Department of Economics and Management
University of Brescia
Italy

WORKING PAPER

Heuristic path generation for the linear constrained system optimum model

Enrico Angelelli
Valentina Morandi
Grazia Speranza

WPDEM 7/2016

Via S. Faustino 74/b, 25122 Brescia – Italy
Tel 00390302988742 Fax 00390302988703
email: paqricerca.dem@unibs.it

Heuristic path generation for the linear constrained system optimum model

Enrico Angelelli (1) *Valentina Morandi* (2) *Grazia Speranza* (3)

(1) *Department of Economics and Management, University of Brescia, Italy, enrico.angelelli@unibs.it*

(2) *Department of Economics and Management, University of Brescia, Italy, valentina.morandi1@unibs.it*

(4) *Department of Economics and Management, University of Brescia, Italy, grazia.speranza@unibs.it*

Abstract

In this paper we propose a heuristic algorithm to solve a linear programming model for the constrained system optimum traffic assignment problem recently proposed in the literature. This kind of models compromise between user equilibrium and system optimum models. However, limiting the set of eligible paths may require a huge amount of paths to be explicitly considered, and, thus make the model computationally intractable. In this paper a heuristic iterative algorithm aimed at generating a near optimal set of feasible paths, and to obtain a near-optimal solution, is presented. Computational experiments highlight that the number of paths generated by the heuristic algorithm is several orders of magnitude smaller with respect to the complete set. Accordingly, the heuristic computational time is orders of magnitude smaller than solving the model with complete enumeration of feasible paths. We also show that, when the model with path complete enumeration can be solved, the heuristic produces very high quality solutions.

Keywords: Traffic congestion; linear constrained system optimum; Heuristic iterative path generation.

1 Introduction

Traffic management has been attracting more and more attention in the last decades as, due to the growing population, the number of vehicles travelling in urban areas has kept increasing at an higher rate than road network development. The effects of the implied traffic congestion are multifold: pollution, stress and delays for public and private transport. All these factors form a vicious circle in which they feed on each other. Several theoretical models and methods have been developed in the attempt to provide tools able

to relieve the traffic congestion problem. Approaches to traffic-flow control can be classified in two main classes: micro and macro traffic control.

Microscopic traffic-flow control models aim at describing the individual behavior of each vehicle while the macroscopic ones aim at describing traffic as a continuous flow subject to global rules. Details on these methodologies can be found in Treiber and Kesting (2013).

Macro traffic models are rooted in the two Wardropian principles: the system optimum and the user equilibrium traffic assignment. After the seminal work by Wardrop (1952) the concepts of system optimum and user equilibrium were formulated in the form of optimization models in Beckmann et al. (1956). The two Wardropian principles assume that travel demand is constant over the whole period of interest. According to Sheffi (1985), this hypothesis is considered to be reliable especially during rush-hour periods when traffic exhibits a steady-state behavior. Furthermore, in both traffic assignments the arc traveling time is assumed to be dependent on the arc flow according to a function called latency function. A review on the mainly used latency functions can be found in Branston (1976) and Rose et al. (1989).

System optimum traffic assignment aims at minimizing the total travel time spent by users on the road network while user equilibrium traffic assignment guarantees that not only users with the same origin and destination (OD pair) will experience the same travel time, but none of them will have any advantage in changing unilaterally their assigned path. The total travel time in a user equilibrium is generally worse than the one provided by a system optimum traffic assignment, but, on the other side, in a system optimum traffic assignment different users with the same origin and destination may be 'unfairly' routed on paths with remarkably different travel times. Moreover, far from being in an equilibrium solution, some user could gain personal advantage at community expenses by unilaterally changing their assigned path.

To the best of our knowledge, the first attempt to find a compromised solution between the two assignments, is due to Jahn et al. (2000) where a traffic assignment model called constrained system optimum has been developed. The set of feasible paths is defined through an a priori measure of their arcs: the Euclidean distance between arc extreme points. For each OD pair the feasible paths are those with length within a fixed percentage of the shortest path, which we call *maximum travel inconvenience*.

Later, the same authors (Jahn et al. (2005)) generalized the a priori arc measure by the concept of *normal length* which can refer to different measures, e.g. Euclidean distance, free-flow travel time, and travel time under the user equilibrium. In this new work, a constrained system optimum model is proposed in which the latency function is the one provided by the Bureau of Public Roads (U.S.), $t_a(x) = t_a^{FF}[1+0.15(\frac{x_a}{u_a})^4]$, where

t_a^{FF} represents the arc free-flow travel time, x_a represents the flow rate assigned to arc a and u_a is a shape parameter, which we call *capacity*, affecting the curvature of the latency function. The problem is solved using a variant of the Frank-Wolfe algorithm, called Partan (see LeBlanc et al. (1985)), in which the search for a descent direction is performed with a linearized version of the model.

A different way to look at both system and users' perspectives is proposed in Angelelli et al. (2016a), where a simple linear model for the so-called proactive route guidance problem is presented. The proactive route guidance approach aims at minimizing the unfairness experienced by users while keeping the network uncongested, if possible, or at its minimum congestion level, otherwise. As normal length, the free-flow travel time is used and, as latency function, a constant function is considered. The hierarchical approach requires the complete enumeration of all feasible paths for each OD pair. The number of paths may be so high to make models computationally intractable. To obviate the generation of all the possible paths, in Angelelli et al. (2016c) a heuristic path generation method, able to generate a small set of paths providing a near optimal solution, is presented. The main feature of the approach presented in Angelelli et al. (2016a) is that it is based on linear programming models only. On the other hand, the main drawback is that the impact of traffic flows on travel time is not considered because the latency function is assumed to be constant.

With the aim to maintain the linearity feature of the proactive route approach and to consider the effect of traffic flows on travel time, in Angelelli et al. (2016b) a linear programming model for the constrained system optimum problem, called *linear constrained system optimum*, is proposed. The linear constrained system optimum model approximates the non-linear latency function on each arc with a n-piecewise linear function and assigns paths to users so as to minimize the total travel time experienced on the constrained path set. The linear constrained system optimum model requires to generate all the feasible paths from origin to destination for each OD pair. As proved in Angelelli et al. (2016a), the number of paths grows exponentially with the instance size. Thus, when large instances are considered, a huge number of paths have to be generated and the model becomes computationally intractable as its size grows accordingly. In the remainder of the paper, we refer to the algorithm that generates all feasible paths as the *complete enumeration* (CE) algorithm.

In this paper a heuristic algorithm for the linear constrained system optimum model is proposed that generates a small set of feasible paths chosen among those able to improve the current solution. Generating suitable paths in a graph is a matter deeply studied in literature (see Ramming (2001) and Prato (2009) for reviews). Most of the path generation techniques are based on the shortest path algorithms as their efficiency is well-known. One of the well-know deterministic generation methods based on the shortest path

calculation is the k-shortest path algorithm proposed in Yen (1971). Other generation methods involve the use of more than one objective function (see Ramming (2001) and Van der Zijpp and Catalano (2005)). The most popular path generation heuristic approaches use either the arc elimination or the arc penalty methods. The arc elimination method removes one or more arcs from the network and performs a new shortest path search on the modified network. This method was first presented in Azevedo et al. (1993), variants can be found in Prato and Bekhor (2006), Prato and Bekhor (2007), Frejinger and Bierlaire (2007) and Angelelli et al. (2016c). The arc penalty methods are based on penalizing some or all the network arc weights and calculating the shortest path on the new network as proposed in de la Barra et al. (1993). Variants can be found in Park and Rilett (1997), Bekhor et al. (2006) and Bekhor and Prato (2006).

