Technische Universität Berlin

FAKULTÄT II
MATHEMATIK UND
NATURWISSENSCHAFTEN
Institut für Mathematik

System-Optimal Routing of Traffic Flows with User Constraints in Networks with Congestion

by

Olaf Jahn Rolf H. Möhring
Andreas S. Schulz Nicolás E. Stier Moses

No. 754/2002

System-Optimal Routing of Traffic Flows with User Constraints in Networks with Congestion

Olaf Jahn[†] Rolf H. Möhring[‡] Andreas S. Schulz[*]
Nicolás E. Stier Moses[**]

November 2002

The design of route-guidance systems faces a well-known dilemma. The approach that theoretically yields the system-optimal traffic pattern may discriminate against some users, for the sake of favoring others. Proposed alternate models, however, do not directly address the system perspective and may result in inferior performance. We propose a novel model and corresponding algorithms to resolve this dilemma. We present computational results on real-world instances and compare the new approach with the well-established traffic assignment model. The quintessence is that system-optimal routing of traffic flow with explicit integration of user constraints leads to a better performance than the user equilibrium while simultaneously guaranteeing a superior fairness compared to the pure system optimum.

Keywords:	Intelligent Transportation Systems, Route Guidance, Traffic Flow, System Optimum, User Equilibrium, Multicommodity Flow, Constrained Shortest Path
Classification (AMS):	90C35; 90B20, 90C25, 90C27, 90C90

[†] Infopark, Berlin, Germany; olaf.jahn@zamioculcas.net
[‡] Technische Universität Berlin, Fakultät II, Institut für Mathematik, MA 6-1, Straße des 17. Juni 136, D-10623 Berlin, Germany; moehring@math.tu-berlin.de
[*] Massachusetts Institute of Technology, Sloan School of Management, E53-361, 77 Massachusetts Avenue, Cambridge, MA 02139-4307, USA; schulz@mit.edu
[**] Massachusetts Institute of Technology, Operations Research Center, E40-130, 77 Massachusetts Avenue, Cambridge, MA 02139-4307, USA; nstier@mit.edu

1. Introduction

More and more cars are equipped with so-called route guidance systems. Visual and acoustic indicators guide the driver to her destination, which she has entered at the beginning of the trip. Typically, these systems compute their routes based on digital maps, possibly up-to-date traffic data, which are broadcast by radio or cellular phone, and obtain the current position with the help of GPS (Global Positioning System) [23].

Current route guidance systems are simple from an algorithmic point of view. They only compute shortest paths (or approximations thereof) to the destination (with respect to travel time or geographic distance). Computational challenges for these approaches arise "solely" from the huge size of the underlying road networks, which is often addressed by using a hierarchical graph representation [12, 43]. One of the benefits of this approach is that the route computation can be perfectly parallelized, i.e., each vehicle can independently compute its own route. More advanced systems incorporate traffic forecast estimated from theoretical models and past experience into the computation (*anticipatory routing*) [9]. It is widely hoped that route guidance systems can help to alleviate the problem of congestion caused by the still increasing amount of road traffic.

1.1. Drawbacks of current route guidance systems

A number of simulations show that route guidance systems indeed decrease the total travel time since inefficiencies caused by human route choice are mostly eliminated [24, 29, 30]. Yet, the majority of these simulations also predict that these benefits will be lost once the number of equipped vehicles exceeds a certain threshold (assuming that the drivers actually follow the routes they are given).[1] In some cases it could even happen that unguided drivers obtain shorter travel times than those with guidance [30].

This phenomenon can be explained by the fact that the perceivable decrease of the overall travel time in the case of a small fraction of equipped vehicles is merely a fortunate byproduct of the system. Current algorithms try to minimize the *individual* journey time of each driver separately, without taking into account the effects of their own route recommendation. Possibly used traffic forecasts treat the influence of guided vehicles as negligible. It is possible that such a system actually causes congestion all by itself. Consider for example a number of vehicles starting at the same point in time in origin A for destination B. Under current systems, they are given exactly the same route recommendation. Even if this route is updated over time as new traffic data become available, it is still the same route for all these vehicles. Therefore, assuming similar mean speed, they always stay together on their trip (*platooning*), which possibly leads to congestion. While Adler and Blue [1] call this phenomenon *oversaturation*, Ben-Akiva, de Palma and Kaysi [10] call it *overreaction*, since "too many" drivers follow the recommended route.

[1] Not all simulations agree with this prediction. Most notably, Hall [22] points to some deficiencies in the models underlying these simulations. He argues that if *accurate* traffic information is used for routing, the increase in road usage will not occur.

Thus, the need for integrated algorithms that actually pay attention to the system-wide performance (which is typically defined as the sum of all individual travel times) has been recognized [6, 23, 26]. Obviously, it is essential that these algorithms are capable of splitting platoons over several paths (*multiple path routing*).

A number of approaches have been proposed to achieve multiple path routing. For instance, Rilett and Van Aerde [36] suggest adding individual random error terms to the road travel times broadcast by a central controller, in order to cause the in-vehicle computers to choose different paths. Lee [29] computes k-shortest paths every 10 minutes and then distributes drivers over them every two minutes, considering the current travel times on these k-shortest paths. These authors report a decrease in the total travel time (compared with single path routing) from their simulation studies, but it is obvious that this decrease is not guaranteed. Furthermore, other solutions are likely to reduce the total travel time even further.

Another popular approach is to route drivers along the paths of a so-called *user equilibrium*, so that no driver can get a quicker path through the network by unilaterally changing her route [41]. The resulting model is called *traffic assignment*. It was originally introduced by Wardrop [42] in order to model natural driver behavior, and it has been studied extensively in the literature. In fact, transportation engineers have used static traffic assignment — traffic is assumed to be in steady state and flow does not change over time — to predict traffic patterns. Sheffi [39] and Patriksson [34] provide a comprehensive treatment of mathematical formulations and algorithms for the static user equilibrium.

While a user equilibrium should satisfy the drivers, it does not necessarily minimize the total travel time of the system. Roughgarden and Tardos [38] investigate the relation between the system optimum and the user equilibrium. In general, the total travel time in equilibrium can be arbitrarily larger compared to the system optimum, although it is never more than the travel time incurred by optimally routing twice as much traffic.

Another unfavorable property of the user equilibrium is its non-monotonicity with respect to the network's capacity. This is illustrated by the *Braess paradox*, where adding a new road to a network with fixed demands actually increases the total travel time of the updated user equilibrium [11, 21, 39].

To overcome one of the unrealistic assumptions of static traffic assignment [8, 18], there has been significant effort towards the dynamic analysis of traffic networks. Unlike static traffic assignment, where the modeling and the solution methods are well established, the *dynamic traffic assignment problem* has been studied from several different perspectives with no single generally accepted model or methodology. The article by Mahmassani and Peeta [31] provides a discussion of the inherent difficulties and corresponding solution attempts.

An intrinsic difficulty in all models is that the path computations depend on the travel times in arcs, which in turn depend on the traffic on these paths. As this relationship is not easy to grasp analytically in the dynamic case, some techniques proposed in the literature for traffic assignment (sometimes even for the static case) use simple feedback loops. They iteratively assign traffic to the network (e.g., by simulation) and compute new arc travel

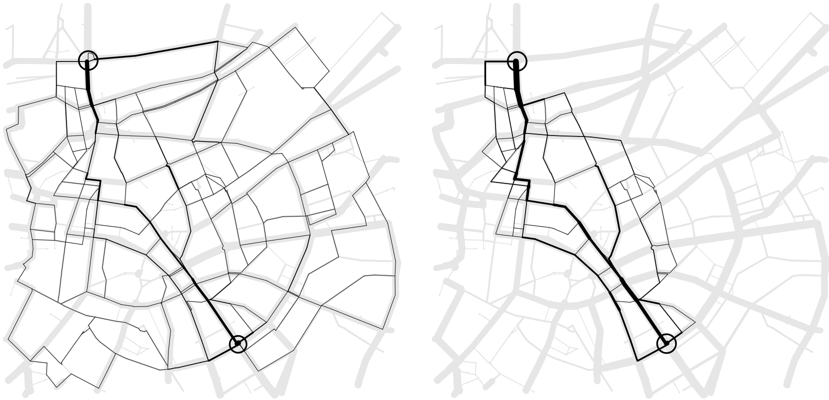


Figure 1: Example without and with restrictions on path lengths: One commodity is routed through the road network between the marked nodes. In the picture on the left, we display the (unconstrained) system optimum. The flow is distributed widely over the network in order to avoid high arc flows that would incur high arc travel times. In the picture on the right, the same amount of flow is routed, but this time with a restriction on the normal length of paths (constrained system optimum). In this example, the normal length has been chosen to be the geographic distance. Line thickness reflects arc capacity (light gray) and arc usage (black).

times based on the last assignment (and perhaps previous ones) [24, 39]. Similar ideas are commonly used in loops of other algorithms as well; for instance, Bell et al. [7] use outdated arc travel times in each iteration of their Frank-Wolfe-type algorithm.

These heuristic approaches are mathematically somewhat unsatisfactory because often it is not clear what the sequence of intermediate solutions actually converges to, or if it even converges at all. Usually "convergence" is established only empirically. Kaufman, Smith and Wunderlich [24] report cycling of their algorithm for certain instances. In [25], they report that this is especially due to single path routing (*all-or-nothing assignment*) used in every iteration. In order to overcome this problem and guarantee convergence, they have to develop a complicated continuous-time model with path splitting, which does not seem to be practical.

1.2. A different approach

From a global perspective, e.g., the traffic authority's point of view, it is certainly desirable to explicitly minimize the total travel time (i.e., to compute a *system optimum*). In particular, the existing road network could henceforth carry more traffic [16, 28]. Yet, users' needs have to be taken into account: Directly implemented, this policy could route some drivers on unacceptably long paths in order to use shorter paths for many other drivers. In fact, the length of a route in the system optimum can be significantly higher than in user equilibrium, even in the pathological case of a single origin-destination pair [37]. This is critical because routes can only be recommended to drivers. It is reasonable to assume that only very few of them would be willing to sacrifice their own short routes for the benefit of others. On the other hand, user acceptance of a route guidance system is important if it is supposed to help in reducing traffic congestion. Therefore, Beccaria and Bolelli [6] have suggested to "find the route guidance strategy which minimizes some global and community criteria with individual needs as constraints." To the best of the authors' knowledge, we provide the first model that integrates these conflicting goals.

We adopt a system optimum approach but honor the individual needs by imposing additional constraints to ensure that drivers are assigned to "acceptable" paths only. More precisely, we introduce the concept of the *normal length* of a path, which can be either its traversal time in the uncongested network or its traversal time in user equilibrium or its geographic distance, or any other appropriate measure. The only condition imposed on the normal length of a path is that it may not depend on the actual flow on the path. Equipped with this definition, we look for a *constrained system optimum* in which no path carrying positive flow between a certain origin-destination (OD) pair is allowed to exceed the normal length of a shortest path between the same OD-pair by more than an "acceptable" factor. Figure 1 demonstrates the effect of length constraints on the system optimum.

Because of the mentioned difficulties with dynamic traffic modeling, in this paper we apply this approach to the static routing problem. After specifying the problem and the used model in Section 2, we present an algorithm for its solution in Section 3. It combines the Frank-Wolfe method with column generation and an algorithm for the constrained shortest path problem. In Section 4, we give computational results obtained with our implementation of this algorithm. The real world input data were kindly provided by the DaimlerChrysler AG, Berlin.

2. Model and Problem Formulation

We consider a static model in which flow represents a traffic pattern at steady state. While working with static flows precludes the direct application of our algorithm to all real-world situations, which are generally of dynamic nature, our model is nonetheless useful for the study of time periods when traffic exhibits a flow-like behavior, e.g., during rush hours [39]. Moreover, results obtained from the static model can provide traffic planners with important bounds on the total travel time, which, due to the imposed restrictions on the normal length of paths, are more realistic than the bounds resulting from both the system optimum and the user equilibrium.

We also assume that all drivers use the route-guidance system and that they actually follow the recommended routes. The former assumption is relatively strong, but unguided drivers can easily be incorporated as a fixed congestion on the network, which would imply, however, that they do not react to the route choices of guided vehicles.

