UNIVERSITY OF BERGAMO

School of Doctoral Studies
Doctoral degree in Analytics for Economics and Business
XXIX Cycle
SSD: Operations research

TITLE:

Linear programming models for a fair and efficient traffic assignment in congested road networks

Advisor:

Prof. Grazia Speranza

Doctor thesis

Valentina MORANDI

Student ID: 1031544

Academic year 2015/2016

Abstract

Solving the road congestion problem is one of the major issues in modern cities since it causes time wasting, pollution, higher industrial costs and huge road maintenance costs. Among possible congestion avoidance methods, traffic assignment is a valuable choice since it does not involve huge investments to expand the road network. Traffic assignments are traditionally devoted to two main perspectives on which the well-known Wardropian principles are inspired : the user equilibrium (user's perspective) and the system optimum (system perspective). User equilibrium is a user-driven traffic assignment in which each user chooses the most convenient path selfishly. It guarantees that fairness among users is respected since, when the equilibrium is reached, all users sharing the same origin and destination will experience the same travel time. The main drawback in a user equilibrium is that the system total travel time is not minimized. On the other hand, the system optimum is a system-wide traffic assignment in which drivers are routed on the network in such a way the total travel time is minimized but users might experience travel times that are higher than the other users travelling from the same origin to the same destination. Thus, there are drawbacks in using one of the two assignments that can be partially overcome by applying users' fairness considerations while minimizing a system-oriented objective. In the last decade, few attempts have been done to present a users' needs and system efficiency trade-off traffic assignment with non-linear programming techniques. In this thesis linear programming models for a fair and efficient traffic assignment in congested road networks are presented.

Acknowledgements

My sincerest thanks to my supervisor, Prof. Grazia Speranza, for introducing me to the operations research world, for helping me during the whole PhD course, for answering to all my questions, for the illuminating advices and for the infinite kindness. I am very grateful to Prof. Enrico Angelelli for introducing me to programming languages, for the shared knowledge and for teaching me the importance of being logical, accurate and meticulous in my research. Thanks to all my colleagues of the OR department of the University of Brescia for always supporting me. Part of this thesis is the result of a joint work with Prof. Martin Savelsbergh, thank you for your advices and suggestions.

Sincerely thanks to Prof. Bertocchi for having coordinated the PhD school with care and wisdom. She helped us in so many ways. Your memory will be invaluable to me.

To my mom and dad, for unconditioned support from the very beginning of my life and for teaching me the importance of studying, understanding and exploring the world. A special thanks to my little brother, Nicola. I hope you will have the same gratification from your studies I had during mine. I wish to thank my aunts, my cousins and the "little princess" Mariachiara. You gave me the strength to continue such a nice job despite the tiredness. Thanks also to Daniela for the help she has given me and still giving me.

To all my friends, I thank you for the support of all these years of study.

Last, I want to thank my beloved Andrea. Thank you for the time that you gave me, the patience and the sacrifices made to achieve my dreams.

Ringraziamenti

I miei piú sentiti ringraziamenti vanno alla Prof.ssa Grazia Speranza per avermi introdotto al mondo della ricerca operativa, per avermi aiutato durante tutto il mio corso di dottorato, per aver risposto a tutte le mie domande, per tutti i consigli illuminanti che mi ha dato e per l'infinita gentilezza mostratami. Sono molto grata al Prof. Enrico Angelelli per avermi introdotto alla programmazione e per avermi insegnato l'importanza di essere coerenti, accurati e meticolosi nel fare ricerca. Ringrazio anche tutto i miei colleghi di Ricerca Operativa dell'Universitá di Brescia per avermi sempre supportato. Parte di questa tesi é il risultato di un lavoro congiunto con il Prof. Martin Savelsbergh che intendo ringraziare per i consigli ed i suggerimenti ricevuti.

Un sentito ringraziamento alla Prof.ssa Marida Bertocchi per avere coordinato la scuola di dottorato con cura e saggezza. Ci ha aiutato in cosí tanti modi e la sua memoria sará preziosa per me.

A mamma e papá, per il supporto incondizionato dal principio della mia vita e per avermi insegnato l'importanza di studiare, capire ed esplorare il mondo. Un ringraziamento speciale al mio fratellino Nicola. Ti auguro di trarre la stessa soddisfazione dai tuoi studi che io ho avuto durante i miei. Desidero inoltre ringraziare le mie zie, i miei cugini e la "piccola principessa" Mariachiara. Mi avete dato la forza di continuare questo splendido percorso nonostante la stanchezza. Grazie anche a Daniela per l'aiuto che mi ha dato e che ancora mi sta dando.

A tutti i miei amici, desidero ringraziarvi per il supporto che mi avete dato in tutti questi anni.

Infine, desidero ringraziare il mio amato Andrea. Grazie per il tempo che mi hai dedicato, per la pazienza e per i sacrifici che hai fatto per permettermi di inseguire i miei sogni.