The heuristic presented in this paper starts with a set of paths containing only the shortest path for each OD pair, and iteratively uses an arc penalty method in order to find new paths that can help in reducing the total travel time. Namely, at each iteration, for each OD pair, the shortest path from origin to destination is sought according to the current value of the latency function determined by the current flow on each arc.

The paper is organized as follows. In Section 2 the linear constrained system optimum model is recalled. In Section 3 the heuristic iterative path generation algorithm for the linear constrained system optimum (HI-GEN) is described. In Section 4 the results of a thorough computational analysis are presented and the benefits of the heuristic LIN-C-SO(n) model are shown. Finally, some conclusions are drawn in Section 5.

2 The linear constrained system optimum model

In order to make the paper self-contained, we recall in this section the linear constrained system optimum model (LIN-C-SO(n)) proposed in Angelelli et al. (2016b). The model is defined on a graph $G = (V,A)$ representing a road network, where vertices V represent road intersections and arcs A represent directed links between intersections. The decision variables x_{ij} represent, for each arc $(i,j) \in A$, the rate of vehicles entering arc (i,j) in a steady state situation. Each arc is also assigned a latency function $t_{ij}(x_{ij})$ representing the time spent by each user traversing arc (i,j) as a function of the entering flow. The latency function considered in Angelelli et al. (2016b) is the one proposed by the U.S. Bureau of Public Roads,

$$t_{ij}(x_{ij}) = t_{ij}^{FF}[1 + 0.15(\frac{x_{ij}}{u_{ij}})^4]$$

, and used in Jahn et al. (2005), but w.l.o.g. any other non-decreasing convex function could be used. Obviously, parameters t_{ij}^{FF} (free-flow travel time) and u_{ij} (capacity) are provided for each arc. The normal length of each arc $(i,j) \in A$ is measured by the free-flow traveling time $t_{ij}^{FF} = t_{ij}(0)$. A set C of OD pairs is also given where each OD pair $c \in C$ is defined by its origin $O_c \in V$, its destination $D_c \in V$ and a positive demand d_c representing the rate of vehicles entering the network in O_c with destination D_c.

The linear constrained system optimum model admits, for each OD pair $c \in C$, the set of feasible paths K_c^{γ} given by those paths whose normal length does not exceed, in percentage, the OD pair shortest path by a fixed maximum travel inconvenience γ. The relative difference between the normal length of path k and the shortest path for OD pair c is called *path travel inconvenience*.

The decision variables y_{ck} represent, for each OD pair $c \in C$ and feasible path $k \in K_c^{\gamma}$, the amount of demand to be routed on path k. Relationship between variables x_{ij} and y_{ck} is expressed by the following equation

$$x_{ij} = \sum_{c \in C} \sum_{k \in K_c^{\gamma}} a_{c,p}^{ij} y_{ck},$$

where $a_{c,k}^{ij}$ is a parameter with value 1 if arc (i,j) is traversed by path $k \in K_c^{\gamma}$ and 0 otherwise.

The objective of the constrained system optimum formulation presented in Jahn et al. (2005) is the minimization of the total travel times in steady traffic conditions

$$\sum_{(i,j) \in A} t_{ij}(x_{ij}) x_{ij}.$$

The idea of the linear constrained system optimum model proposed in Angelelli et al. (2016b) is to approximate each non-linear term $F_{ij}(x_{ij}) = t_{ij}(x_{ij})x_{ij}$ of the objective function by a piecewise linear convex function on a fixed interval $[0, U_{ij}]$, where U_{ij} is a given upper bound on the flow x_{ij}. This given upper bound is assumed to be high enough to keep the model feasible also considering $\gamma = 0\%$. Given an accuracy parameter n, the range $[0, U_{ij}]$ is partitioned in n intervals at break-points $B = \{b_{ij}^0 = 0, b_{ij}^1, ..., b_{ij}^{n-1}, b_{ij}^n = U_{ij}\}$ with corresponding values $F = \{f_{ij}^0 = F_{ij}(0) = 0, f_{ij}^1 = F_{ij}(b_{ij}^1), ..., f_{ij}^n = F_{ij}(U_{ij})\}$. The width of the intervals is given by $\Delta_{ij}^h = b_{ij}^h - b_{ij}^{h-1}$ $(h = 1, \ldots, n)$.

The piecewise linear approximation of $F_{ij}(x)$ is given by $\sigma_{ij}(x)$:

$$F_{ij}(x) \approx \sigma_{ij}(x) = \begin{cases} f_{ij}^0 + \frac{f_{ij}^1 - f_{ij}^0}{\Delta_{ij}^1}(x - b_{ij}^0) & x \in [b_{ij}^0, b_{ij}^1] \\ f_{ij}^1 + \frac{f_{ij}^2 - f_{ij}^1}{\Delta_{ij}^2}(x - b_{ij}^1) & x \in (b_{ij}^1, b_{ij}^2] \\ ..... & .... \\ f_{ij}^{n-1} + \frac{f_{ij}^n - f_{ij}^{n-1}}{\Delta_{ij}^n}(x - b_{ij}^{n-1}) & x \in (b_{ij}^{n-1}, b_{ij}^n]. \end{cases}$$

The resulting model is the linear constrained system optimum LIN-C-SO(n) formulated as follows:

The LIN-C-SO(n) model

$$\min \quad \sum_{(ij)\in A} \sigma_{ij}$$

$$x_{ij} = \sum_{h=1}^{n} \lambda_{ij}^h \qquad \forall (i,j) \in A \tag{1}$$

$$\sigma_{ij} = \sum_{h=1}^{n} \frac{f_{ij}^h - f_{ij}^{h-1}}{\Delta_{ij}^h} \lambda_{ij}^h \qquad \forall (i,j) \in A \tag{2}$$

$$0 \le \lambda_{ij}^h \le \Delta_{ij}^h \qquad \forall (i,j) \in A \quad \forall h = 1,...,n \tag{3}$$

$$x_{ij} = \sum_{c\in C} \sum_{k \in K_c^\gamma} a_{ij}^{kc} y_{ck} \qquad \forall (i,j) \in A \tag{4}$$

$$d_c = \sum_{k \in K_c^\gamma} y_{ck} \qquad \forall c \in C \tag{5}$$

$$x_{ij} \ge 0 \qquad \forall (i,j) \in A \tag{6}$$

$$y_{ck} \ge 0 \qquad \forall c \in C \quad \forall k \in K_c^\gamma. \tag{7}$$

Auxiliary variable λ_{ij}^h represents a portion of the flow x_{ij} in interval $[b_{ij}^{h-1}, b_{ij}^h]$ as in constraints (3). The arc flow on arc (i,j) is obtained by summing over all λ_{ij}^h variables in constraints (1). Constraints (2) set the arc travel time as the sum over all the pieces in the piecewise function of the h-th slope multiplied by the corresponding λ_{ij}^h. Constraints (4) set the flow rate x_{ij} equal to the sum of flows on paths containing the arc (i,j). Constraints (5) guarantee that the demands d_c are completely routed on their feasible paths. Constraints (6)-(7) guarantee that variables x_{ij} and y_{ck} are non-negative. Table 1 summarizes the notation.