Formulations of routing problems are often arc-based with one flow variable per arc. Since a route guidance system eventually has to propose paths to the drivers, arc flows

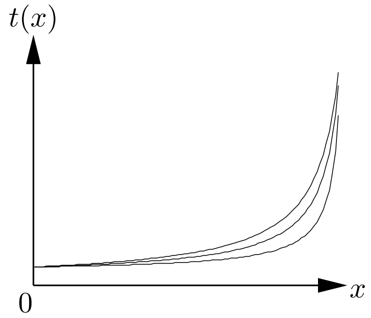


Figure 2: Typical link delay functions; x is the arc flow, and $t(x)$ is the travel time.

have to be decomposed in a postprocessing step into path flows. As this decomposition is in general not unique, it is virtually impossible to model path constraints in an arc-based formulation. Therefore, we use a path-based problem formulation with one variable for each feasible path. This enables us to impose restrictions on paths very easily. As we cannot enumerate all these variables efficiently, we only generate them when they are needed to carry flow.

Let the road network be given by a directed multigraph $G = (V, A)$ with three attributes on each arc $a \in A$: the capacity $u_a \geqslant 0$ (measured in vehicles per time unit), the link delay function $t_a \colon [0, u_a] \to \mathbb{R}_0^+$ denoting the arc travel time depending on the traffic flow on this arc, and the normal length $\ell_a \geqslant 0$ of this arc. We require t_a to be monotonically increasing, twice continuously differentiable, and convex. These requirements are naturally met by common link delay functions [14, 39] (Figure 2). We use the function of the U.S. Bureau of Public Roads,

$$t_a(x_a) := t_a^0 \left(1 + \alpha \left(\frac{x_a}{c_a} \right)^\beta \right),$$

with the travel time $t_a^0 > 0$ in the uncongested network and tuning parameters $\alpha \geqslant 0$, $\beta \geqslant 0$ and $c_a > 0$. As suggested in [39], we set $\alpha = 0.15$ and $\beta = 4.0$. The parameter c_a is called "the practical capacity" of arc a. We use a value slightly greater than the real capacity u_a of arc $a \in A$. Other functions, like Davidson's function [39], could have been used as well.

Vehicles with the same origin and destination are modeled as one commodity. For each commodity k, let $(s_k, t_k) \in V \times V$, $s_k \neq t_k$, denote the pair of origin node and destination node, and let $b_k > 0$ be the amount of flow to be routed for k (vehicles per time unit). In addition, we want the normal length of all paths assigned to k not to exceed a given value $L_k > 0$. We describe the proper choice of the metric underlying the normal length together with the choice of values for the parameters L_k in Section 4. The set of all commodities is called C. We define path sets

$$P_k := \{p \mid p \text{ is a directed path from } s_k \text{ to } t_k\},$$
$$P'_k := \{p \in P_k \mid \ell(p) \leqslant L_k\},$$

with the *normal path length*

$$\ell(p) := \sum_{a \in p} \ell_a,$$

and the set of all feasible paths

$$P' := \bigcup_{k \in C} P'_k.$$

Let $\Phi \in \{0,1\}^{|A| \times |P'|}$ be the arc-path incidence matrix such that the vectors x^A and x^P of arc and path flows are related by $x^A = \Phi x^P$. Similarly, let $\Psi \in \{0,1\}^{|C| \times |P'|}$ be the commodity-path incidence matrix, where the entry in row k and column p is 1 if $p \in P'_k$ and 0, otherwise.

We want to minimize the total travel time, which can be stated as the sum of the lengths of all paths, weighted with the flow on each path. With the vectors $t^A(x^A) \in \mathbb{R}^A$, $b \in \mathbb{R}^C$ and $u \in \mathbb{R}^A$ respectively composed of the arc travel times $t_a(x_a)$, the demand rates b_k and the capacities u_a, the problem can be written as

$$\begin{aligned} \mathcal{P}\colon \quad & \min \quad z^P(x^P) := t^A(\Phi x^P)^\intercal \Phi x^P \\ & \text{s.t.} \\ & \Psi x^P = b \\ & \Phi x^P \leqslant u \\ & \ \ x^P \geqslant 0\ . \end{aligned} \tag{1}$$

Problem $\mathcal{P}$ is a minimum cost multicommodity flow problem with path constraints and a separable convex nonlinear objective function. Of course, the set P' of all feasible paths is given only implicitly (by the L_k, $k \in C$); thus, $\mathcal{P}$ is a generalization of the well known constrained shortest path problem (see also Section 3.2), which renders it NP-hard. Note that the flow variables x^P are not required to be integral since they describe abstract traffic flows.

3. Algorithms

The algorithm we propose for solving $\mathcal{P}$ is an adaption of the Frank-Wolfe algorithm [17], which is applicable since the gradient of the objective function can be expressed in terms

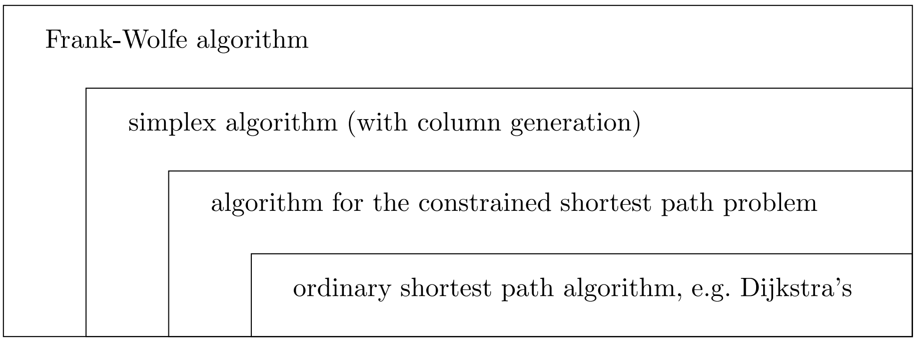


Figure 3: The nested structure of the algorithm.

of arcs flows, as we show in Section 3.3. The algorithm solves a linearized version of $\mathcal{P}$ in every iteration. The subroutine for the linear case in turn repeatedly calls an algorithm for solving the constrained shortest path problem. Figure 3 provides an overview of the nested structure of the entire algorithm, which always converges to a global minimum.

3.1. The linear problem

Let us first consider $\mathcal{P}$ with constant link delay functions $t_a \geqslant 0$. Then $\mathcal{P}$ becomes

$$\begin{aligned} \mathcal{P}_{\text{lin}}\colon\quad & \min \quad (t^A)^\intercal \Phi x^P \\ & \text{s.t.} \\ & \Psi x^P = b \\ & \Phi x^P \leqslant u \\ & x^P \geqslant 0 \end{aligned} \tag{2}$$

with $x^P \in \mathbb{R}^{P'}$. Again, P' is given only implicitly by means of the L_k. Therefore, $\mathcal{P}_{\text{lin}}$ is a linear program with additional constraints for the paths. This still renders it NP-hard in the original input size. Since we have got one variable for each feasible path, it is easy to ensure that all paths fulfill the normal length restriction (or other restrictions we might think of), as we show below. It is not practical to hold all variables in memory at the same time, as there could be exponentially many. Therefore, we use the simplex algorithm with *column generation* to generate new paths when we need to route flow through them. While column generation is a commonly used technique to solve linear multicommodity flow problems with path constraints (see, e.g., [15] for a survey of applications in routing and scheduling), we outline it here for the sake of completeness.

Note first that at most $|C|+|A|$ variables have nonzero value in a basic feasible solution to $\mathcal{P}_{\text{lin}}$. The key idea is therefore to restrict $\mathcal{P}_{\text{lin}}$ to a small subset of variables with index set $\bar{P}' \subset P'$, where $|\bar{P}'| > |C|+|A|$, which contains the current basis and at least one additional

Input: An instance of $\mathcal{P}_{\text{lin}}$ and a feasible basis B
Output: An optimal solution of this instance

$\bar{P}' := B \cup \{$at least one additional variable from $P'\}$
loop
  Solve $\mathcal{P}_{\text{lin}}$ restricted to $\bar{P}'$.
  Let σ_k and π_a be the simplex multipliers of the found optimal solution.
  for all $k \in C$
    find shortest path p_k in P'_k w.r.t. the arc costs $t_a - \pi_a$.
  end for
  if $t(p_k) \geqslant \sigma_k$ for all $k \in C$ **then**
    return values of basic variables
  else
    Remove one or more non-basic variables from $\bar{P}'$.
    Choose at least one path p_k with $t(p_k) < \sigma_k$ and add it to $\bar{P}'$.
  end if
end loop

Figure 4: The revised simplex algorithm with column generation.

variable, and swap variables in and out as needed. If one solves this so-called *restricted master problem* $\mathcal{P}_{\text{lin}}$ over $\bar{P}'$ to optimality employing the revised simplex algorithm, it returns values for the path flows x_p, $p \in \bar{P}'$, as well as optimal values for the dual variables σ_k, $k \in C$, and π_a, $a \in A$, associated with the equality and inequality constraints of $\mathcal{P}_{\text{lin}}$. It follows from linear programming theory that the solution x^P, with $x_p = 0$ for all $p \in P' \backslash \bar{P}'$, is also optimal for the *whole* problem over P' if and only if the following two conditions hold:

$$\sigma^\mathsf{T}\Psi + \pi^\mathsf{T}\Phi \leqslant (t^A)^\mathsf{T}\Phi \tag{3}$$

$$\pi \leqslant 0 \tag{4}$$

Here, σ and π are the vectors of the σ_k and π_a, respectively. Since x^P is optimal for the restricted problem, all columns of (3) belonging to variables of $\bar{P}'$ satisfy this condition. Furthermore, π satisfies (4). It follows from the principle of the simplex algorithm that all variables belonging to columns that violate (3) are eligible for entry into the basis (and therefore into $\bar{P}'$), while non-basic variables in $\bar{P}'$ can be removed from $\bar{P}'$.

It remains to be discussed how we can check whether any variable not in $\bar{P}'$ violates (3) or not, without enumerating them all. A single column of (3), belonging to $p \in P'_k$, can be written as

$$\begin{aligned}
&\sigma_k + \sum_{a\in p} \pi_a \leqslant \sum_{a\in p} t_a \\
\Leftrightarrow \quad & t(p) := \sum_{a\in p} (t_a - \pi_a) \geqslant \sigma_k. \qquad (5)
\end{aligned}$$

This means that for every commodity $k \in C$, all paths in $p \in P'_k$ must have length $t(p)$ of at least σ_k with respect to the modified arc lengths $t_a - \pi_a \geqslant 0$. This in turn means that (3) can be checked by computing a shortest path in P' for every commodity. To be precise, it is generally not necessary to compute an actual shortest path. In the case that (5) is violated it suffices to obtain a path with $t(p) < \sigma_k$. The resulting algorithm is summarized in Figure 4. We obtain an initial solution by solving a so-called first phase with additional artificial variables and a modified objective function [13].

Since P' contains only the paths $p \in P_k$ satisfying the constraints $\ell(p) \leqslant L_k$ on their normal length, we have to solve a so-called *constrained shortest path problem* for every commodity. This problem is significantly harder than the ordinary shortest path problem and will be discussed in the next section.

It is obvious that we can impose other constraints on the available paths as long as there is an algorithm that can solve the associated constrained shortest path problem. Notice that the $|C|$ shortest path computations are completely independent and can be done in parallel: A coordinator solves the restricted master problem and then deals out the current arc costs $t_a - \pi_a$ for all $a \in A$ to a number of helpers, which compute the constrained shortest paths and report back their results.

If we forego the capacity constraints (1) and (2) in $\mathcal{P}$ and $\mathcal{P}_{\text{lin}}$, respectively, $\mathcal{P}_{\text{lin}}$ decomposes into $|C|$ independent constrained shortest paths problems. This renders the simplex algorithm unnecessary. In Section 4 we present computational results obtained with and without obeying the capacity constraints.

3.2. The constrained shortest path problem

In the constrained shortest path problem, every arc a of a given directed graph (V, A) carries two parameters, a traversal time t_a and a length ℓ_a. Given an origin-destination pair (s, t), the objective is to compute a quickest path from s to t whose length does not exceed a given bound L. With the notation introduced above, it can we written as

$$\begin{aligned}
\min \quad & t(p) \\
\text{s.t.} \quad & \\
& \ell(p) \leqslant L \\
& p \text{ is a path } s \rightsquigarrow t.
\end{aligned}$$

This problem is known to be (weakly) NP-hard [19].

We have implemented and tested two algorithms for this problem. The first algorithm follows a branch-and-bound scheme by Beasley and Christofides [5], which is quite similar to the one in Ribeiro and Minoux [35]. It employs depth-first-search from node s. Bounds for the remaining paths to t are computed using Lagrangian relaxation.