Contents

Introduction

Road congestion is becoming a serious problem in metropolitan areas where the traffic demand is steadily growing. Congestion is a significant burden in terms of wasted time, pollution, industrial costs and road maintenance and alleviating traffic volumes will become more and more urgent as the population grows. According to a new report by INRIX and Centre for Economics and Business Research (CEBR), the annual cost of traffic congestion and gridlocks on individual households and national economies in the U.S., U.K., France and Germany will raise to \$293 billion dollars in 2030 with a cumulative value of congestion cost, from now to 2030, near \$4.4 trillion. In some cases the road network has been extended to accommodate the growing demand but this is not always convenient in terms of costs/benefits trade-off. Regulating congestion without extending the current road network can be done using Intelligent Transportation Systems (ITS), such as ramp metering, reversible lanes, limited access roads, bus lanes, carpooling lanes, express toll lanes, congestion pricing mechanisms, variable message signs, etc., or coordinating traffic assignment. Traffic assignment concerns assigning paths to users in order to optimize an objective function depending on the goal the traffic regulator wants to achieve. Traffic assignment may be dynamic or static. Static traffic assignment is mainly used in studying the rush hour period since, during that period, traffic shows a steady-state behaviour where the main assumption is that demand is constant over time. When traffic demand exhibits a time-dependent behaviour only dynamic traffic assignment techniques have to be considered as reliable. Static traffic assignments are traditionally divided into two main approaches inspired by the well-known Wardropian principles: the user equilibrium and the system optimum. User equilibrium is a user-driven traffic assignment in which each user chooses the most convenient path selfishly. It guarantees that fairness among users is respected since, when the equilibrium is reached, all users sharing the same origin and destination will experience the same travel time. The main drawback in a user equilibrium is that the total travel time is not minimized. On the other hand, the system optimum is a system-wide traffic assignment in which drivers are routed on the network in such a way the total travel time is minimized but users might experience travel times that are higher than the other users travelling from the same origin to the same destination. Thus, there are drawbacks in using one of the two assignments that can be partially overcome by applying users' fairness considerations while minimizing a system-oriented objective. In literature there are a few attempts to overcome the limitations of the two assignments with non-linear programming techniques. The thesis focuses on linear programming based methods for coordinated static traffic assignment that consider the system perspective while focusing also on users' fairness. To this aim, in this thesis two different methodologies will be explained and deeply discussed: the proactive route guidance approach and the constrained system optimum problem. The former focuses on minimizing the average inconvenience experienced by users while guaranteeing a maximum network congestion level as low as possible. The latter focuses on minimizing the total travel time spent by users on the network. Both methodologies account for users' fairness by limiting the set of feasible paths for each Origin-Destination (OD) pair to the ones that do not exceed a fixed percentage of the OD pair shortest path length. The proposed methodologies differ for the assumptions made since the first one aims at eliminating congestion and it assumes that the arc travel time is constant while the second assumes that arc travel time depends on the flow that is currently traversing the arc. The thesis is structured as follows. First, a literature review on traffic assignment methods for the system optimization with users' fairness constraints is proposed. In the following chapters the proactive route guidance approach and the linear constrained system optimum problem are proposed. Since in solving the proactive route guidance approach and the linear constrained system optimum an exponential the complete enumeration of all feasible paths for each OD pair is required, complexity issues due to the exponential number of paths raise when the instance grows. Thus, for each methodology, a heuristic algorithm able to generate only a subset of feasible paths and to obtain a near-optimal solution is proposed. Then, the features of the road network instance generator used for the computational experiments are described.

0.1 Outline of the thesis

The thesis focuses on linear programming models optimizing the system while considering users' fairness. Chapter 1 is devoted to a review of the main traffic assignment models for the system optimization with users' fairness constraints. Chapter 2 introduces the proactive route guidance approach that aims at minimizing the average experienced user inconvenience while keeping the network uncongested or at the least possible congestion level. An exact solution method based on a hierarchy of linear programming models and requiring the complete path enumeration is proposed. In Chapter 3 a heuristic method able to find a suboptimal solution for the proactive route guidance approach is shown. In Chapter 4 the constrained system optimum problem aiming at minimizing the total travel time while choosing only convenient paths is introduced and a linear programming model, called linear constrained system optimum model, is proposed. In Chapter 5 a heuristic method able to find a suboptimal solution for the linear constrained system optimum model is presented. Finally, in Chapter 6 the instance generator, designed and implemented in order to create instances, has been described.

0.1.1 Chapter 1: System optimal traffic assignment with users' constraints: a literature review

Traditionally, traffic assignment has been studied from two points of views: user fairness and system optimality. User fairness is related to assigning paths to users in such a way no user will experience a shorter travel time with respect to the other users of the same OD pair. System optimality is achieved by optimizing the whole network performance assigning paths to users. Both perspectives present drawbacks and a trade-off between the two seems to be a valuable way to assign traffic. To this aim, the main attempts in considering the two perspectives are presented in the literature review.

0.1.2 Chapter 2: Proactive route guidance to avoid congestion

In this chapter the proactive route guidance approach based on linear programming models is proposed. The system optimality is achieved by minimizing the average inconvenience experienced by users while keeping the maximum network congestion level under the congestion threshold. User fairness is taken into account by limiting the set of available paths only to paths whose relative increase with respect to the shortest path is within a pre-definite limit. The proposed approach relies on the complete enumeration of all feasible paths from origin to destination for each OD pair and the set of available paths is generated by means of an ad-hoc algorithm. This chapter has given rise to the journal paper (Angelelli et al. (2016a)) published on Transportation Research B: Methodological.

0.1.3 Chapter 3: Heuristic path generation for the proactive route guidance approach

The proactive route guidance approach presented in Chapter 2 is based on a complete enumeration of all feasible paths for each origin-destination pair. When the instance size grows, the number of generated paths grows exponentially and, at a certain point, the approach becomes computationally intractable. To overcome this limitation, in Chapter 3 a heuristic path generation method is implemented in order to find a suboptimal solution of the proactive route guidance approach. Experiments show that the heuristic path generation produces very small optimality gaps. This chapter has given rise to a paper (Angelelli et al. (2016c)) that has been submitted for publication.

