-II), and thus the proposed mechanism solves (MSC-II). The next theorem summarizes the preceding discussion.

Theorem 2. *The above dynamic scheme of resource pricing, LOPs' bid updating and capacity proportion updating assures the monotonic convergence of the* (MNET-II) *problem to the optimal solution. The algorithm may start from any initial state of resource prices, LOPs' bids and capacity proportion vector.*

3 Experimental Study of the Multiple-Line Pool Cases

In this section we present the experimental results for the multiple-line pool case where the LOPs have different utilities per pool. We have implemented a discrete version of the decentralized mechanism, whose pseudocode follows.

```
$f_k(0) = \frac{1}{|K|}$;
repeat
  $t = t + 1$;
  for all $k \in K$ do
    Solve an instance of the single-pool case for each line pool $k$;
  end for
  $\mathrm{cost}_k(t) = \boldsymbol{c}^T \cdot \boldsymbol{\Lambda}_{\star,k}(t)$;
  $\zeta = \frac{\sum_{k \in K} \mathrm{cost}_k}{|K|}$;
  for all $k \in K$ do
    $\dot{f}_k(t) = \frac{\max\{0, \mathrm{cost}_k(t) - \zeta(t)\}}{\zeta(t)}$;
    $f_k(t) = f_k(t-1) + 0.1 \cdot \dot{f}_k(t)$;
  end for
  $\mathrm{total\_f} = \sum_{k \in K} f_k(t)$;
  for all $k \in K$ do
    $f_k(t) = \frac{f_k(t)}{\mathrm{total\_f}}$;
  end for
until equal_costs($\mathbf{cost}(t)$)
```

The algorithm was implemented in C++ using the GNU g++ compiler (version 4.4) with the second optimization level (-O2 switch) on. Experiments were performed on synthetic and real-world data.

Synthetic data consisted of grid graphs having a number of 7 nodes on the vertical axis and a number of nodes in $[120, 360]$ along the horizontal axis; i.e., the size of the grid graphs varied from 7×120 to 7×360. The capacity of each edge was randomly chosen from $[10, 110)$. Four line pools were defined. In each pool, there were three LOPs, each one interested in a different line. Those lines had the first edge $((0, 3), (1, 3))$ in common. The next edges of each line were randomly chosen each time.

Real-world data concern parts of the German railway network (mainly intercity train connections), denoted as R1 (280 nodes and 354 edges) and R2 (296 nodes and 393 edges). The capacities of the edges were in $[8, 16]$. The total number of lines varies from 100 up to 1000, depending on the size of the networks. For each network, we defined four line pools. The second, third and fourth pool differed from the first in about 10% of the lines (the new lines in each pool were randomly selected from the available lines in each network).

In the experiments we measured the number of iterations needed to find the correct vector $\boldsymbol{f}$ of capacity proportions (we did not concentrate on the solutions of the single-pool case, used as a subroutine, since this case was investigated in [1]). We investigated the following four scenarios:

S1: $U_{p,1}(x_{p,1}) = 10^4\sqrt{x_{p,1}}$ and $U_{p,2}(x_{p,2}) = 10^4\sqrt{x_{p,2}}$, $\forall p \in P$.
S2: $U_{p,1}(x_{p,1}) = \frac{3}{4} \cdot 10^4 \cdot \sqrt{x_{p,1}}$ and $U_{p,2}(x_{p,2}) = \frac{4}{5} \cdot 10^4 \cdot \sqrt{x_{p,2}}$, $\forall p \in P$.
S3: $U_{p,1}(x_{p,1}) = 10^4 \cdot \sqrt{x_{p,1}}$ and $U_{p,2}(x_{p,2}) = \frac{1}{2} \cdot 10^4 \cdot \sqrt{x_{p,2}}$, $\forall p \in P$.
S4: $U_{p,1}(x_{p,1}) = 10^4 \cdot \sqrt{x_{p,1}}$ and $U_{p,2}(x_{p,2}) = \frac{1}{4} \cdot 10^4 \cdot \sqrt{x_{p,2}}$, $\forall p \in P$.

(a)

#Lines	S1	S2	S3	S4
100	9	33	127	178
200	12	33	127	178
300	19	29	128	178

(b)

#Lines	S1	S2
100	33	52
200	26	49
300	1	40
400	6	34
500	1	37

Table 1. Number of updates of $\boldsymbol{f}$ for different utility functions and number of lines per pool ($|K| = 2$) for R1 (a) and R2 (b).