Linear constrained system optimum notation

Sets

V	set of vertices
A	set of arcs
C	set of OD pairs
K_c^γ	set of eligible paths for $c \in C$ with maximum travel inconvenience γ

Parameters

γ	maximum travel inconvenience
d_c	flow rate for OD pair $c \in C$
u_{ij}	capacity of arc $(i,j) \in A$
U_{ij}	maximum flow allowed on arc $(i,j) \in A$
t_{ij}^{FF}	free-flow travel time of arc $(i,j) \in A$
a_{ij}^{kc}	1 if path $k \in K_c^\gamma$ contains arc $(i,j) \in A$, 0 otherwise
n	number of intervals
b_{ij}^h	h-th breakpoint related to the range $[0, U_{ij}]$, $(h = 0, \ldots, n)$)
f_{ij}^h	value of the function $F(x_{ij})$ at breakpoint b_{ij}^h, $(h = 0, \ldots, n)$
$\Delta_{ij}^h = b_{ij}^h - b_{ij}^{h-1}$	$h-th$ interval size, $(h = 1, \ldots, n)$

Decision variables

y_{ck}	flow rate of OD pair $c \in C$ routed on path $k \in K_c^\gamma$
x_{ij}	total flow rate entering arc $(i,j) \in A$: $x_{ij} = \sum_{c \in C} \sum_{k \in K_c^\gamma} a_{ij}^{kc} y_{ck}$

Auxiliary decision variables

σ_{ij}	total travel time multiplied by x_{ij} evaluated using the piecewise function
λ_{ij}^h	amount of vehicles in the $h-th$ interval of piecewise function of arc (i,j)

Table 1. Notation related to the LIN-C-SO(n) model

3 The HI-GEN algorithm

The *Heuristic Iterative path GENeration* (HI-GEN) algorithm aims at generating a near optimal solution of model LIN-C-SO(n) by solving a sequence of restricted versions of it. The idea is to start with a solution provided by a minimal set of paths and iteratively add a few paths to the current set in order to improve the current solution. Algorithm HI-GEN stops when no improvement can be guaranteed.

More in detail, each arc (i, j) of the network graph is first assigned a travel time equal to its normal length which is consistent with its latency function with null flow; the shortest/fastest paths from origin to destination of every OD pair $c \in C$ initialises a current path set, and a restricted model R-LIN-C-SO(l) is built similar to LIN-C-SO(n) by substituting the whole set of feasible paths with the current path set. Parameter $l < n$ indicates that a less accurate approximation of the latency function to be used in the objective function of the optimization model. This is done to make the model faster to solve. Model R-LIN-C-SO(l) is thus solved producing a first traffic assignment. Note that a feasible solution for R-LIN-C-SO(l) is also feasible for LIN-C-SO(n). The traffic flows of the solution are used to redefine the traveling time of each arc according to its latency function. On this modified graph a new shortest path is computed for every OD pair and added to the current path set. The process is iterated as far as at least one new and feasible path is generated. Otherwise, the algorithm stops.

The HI-GEN algorithm is sketched in Algorithm 1.

Algorithm 1: HI-GEN algorithm

input : G : graph of the road network,
C : set of OD pairs,
γ : maximum travel inconvenience,
n, l : approximation levels of the latency function
output: x : heuristic solution of LIN-C-SO(n)
global : G, C, γ, P^{Curr}

– $P^{Curr} := \emptyset$;
– $L :=$ set of shortest paths for each $c \in C$ from origin O_c to destination D_c with respect to the arc normal length;

while $L \neq \emptyset$ **do**
  – $P^{Curr} := P^{Curr} \bigcup L$;
  – $x :=$ optimal solution of R-LIN-C-SO(l) on path set P^{Curr};
  – $L := findCheaperPaths(x)$;
– $x :=$ optimal solution of R-LIN-C-SO(n) on path set P^{Curr};
– **return** x

P^{Curr} represents the current path set and first initialized as empty while L is an auxiliary set containing paths found at each iteration. It is initialized with the shortest path with respect to the normal length for each OD pair. While the auxiliary set is not empty, the algorithm will go through three steps. First, the auxiliary set L is added to P_{Curr} and, then, the R-LIN-C-SO(l) is run on the augmented P_{Curr} set. The latter step allow us to find optimal flows that are used in the $L := findCheaperPaths(x)$ function in order to find new paths to include in the current path set. When the algorithm is not able to find new paths, the R-LIN-C-SO(n) is run considering the P_{curr} set constructed during the iterations.

3.1 Searching for an improving set of paths

At each iteration of the HI-GEN algorithm, the R-LIN-C-SO(l) model provides the current approximation x of the optimal solution of model LIN-C-SO(n). The objective of the routine $findCheaperPaths$ is to find, for at least one OD pair $c \in C$, a path p with the two following properties: $p \in K_c^\gamma$, and travel time on p in the current traffic assignment x is less than any other path already in the current path set for OD pair c. We first search the shortest path according to the arc latency function valued by solution x (i.e. $t_{ij} = t_{ij}(x_{ij})$), then we check the feasibility of such path. If no path with the required properties is found, the routine fails and returns an empty set.

Routine $findCheaperPaths$ is sketched in Algorithm 2.

Algorithm 2: findCheaperPaths

input : x : traffic flow
output: L improving path set
global : G, C, γ, P^{Curr}

– $L := \emptyset$;

for $c \in C$ **do**
  – $p :=$ shortest path in G from O_c to D_c with respect to arc lengths $t_{ij} = t_{ij}(x_{ij})$;
  – $nlp :=$ length of p with respect to arc normal length $t_{ij} = t_{ij}(0)$;
  – $nlsp :=$ length of shortest path from O_c to D_c with respect to arc normal length $t_{ij} = t_{ij}(0)$;
  if $nlp \leq (1+\gamma) nlsp \wedge p \notin P^{Curr}$ **then**
    – $L := L \bigcup \{p\}$;

– **return** L

The set L represents an auxiliary path set in which new paths are added and is initialized as empty. For each OD pair $c \in C$, the algorithm searches the shortest path considering $t_{ij} = t_{ij}(x_{ij})$ as arc length and if feasible it can be added to L. The feasibility check has been constructed according to the one used in

constructing the path in the LIN-C-SO(n) model, i.e. using the normal lengths as arc lengths.