0.1.4 Chapter 4: System optimal routing of traffic flows with user constraints using linear programming

In the literature, the most known attempt to consider users' fairness while achieving the system optimality is the constrained system optimum problem. The constrained system optimum problem consists in minimizing the total travel time spent on the network using a limited path set containing, for each user, only to paths whose relative increase with respect to the shortest path is within a pre-definite limit. In the literature, all approaches to the constrained system optimum are non-linear. In this chapter a linear programming model to solve the constrained system optimum problem, called linear constrained system optimum, is presented. This chapter has given rise to a paper (Angelelli et al. (2016b)) that has been submitted for publication.

0.1.5 Chapter 5: Heuristic path generation for the linear constrained system optimum model

The linear programming formulation for the constrained system optimum presented in Chapter 4 is based on a complete enumeration of all feasible paths for each OD pair. When big size instances are considered, the number of generated paths becomes huge and, as for the proactive route guidance approach proposed in Chapter 2, the problem results computationally intractable. In Chapter 5 a heuristic path generation method is implemented in order to find a suboptimal solution of the linear constrained system optimum model that drastically reduces time and memory use with respect to the complete enumeration. Experiments show that the heuristic path generation produces very small optimality gaps. This chapter has given rise to a technical report (Angelelli et al. (2016d)) and it will be submitted for publication.

0.1.6 Chapter 6: On the instance generation

Generating instances for road network problems is a big issue. Either extracting a network from a real map or constructing instances from scratch should lead to huge problems. In this thesis an instance generator able to model different size of road networks, hierarchies of roads (number of lanes, safety distance, maximum speed allowed, etc.), to consider delays due to traffic lights and other regulations, different demand patterns and origin/destination dispersions is presented. A description of the parameters used in generating instances for the previous works is provided.

1. System optimal traffic assignment with users' constraints: a literature review

Traffic congestion on road networks has been recognized as one of most urgent problems in modern cities because it causes long travel times, vehicular queueing and, consequently, users' frustration. Delays also reduce productivity and, consequently, increase operational costs. Traffic congestion also injures the environment since it increases air pollution and CO_2 emissions. Furthermore, congestion influences a lot of economic decisions as the choice of the living place, the working place and the travelling mode for the population. In addition, congestion continues to increase because of the growing population and the increased motorization level. In many cases it is not convenient, or even not possible, to improve the road network by adding new roads or increasing existing roads capacity. Hence, alternative methods to reduce congestion have to be implemented. From the point of view of the ITS (Intelligent Transportation System) technologies, several congestion reduction methods have been implemented such as ramp metering, variable message signs and vehicle to vehicle communication. Reviews on these congestion reduction methods can be found in Papageorgiou and Kotsialos (2000), Peeta et al. (2000), Sichitiu and Kihl (2008) and Luo and Hubaux (2004). Also traffic limitations can be useful in order to reduce congestion. Common traffic limitations are limited access roads, carpooling lanes, express toll lanes, congestion pricing mechanisms, even and odd plate days and reversible lanes. In some cases the use of ITS technologies and/or traffic limitations has completely or partially solved the congestion problem. Another way to reduce congestion is traffic coordination, i.e. considering the whole system welfare and assigning paths to users in such a way the congestion problem is solved or, at least, kept at a minimum level. Traffic coordination may become an useful tool to reduce congestion also in view of a massive use of the autonomous vehicles in the future. A centralized system may optimize the network performance and paths may be assigned to vehicles according to an optimal assignment. However, traffic coordination can be applied also on current road networks if individual needs are taken into account. A centralized system optimizing network performance that assigns paths to user without any consideration about fairness among users will not be accepted by users. This is well known. Thus, coordinated traffic assignment on real road network has to be efficient from the system perspective but also fair for users. The literature review focuses on methods achieving efficient and fair traffic assignment that is the subject of the thesis. In Section 1.1 congestion definitions and measures are presented. In Section 1.2 the traditional approaches to traffic assignment are presented. Finally, in Section 1.3 coordinated traffic assignment approaches incorporating system and users' needs are presented.

1.1 Congestion definitions

Traffic congestion is the result of the imbalance between the network capacity and the demand. According to Falcocchio and Levinson (2015) congestion in transportation occurs when the occupancy of spaces by vehicles or people reaches unacceptable levels of discomfort or delay. Congestion phenomena are divided into two main categories: the recurring and the non-recurring congestion. According to Stopher (2004) and Falcocchio and Levinson (2015) the recurring congestion is the delay that travellers regularly experience during certain periods of time (for example, the rush hour during the morning commute). The non-recurring congestion is a delay due to not predictable events that disrupt the traffic flow such as car breakdowns, crashes, works in progress and bad weather conditions. We will focus only on recurring congestion since the scope of the thesis is to develop methods to eliminate its effects. But how is the recurring congestion measured? When does the recurring congestion appear? The recurring congestion level is usually measured by means of intensity, duration and variability measures. Since the whole thesis is devoted to solving the congestion problem during the rush hour and, according to Sheffi (1985), traffic during the rush hour exhibits a steady-state behaviour, we consider only intensity measures as the demand for transportation is constant over time. The intensity indicators are usually constructed in such a way that, when the indicator value exceeds a fixed threshold, congestion occurs. According to Falcocchio and Levinson (2015) different thresholds can be used to detect if congestion occurs and each threshold is suitable for specific traffic situations. In order to compare experienced speeds or experienced travel times with a fixed value, in congestion detection two measures called $FreeFlowSpeed$ and $FreeFlowTravelTime$ are used and are defined as the experienced speed and travel time when the network is empty, respectively. We define as $ExperiencedSpeed$ and $ExperiencedTravelTime$ the user experienced speed and travel time, respectively. An example of intensity measure is the congestion delay rate which is the difference between the inverse experienced speed and inverse free-flow speed (min/km), i.e. $\frac{1}{ExperiencedSpeed} - \frac{1}{FreeFlowSpeed}$. This measure is often used in measuring congestion on arterial roads and highways. Sometimes congestion level is obtained by aggregate data on how many hours are spent each day by each driver at a certain $ExperiencedSpeed$ with respect to the $FreeFlowSpeed$. This is an averaged result and is only used for survey purposes. Another example of intensity measure is the travel time index, i.e. the ratio between the $FreeFlowSpeed$ and the $ExperiencedSpeed$. Since in large cities networks it is not realistic to travel at the $FreeFlowSpeed$ during the peak