We report on experiments with the R1 network and two line pools for all four scenarios, and on R2 for scenarios S1 and S2 (similar results hold for the other scenarios). Table 1 shows the results for 100, 200 and 300 lines per pool for R1, and for 100 to 500 lines per pool for R2. For S1 (same utility functions), we observe a small number of necessary updates to the capacity proportion vector

$\boldsymbol{f}$, until the system reaches the optimum. The main reason for this is the use of the same utility function for every pool by the LOPs, because the algorithm starts with the initial values $f_k = \frac{1}{|K|}$ and the optimal values in this case are quite close to these initial values. For the other scenarios with different utility functions per pool (S2, S3, S4), we observe a larger number of the updates required. We also observe that the more different the utility functions of each LOP in the two pools are, the larger the number of updates required to reach the optimum.

Another interesting observation in the case of the different utility functions per line pool, is that the number of updates of $\boldsymbol{f}$ is almost equal. This is due to the fact that the difference in utility functions across line pools has a more significant effect on the required number of updates than the difference in lines among the pools (in other words, more steps are required to reach the optimal values due to the different utility functions than due to the different costs of the line pools).

In conclusion, the number of updates required by our mechanism to converge (to the optimal values of vector $\boldsymbol{f}$) depends largely on the exact parameters of the system of differential equations (1).

4 Experimental Study of Disruptions in the Network

We turn now to a different experimental study. We assume that the network is currently operating at optimality and that a few disruptions occur. These disruptions affect the capacity of some edges. This can be due to technical problems leading to reducing the capacity of those edges, or to increasing their capacity for a particular period to handle increased traffic demand (e.g., during holidays, or rush hours) by "releasing" more infrastructure.

We examine the behavior of the algorithms for the single and multiple pool cases in such situations. We investigated three disruption scenarios:

- D1: Reducing the capacity of a certain number of edges (chosen among the congested ones).
- D2: Increasing the capacity of a certain number of edges (chosen among the congested ones).
- D3: Reducing the capacity of a certain number of edges, while increasing the capacity of an equal number of a different set of edges (chosen among the congested ones).

We start from a known optimal solution to the problem. Then, we add disruptions to a few edges and apply the algorithm. The relative and absolute error for the differential equations were set to 0.1.

These scenarios were tested on grid graphs and on the R1 network (similar results hold for R2). For the grid graphs, the lines were chosen randomly, but all of them shared the same first edge. The number of lines in each pool were 10 and the capacities of the edges were chosen randomly in $[4, 20]$.

(a)

Disruption	p	D1	D2	D3
10%	120	1292	340	9983
	180	1235	395	550
	240	317	453	407
	300	4005	556	1337
	360	163	8484	542
50%	120	403	480	1022
	180	248	1116	875
	240	409	498	533
	300	3966	1284	1180
	360	751	658	712

(b)

Disruption	#lines	D1	D2	D3
10%	100	10335	90085	464
	200	32466	2806	5033
	300	4171	276	5208
50%	100	8409	1057	1506
	200	1042	1109	4314
	300	5430	974	1058

Table 2. (a) Required number of updates of $\boldsymbol{\Lambda}$ for grid graphs with sizes $7 \times p$, when the algorithm starts from a previous optimal state, until the system reaches the equilibrium point after the disruptions under scenarios D1, D2, and D3. (b) Required number of updates of $\boldsymbol{\Lambda}$ for R1, when the algorithm starts from a previous optimal state, until the system reaches the equilibrium point after the disruptions under scenarios D1, D2, and D3.

Single pool case. For this case, we chose randomly, among the congested ones, 4 edges in the case of grid graphs and 10 edges in the case of R1. Their capacity was reduced (or increased) by 10% and 50%. In the experiments we measured the number of updates required for finding the optimal values of $\boldsymbol{\Lambda}$ (resource prices per-unit-of-frequency).

The number of iterations required for finding the optimal values of $\boldsymbol{\Lambda}$ for grid graphs and R1, when we start from a previous optimal solution, is presented in Tables 2(a) and 2(b). For comparison, the number of the required updates of $\boldsymbol{\Lambda}$ when we start from a random initial state is given in Tables 3(a) and 3(b). We observe the significantly less number of updates required when we start from a previous optimal solution. This is due to the fact that the disruptions caused are not very big, and hence the new optimal solution is quite close to the previous one. There are, however, one or two exceptions; i.e., we observe in these cases a smaller number of updates when we start from a random initial solution. This happens, because the algorithms for solving differential equations are arithmetic methods that depend greatly on the exact parameters given. This results in a few pathological cases such as these. One can conclude, though, that in general the use of the previous optimal solution leads to a smaller number of required updates for $\boldsymbol{\Lambda}$.