3.2 An example

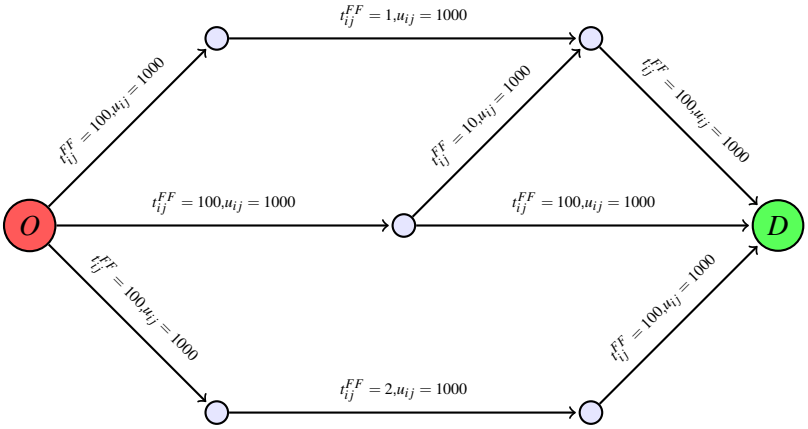
The HI-GEN algorithm aims at producing a traffic assignment on a small path set that is very near to the assignment produced by the LIN-C-SO(n) model. In Figures 1, 2 and 3 we provide an example instance with $l = 100$ and $\gamma = 30\%$. In Figure 1(a) the free-flow travel times and capacities for each arc are shown. At the beginning of the first iteration path set P^{Curr} contains the shortest path $path_1$ from origin O to destination D (Figure 1(b)). Figure 1(c) illustrates the solution provided by R-LIN-C-SO(l) solved on P^{Curr}. The obtained flows x_{ij} are used by the $findCheaperPaths$ function to compute the shortest path $path_2$ on the network with updated travel times $t_{ij}(x_{ij})$ (Figure 2(a)). In the second iteration $path_2$ is added to P^{Curr} and R-LIN-C-SO(l) model is solved on the new path set; Figure 2(b) depicts the new solution. Note that travel time on $path_1$ is 202.4 while travel time on $path_2$ is 202.73. A new shortest path $path_3$ is found in the network with updated flows (Figure 2(c)) and added to P^{Curr} at the beginning of the third iteration. The solution of the R-LIN-C-SO(l) model is shown in Figure 3(a). A new run of $findCheaperPaths$ function returns again $path_1$ (Figure 3(b)) and, hence, fails (no new path found as in Figure 3(b)), iterations stop and the R-LIN-C-SO(n) model is run in order to find the HI-GEN solution as in Figure 3(c) There are three paths in P^{Curr} and 350 units are assigned to $path_1$, 350 to $path_2$ and 300 to $path_3$. The total travel time is 201339.09 seconds.

Finally, in Figure 4 the LIN-C-SO(n) solution is provided. The optimal solution assigns 374.5 units to $path_1$, 339.5 units to $path_2$ and 286 units to $path_3$. Total travel time, evaluated by means of the non linear latency function with the optimal solution as input, is 201331.49.

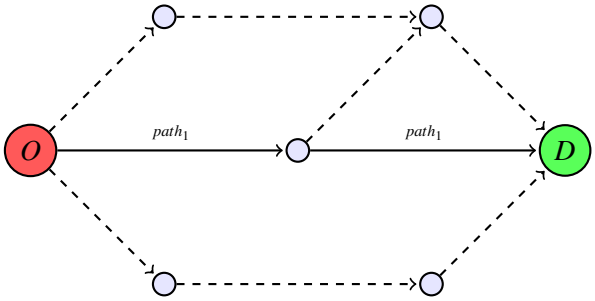
4 Computational results

The HI-GEN and the CE algorithms were implemented in Java, and the optimization models were solved by CPLEX 12.6.0. The experiments were run on a Windows 64-bit computer with Intel Xeon processor E5-1650, 3.50 GHz, and 64 GB Ram. In all experiments, parameters U_{ij} are set as four times the capacity u_{ij} of the corresponding arc. Break-points are uniformly distributed in $[0, U_{ij}]$ so that $\Delta_{ij}^h = U_{ij}/n$ for $h = 1, \ldots, n$.

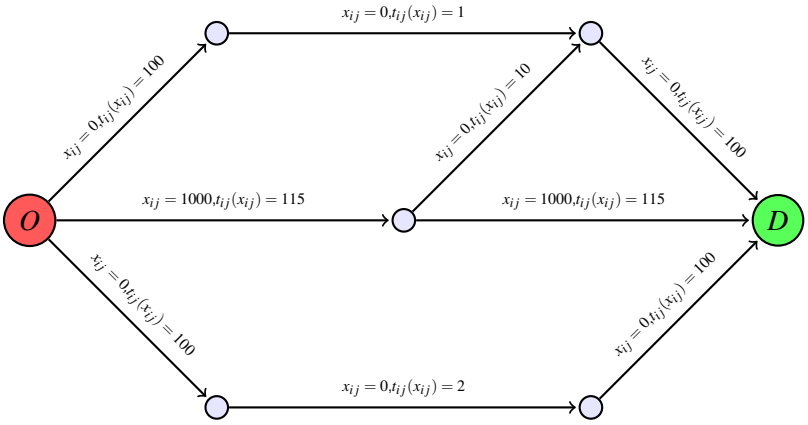
Experiments, devoted to compare algorithm CE to algorithm HI-GEN, are organized in two parts. In the first part, experiments are carried out on 40 network graphs with 150 nodes, values of γ range from 5% to 25% with step 5%, the accuracy level n used in LIN-C-SO(n) is fixed to 1000 while several values for the



(a) Initial network with t_{ij}^{FF} and u_{ij} for each arc

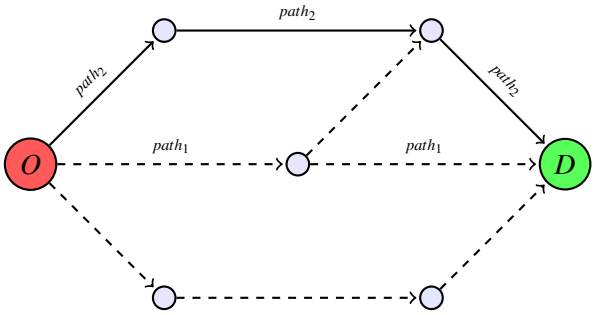


(b) The shortest path for OD pair O

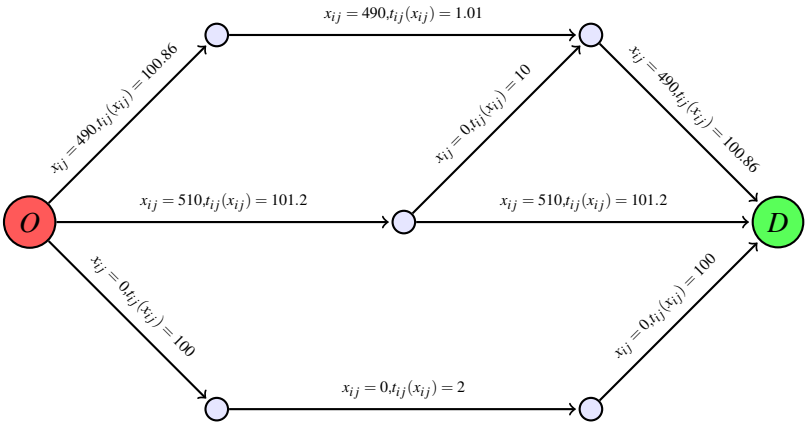


(c) Current solution and modified arc travel times

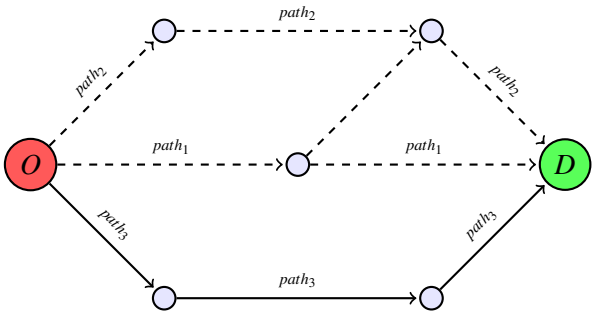
Figure 1. An example with $l = 100$ (continues in Figure 2)