hour, then comparing *ExperiencedSpeed* with *FreeFlowSpeed* in metropolitan areas could be not reliable. However, this measure is appropriate for observing congestion behaviour over time in the same area (with the same *FreeFlowSpeed* value). The USA Transportation Research Board and Ryus et al. (2011) classifies congestion on roads with respect to the Level of Service (LoS), i.e. grades from A (free-flow) to F (forced breakdown congestion). The level of service is evaluated taking into account *ExperiencedSpeed* and travel time, delays, freedom to manoeuvre, safety, driving comfort and other parameters. For each of the different road classes they provide a speed-flow curve able to capture the current level of service depending on the *ExperiencedSpeed*.
Once the congestion level on a road network has been measured, a policy for the congestion avoidance has to be implemented. When dealing with recurrent congestion, collecting information is crucial in order to estimate the demand that will travel on the road segment. In Ben-Elia et al. (2013) an experiment with different levels of information accuracy is described and the negative effect of low information levels is demonstrated. However, even with full information provided, in case of bottlenecks it is necessary to reconsider network design features. This is the case of ramp metering studies (see Kachroo and Özbay (2011) for details). Many cities have developed congestion charging strategies for congestion reduction (see de Palma and Lindsey (2011) for details). Congestion charging can be developed in several ways like the facility-based strategies and cordons congestion charging. The facility-based congestion charging regards tolling roads, bridges and tunnels only on a few facilities. It can be a single point toll or a distance-based toll. Cordons congestion charging is an area-based charging method in which vehicles pay a toll to cross a cordon in the inbound or outbound direction or both. Another congestion charging scheme is the zonal scheme in which vehicles pay a fee to enter or exit a zone or to travel inside the zone. In de Palma and Lindsey (2011) some rules on which congestion charging scheme is suitable for each case of study era offered and a good review on congestion pricing technologies is provided. In Stopher (2004) a congestion charging situation, in which tolls are applied, is considered on an area that is most likely congested and a congestion charging situation in which tolls are distance-based. It points out that the latter is fairer than the former because tolls are spread along the journey in a progressive way and depend on how long the travel is.
Also congestion detection is a big issue for traffic regulators. In recent years many devices have been developed in order to estimate vehicle speed, safety distance between vehicles and other congestion parameters. Main methods are RFID sensors, CCTV cameras and vehicle to vehicle communication systems but, as pointed out in the introductory section, when the network infrastructure cannot be expanded or a no-toll policy is observed, the traffic coordination is a valuable choice in order to relieve or alleviate congestion on road network.

1.2 The traffic assignment problem

According to Patriksson (2015), transportation planning is usually divided into five steps: goal definition, base year inventory, model analysis, travel forecast and network evalu-

ation. The goal definition step is related to defining the participation, the committee structure and staff involved in the road network study. In this phase an agreement on goals and objectives of the work has to be defined. In the base year inventory step, all the data related to the network (arcs, capacities, etc.) and demand patterns has to be collected. Model analysis phase is devoted to finding the relationship between measured quantities (traffic flows and road congestion, for instance). Model analysis is the result of four different phases: trip generation, trip distribution, modal split and traffic assignment. Trip generation consists of finding the number of trips that originate and terminate in different zones of the studied area. Usually this phase is carried out considering socio-economic, geographic and land use features and the different zones are categorized by the main purpose as work, leisure or shopping areas. The result of the trip generation phase is the Origin-Destination (OD) matrix. In trip distribution phase some formulas to predict the demand of travellers from an origin zone to a destination zone have to be developed. A demand value is associated with each OD pair in the OD matrix. Modal split is a phase in which the mean of transport used by each traveller is determined. The number of travellers that choose a particular mean of transport depends mainly on travel cost in terms of monetary cost or travel time but, sometimes, also socio-economic factors affect the choice. Traffic assignment is devoted to assigning the demand from an origin to a destination to routes in a transportation network. This phase is particularly relevant because an estimate of traffic volumes and travel time is returned. Finally, the network evaluation is a phase in which alternative transportation networks and facilities benefits are evaluated and compared. In this literature review we will focus only on the traffic assignment phase of the model analysis step.
Traffic assignment is a method that assigns OD demands to paths on a transportation network. As input of traffic assignment an OD matrix, representing all the OD pairs with demands, and the network representation (usually a capacitated network) are required and the output is an estimate of the traffic flows on each link. The first attempt to implement a traffic assignment scheme dates back to just after the World War II: the so-called all-or-nothing assignment proposed in Campbell (1950). The main assumptions of the all-or-nothing assignment are that the road travel time is not correlated to the flow on the road and that all the demand of an OD pair is entirely assigned to the shortest path for that OD pair. The traffic scientists realised the all-or-nothing assignment was not realistic in modelling congested road networks and, thus, they tried taking into account congestion effects in computing travel times. The result was the so-called latency function, i.e. a function in which arc travel time depends on the number of vehicles entering the arc and on the arc capacity rate. A survey on the used latency functions is proposed in Branston (1976) and includes:

- $t = t_0 e^{\frac{x}{c}}$;
- $t = t_0 \alpha^{\beta \frac{x}{c}}$ where α and β are parameters;
- $t = t_0[1 + \alpha(\frac{x}{c})^{\beta}]$ where α and β are parameters;
- $t = \begin{cases} \frac{d}{S_0} & x \geq \delta \\ \frac{d}{S(x)} & x \leq \delta \end{cases}$ where d is the distance, S_0 is the free-flow speed and $S(x)$ the experienced speed with flow greater than δ. δ should be considered as the conges-

tion threshold.

The most used latency function is the one proposed by the Bureau of Public Roads (BPR) $t = t_0[1 + \alpha(\frac{x}{c})^{\beta}]$ with $\alpha = 0.15$ and $\beta = 4$.
In Wardrop (1952), two principles on traffic assignment flow distribution, called Wardropian principles, related to the user's perspective and to the system perspective are proposed. Wardropian principles have some underlying assumptions: the drivers have a complete information about the available paths and the network flows are stable over time. When the information is incomplete the resulting traffic assignment is said to be stochastic and when the network flows vary over time the resulting traffic assignment is said to be dynamic. Since the thesis focuses on traffic assignment methods during the rush hour and with a complete information, the literature review goes only through static and deterministic traffic assignment. The first Wardropian principle is based on the assumption that all users minimize their travel times when travelling from an origin to a destination and travel times are equal on all used routes and lower than on any unused route. The resulting assignment is called *user equilibrium*. The second is based on the assumption that traffic spreads on the road network in such a way the total travel time is minimized and the resulting assignment is called *system optimum*. When all drivers individually decide the route they will use in travelling from origin to destination as in the user equilibrium, there are no drivers that can unilaterally choose another route because all used route from an origin to a destination are characterized by the same average travel time, as stated in the Wardropian first principle. This is because each driver decides to use the least duration path and, at the end, all routes used for the same OD pair have the same travelling times since, if not, some users will move to the quickest path until the travel times of that path equates the travel time of the slowest path. In fact, the equilibrium is reached when no users will move to another quicker path. In Beckmann et al. (1956) the mathematical models for the user equilibrium and the system optimum traffic assignments have been developed in form of a convex non-linear optimization model with linear constraints. The arc flows are considered continuous since, when demand values are high enough, relaxing an integer variable into a continuous one does not affect the solution in a substantial way. According to Beckmann et al. (1956) and Sheffi (1985), in a traffic assignment problem formulation the set of constraints ensures that the demand of all the OD pairs has to be routed, the flows have to be non-negative and the arc flows equate the sum of all the path flows traversing that arc. The traversing time of a path depends on the arc flow according to its arc latency function $t_a = t_a(x_a)$, where x_a represents the flow on arc a. The latency function is usually assumed convex and non-decreasing and the path latency is usually defined as the sum over all traversed arcs of the arc latency function. The user equilibrium objective function is formulated in Beckmann et al. (1956) as the sum over all arcs of the integral between 0 and the arc flow of the arc latency function. In Sheffi (1985) a proof of existence and uniqueness of the user equilibrium and a proof of the correspondence between the user equilibrium definition and the proposed model are provided. The system optimum traffic assignment model is also provided in Beckmann et al. (1956) where the objective function is the sum over all arcs of the arc latency function multiplied by the flow on the arc. The system optimum traffic assignment assigns paths to users in order to minimize the total travel time but some users could be routed on

paths that much longer than other paths assigned to users of the same OD pair. Thus, the system optimum is efficient from the point of view of the minimization of the total travel time but no consideration on users' fairness are taken into account. Conversely, the user equilibrium considers users' fairness but it does not guarantee the achievement of the system optimality. Literature on Wardropian traffic assignment methods is wide and references can be found in Potts and Oliver (1972), Newell (1980), Sheffi (1985), Bell and Iida (1997) and Patriksson (2015). In addition, references on dynamic traffic assignment can be found in Merchant and Nemhauser (1978) and in Peeta and Ziliaskopoulos (2001). References on stochastic traffic assignment can be found in Daganzo and Sheffi (1977) and in Sheffi and Powell (1981).
Bounds on the inefficiency of the user equilibrium are discussed in Section 1.2.1 while methods to alleviate the unfairness of the system optimum imposing constraints on users' satisfaction are shown in Section 1.3.