Multiple pool Case. We created two pools for these experiments. In the case of grid graphs, the lines in each pool were chosen randomly, and in the case of R1 there was a 10% difference in the lines between the two pools. In these experiments we measured the number of updates of the bids of the LOPs (bid

(a)

Case of Disruption	p	#Updates of $\boldsymbol{\Lambda}$
10%	120	6701
	180	6643
	240	7835
	300	6813
	360	5854
50%	120	7381
	180	7246
	240	6468
	300	6197
	360	7617

(b)

#Lines	#Updates of $\boldsymbol{\Lambda}$
100	12393
200	6641
300	7817

Table 3. (a) Number of updates of $\boldsymbol{\Lambda}$ for grid graphs of size $7 \times p$, when the algorithm starts from a random initial state. (b) Number of updates of $\boldsymbol{\Lambda}$ for R1, when the algorithm starts from a random initial state.

vector $\boldsymbol{w}$). In none case there was a need to update the capacity proportion vector $\boldsymbol{f}$.

The results are shown in Tables 4(a) and 4(b). One can see that only rarely there is a need to update the bid vector $\boldsymbol{w}$. Especially for the R1 network, we had to introduce disruptions of 90% of the original capacity to get the bid vector to be updated. Hence, the algorithm reaches the optimal solution quite fast. The important observation is that, starting from the previous optimal solution, we avoid the update of the capacity proportion vector $\boldsymbol{f}$, which is the most expensive operation.

5 Conclusions

We have studied a variant of the robust multiple-pool line planning problem defined in [1], where the LOPs have different utility functions per pool. We have shown that a dynamic, decentralized mechanism exists for this problem that eventually converges to the optimal solution.

We have also studied the above mechanism experimentally, showing that the exact behavior of the algorithm greatly depends on the exact input parameters; however, the convergence is in general quite fast.

Moreover, we studied the case that disruptions take place in the network. We have seen that in most cases it is much better to take advantage of the previous (optimal) solution to bootstrap the algorithm.

(a)

Disruptions	p	D1	D2	D3
10%	120	0	0	0
	180	0	0	0
	240	0	0	0
	300	0	0	0
	360	0	0	0
50%	120	0	2	1
	180	0	2	0
	240	0	0	0
	300	0	1	2
	360	0	2	2

(b)

Disruption	#Lines	D1	D2	D3
10%	100	0	0	0
	200	0	0	0
	300	0	0	0
50%	100	0	0	0
	200	0	0	0
	300	0	0	0
90%	100	0	3	0
	200	0	2	2
	300	0	0	0

Table 4. (a) Required number of updates of $\boldsymbol{w}$ for grid graphs of sizes $7 \times p$ for scenarios D1, D2, D3, when the algorithm starts from a previous optimal state, so that the system returns to an equilibrium point after a disruption. (b) Required number of updates of $\boldsymbol{w}$ for R1 for scenarios D1, D2, D3, when the algorithm starts from a previous optimal state, so that the system returns to an equilibrium point after a disruption.

References

1. A. Bessas, S. Kontogiannis, and C. Zaroliagis, “Incentive-Compatible Robust Line Planning”, In *Robust and Online Large-Scale Optimization*, Chapter 4, Springer 2009, pp. 85-118.
2. A. Bessas, S. Kontogiannis, and C. Zaroliagis, “Robust Line Planning in case of Multiple Pools and Disruptions”, `http://arxiv.org/abs/1101.2770`. January 2011.
3. H. Dienst, “Linienplanung im spurgeführten Personenverkehr mit Hilfe eines heuristischen Verfahrens”, PhD thesis, Technische Universität Braunschweig, 1978.
4. J. Goossens, C. van Hoesel, and L. Kroon, “A branch and cut approach for solving line planning problems”, *Transportation Science* 38 (2004), pp. 379393.
5. F. Kelly, “Charging and rate control for elastic traffic”, *European Transactions on Telecommunications*, 8 (1997), pp. 33-37.
6. F. Kelly, A. Maulloo, and D. Tan, “Rate control in communication networks: shadow prices, proportional fairness and stability”, *Journal of the Operational Research Society*, 49 (1998), pp. 237-252.
7. A. Schöbel and S. Scholl, “Line Planning with Minimal Traveling Time”, In *Proc. 5th Workshop on Algorithmic Methods and Models for Optimization of Railways* – ATMOS 2005.
8. A. Schöbel and S. Schwarze, “A Game-Theoretic Approach to Line Planning”, in *Proc. 6th Workshop on Algorithmic Methods and Models for Optimization of Railways* – ATMOS 2006.
9. S. Scholl, “Customer-oriented line planning”, PhD thesis, Technische Universität Kaiserslautern, 2005.