(a) A cheaper path for O

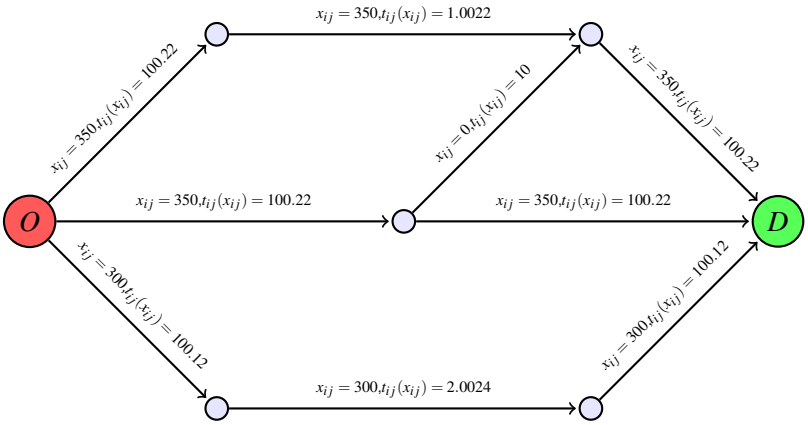


(b) Current solution and modified arc travel times

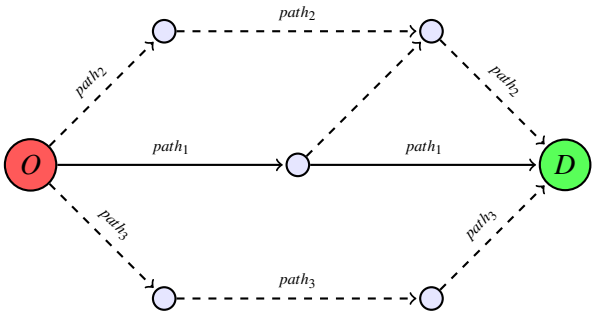


(c) Another cheaper path for O

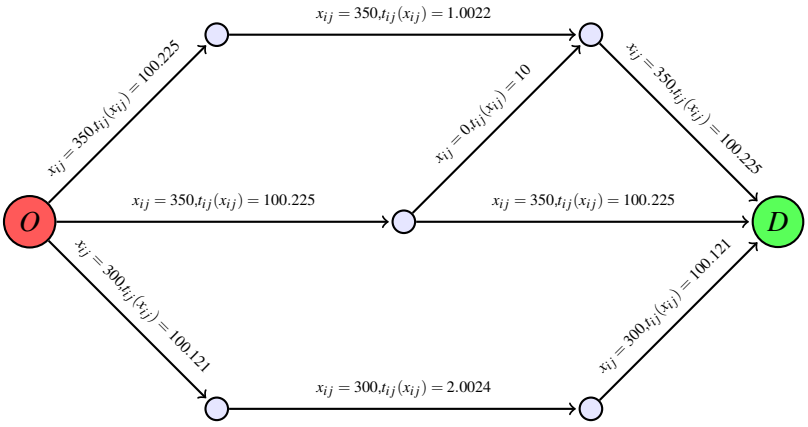
Figure 2. An example with $l = 100$



(a) Current solution and modified arc travel times

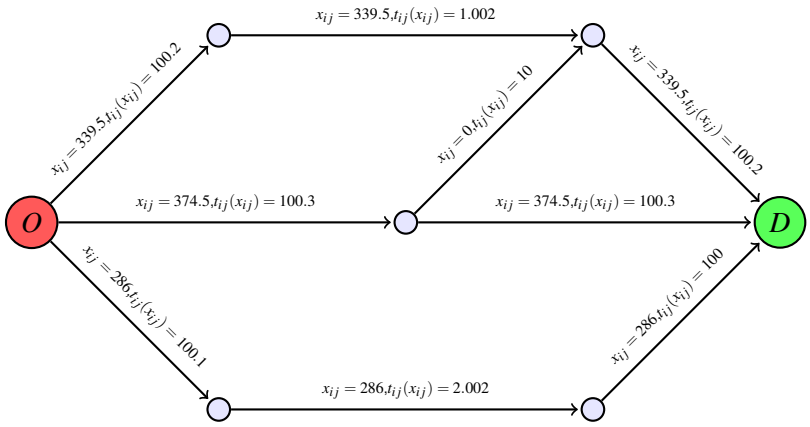


(b) No new cheaper paths can be found



(c) Solution of the R-LIN-C-SO(n) model

Figure 3. An example with $l = 100$



(a) Optimal solution of the LIN-C-SO(n) model

Figure 4. Solution of the LIN-C-SO(n) CE

accuracy l used in algorithm HI-GEN are tested: $l = 100, 300, 500, 700, 1000$. The second part is devoted to comparing algorithm CE to HI-GEN when instance size increases. Experiments were carried out using 8 network graphs, with up to 330 nodes, and values of γ ranging from 5% to 25% with step 5%. Here, the accuracy level l used in HI-GEN algorithm is $l = 100$.

The instances were generated taking into account different demand patterns and point attractiveness distributions as explained thoroughly in Angelelli et al. (2016a) and are available at `http://or-brescia.unibs.it/instances`. The statistics collected for each experiment are described in Section 4.1. Results for the 150 nodes network graphs are presented and discussed in Section 4.2. Results for the increasing size of network graphs are presented and discussed in Section 4.3.

4.1 Statistics

A number of statistics on the CE and the HI-GEN algorithms are collected or computed on all the tested instances.

- COMPUTATIONAL TIME. The computational time is computed considering the total computational time. In particular, the computational time accounted for CE algorithm includes the time needed to generate paths and solve the LIN-C-SO(n) model.
- SOLUTION QUALITY.
  - total travel time θ_{CE} and θ_{HI-GEN}: solution value produced by CE and HI-GEN, respectively;

– optimality gap $\Theta_\gamma = \frac{\theta_{HI-GEN} - \theta_{CE}}{\theta_{CE}}$.

- NETWORK CONGESTION. Fraction of arcs falling in each of the following four classes for different γ values:
  - unused arcs ($x_{ij}/u_{ij} = 0$);
  - non-congested arcs ($0 < x_{ij}/u_{ij} \leq 1$);
  - lightly congested arcs ($1 < x_{ij}/u_{ij} \leq 1.5$);
  - heavily congested arcs ($1.5 < x_{ij}/u_{ij}$).
- MEMORY USAGE. The number of generated paths.