1.2.1 The price of anarchy

In Murchland (1970) a paradox on traffic assignment, called Braess's paradox, has been provided that shows that including a new road in the road network could lead to worsening in terms of individual travel times. Braess's paradox is stated as follows: "For each point of a road network, let there be given the number of cars starting from it, and the destination of the cars. Under these conditions one wishes to estimate the distribution of traffic flow. Whether one street is preferable to another depends not only on the quality of the road, but also on the density of the flow. If every driver takes the path that looks most favourable to him, the resultant running times need not be minimal. Furthermore, it is indicated by an example that an extension of the road network may cause a redistribution of the traffic that results in longer individual running times". Examples can be found in Sheffi (1985). Braess's paradox is a clear example that achieving the user equilibrium does not imply that the total travel time is minimized, showing in particular an unexpected fact that adding a new road segment improves the total travel time in a system optimum model but may result a worsening in individual travel times. The difference in terms of total travel time between the two assignments is a measure of the inefficiency of the user equilibrium and is called price of anarchy. The price of anarchy is defined as the worst-case ratio of the total travel time produced by a user equilibrium assignment over the total travel time under a system-optimum assignment considering a specific latency function. The literature on the price of anarchy is wide and upper bounds have been found for affine and non-negative coefficient polynomial arc latency functions. Considering an instance with a latency function l drawn from a family L of non-decreasing continuous functions, the price of anarchy is bounded from above by $\alpha(L)$, i.e. $\sum_a x_a^{UE} t_a(x^{UE}) \leq \alpha(L) \sum_a x_a^{SO} t_a(x^{SO})$. According to Roughgarden and Tardos (2002), in a single OD pair case and for linear latency functions, the price of anarchy is bounded to $\alpha(L) = \frac{4}{3}$ and bounds for other function families have been derived. The maximum latency price of anarchy is an alternative way to measure the price of selfish routing. In Lin et al. (2011) and in Bayram et al. (2015) the price of selfish routing with respect to the maximum latency experienced by a user, i.e. the user equilibrium total travel time is compared with the total travel time resulting from a min-max path latency model, is studied. The min-max path latency model minimizes the maximum path latency over all experienced paths under the same constraints of the user equilibrium. Conversely to the classical price of anarchy, in Correa et al. (2007) it is proved that, even for linear latency functions, the maximum latency price of anarchy can be unbounded. Another measure of the price of anarchy is related to Braess's paradox and is called Braess's ratio. Let $L_i(G)$ be the averaged path latency of the $i-th$ OD pair on a graph G. Let $H \subseteq Q$ a subgraph obtained removing arcs from G paying attention in having at least a path for each commodity. Braess's ratio is: $\beta(G) = \max\limits_{H \subseteq G} \min\limits_{i=1,\dots,k} \frac{L_i(G)}{L_i(H)}$. In Lin et al. (2011) it is shown that the maximum latency price of anarchy is an upper bound for the Braess's ratio.

1.3 System optimal traffic assignment with users' constraints

In order to limit the unfairness produced minimizing a system-wide objective, the paths set can be bounded by users' constraints that allow only those paths that guarantee a certain level of fairness among users in the same OD pair. In order to generate only those paths that guarantee a certain fairness level, an arc length measure called normal length is introduced. The normal length is an a priori estimate of the path duration that can be defined as the geographical distance, as the free-flow travel time or as the travel time under user equilibrium. The geographical distance and the free-flow travel time are normal length measures that do not depend on the demand volume spread on the network while the travel time under user equilibrium depends on flow accommodated on each arc by a user equilibrium traffic assignment and, hence, also on the global demand that intends to travel on the network. The path set is generated choosing only those paths that are within a certain percentage, called users' constraints percentage, of the shortest path for each OD pair evaluated using the normal length. Since the system-wide minimization is done on a restricted path set, the optimal value of the traffic assignment is a lower bound for the optimal value obtained without users' constraints. In the literature on system optimal traffic assignment with users' constraints, there are some examples of system-oriented objectives as the minimization of the total travel time and the minimization of the weighted geometric path duration and different latency functions have been used as the Davidson's function $t_a(x) = t_a^{FF}(1 + \frac{\alpha x_a}{u'_a - x_a})$, where t_a^{FF} is the arc free-flow travel time and u'_a and α are tuning parameters, and the one provided by the U.S. Bureau of Public Road, $t_a(x) = t_a^{FF}[1 + 0.15(\frac{x_a}{u_a})^4]$, where t_a^{FF} represents the arc free-flow travel time and u_a represents the maximum rate of vehicles that can enter an arc without experiencing substantial delays due to congestion.