The second algorithm is a generalization of Dijkstra's algorithm to the case of two objective functions by Aneja, Aggarwal, and Nair [2]. It fans out from the start node s and labels each reached node $v \in V$ with labels of the form $(d_t(v), d_\ell(v))$. Such a label consists of the traversal time $d_t(v)$ and the distance $d_\ell(v)$, respectively, of the paths that have reached v so far. During the course of this algorithm several labels may have to be stored for each node, namely the Pareto-optimal labels of all paths that have reached it. This labeling algorithm can be interpreted as a special kind of branch-and-bound with a search strategy similar to breadth-first-search. Starting from a certain label of v, we obtain lower bounds for the remaining path from v to t by separately computing ordinary shortest path distances from v to t with respect to travel times t_a and lengths ℓ_a, respectively. If one of these bounds is too large, we can dismiss the label.

Both algorithms need to compute ordinary shortest paths many times. We use Dijkstra's algorithm for this task, since all arc weights are guaranteed to be nonnegative. Computational experience indicates that at least for our kind of problems the labeling algorithm is by far faster than the branch-and-bound method of Beasley and Christofides. This finding is also supported by a study on computing Pareto-optimal shortest paths in the context of finding good train connections with respect to multiple criteria [32].

3.3. The nonlinear problem

The original problem $\mathcal{P}$ as stated at the end of Section 2 is a nonlinear optimization problem with linear constraints over an implicitly given variable set. We solve it by using the convex combination algorithm by Frank and Wolfe [17] in the version described by Klessig [27]. It is a feasible direction method for problems with convex objective function and convex domain, especially suitable when the linearized problem to be solved in each iteration decomposes nicely, as is the case with $\mathcal{P}$. We summarize this algorithm for the reader's convenience.

Consider a minimization problem $\min\{z(x) \mid x \in \Omega\}$ with $z\colon \mathbb{R}^n \to \mathbb{R}$ convex and twice continuously differentiable, and $\Omega \subset \mathbb{R}^n$ convex. The feasible direction method is outlined in Figure 5. It can be shown that this algorithm terminates with an optimal solution, or that at least every accumulation point of the generated sequence is optimal if the computed step sizes guarantee a certain decrease of the objective value.

In order to compute the feasible direction in line 2 of Figure 5, we have to solve a linearized version of the problem. The step size θ_i in line 6 is best computed by bisection [4] as $\theta_i \approx \operatorname{argmin}\{z(x_i + \theta(y_i - x_i)) \mid \theta \in (0, 1]\}$. Our tests with the method of Armijo [3, 27]

Input: Convex problem $\min\{z(x) \mid x \in \Omega\}$ and an initial feasible solution $x_1 \in \Omega$.
Output: An optimal solution

---

for $i = 1, 2, \ldots$
  Compute feasible direction $y_i - x_i$ by $y_i := \operatorname{argmin}\{\nabla z(x_i)^\intercal (y - x_i) \mid y \in \Omega\}$.
  if $\nabla z(x_i)^\intercal (y_i - x_i) = 0$ **then**
    return x_i.
  end if
  Calculate step-size $\theta_i \in (0, 1]$ by line search.
  $x_{i+1} := x_i + \theta_i (y_i - x_i)$.
end for

Figure 5: The Frank-Wolfe algorithm.

generally produced worse results.[2] This method trades less objective function evaluations during a line search for worse descent of the objective value, compared with minimization by bisection (Figure 6). But since the time spent with step size computations either way is negligible in our algorithm, bisection should be preferred as it yields better objective values in the same number of iterations.

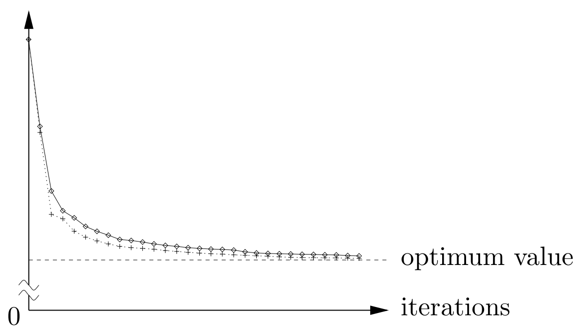


Figure 6: Characteristic descent of the objective value $z(x_i)$ during a run of the convex combinations algorithm. The lower curve is computed with line search done by bisection, the upper curve with the method of Armijo. The rapid descent in the first few iterations is typical.

Because of the convexity of the objective function z, $z(x') \geqslant z(x) + \nabla z(x)^\intercal (x' - x)$ holds for all $x, x' \in \Omega$. Therefore it is easy to see that we can obtain bounds in each iteration for

[2] The step size is computed as $\theta_i = \beta^{k_i}$ with

$$k_i := \operatorname{argmin}\{k \in \mathbb{N} \cup \{0\} \mid z(x_i + \beta^k (y_i - x_i))) \leqslant z(x_i) + \alpha \beta^k \nabla z(x_i)^\intercal (y_i - x_i)\}.$$

$\alpha, \beta \in (0, 1)$ are tuning parameters. Klessig suggests $\alpha = \beta = 0.5$.

its optimal value $z^* = z(x^*)$ by

$$\begin{aligned} z(x_i) \geqslant z^* \geqslant z(x_i) + \nabla z(x_i)^\mathsf{T}(x^* - x_i) \geqslant z(x_i) + \nabla z(x_i)^\mathsf{T}(y_i - x_i) \\ \Rightarrow \quad z(x_i) - z^* \leqslant |\nabla z(x_i)^\mathsf{T}(y_i - x_i)|. \end{aligned} \tag{6}$$

Here, y_i is the feasible direction computed in line 2 of the algorithm presented in Figure 5. Since the right-hand side is (at least partially) computed anyway in line 2, this gives us the value of the remaining gap at little additional cost. It enables us to terminate the algorithm when the desired precision is reached.

We now apply the algorithm to our original problem $\mathcal{P}$ with objective function

$$z^P(x^P) := t^A(\Phi x^P)^\mathsf{T}\Phi x^P$$

and feasible set

$$\Omega = \{x^P \in \mathbb{R}^{P'} \mid \Psi x^P = b, \Phi x^P \leqslant u, x^P \geqslant 0\}.$$

We encounter the difficulty that the gradient $\nabla z^P(x_i^P)$ needed for the feasible direction has got $|P'|$ entries, which are impractically many. Fortunately, if we express z^P in terms of arc flows $x^A = \Phi x^P$ as

$$z^A(x^A) := t^A(x^A)^\mathsf{T}x^A$$

we obtain

$$z^P(x^P) = z^A(\Phi x^P).$$

Here, $t^A(x^A)$ is by definition the vector of the travel time functions $t_a(x_a)$, $a \in A$. Since each $t_a(x_a)$ is convex and twice continuously differentiable, and Φ is a linear mapping, it is easy to prove that both z^P and z^A are convex and twice continuously differentiable. It follows from basic calculus that

$$\nabla z^P(x^P) = (\nabla z^A(\Phi x^P)\Phi)^\mathsf{T},$$

and with $y^P \in \mathbb{R}^{P'}$ that

$$\nabla z^P(x^P)^\mathsf{T} y^P = \nabla z^A(\Phi x^P)^\mathsf{T}\Phi y^P.$$

Consequently, we can solve the minimization problem for the feasible direction using only $\nabla z^A(\Phi x^P) \in \mathbb{R}^A$ in line 2 of Figure 5:

$$\begin{aligned} y_i &= \operatorname{argmin}\{\nabla z^P(x_i^P)^\mathsf{T}(y_i^P - x_i^P) \mid y^P \in \Omega\} \\ &= \operatorname{argmin}\{\nabla z^P(x_i^P)^\mathsf{T} y_i^P \mid y^P \in \Omega\} \\ &= \operatorname{argmin}\{\nabla z^A(\Phi x_i^P)^\mathsf{T}\Phi y_i^P \mid y^P \in \Omega\}, \end{aligned}$$

which is again a problem of the linear type $\mathcal{P}_{\text{lin}}$. In Section 2, we assumed the link delay functions $t_a(x_a)$ to be monotonically increasing. This means, because of $x^P \geqslant 0$, that the arc costs $\nabla z^A(\Phi x_i^P)$ of this linear problem are indeed nonnegative. Hence, we can use the simplex algorithm with column generation of Section 3.1 to determine an improving direction of the nonlinear problem.

Note that the direction y_i^P is only described by its components that actually carry flow. In the convex combinations step in line 7 we add two vectors represented in this manner. The result is again a (relatively) sparse vector, stored accordingly. Finally, an initial solution x_1^P needed by the algorithm can easily be found by solving a linear problem $\mathcal{P}_{\text{lin}}$ with arbitrary arc travel times $t_a > 0$.

4. Computational Study

The computational study is divided into three parts. First, we discuss which normal length should be used in practice. Next, we analyze efficiency vs. fairness of solutions to instances that arise from real-world networks. Finally, we report on the performance and convergence properties of the algorithm.

The instances used in this study represent different parts of the actual road network of the city of Berlin, Germany. Commodities and corresponding demand rates stem from origin-destination polls conducted in Berlin. Table 1 respectively shows the size of the network and the number of commodities for each instance.

name	$\|V\|$	$\|A\|$	$\|C\|$
frie	224	523	506
neuk	1890	4040	3166
3nei	975	2184	9801
berl	12100	19570	49689

Table 1: Problem instances used in the experiments. Columns represent the number of vertices, arcs and commodities, respectively.

Recall that our model requires a maximal normal length L_k to define the allowable paths for each OD-pair k. The value of L_k is computed by multiplying the length of a shortest path (in terms of normal arc lengths) between OD-pair k with a constant factor $f \geqslant 1$. Naturally, the bigger the factor is, the larger is the feasible region. Consequently, optimal solutions to model $\mathcal{P}$ are closer to (unconstrained) system optimality when f is larger. We denote by CSO_f the optimal solution to the problem with factor f.

There are two aspects that define the quality of a solution. The total travel time in the system is of importance to the traffic authority; it is directly measured in the objective function of problem $\mathcal{P}$. The second aspect is the users' perception of their route assignment; to evaluate it, we introduce three different notions of *unfairness* of a solution; each compares the travel times of different users in the system with each other. Ideally, one would prefer different users with the same OD-pair to have similar travel times.

For a given flow, we define the unfairness of a particular traveler as follows:

Normal unfairness: ratio of the length of her path to the length of the shortest path between the same OD-pair, both measured with respect to normal arc lengths.

Loaded unfairness: ratio of her travel time to the travel time of the fastest traveler for the same OD-pair in the optimal solution (measured in terms of the current flow).

User equilibrium (UE) unfairness: ratio of her travel time (w.r.t. the current flow) to the travel time for the same OD-pair in user equilibrium (which is the same for every user of that OD-pair).

It follows from these definitions that, for any flow, the normal unfairness and the loaded unfairness are always greater than or equal to 1; the UE unfairness can be any nonnegative number. Note also that the normal unfairness of CSO_f cannot be bigger than f, for any factor f. Moreover, in equilibrium the value of each unfairness measure is equal to 1, for every user. (For the normal unfairness, this is only true, of course, if the normal length of an arc is equal to its traversal time in user equilibrium.)

Of the three different notions, the loaded unfairness is most important because it measures the variation in the travel times of users with the same OD-pair in the solution. The UE unfairness, which was introduced in [37] in the single-commodity context, is of interest, too, because it indicates how the travel times of the solution relate to those in user equilibrium. In practice though, a user does typically not know the travel times in equilibrium; it is arguably more important to her how her travel time compares to the actual travel time of others. Our ultimate goal is to achieve a good total travel time while keeping unfairness low. Although the algorithm proposed above can only control the normal unfairness directly, with the "correct" choice of normal lengths we indirectly influence the other two, as we are about to see.

4.1. Choice of normal length

Initially, we considered three possible ways to define the normal length of an arc: *geographic distances*, travel times when the network is *uncongested* (zero flow), and travel times when the network is in *user equilibrium*. Recall that normal lengths can only be static; for instance, it is not possible to consider travel times under the current solution with the methodology described in Section 3. The advantage of keeping the model simple is a fast algorithm that still produces solutions with small total travel time and low unfairness, as we will see soon.