4.2 Comparison of the HI-GEN solution with the CE solution

As already mentioned, in this section we summarize the results obtained on 40 networks with 150 nodes and values of γ ranging from 5% to 25% with step 5%. In Table 2, the CE and HI-GEN computational times are shown for different γ values and for different accuracy levels l. Algorithm HI-GEN is less time consuming than CE and time savings grow as growing values of γ are considered. In fact, considering an accuracy level $l = 100$ and $\gamma = 5\%$ algorithm CE, on average, spends about twice the time needed by HI-GEN. However, considering an accuracy level $l = 100$ and $\gamma = 25\%$, on average, algorithm CE uses 107 seconds while HI-GEN spends 6.8 seconds, i.e. only 6.4% of CE.

	HI-GEN algorithm l					CE algorithm n
	100	300	500	700	1000	1000
$\gamma = 5\%$	5.9	8.0	10.2	13.3	15.2	9.8
$\gamma = 10\%$	5.9	8.1	10.6	13.8	15.4	15.8
$\gamma = 15\%$	6.1	7.9	10.3	13.5	14.9	27.0
$\gamma = 20\%$	6.6	7.7	10.3	13.3	14.8	44.1
$\gamma = 25\%$	6.8	7.9	14.6	13.5	16.0	107.1

Table 2. Computational time (sec)

In Tables 3 and 4 statistics on the optimality gap Θ_γ are shown. In Table 3, the average Θ_γ over all instances and for different γ and l values is shown. Note that, for $\gamma = 5\%$, the average Θ_γ is around 0.278% for all values of l. For higher values of γ, the average Θ_γ reduces until an average value 0.096% is obtained

with $\gamma = 25\%$. From the point of view of the average Θ_γ, differences in considering different l values are negligible. In Table 4 the maximum Θ_γ over all instances and for different γ and l values is shown. The maximum value of Θ_γ follows the behaviour of the average Θ_γ, i.e. it decreases with the increasing of γ. Note that, for $\gamma = 5\%$, the maximum Θ_γ is around 1.14% for all values of l while with $\gamma = 25\%$ the maximum Θ_γ is 0.212% with $l = 100$ and 0.209% with all the other l values. In order to statistically test the differences in the average optimality gapwith different levels of l on the proposed instances, we have used a t-student test with, as null hypothesis, the equivalence between the results. We have obtained that, with a significance value of 0.05, the hypothesis of no difference between the average errors in using different levels of l is accepted.

	HI-GEN algorithm l				
	100	300	500	700	1000
$\gamma = 5\%$	0.2791	0.2780	0.2784	0.2784	0.2783
$\gamma = 10\%$	0.1675	0.1645	0.1646	0.1646	0.1646
$\gamma = 15\%$	0.1263	0.1262	0.1260	0.1261	0.1260
$\gamma = 20\%$	0.1023	0.1016	0.1011	0.1014	0.1014
$\gamma = 25\%$	0.0960	0.0955	0.0955	0.0958	0.0956

Table 3. Average optimality gap Θ_γ (%)

	HI-GEN algorithm l				
	100	300	500	700	1000
$\gamma = 5\%$	1.1444	1.1417	1.1417	1.1417	1.1417
$\gamma = 10\%$	0.6018	0.6018	0.6018	0.6019	0.6018
$\gamma = 15\%$	0.3910	0.3894	0.3900	0.3900	0.3900
$\gamma = 20\%$	0.2574	0.2545	0.2497	0.2497	0.2497
$\gamma = 25\%$	0.2123	0.2090	0.2090	0.2090	0.2090

Table 4. Maximum optimality gap Θ_γ (%)

In Table 5, the Θ_γ distribution for different l and γ values is shown. For each γ value, instances are classified into five different classes calculated as classes of percentage of the maximum Θ_γ value over all instances. For each class, we indicate the percentage of instances with a Θ_γ falling in that class. Considering $\gamma = 5, 10, 20\%$, there are no differences among different values of l while considering $\gamma = 15, 25\%$ there are very few differences. Note that, for all l values and $\gamma = 5, 10, 15\%$, 75% of the instances are experiencing Θ_γ values that are lower than 40% of the maximum Θ_γ. For all l values and $\gamma = 20, 25\%$, half of the instances

are experiencing Θ_γ values that are lower than 40% of the maximum Θ_γ. Globally, less than 15% of the instances are experiencing Θ_γ values that are higher than 80% of the maximum error. In order to statistically test the differences in optimality gapdistribution in using different levels of l on the proposed instances, we have used a t-student test with, as null hypothesis, the equivalence between the results. We have obtained that, with a significance value of 0.05, the hypothesis of no difference in the optimality gapdistribution in using different levels of l is accepted.

	Classes	γ				
		5%	10%	15%	20%	25%
$l = 100$	0-20 %	70	60	45	35	25
	20-40 %	10	20	30	15	25
	40-60 %	0	10	10	30	25
	60-80 %	10	0	5	5	10
	80-100 %	10	10	10	15	15
$l = 300$	0-20 %	70	60	45	35	20
	20-40 %	10	20	25	15	30
	40-60 %	0	10	15	30	20
	60-80 %	10	0	5	5	15
	80-100 %	10	10	10	15	15
$l = 500$	0-20 %	70	60	45	35	15
	20-40 %	10	20	25	15	35
	40-60 %	0	10	15	30	25
	60-80 %	10	0	5	5	10
	80-100 %	10	10	10	15	15
$l = 700$	0-20 %	70	60	45	35	15
	20-40 %	10	20	25	15	35
	40-60 %	0	10	15	30	25
	60-80 %	10	0	5	5	10
	80-100 %	10	10	10	15	15
$l = 1000$	0-20 %	70	60	45	35	15
	20-40 %	10	20	25	15	35
	40-60 %	0	10	15	30	25
	60-80 %	10	0	5	5	10
	80-100 %	10	10	10	15	15

Table 5. Θ_γ distribution (%) with respect to the maximum Θ_γ

In order to compare the memory usage of algorithm HI-GEN with respect to CE, we report the number of paths generated by the two methods and for different values of l in Table 6. The number of paths generated with the CE rapidly grows with increasing values of γ. For $l = 100$ and $\gamma = 25\%$ the percentage of paths generated by the HI-GEN algorithm is 0.39% of the paths generated by the CE. Considering higher values

of l, memory usage remains almost steady.