The first attempt to model a system optimal traffic assignment with users' constraints was the so-called constrained system optimum proposed in Jahn et al. (2000). Constrained system optimum approach proposed in Jahn et al. (2000) minimizes the total travel time and the user perspective is taken into account by bounding the path set using as normal length the geographical distance. In order to compute the total travel time they proposed as latency function the Davidson's one. The road network is represented by a directed graph where the arc capacity u_a and the geographical length l_a are defined on each arc. The network is considered as capacitated and, hence, it could happen that a certain amount of demand is too high be routed on eligible paths. Since the optimization problem is not linear, they propose as solution method a variant of the Frank-Wolfe algorithm called Partan (see LeBlanc et al. (1985) for details). In the proposed algorithm the search for a descent direction is made using a linearized version of the problem in which the travel time on each arc is considered constant. They do not explicitly enumerate all possible paths from origin to destination and a column generation technique is implemented. The subproblem associated with the linearized version is a constrained shortest path problem, known to be NP-hard. Later, in Jahn et al. (2005), a constrained system optimum formulation is proposed in which the USA Bureau of Public Road latency function is used and the user equilibrium travel time is used as normal length. They compare the performance of the different normal lengths and point out that using the travel time under user equilibrium reflects the impact of traffic volumes on the network and, hence, the path set depends on the amount of traffic that intends to travel through the network. Using the user equilibrium travel time as normal length allows us to derive the following bound on the optimal value of the constrained system optimum: the total travel time obtained using a constrained system optimum with any chosen users' constraints percentage is always lower than the total travel time under user equilibrium. The first Wardropian principle, i.e. the user equilibrium one, states that each used path under user equilibrium has the same travel time. If the user equilibrium travel time is used as normal length and a user equilibrium assignment is performed on that road network, the result is, trivially, that all demand travel on the shortest paths from origin to destination. Each of these used paths is a shortest path and, hence, is feasible also considering a 0% users' constraints percentage. Given this, the user equilibrium solution is always a feasible solution for the constrained system optimum for each users' constraints percentage value. Hence, total travel times under user equilibrium is always higher or equal to the optimal value of the constrained system optimum. On the other hand, using this traffic dependent normal length, comparisons between different levels of traffic on the same road network are not reliable since the path set changes. As methodology they propose the Partan algorithm used in Jahn et al. (2000). They provide a wide computational study in which they test the model on seven real road networks where demands are generated using estimations of real data. Results in terms of experienced unfairness are evaluated by means of the following measures: the loaded unfairness (experienced travel time with respect to the fastest path for the same OD pair), the normal unfairness (normal length of the experienced path with respect to the normal length shortest path), the user equilibrium unfairness (path experienced travel time with respect to the fastest path travel time under user equilibrium) and the free-flow unfairness (path experienced travel time with respect to the fastest path travel time under free-flow conditions). They show that, using a large enough value as users' constraints percentage, the total travel time observed is very close to the one obtained using all possible paths as in a system optimum assignment. Contrary to the model presented in Jahn et al. (2000), the network is considered uncapacitated, i.e. infinite flows can be assigned to each arc. A theoretical framework on the constrained

system optimum and, in particular, on the work shown in Jahn et al. (2005), is provided in Schulz and Stier-Moses (2006) where theoretical considerations on efficiency and fairness have been developed. As in the literature on the system optimum they derive results for the price of anarchy, i.e. show how bad a user equilibrium traffic assignment acts in terms of total travel time with respect to the constrained system optimum one. Given a class of different latency functions Ω, the price of anarchy is formally defined as $\alpha^{\phi}(\Omega) = \sup_{latency \in \Omega} \frac{totalTravelTime_{UE}}{totalTravelTime^{\phi}_{CSO}}$, where ϕ is the users' constraints percentage used in the constrained system optimum. They have proved results either considering as normal length the free-flow travel time and the user equilibrium travel time. Considering the free-flow travel time as normal length, they have proven that if the users' constraints percentage is within 0% and 100% and the considered latency function is affine, then the price of anarchy is $\alpha^{\phi}(\Omega_{affine}) \leq (2-\phi)^{-1}$. If $\phi < \frac{5}{4}$, then $\alpha^{\phi}(\Omega_{affine}) < \frac{4}{3}$. Considering the user equilibrium travel time as normal length, they have proven that price of anarchy generated by the constrained system optimum with users' constraints percentage is the same of the system optimum. They derive results also on fairness among users.
A few other approaches to system-optimality with users' constraints have been proposed in the literature. In Lujak et al. (2015) a model in which the weighted geometric mean of the path duration is used as objective function is presented. The unfairness among users of the same OD pair is bounded using as normal length the free-flow travel time and also unfairness among users of different OD pairs is considered. For each OD pair they define the OD pair average path duration cost as the geometric mean, over all OD pair paths, of the flow on path multiplied by the path travel time. Once the OD pair average path duration cost is defined for each OD pair, they compute the weighted geometric mean of the path duration as the sum over all the OD pairs of the respective normalized mean path durations. The constraints set imposes that the demand is completely routed and that arc capacities are not exceeded. The unfairness among different OD pairs is regulated by a constraints set called envy-free constraints. These constraints guarantee that there is no OD pair that envies any other OD pair for paying less than the cost paid by the other OD pairs raised to a fixed power. Computational results and comparisons with the system optimum and the user equilibrium show that the proposed model produces average travel times that are better than the ones produced by a user equilibrium and near to the system optimum values. In Bayram et al. (2015) a shelter location problem, in which a constrained system optimum traffic assignment is integrated, is shown. The total evacuation time is minimized under optimal location of the shelters but people can be assigned to a certain path only if the path length is within a certain percentage of the shortest path to the shelter. Constraints set guarantees that the demand is completely routed and that at least a fixed number of shelter will be opened. They propose also a version in which the shelters are capacitated since this is more reliable in real world cases. Also in communication networks the constrained system optimum model is used. In Holmberg and Yuan (2003) the main issue is to avoid paths with high dispatching delays. The time delay is calculated by summing up the estimated link delays of each arc that belongs to the considered path. For this reason a limit on the cost per unit of flow on each path is calculated. These limits can also include distortion on the network and link failure. Another little difference is in the forcing constraints, i.e. the ones that say if a path has to be used or not. Since binary variables are used to recognize if a path is used or not, a relaxation is proposed. The problem is solved with a column generation

technique.