Geographic distances and travel times in the uncongested network are highly correlated; therefore, one cannot expect significant differences between solutions resulting from either choice of normal length. It is shown in [40], and our runs confirm it, that user equilibria are better than constrained system optima when the factor f used in determining the maximal path lengths L_k is small. Consequently, to obtain an improvement in the total travel time, bigger factors must be considered. But this gives rise to high unfairness, which is undesired. For instance, the graph on the left in Figure 7 shows the value of the objective function

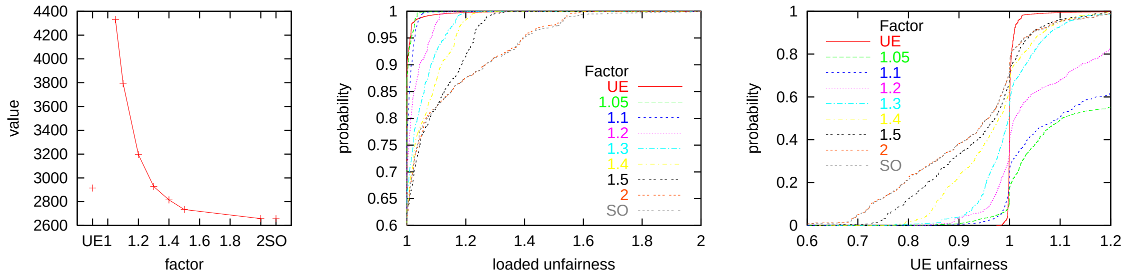


Figure 7: Objective values and unfairness distributions for instance `neuk` and normal lengths equal to travel times in the uncongested network.

for different factors in the scenario in which one uses the travel time in the uncongested network to define the normal length. Factors less than 1.4 are not useful because the total travel time of the corresponding solutions is greater than the total travel time in user equilibrium. The other two graphs in Figure 7 depict the distribution of unfairness for the different factors; for instance, for factor $f = 1.5$, 80% of all users will experience a loaded unfairness of less than 1.1. This value increases to 1.2 if one considers 90% of all users. In fact, for factors greater than 1.4, the distributions are quite similar to those of the system optimum, which especially means that these solutions are not very close to being in equilibrium. Notice also, in the graph on the right, that for small factors most users end up traveling longer than they would in user equilibrium. This happens because there are not enough alternatives (i.e., paths between OD-pairs), which explains the poor quality of the solutions under this choice of normal length.

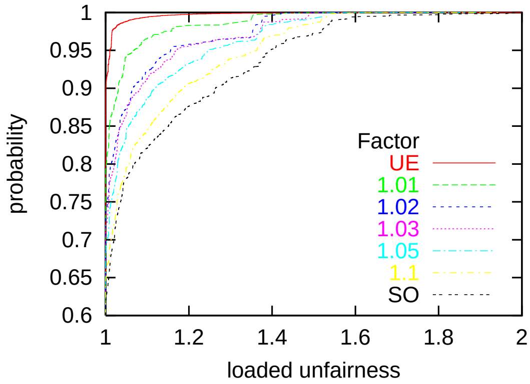
We therefore propose to make use of the travel times in user equilibrium when defining normal arc lengths, which indeed results in high-quality solutions. First, note that, for any factor f, the user equilibrium itself is a feasible solution to the constrained system optimum problem. Therefore, for all f,

$$z(CSO_f) \leqslant z(UE),$$

which guarantees that the optimal solution to problem $\mathcal{P}$ is not worse than the user equilibrium in terms of the total travel time in the system.

In Figures 8 and 10 below, we display the graphs corresponding to the ones in Figure 7 for this choice of normal length. Most notably, total travel times are distinctively smaller than in equilibrium, while the fraction of users traveling longer than in equilibrium is substantially smaller.

From the previous discussion, the normal length defined as the travel time in user equilibrium seems to be the best choice. For this reason, we limit our analysis in the sequel to this version of normal length.



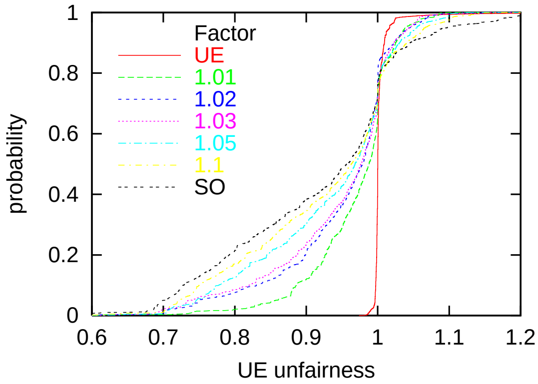


Figure 8: Distribution of unfairness for instance neuk and different factors. The graph on the left corresponds to loaded unfairness, and the one on the right to UE unfairness.

4.2. Quality of constrained system optima

Table 2 exhibits the output of the algorithm for the four instances presented in Table 1 and different factors. Every row represents one run for the factor reported in the first column. The column *value* is the total travel time in the solution; *paths* is the number of paths with positive flow. In the column *unfairness*, we report the 99th percentiles of the corresponding distributions. For example, the second row for instance frie presents the characteristics of the constrained system optimum with factor $f = 1.01$. The total travel time is 629, and users are distributed over 1313 different paths. (Recall that frie has 506 distinctive OD-pairs.) Some users experience a 9.1% longer travel time than they would in user equilibrium; the maximal difference between travel times for the same OD-pair is 66.5%. Note that the latter two values respectively are 26.1% and 108.4% in system optimum.

As we discussed before, when the factor is bigger the objective value is closer to the total travel time in the (unconstrained) system optimum and the unfairness is higher, and vice-versa. We will argue that the constrained system optimum with factor 1.02, denoted by $CSO_{1.02}$, offers a good trade-off for the real-world instances available to us. Figure 8 depicts the unfairness distributions corresponding to the runs of instance neuk. As before, we do not include the normal unfairness because we explicitly constrain it to be less than the factor used. To facilitate a comparison across runs, Figure 9 plots the different notions of unfairness, percentiles and factors, again for the instance neuk.

- In the four instances shown in Table 2, the gap between the total travel time in $CSO_{1.02}$ and the system optimum is approximately a third of the gap between the user equilibrium and the system optimum. For instance neuk, we conclude from

factor	value	paths	unfairness		
			normal	loaded	UE
frie					
UE	683	1687	1.011	1.021	1.020
1.01	629	1313	1.008	1.665	1.091
1.02	622	1277	1.017	1.694	1.116
1.03	614	1464	1.029	1.711	1.091
1.05	612	1579	1.046	1.779	1.090
1.10	594	1664	1.097	1.925	1.114
1.20	592	2013	1.175	2.070	1.182
1.30	591	2215	1.213	2.069	1.229
SO	591	2509	1.226	2.084	1.261
neuk					
UE	2915	7109	1.060	1.056	1.063
1.01	2800	4604	1.008	1.348	1.079
1.02	2738	5349	1.017	1.375	1.075
1.03	2726	5912	1.028	1.424	1.069
1.05	2694	6421	1.045	1.456	1.101
1.10	2676	8410	1.093	1.517	1.125
1.20	2657	9746	1.171	1.545	1.183
1.30	2657	10325	1.195	1.538	1.196
SO	2657	11055	1.197	1.546	1.201
3nei					
UE	1849	41110	1.039	1.051	1.049
1.01	1772	32514	1.007	1.304	1.051
1.02	1764	35552	1.017	1.307	1.050
1.03	1756	36150	1.026	1.312	1.049
1.05	1735	37749	1.046	1.358	1.061
1.10	1729	48376	1.088	1.454	1.085
1.20	1729	57864	1.136	1.484	1.128
1.30	1728	61336	1.140	1.490	1.132
SO	1728	64031	1.140	1.513	1.139
berl					
UE	16255	174547	1.058	1.058	1.058
1.01	16273	111853	1.009	1.156	1.913
1.02	15827	146866	1.018	1.215	1.119
1.03	15690	178858	1.028	1.242	1.071
1.05	15648	232243	1.046	1.269	1.062
1.10	15604	274019	1.087	1.330	1.086
1.20	15589	306362	1.135	1.376	1.124
1.30	15582	321901	1.151	1.413	1.138
SO	15562	334447	1.168	1.473	1.162

Table 2: Results of the runs on the four instances with different factors. The columns represent the factor used, total travel time, number of paths with positive flow in optimal solution and 99th percentile of the three unfairness.

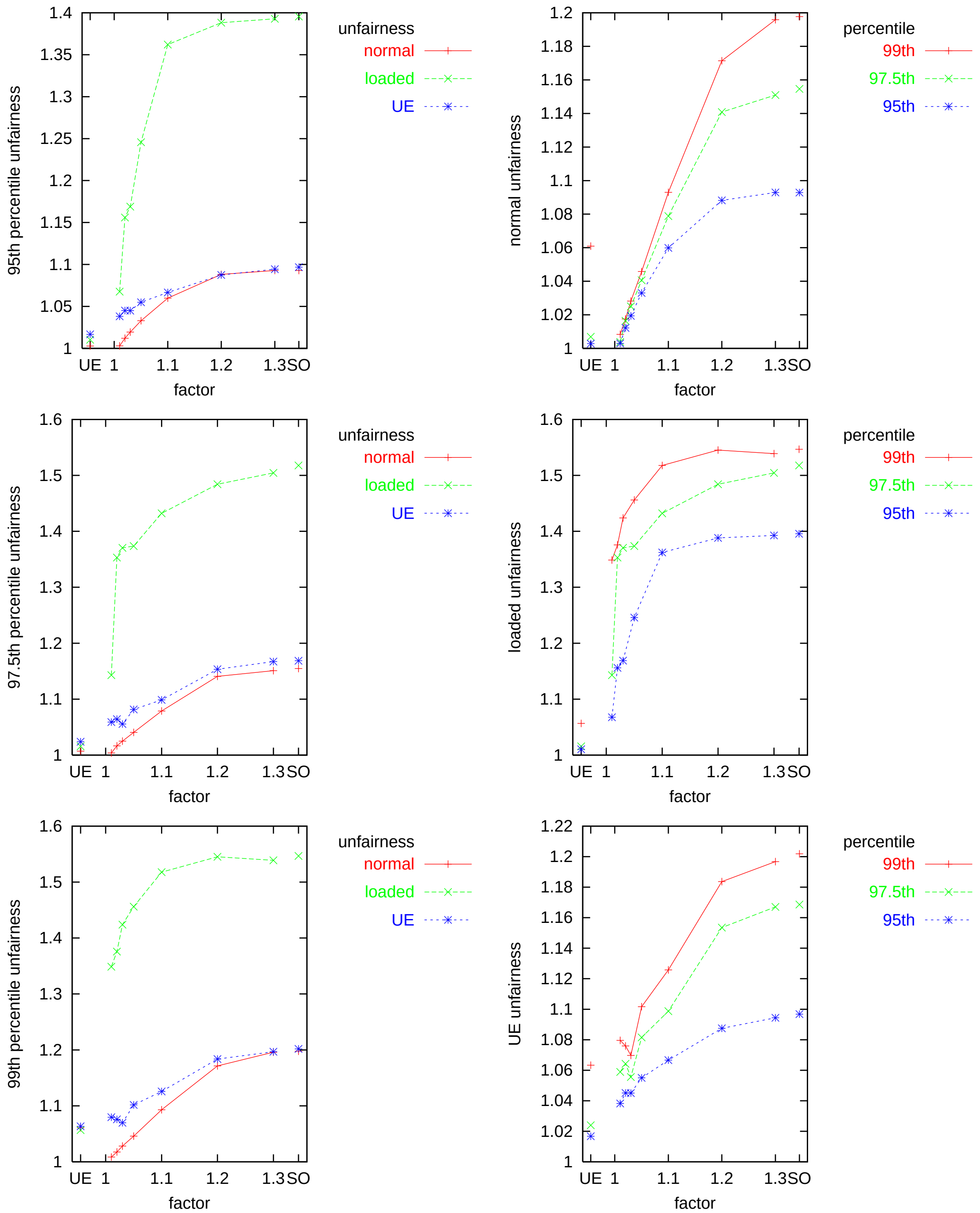


Figure 9: Unfairness over the different factors and percentiles for instance neuk.

Table 2 that the gap for the user equilibrium is $2915 - 2657 = 258$, and the gap for $CSO_{1.02}$ is $2738 - 2657 = 81$.

- The loaded unfairness of $CSO_{1.02}$ is approximately 40% smaller than the loaded unfairness of the system optimum. (This reduction is even higher if one considers the 95th percentile of the unfairness). For example, Table 2 shows for instance neuk that in the system optimum some people are assigned paths of 54.6% higher travel time than others. The corresponding figure for $CSO_{1.02}$ is 37.5%.
- From the "loaded unfairness" graph in Figure 8, we see that in the system optimum, 18% of the users travel 10% longer than the shortest path, whereas for $CSO_{1.02}$, only 8% of the users exceed that number.
- The "UE unfairness graph" in Figure 8 shows that most users (around 75%) travel less than they would in equilibrium. Actually, for factor 1.02, only 0.4% of the users travel 10% more than in equilibrium. Compare this number to the 5% that travel at least 10% longer under the system optimum.