		γ				
		5%	10%	15%	20%	25%
CE algorithm $n = 1000$	paths	16365	43305	91046	160740	452059
$l = 100$	HI-GEN paths	1637	1685	1706	1730	1744
	% HI-GEN on CE	10.00	3.89	1.87	1.08	0.39
$l = 300$	HI-GEN paths	1642	1688	1711	1731	1749
	% HI-GEN on CE	10.03	3.90	1.88	1.08	0.39
$l = 500$	HI-GEN paths	1643	1686	1710	1731	1747
	% HI-GEN on CE	10.01	3.89	1.87	1.08	0.39
$l = 700$	HI-GEN paths	1643	1687	1711	1732	1746
	% HI-GEN on CE	10.00	3.90	1.88	1.08	0.39
$l = 1000$	HI-GEN paths	1643	1687	1711	1732	1747
	% HI-GEN on CE	10.01	3.89	1.87	1.08	0.39

Table 6. Number of generated paths by CE and HI-GEN algorithms

In Figure 7 the distribution of arc congestion level in different congestion classes is shown. Note that these values are averaged over all instances. Considering all values of parameter l, there are negligible differences among arc congestion distributions. This means that the percentage of arcs experiencing a certain congestion level remains almost steady when the accuracy level in searching paths is increased (or decreased). Regarding the arc congestion level produced by algorithm CE, there are relevant differences when compared to the HI-GEN one. The unused arc percentage is larger than the HI-GEN one while the non-congested percentage is lower than the HI-GEN one. Furthermore, CE assigns demands to paths in such a way no heavily congested arcs exist while a small heavily congestion percentage is produced by the HI-GEN assignment when $\gamma = 5\%$. Even though the objective function does not explicitly reduces the arc utilization, the increase of the arc utilization affects negatively the arc travel time and, therefore, the model, implicitly, tends to keep the arc utilization at low levels.

4.3 The HI-GEN algorithm on increasing size instances

Experiments were carried out considering 8 networks, with a number of nodes that ranges from 120 to 330 and with γ ranging from 5% to 25% with step 5%. We use a single value of accuracy level $l = 100$ since, as shown in Section 4.2, differences among different accuracy levels in generating paths are negligible. In Tables 8 and 9 the collected statistics are presented. At each step the number of nodes in the network is

		Congestion distribution			
		Unused	Non-congested	Lightly congested	Heavily congested
$\gamma = 5\%$	l=100	10.84	88.49	0.67	0.01
	l=300	10.82	88.51	0.67	0.01
	l=500	10.81	88.52	0.67	0.01
	l=700	10.81	88.52	0.67	0.01
	l=1000	10.81	88.52	0.67	0.01
	CE n=1000	12.06	87.79	0.15	0
$\gamma = 10\%$	l=100	10.92	88.60	0.48	0
	l=300	10.91	88.62	0.48	0
	l=500	10.91	88.62	0.48	0
	l=700	10.91	88.62	0.48	0
	l=1000	10.90	88.63	0.48	0
	CE n=1000	12.22	87.75	0.03	0
$\gamma = 15\%$	l=100	10.87	88.86	0.28	0
	l=300	10.87	88.86	0.28	0
	l=500	10.87	88.86	0.28	0
	l=700	10.87	88.86	0.28	0
	l=1000	10.86	88.87	0.28	0
	CE n=1000	12.24	87.74	0.02	0
$\gamma = 20\%$	l=100	10.87	89.03	0.10	0
	l=300	10.87	89.03	0.10	0
	l=500	10.84	89.04	0.11	0
	l=700	10.85	89.03	0.11	0
	l=1000	10.85	89.03	0.11	0
	CE n=1000	12.26	87.72	0.02	0
$\gamma = 25\%$	l=100	10.89	89.03	0.08	0
	l=300	10.90	89.02	0.08	0
	l=500	10.90	89.02	0.08	0
	l=700	10.88	89.04	0.08	0
	l=1000	10.90	89.02	0.08	0
	CE n=1000	12.26	87.72	0.02	0

Table 7. Arc utilization distribution (%)

	Stats	γ 5%	10%	15%	20%	25%
120 nodes	Paths CE	5351	12727	21227	30520	45757
	Paths HI-GEN	1285	1524	1676	1681	1696
	Time CE (sec)	4(1)	7(2)	9(3)	12(5)	14(7)
	Time HI-GEN (sec)	4	4	6	5	5
	optimality gap Θ_γ (%)	3.37	1.24	0.63	0.59	0.59
150 nodes	Paths CE	17364	46817	99225	177344	525756
	Paths HI-GEN	1678	1776	1835	1846	1873
	Time CE (sec)	10(4)	16(8)	29(17)	48(30)	133(79)
	Time HI-GEN (sec)	6	6	7	6	7
	optimality gap Θ_γ (%)	1.01	0.52	0.32	0.19	0.17
180 nodes	Paths CE	14875	68531	205590	539530	1529343
	Paths HI-GEN	1474	1546	1598	1626	1639
	Time CE (sec)	12(4)	27(14)	61(37)	153(94)	420(247)
	Time HI-GEN (sec)	6	6	6	6	6
	optimality gap Θ_γ (%)	3.75	0.27	0.23	0.23	0.23
210 nodes	Paths CE	17176	106047	416845	1366050	4652132
	Paths HI-GEN	1418	1476	1512	1518	1520
	Time CE (sec)	16(6)	43(25)	135(83)	422(252)	1513(812)
	Time HI-GEN (sec)	9	7	7	7	7
	optimality gap Θ_γ (%)	0.99	0.36	0.25	0.25	0.25

Table 8. Results for different number of nodes in the network

	Stats	γ 5%	10%	15%	20%	25%
240 nodes	Paths CE	25860	191367	930180	3779845	–
	Paths HI-GEN	1388	1434	1443	1444	1444
	Time CE (sec)	25(12)	83(53)	344(213)	1422(787)	–
	Time HI-GEN (sec)	12	9	9	9	17
	optimality gap Θ_γ (%)	0.27	0.16	0.15	0.15	–
270 nodes	Paths CE	37677	320042	1663763	7036847	–
	Paths HI-GEN	1324	1336	1337	1337	1337
	Time CE (sec)	34(18)	146(91)	651(379)	3076(1537)	–
	Time HI-GEN (sec)	14	10	11	16	42
	optimality gap Θ_γ (%)	0.12	0.12	0.12	0.12	–
300 nodes	Paths CE	60738	649617	3997117	–	–
	Paths HI-GEN	1482	1503	1503	1503	1503
	Time CE (sec)	53(32)	304(192)	1919(984)	–	–
	Time HI-GEN (sec)	14	13	19	13	14
	optimality gap Θ_γ (%)	0.07	0.06	0.06	–	–
330 nodes	Paths CE	98367	1383200	10074563	–	–
	Paths HI-GEN	1478	1523	1525	1527	1527
	Time CE (sec)	87(55)	705(422)	*(3396)	–	–
	Time HI-GEN (sec)	15	21	17	15	21
	optimality gap Θ_γ (%)	0.12	0.08			

Table 9. Results for different number of nodes in the network

increased by 30. Note that, for instances with a number of nodes greater than 150 and for some values of γ, the statistics for algorithm CE are not shown because either the path generation procedure ran out of memory or the solver running time exceeded a time threshold of 7200 seconds. Entries are denoted with '*' when the solver ran out of memory and with '-' when the time threshold was exceeded. The number of paths generated by algorithm CE grows dramatically faster with respect to algorithm HI-GEN as the number of nodes increases. In rows 'Time CE (sec)' the global computational time is shown (time spent to generate the set of feasible paths in brackets), while in rows 'Time HI-GEN (sec)' the HI-GEN computational time is shown. Rows 'Paths CE' and rows 'Paths HI-GEN ' represent, respectively, the number of generated paths. Finally, rows 'optimality gap Θ_γ (%)' represent the optimality gap produced by HI-GEN. With $\gamma = 25\%$, for the 120 nodes instance the number of paths generated by the HI-GEN algorithm is approximately 3.7% of the number of generated paths by the CE while for the 150 nodes instance the percentage is approximately 0.36%. When the instance size grows to 180 nodes the percentage decreases to 0.1% and, when it grows to 210 nodes, the percentage is 0.03%. This means that the HI-GEN algorithms produces a number of paths that is less than 3 orders of magnitude of the paths generated by CE. With higher instance sizes there is no available data for $\gamma = 25\%$ since the solver exceeded the time threshold or ran out of memory. However, regardless the value of γ, this percentage continues to decrease with the increase of the instance size.