4.3. Convergence and performance of the algorithm

The algorithm was implemented in C++. We used CPLEX 7.1 callable library to solve the restricted linear programs. The computing platform was a Pentium IV based computer running at 2.4 GHz with 1 GB RAM. Besides running the standard version of the algorithm, we also used the version that neglects arc capacities, which leads to a simpler algorithm, as noted at the end of Section 3.1. In the latter case, CPLEX is not used at all.

Computational results for the instances described in Table 1 are presented in Tables 3 and 4. These tables contain three columns. Each row reports a different run, characterized by the information in the first column: the factor and the desired distance to optimality used to stop the computation. Runs that exceeded four hours were terminated and are reported as "exceeded". The second column corresponds to the standard algorithm described in Section 3. Within this column, we respectively report the number of iterations, the number of paths with positive flow in the solution and the running time in seconds needed to achieve the desired gap. The third column corresponds to the alternative algorithm that ignores arc capacities. In this case, we also report the number of capacity violations, the maximal violation and the average violation.

The results show that instances with a few thousand nodes, arcs and commodities can be solved on an average PC. Bigger instances like berl take longer but can be solved without difficulty. In our experience, most of the running time is spent computing constrained shortest paths, which implies that improved algorithms for this subproblem would yield greatly improved overall performance. Empirically, we found it advantageous to add in every iteration as many new columns to the restricted master problem as possible. We observed a reduction in computation time by factors of about 50, compared to adding only a single column to $\bar{P}'$, and removing another one as needed.

		With Capacities			Capacities Relaxed			
factor	gap	iter.	paths	sec.	iter.	paths	sec.	cap.viol. ♯/max/avg
				frie				
UE	0.50	85	1278	2	52	1687	1	5,87.52,40.74
UE	1.00	49	1190	1	29	1668	0	5,82.92,37.76
UE	2.00	36	1121	1	19	1627	0	5,66.68,33.61
UE	4.00	24	962	0	11	1531	0	5,99.51,38.73
1.01	0.50	infeasible			118	1313	2	7,95.44,42.55
1.01	1.00	infeasible			55	1265	1	7,100.3,46.67
1.01	2.00	infeasible			27	1195	0	7,102.8,50.95
1.01	4.00	infeasible			10	1119	0	7,135.5,67.14
1.03	0.50	infeasible			101	1464	2	3,5.747,5.747
1.03	1.00	infeasible			49	1425	1	3,9.089,9.089
1.03	2.00	infeasible			24	1331	0	4,16.91,14.3
1.03	4.00	infeasible			9	1197	0	5,89.52,31.66
1.10	0.50	229	1610	6	100	1664	1	0,0,0
1.10	1.00	116	1512	3	31	1527	0	1,4.838,4.838
1.10	2.00	51	1375	1	13	1366	0	1,12.51,12.51
1.10	4.00	36	1215	1	8	1259	0	1,11.58,11.58
SO	0.50	461	2131	12	180	2509	3	0,0,0
SO	1.00	254	1994	7	94	2423	2	0,0,0
SO	2.00	124	1841	3	54	2348	1	0,0,0
SO	4.00	66	1614	2	27	2209	0	0,0,0
				neuk				
UE	0.50	72	6621	77	44	7109	44	1,29.12,29.12
UE	1.00	54	6573	62	24	7059	29	1,31.52,31.52
UE	2.00	40	6496	49	15	6994	23	1,40.12,40.12
UE	4.00	31	6335	41	8	6822	17	1,61.37,61.37
1.01	0.50	101	4600	61	69	4604	36	0,0,0
1.01	1.00	65	4553	39	34	4539	22	0,0,0
1.01	2.00	39	4439	25	15	4445	14	1,3.379,3.379
1.01	4.00	27	4384	19	7	4334	11	1,7.218,7.218
1.03	0.50	107	5904	72	44	5912	28	0,0,0
1.03	1.00	65	5746	44	24	5785	20	1,2.968,2.968
1.03	2.00	39	5564	29	12	5515	14	1,13.89,13.89
1.03	4.00	24	5328	21	6	5283	12	1,21.37,21.37
1.10	0.50	175	8266	102	72	8410	35	1,5.675,5.675
1.10	1.00	110	7962	67	38	8141	25	1,10.46,10.46
1.10	2.00	82	7655	54	20	7800	18	1,17.6,17.6
1.10	4.00	51	7015	39	8	7049	14	1,34.64,34.64
SO	0.50	214	10533	225	100	11055	92	0,0,0
SO	1.00	153	10418	167	62	10874	63	0,0,0
SO	2.00	115	10146	133	39	10708	45	0,0,0
SO	4.00	78	9786	97	22	10233	31	0,0,0

Table 3: Results of the runs for instances frie and neuk with varying gaps.

		With Capacities			Capacities Relaxed			
factor	gap	iter.	paths	sec.	iter.	paths	sec.	cap.viol. ♯/max/avg
3nei								
UE	0.50	808	33305	599	30	41110	157	4,164.1,83.17
UE	1.00	719	29550	428	17	39148	119	4,141.2,87.3
UE	2.00	676	25384	341	10	35820	103	4,124.8,79.46
UE	4.00	655	20852	292	6	32463	89	4,159.3,90.09
1.01	0.50	245	30934	685	29	32514	145	1,9.115,9.115
1.01	1.00	100	26006	314	12	29499	111	1,36.73,36.73
1.01	2.00	66	22817	211	6	26082	87	1,56.78,56.78
1.01	4.00	52	20593	172	4	23234	67	1,56.7,56.7
1.03	0.50	315	34263	854	38	36150	168	1,0.7908,0.7908
1.03	1.00	133	29898	405	16	32462	126	1,41.76,41.76
1.03	2.00	83	26723	273	8	28866	100	1,73.27,73.27
1.03	4.00	60	23698	204	4	25393	73	1,118.2,118.2
1.10	0.50	394	46131	891	44	48376	190	0,0,0
1.10	1.00	206	39823	493	20	42992	154	0,0,0
1.10	2.00	124	34137	342	9	35787	119	1,50.05,50.05
1.10	4.00	80	27564	238	6	32011	90	1,69.4,69.4
SO	0.50	1117	55104	1312	71	64031	228	0,0,0
SO	1.00	905	50917	947	40	59901	226	0,0,0
SO	2.00	786	45683	697	22	54250	190	1,10.2,10.2
SO	4.00	728	39369	541	12	48326	159	1,14.82,14.82
berl								
UE	0.50	exceeded			26	174547	5487	5,703.8,199.5
UE	1.00	94	144259	13621	16	162011	4306	8,645.3,122.9
UE	2.00	65	130630	10137	10	148770	3456	6,750.8,180.3
UE	4.00	39	108829	6589	5	129652	2613	8,536.6,99.1
1.01	0.50	infeasible			14	111853	3456	2,329.8,235.2
1.01	1.00	infeasible			8	104594	2867	2,338.2,240.4
1.01	2.00	infeasible			4	92776	2155	3,365,170.3
1.01	4.00	infeasible			3	88399	1919	2,365.6,255.9
1.03	0.50	infeasible			29	178858	6263	3,262.6,120.9
1.03	1.00	infeasible			14	151885	4121	3,266.5,123
1.03	2.00	infeasible			7	128420	2973	3,254.3,122
1.03	4.00	infeasible			4	110821	2312	3,310.4,145
1.10	0.50	exceeded			54	274019	9106	2,259.7,164.7
1.10	1.00	exceeded			27	227212	6641	3,259.9,111.3
1.10	2.00	exceeded			12	181369	4610	3,272.7,114.9
1.10	4.00	exceeded			7	154032	3135	3,331.6,137.1
SO	0.50	exceeded			76	334447	13391	3,257.2,106.9
SO	1.00	exceeded			45	297940	9543	2,253.2,158.2
SO	2.00	exceeded			27	262212	7010	3,255.9,106.2
SO	4.00	exceeded			14	220726	4888	3,272.9,112.8

Table 4: Results of the runs for instances 3nei and berl with varying gaps.

Figure 10 shows a detailed study of the effects of varying the gap and the constraint factor for the instance neuk and the algorithm that ignores arc capacities. (The corresponding analysis for the other instances can be found in the Appendix). As expected, a smaller desired gap leads to a decrease of the objective value, while tighter length constraints (factor f close to one) yield a sharp increase of the objective value. There is an exponential increase in the running time when the gap is made smaller.

The problem becomes more difficult when the factor grows because additional paths become allowable. However, the constrained shortest path problem becomes easier because the normal lengths are less binding. In this trade-off, both the total work and the number of iterations increase, but the work per iteration decreases.

4.3.1. Comparison of the cases with and without arc capacities

As described in Section 3.1, when arc capacities are ignored, the linear subproblem to be solved in each iteration of the convex combinations algorithm (Figure 5) decomposes into $|C|$ constrained shortest paths problems. Naturally, this makes each iteration much faster, compared to the case with arc capacities, in which each linear subproblem is solved by the simplex algorithm, which in turn consists of several iterations of constrained shortest path computations. Consequently, there is a huge speed-up when capacities are relaxed. For example, for the instance berl, which is the largest, we could find an optimal solution in under four hours with this alternative only. For the remaining instances, the running time of the algorithm with capacities relaxed was on average a third of the running time of the standard algorithm.

5. Summary and Conclusion

When designing a route guidance system, it is desirable to explicitly aim at reducing the total (and therefore the average) travel time by putting it into the objective function of the underlying optimization problem. Yet, without further constraints, this would include the possibility that some vehicles are assigned to fairly long paths in order to make the shorter paths available to other drivers. Obviously, this phenomenon would render such a system unacceptable for several drivers, jeopardizing the desired effect of improved system performance.

We propose to capture this aspect of human behavior by imposing constraints on exercised paths to eliminate lengthy detours. While it may be ideal to explicitly enforce that travel times of recommended routes between the same origin-destination pair do not deviate drastically from each other, our computational results justify the use of a computationally simpler model, in which the deviation is not measured with respect to the actual flow but with respect to a “normal length”. Another plus of the latter tactic is that drivers with different origin-destination pairs can be treated equally. Our computational study suggests that the travel time in user equilibrium is an excellent choice to define the normal length.

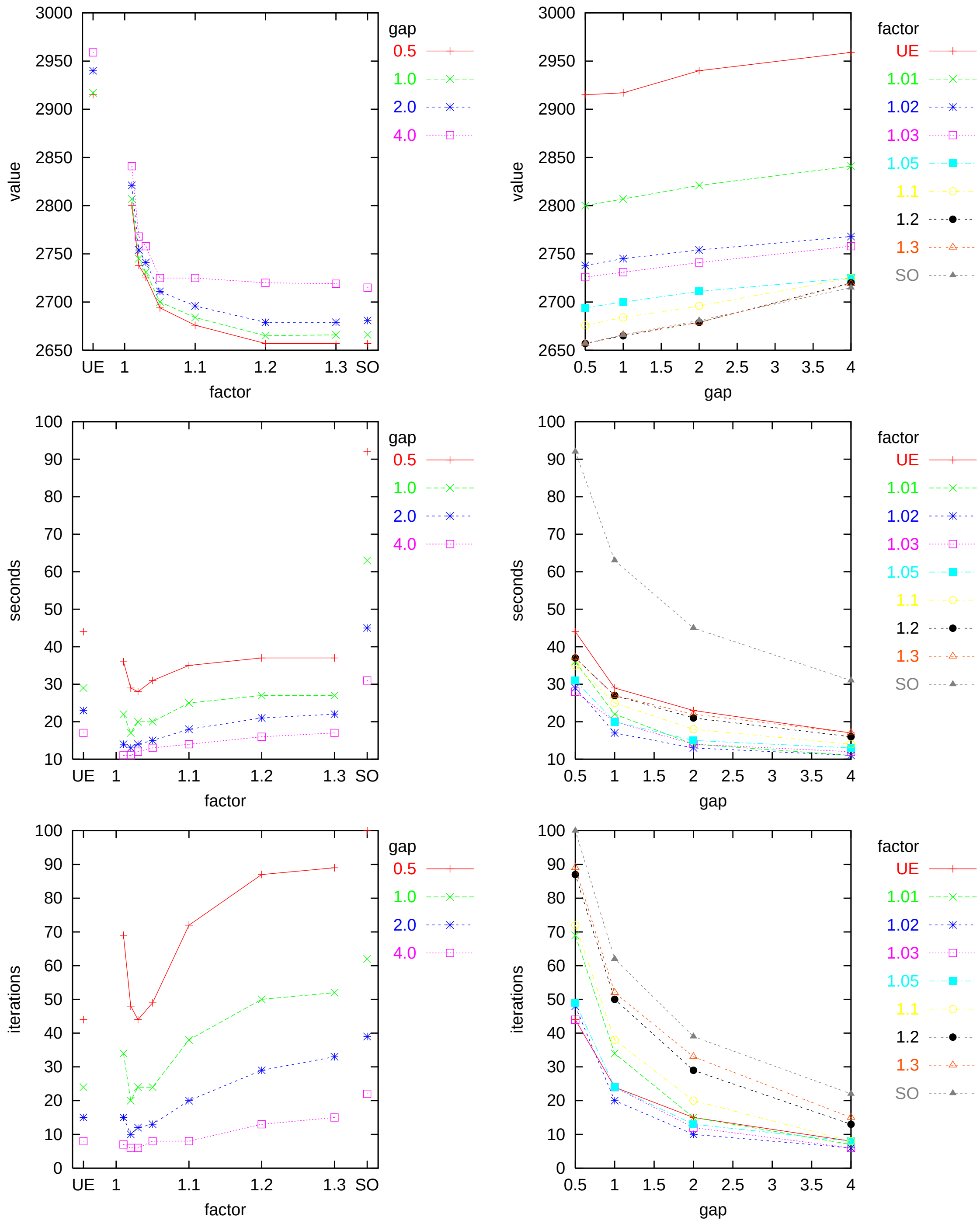


Figure 10: Output of the algorithm over different gaps and factors for instance neuk.

In fact, it turns out that this approach offers significant advantages over both the traditionally considered user equilibrium ("traffic assignment") and the system optimum. On the one hand, it guarantees a superior fairness for the individual user compared to the system optimum, in which individual travel times between the same origin-destination pair may deviate substantially from each other. On the other hand, the total travel time of a constrained system optimum is still close to that in the (unconstrained) system optimum and thus much better than in user equilibrium. This shows that optimal route guidance with fairness guarantees is in principle feasible.

Apart from the proof of concept, we consider our algorithm practical for problems with some thousand nodes, arcs, and commodities. Yet, future work should incorporate the dynamic aspect of traffic and the behavior of unguided users in a more sophisticated way.

Acknowledgments

The authors are grateful to Dr. Stefan Gnutzmann (DaimlerChrysler AG, Berlin), who excited our work on route guidance, and to Dr. Stefan Gnutzmann and Valeska Naumann (DaimlerChrysler AG, Berlin) for several stimulating discussions and support.

The second author acknowledges support by grant 03-MOM4B1 "Models and Algorithms for Dynamic Route Guidance in Traffic Networks" of the German Ministry for Science and Education (BMBF).

The third author gratefully acknowledges support by a General Motors Innovation Grant.

References

[1] J.L. Adler and V.J. Blue. Toward the design of intelligent traveler information systems. *Transportation Research C*, 6:157–172, 1998.

[2] Y.P. Aneja, V. Aggarwal, and K.P.K. Nair. Shortest chain subject to side constraints. *Networks*, 13:295–302, 1983.

[3] L. Armijo. Minimization of functions having Lipschitz continous first partial derivatives. *Pacific Journal of Mathematics*, 16:1–3, 1966.

[4] M.S. Bazaraa, H.D. Sherali, and C.M. Shetty. *Nonlinear Programming. Theory and Algorithms.* John Wiley & Sons, New York, 2nd edition, 1993.

[5] J.E. Beasley and N. Christofides. An algorithm for the resource constrained shortest path problem. *Networks*, 19:379–394, 1989.

[6] G. Beccaria and A. Bolelli. Modelling and assessment of dynamic route guidance: the MARGOT project. In *Proceedings of the IEEE Vehicle Navigation & Information Systems Conference*, 1992, Oslo, pages 117–126.

[7] M.G.H. Bell, C.M. Shield, J.M. Anderson, and F. Busch. Assignment in the integration of urban traffic control and dynamic route guidance. In [20], pages 39–57.

[8] M.E. Ben Akiva. Dynamic network equilibrium research. *Transportation Research A*, 19:429–431, 1985.

[9] M.E. Ben-Akiva, E. Cascetta, and H. Gunn. An on-line dynamic traffic prediction model for an inter-urban motorway network. In [20], pages 83–122.

[10] M.E. Ben-Akiva, A. de Palma, and I. Kaysi. Dynamic network models and driver information systems. *Transportation Research A*, 25:251–266, 1991.

[11] D. Braess. Über ein Paradoxon aus der Verkehrsplanung. *Unternehmensforschung*, 12:258–268, 1968.

[12] Y.-L. Chou, H.E. Romeijn, and R.L. Smith. Approximating shortest paths in large-scale networks with an application to intelligent transportation systems. *INFORMS Journal on Computing*, 10:163–179, 1998.

[13] V. Chvátal. *Linear Programming.* W.H. Freeman and Company, New York, 1983.

[14] S. Cohen. Flow variables. In [33], pages 139–143.

[15] J. Desrosiers, Y. Dumas, M.M. Solomon, and F. Soumis. Time constrained routing and scheduling. In M.O. Ball, T.L. Magnanti, C.L. Monma, and G.L. Nemhauser, editors, *Network Routing*, volume 8 of *Handbooks in Operations Research and Management Science*, pages 35–139. Elsevier Science, Amsterdam, 1995.

[16] M.C. Ferris and A. Ruszczyński. Robust path choice and vehicle guidance in networks with failures. Technical Report 97–04, Computer Sciences Department, University of Wisconsin, April 1997. `ftp://ftp.cs.wisc.edu/math-prog/tech-reports/97-04.ps.Z`.

[17] M. Frank and P. Wolfe. An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, 3:95–110, 1956.

[18] T.L. Friesz. Transportation network equilibrium, design and aggregation: Key development and research opportunities. *Transportation Research A*, 19:413–427, 1985.

[19] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* Freeman and Company, New York, 1979.

[20] N.H. Gartner and G. Improta, editors. *Urban Traffic Networks. Dynamic Flow Modelling and Control.* Springer, Berlin, 1995.

[21] J.N. Hagstrom and R.A. Abrams. Characterizing Braess's paradox for traffic networks. In *Proceedings of IEEE Conference on Intelligent Transportation Systems*, pages 837–842, 2001.

[22] R.W. Hall. Route choice and advanced traveller information systems on a capacitated and dynamic network. *Transportation Research C*, 4:289–306, 1996.

[23] J.J. Henry, C. Charbonnier, and J.L. Farges. Route guidance, individual. In [33], pages 417–422.

[24] D.E. Kaufman, R.L. Smith, and K.E. Wunderlich. An iterative routing/assignment method for anticipatory real-time route guidance. In *Proceedings of the IEEE Vehicle Navigation & Information Systems Conference*, 1991, Dearborn, pages 693–700.

[25] D.E. Kaufman, R.L. Smith, and K.E. Wunderlich. User-equilibrium properties of fixed points in dynamic traffic assignment. *Transportation Research C*, 6:1–16, 1998.

[26] I. Kaysi, M.E. Ben-Akiva, and A. de Palma. Design aspects of advanced traveler information systems. In [20], pages 59–81.

[27] R.W. Klessig. An algorithm for nonlinear multicommodity flow problems. *Networks*, 4:343–355, 1974.

[28] S. Lafortune, R. Sengupta, D.E. Kaufman, and R.L. Smith. A dynamical system model for traffic assignment in networks. In *Proceedings of the IEEE Vehicle Navigation & Information Systems Conference*, 1991, Dearborn, pages 701–708.

[29] C.-K. Lee. A multiple-path routing strategy for vehicle route guidance systems. *Transportation Research C*, 2:185–195, 1994.

[30] H.S. Mahmassani and R. Jayakrishnan. System performance and user response under real-time information in a congested traffic corridor. *Transportation Research A*, 25:293–307, 1991.

[31] H.S. Mahmassani and S. Peeta. System optimal dynamic assignment for electronic route guidance in a congested traffic network. In [20], pages 3–37.

[32] M. Müller-Hannemann and K. Weihe. Pareto shortest paths is often feasible in practice. In G. Brodal, D. Frigioni, and A. Marchetti-Spaccamela, editors, *Proceedings of the 5th International Workshop on Algorithm Engineering*, Lecture Notes in Computer Science 2141, pages 185–197. Springer, 2001.

[33] M. Papageorgiou, editor. *Concise Encyclopedia of Traffic & Transportation Systems.* Pergamon Press, Oxford, 1991.

[34] M. Patriksson. *The Traffic Assignment Problem: Models and Methods.* VSP, Utrecht, The Netherlands, 1994.

[35] C. Ribeiro and M. Minoux. Solving hard constrained shortest path problems by Lagrangean relaxation and branch-and-bound algorithms. *Methods of Operations Research*, 53:303–316, 1986.

[36] L.R. Rilett and M.W. Van Aerde. Modelling distributed real-time route guidance strategies in a traffic network that exhibits the Braess paradox. In *Proceedings of the IEEE Vehicle Navigation & Information Systems Conference*, 1991, Dearborn, pages 577–587.

[37] T. Roughgarden. How unfair is optimal routing? In *Proceedings of the 13th Annual ACM-SIAM Symposium on Discrete Algorithms*, 2002, San Francisco, CA, pages 203–204.

[38] T. Roughgarden and É. Tardos. How bad is selfish routing? *Journal of the ACM*, 49:236–259, 2002.

[39] Y. Sheffi. *Urban Transportation Networks*. Prentice-Hall, New Jersey, 1985.

[40] A.S. Schulz and N. Stier Moses. On the performance of user equilibria in traffic networks. *Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms*, 2003. Baltimore, MD, January 12–14.

[41] T. van Vuren and D. Watling. Multiple user class assignment model for route guidance. In *In-Vehicle Information Systems: Modeling Traffic Networks and Behavioral Considerations 1991*, volume 1306 of *Transportation Research Record*, pages 22–32. Washington, D.C., 1991.

[42] J.G. Wardrop. Some theoretical aspects of road traffic research. *Proceedings of the Institution of Civil Engineers*, 1:325–362, 1952.

[43] T.A. Yang, S. Shekhar, B. Hamidzadeh, and P.A. Hancock. Path planning and evaluation in IVHS databases. In *Proceedings of the IEEE Vehicle Navigation & Information Systems Conference*, Dearborn, pages 283–290.

A. Charts corresponding to other instances

The unfairness distributions for the other instances are depicted in Figure 11. Figures 12, 13 and 14 display the output of the algorithm for these instances. Moreover, Figures 15, 16 and 17 analyze the unfairness of the resulting route assignments under varying paramters.