Regarding the computational time, the time required by algorithm CE grows faster with respect to HI-GEN as the number of nodes increases. With $\gamma = 25\%$, for the 120 nodes instance the HI-GEN algorithm computational time is approximately 36% of the CE computational time while for the 150 nodes instance it is approximately 5.2%. When the instance size grows to 180 nodes the percentage decreases to 1.4% and when it grows to 210 nodes the percentage is 0.5%. This means that algorithm HI-GEN takes a time that is less than 2 orders of magnitude of the time required by CE. As for the number of paths, with higher instance sizes there is no available data for $\gamma = 25\%$ but the percentage continues to decrease with the increase of the instance size. For example, with a 330 nodes instance and $\gamma = 15\%$ the computational time required by HI-GEN is 0.5% of the time required only for the generation of the feasible path set by algorithm CE.

After the analysis of HI-GEN memory and time saving, some conclusions also on its effectiveness in generating high quality solutions can be derived. The optimality gap Θ_γ seems to reduce when the instance size grows. The maximum experienced value of Θ_γ is around 3.4% with 120 nodes and with $\tau = 5\%$ and the best value is experienced with a 300 nodes instance and $\gamma = 10\%$ where Θ_γ is 0.06%. The trend is a decrease of Θ_γ with the increase of γ and with the increase of instance size. The results suggest that the larger an

instance is the smaller is the HI-GEN optimality gap.

5 Conclusions

.

In this paper a heuristic algorithm generating the path set for the linear constrained system optimum model is presented. The computational complexity of the model is mainly affected by the number of paths that, in the worst case, grows exponentially with the number of nodes of the network. The computational experiments show that algorithm HI-GEN reduces by orders of magnitude the number of generated paths and, consequently by orders of magnitude the amount of memory usage and computational time. The results also show that the quality of the solutions produced by HI-GEN is very high since the HI-GEN solution is very close to the one obtained generating all feasible paths. On larger instances HI-GEN is able to return a solution in a few second where LIN-C-SO(n) is computationally intractable.

References

E. Angelelli, I. Arsik, V. Morandi, M. Savelsbergh, and G. Speranza. Proactive route guidance to avoid congestion. *Transportation Research Part B: Methodological*, 94:1–21, 2016a. doi: http://dx.doi.org/10.1016/j.trb.2016.08.015.

E. Angelelli, V. Morandi, M. Savelsbergh, and G. Speranza. System optimal routing of traffic flows with user constraints using linear programming. Technical Report 5-2016, University of Brescia, Department of Economics and Magagement, 2016b. URL http://www.unibs.it/dipartimenti/economia-e-management/ricerca/working-paper. (submitted).

E. Angelelli, V. Morandi, and G. Speranza. Heuristic path generation for the proactive route guidance problem. Technical Report 3-2016, University of Brescia, Department of Economics and Magagement, 2016c. URL http://www.unibs.it/dipartimenti/economia-e-management/ricerca/working-paper. (submitted).

J. Azevedo, M. Costa, J. Madeira, and E. Martins. An algorithm for the ranking of shortest paths. *European Journal of Operational Research*, 69:97–106, 1993.

M. Beckmann, C. McGuire, and C. B. Winsten. Studies in the economics of transportation. Technical report, RAND Corporation, 1956.

S. Bekhor and C. Prato. Effects of choice set composition in route choice modeling. In *Proceedings of the 11th Conference of the International Association for Travel Behavior Research*, pages 16–20. Kyoto Japan, 2006.

S. Bekhor, T. Toledo, and J. Prashker. Implementation issues of route choice models in path-based algorithms. In *11th international conference on travel behaviour research, Kyoto, Japan*, 2006.

D. Branston. Link capacity functions: A review. *Transportation Research*, 10:223–236, 1976.

T. de la Barra, B. Perez, and J. Anez. Multidimensional path search and assignment. In *PTRC Summer Annual Meeting, 21st, 1993, University of Manchester, United Kingdom*, 1993.

E. Frejinger and M. Bierlaire. Capturing correlation with subnetworks in route choice models. *Transportation Research Part B: Methodological*, 41:363–378, 2007.

O. Jahn, R. H. Möhring, and A. S. Schulz. Optimal routing of traffic flows with length restrictions in networks with congestion. In *Operations Research Proceedings 1999*, pages 437–442. Springer, 2000.

O. Jahn, R. Möhring, A. Schulz, and N. Stier-Moses. System-optimal routing of traffic flows with user constraints in networks with congestion. *Operations research*, 53:600–616, 2005.

L. J. LeBlanc, R. V. Helgason, and D. E. Boyce. Improved efficiency of the frank-wolfe algorithm for convex network programs. *Transportation Science*, 19:445–462, 1985.

D. Park and L. Rilett. Identifying multiple and reasonable paths in transportation networks: A heuristic approach. *Transportation Research Record: Journal of the Transportation Research Board*, pages 31–37, 1997.

C. Prato and S. Bekhor. Applying branch-and-bound technique to route choice set generation. *Transportation Research Record: Journal of the Transportation Research Board*, pages 19–28, 2006.

C. Prato and S. Bekhor. Modeling route choice behavior: how relevant is the composition of choice set? *Transportation Research Record: Journal of the Transportation Research Board*, 2007.

C. G. Prato. Route choice modeling: past, present and future research directions. *Journal of Choice Modelling*, 2:65–100, 2009.

M. S. Ramming. *Network knowledge and route choice*. PhD thesis, Massachusetts Institute of Technology, 2001.

G. Rose, M. A. Taylor, and P. Tisato. Estimating travel time functions for urban roads: options and issues. *Transportation Planning and Technology*, 14:63–82, 1989.

Y. Sheffi. *Urban transportation networks: equilibrium analysis with mathematical programming methods*. Prentice-Hall, 1985.

M. Treiber and A. Kesting. Traffic flow dynamics. *Traffic Flow Dynamics: Data, Models and Simulation, Springer-Verlag Berlin Heidelberg*, 2013.

N. Van der Zijpp and S. F. Catalano. Path enumeration by finding the constrained k-shortest paths. *Transportation Research Part B: Methodological*, 39:545–563, 2005.

J. G. Wardrop. Road paper. some theoretical aspects of road traffic research. *Proceedings of the institution of civil engineers*, 1:325–362, 1952.

J. Y. Yen. Finding the k shortest loopless paths in a network. *Management Science*, 17:712–716, 1971.