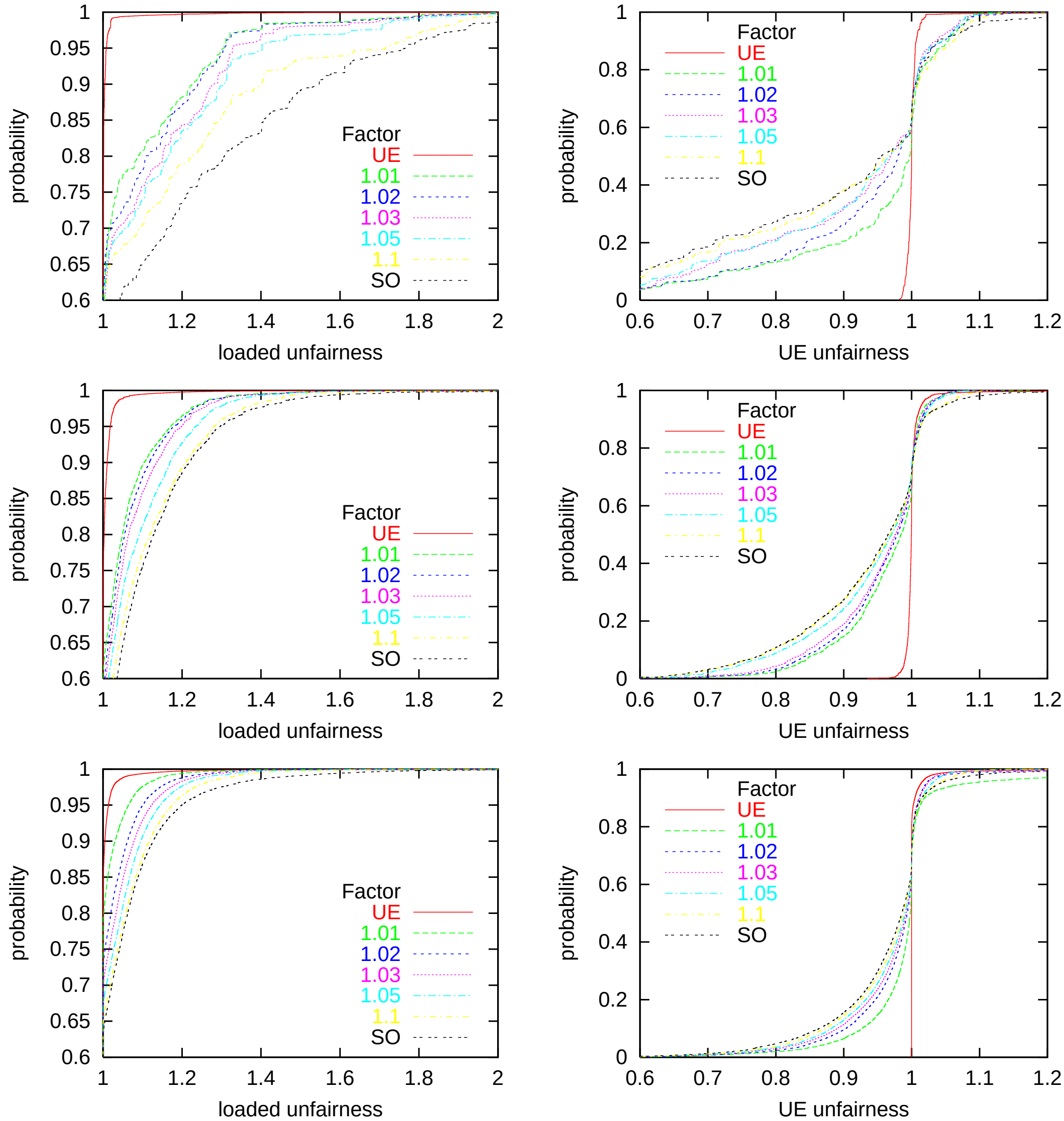


Figure 11: Distribution of unfairness for instances frie, 3nei and berl from top to bottom. The graphs on the left correspond to loaded unfairness and the ones on the right to UE unfairness.

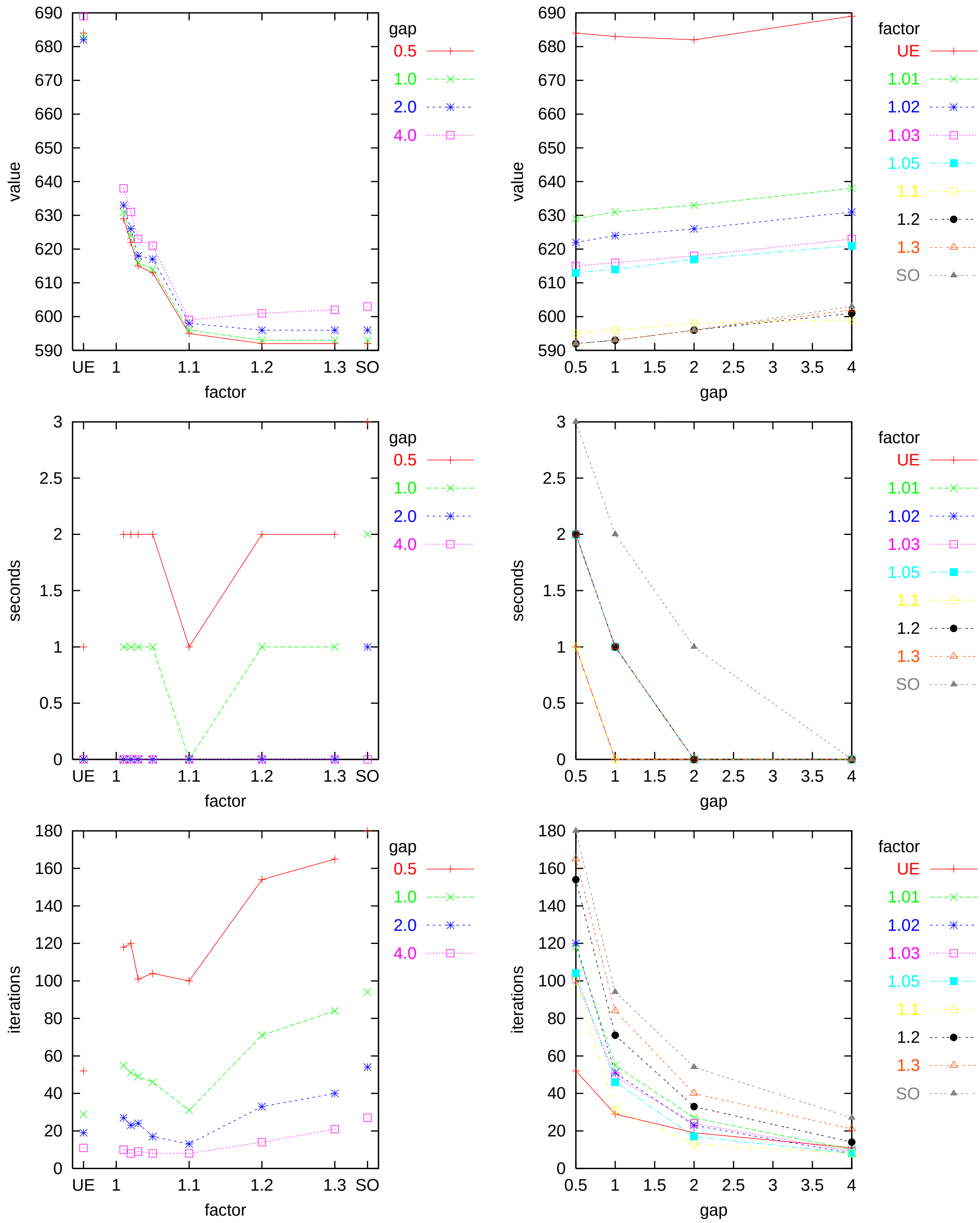


Figure 12: Parameter analysis for instance frie.

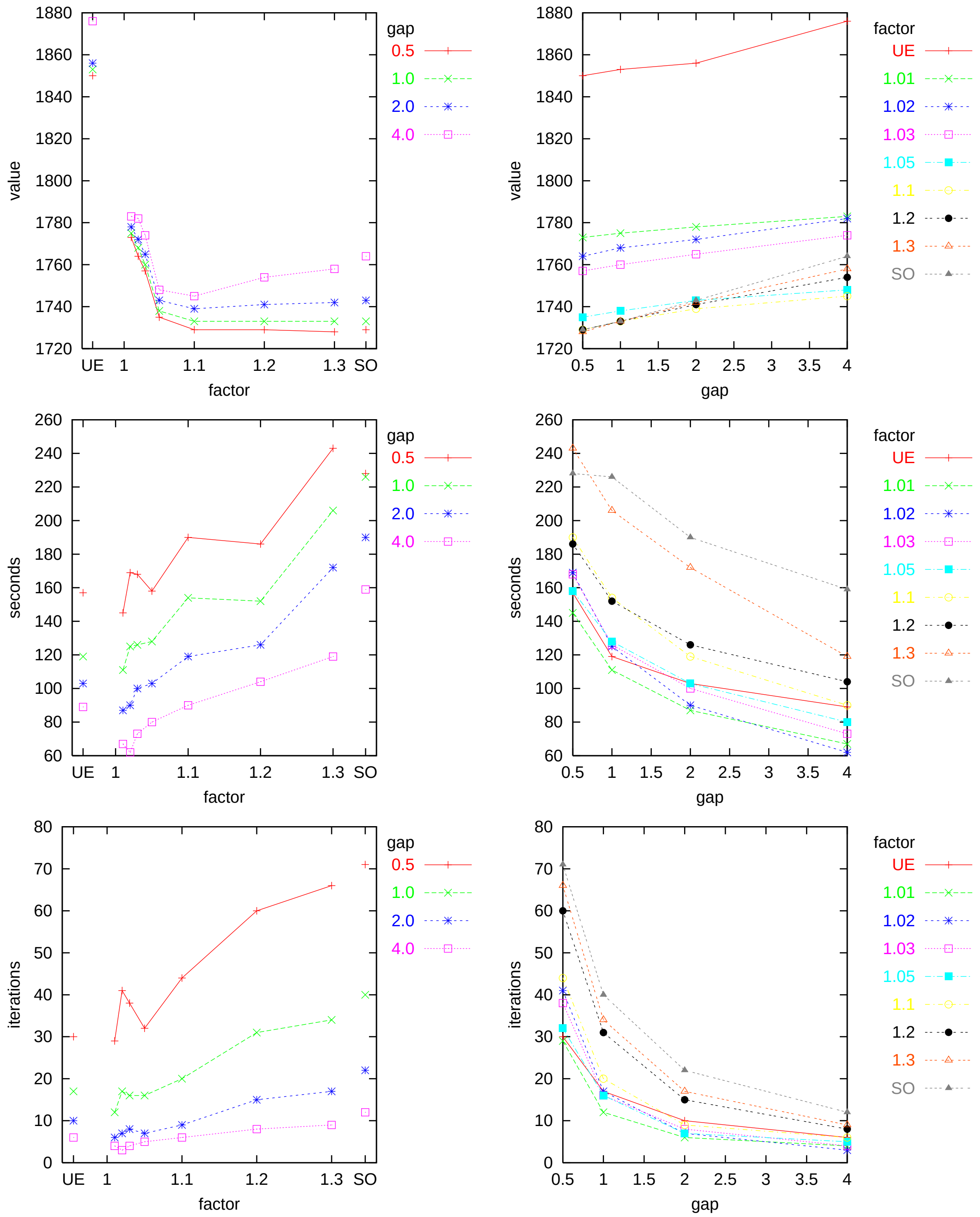


Figure 13: Parameter analysis for instance 3nei.

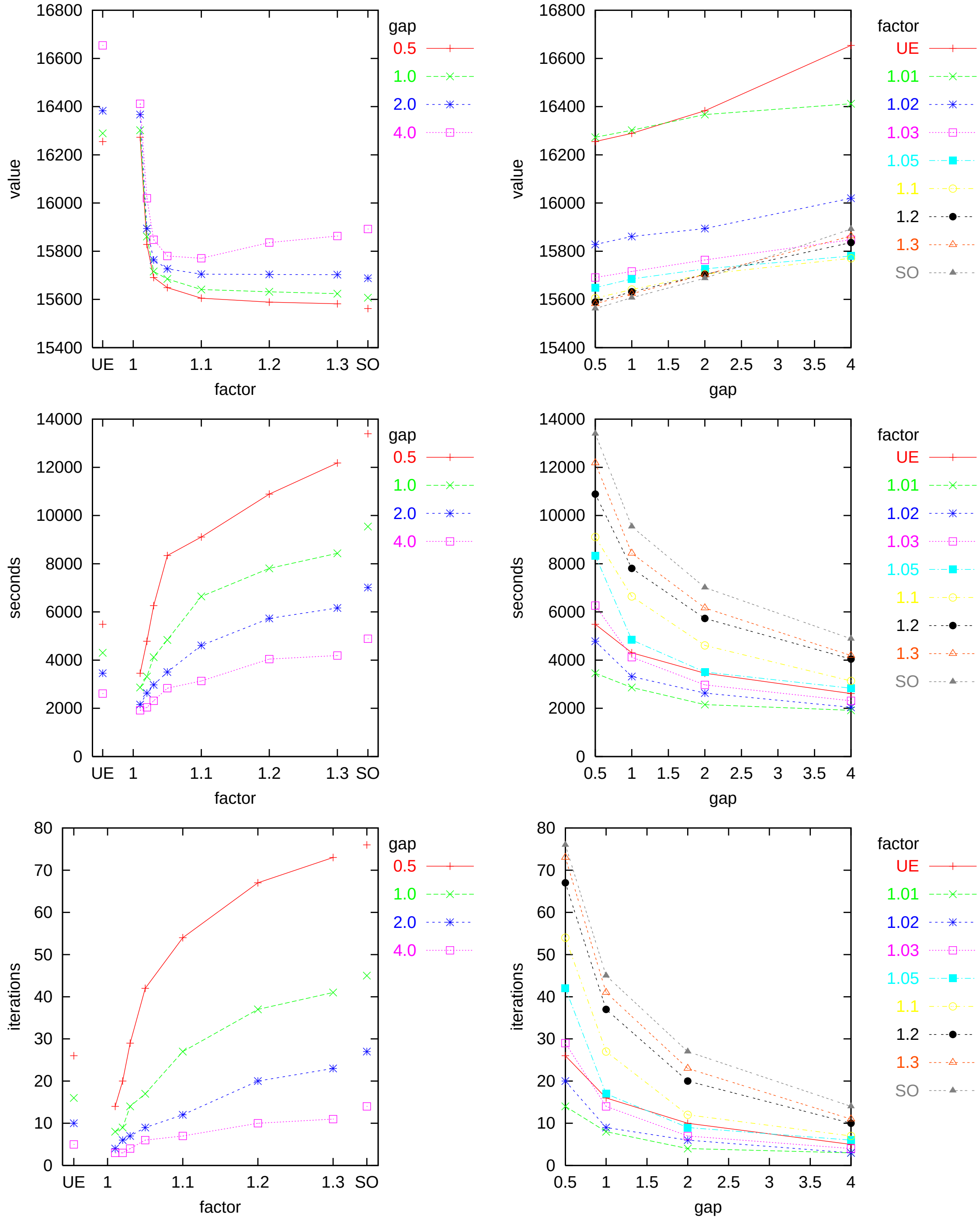


Figure 14: Parameter analysis for instance berl.

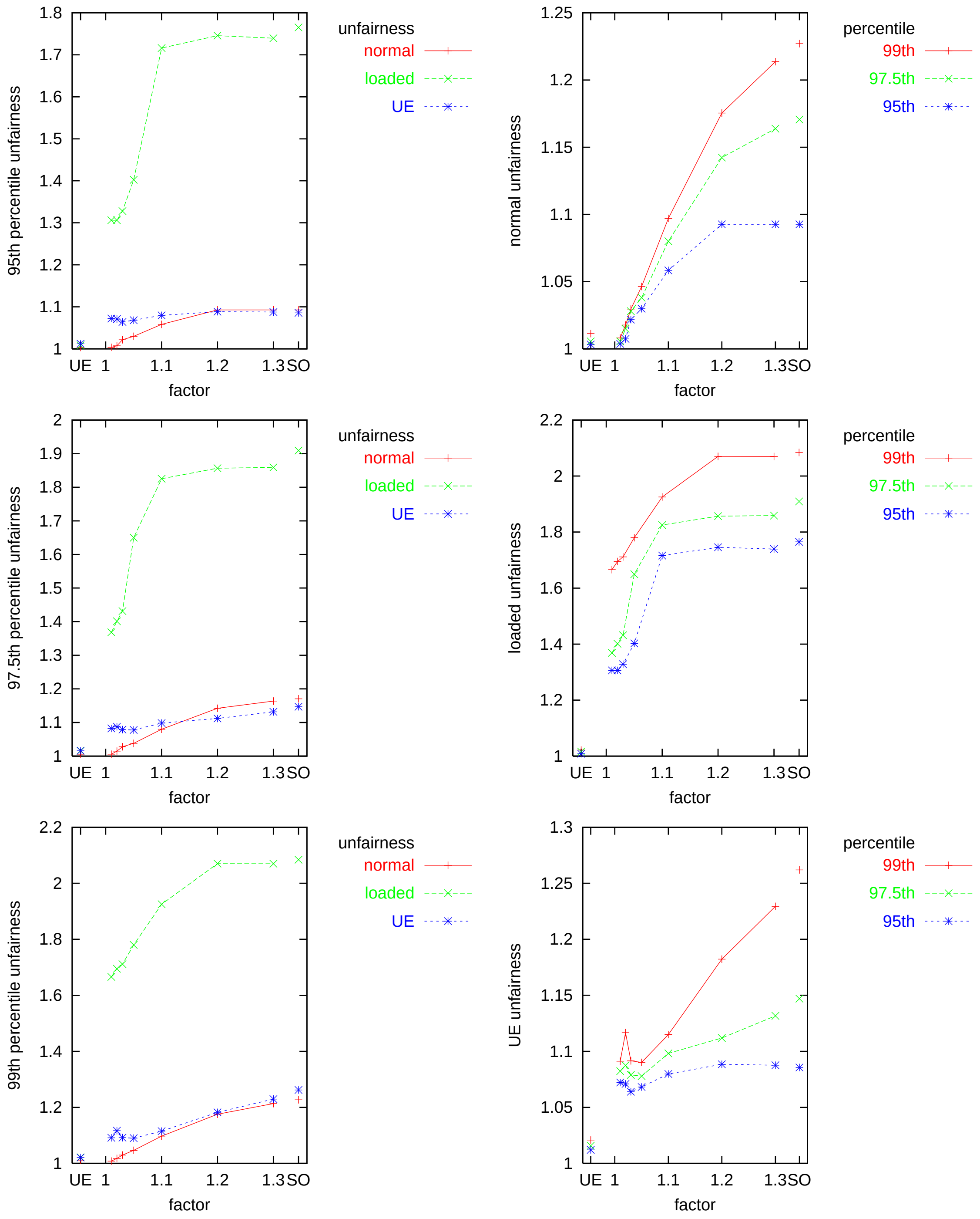


Figure 15: Unfairness over different factors and percentiles for instance frie.

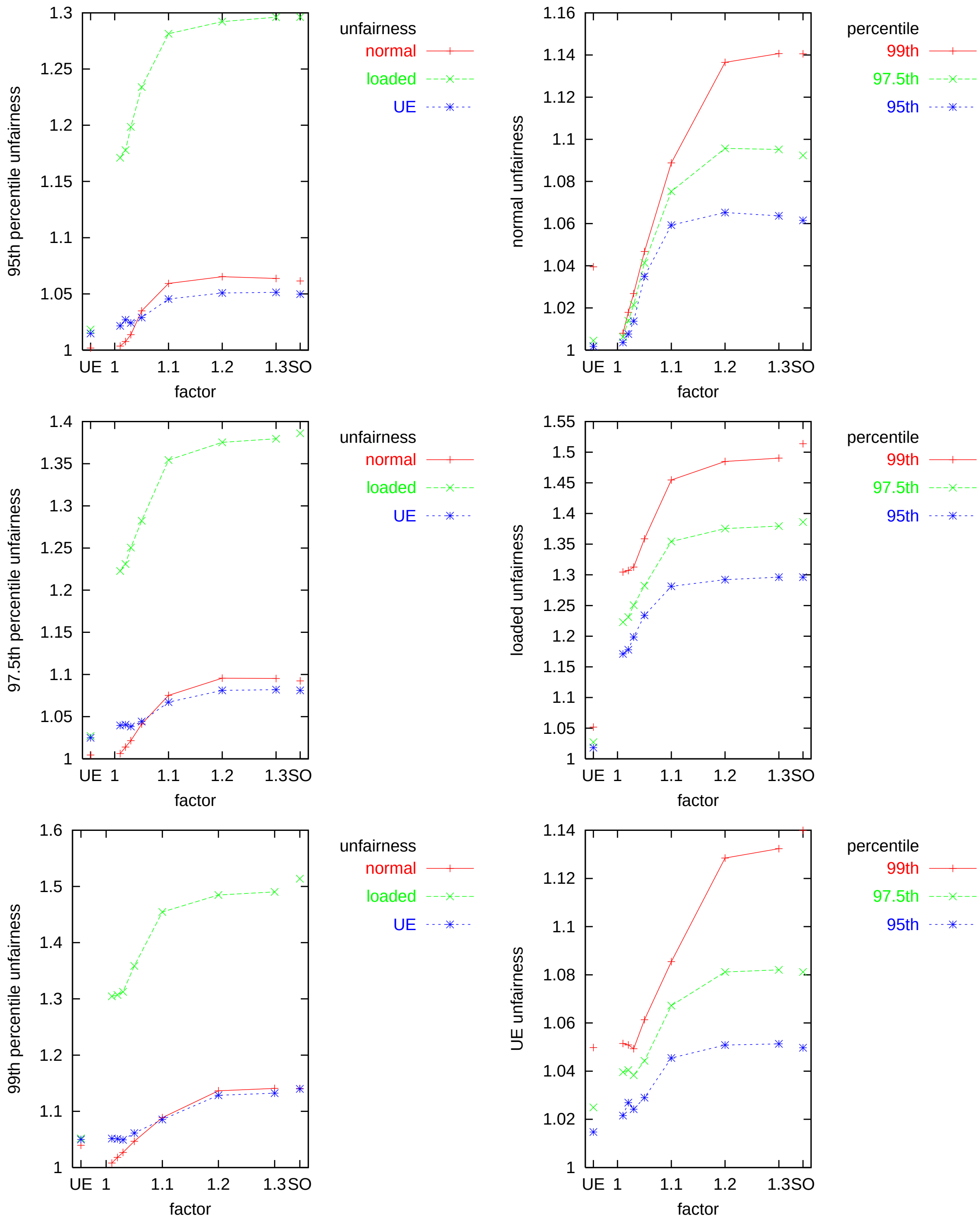


Figure 16: Unfairness over different factors and percentiles for instance 3nei.

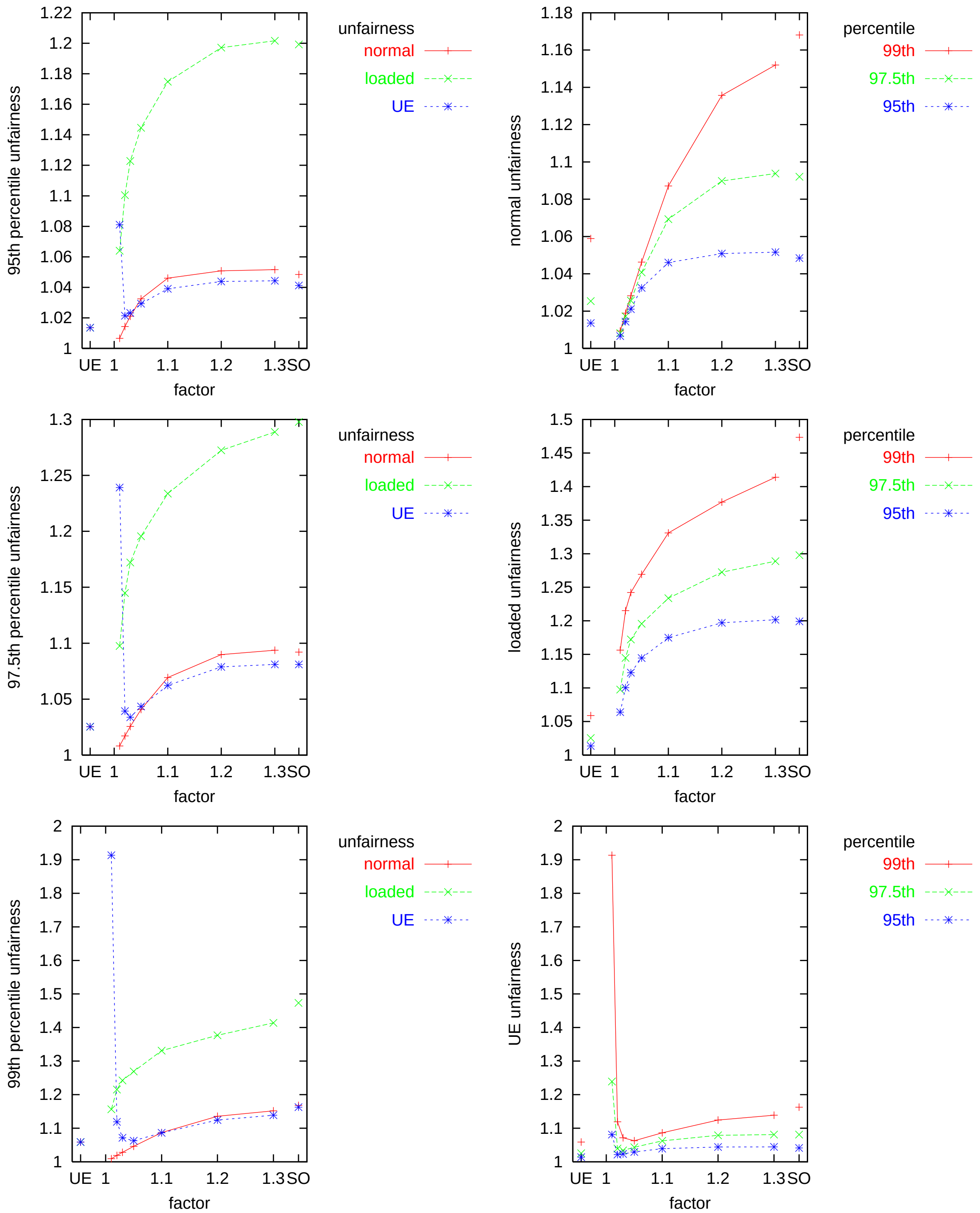


Figure 17: Unfairness over different factors and percentiles for instance berl.