2. Proactive route guidance to avoid congestion

Abstract

We propose a proactive route guidance approach that integrates a system perspective: minimizing congestion, and a user perspective: minimizing travel inconvenience. The approach assigns paths to users so as to minimize congestion while not increasing their travel inconvenience too much. A maximum level of travel inconvenience is ensured and a certain level of fairness is maintained by limiting the set of considered paths for each Origin-Destination pair to those whose relative difference with respect to the shortest (least-duration) path, called travel inconvenience, is below a given threshold. The approach hierarchically minimizes the maximum arc utilization and the weighted average travel inconvenience. Minimizing the maximum arc utilization in the network, i.e., the ratio of the number of vehicles entering an arc per time unit and the maximum number of vehicles per time unit at which vehicles can enter the arc and experience no slowdown due to congestion effects, is a system-oriented objective, while minimizing the weighted average travel inconvenience, i.e., the average travel inconvenience over all eligible paths weighted by the number of vehicles per time unit that traverse the path, is a user-oriented objective. By design, to ensure computational efficiency, the approach only solves linear programming models. In a computational study using benchmark instances reflecting a road infrastructure encountered in many cities, we analyze, for different levels of maximum allowed travel inconvenience and, the minimum maximum arc utilization and the weighted average travel inconvenience. We find that accepting relatively small levels of maximum travel inconvenience can result in a significant reduction, or avoiding, of congestion.

Keywords. Congestion; Proactive route guidance; Travel inconvenience; Linear programming

2.1 Introduction

The fraction of the population living in urban areas continues to grow. As a consequence, traffic in urban areas is increasing and the inability to significantly increase road network infrastructure is making the issue of traffic control and coordination more and more relevant and pressing. Congestion is a common phenomenon experienced in cities and towns around the world, and causes delays, stress, and pollution. The negative impact of congestion on the economy, the society, the environment, and on people's health is enormous. Government, industry, and private citizens are all interested in ways to reduce the negative externalities of transportation. Technology has always been an integral part of attempts to alleviate congestion, but recent and anticipated technological and automotive advances offer enormous and exciting new opportunities.

Traditionally, traffic information has been communicated to drivers via radio or by means of Variable Message Signs. The drawback of these systems is that the information being communicated is the same for all drivers and, as such, has only limited value in globally coordinating traffic. The most common modern in-vehicle device aimed at helping drivers guide a vehicle in a road network is a car navigation system based on a digitalized road network map and a global positioning system (GPS) aerial. The GPS aerial allows the vehicle to be localized on the map, and embedded optimization software allows the selection of the best route to the destination. Based on the available information on the status of the road network, the navigation system may provide an optimal route to the destination with respect to the user's preference, which can be the shortest path in terms of distance or travel time, or the least expensive path in terms of fuel consumption or, even, emissions produced. Challenges (and frustrations) occur when the road utilization on the route proposed by the navigation system exceeds its capacity and congestion occurs. In fact, and especially during peak hours, congestion often occurs because the paths of many vehicles traverse the same sections of the road network. Recently, navigation systems have been integrated with real-time traffic data acquisition systems that allow detection of traffic jams and/or road interruptions and offer the potential to reroute the drivers to different paths. Unfortunately, these systems typically do not (yet) consider the system-wide impact of the directions they provide to the drivers. The navigation devices, again, provide drivers with the same information and route guidance, which, in many cases, simply shifts the congestion to other parts of the road network.

The drawbacks of user-optimal paths, which result in a user equilibrium state of the traffic network, compared to a system-optimal set of user paths have long been investigated and understood (see, for example, Mahmassani and Peeta (1993)). It is also well-known that, with a system-optimal set of user paths, some users may end up being assigned to paths that are much longer, in distance or time, than the shortest possible path between their origin and destination. This unfairness, among others, results in users not following route guidance, especially when suggested routes deviate substantially from a user's preference and are (expected to take) much longer.

However, technological and automotive advances may change the situation favorably. The anticipated introduction of autonomous or self-driving vehicles may drastically alter the landscape. Massive adoption of self-driving vehicles will have several benefits in terms of congestion. First and foremost, because the safe separation distance between two self-driving vehicles will be much smaller, their introduction will implicitly alter the capacity of the road network. Secondly, drivers will get used to trusting their vehicle to get them from their origin to their destination and, as a consequence, will be more likely to accept a route that deviates from the preferred (shortest) route. The latter, forms the motivation and underpinning of our research. In particular, we focus on an environment in which a specific origin-destination path can be assigned to each (self-driving) vehicle and in which that vehicle will follow the assigned path. (We will conduct computational experiments to study the sensitivity to the 100% compliance assumption.)

We propose a centralized proactive route guidance approach that integrates a system perspective, focused on reducing, ideally avoiding, congestion, and a user perspective, focused on minimizing the experienced travel inconvenience. The goal is to reduce or avoid congestion without causing an excessive increase in the length (or duration) of the paths traveled by individuals, when compared to the shortest (least-duration) paths between their origin and destination.

Our starting point is a road network and an Origin-Destination (OD) matrix specifying the number of trips that are estimated to take place between each origin and each destination. The problem of forecasting traffic on a road network has been well studied (see, for example, Sheffi (1985), Ben-Akiva and Lerman (1985), Florian and Hearn (1999), and more recently de Dios Ortuzar and Willumsen (2011)) and traditionally has been modeled in four steps: trip generation, trip distribution, mode choice, and traffic assignment. In particular, the zonal interchange analysis of trip distribution provides the so called Origin-Destination (OD) matrix, that is the matrix that provides for each OD pair the number of trips with the same origin and destination. Starting from Merchant and Nemhauser (1978), researchers have also studied a dynamic traffic assignment problem which presents additional challenges (see Papageorgiou (1990), Peeta and Ziliaskopoulos (2001) and more recently Ben-Akiva et al. (2012)).

The time period of interest is the rush hour, which in large cities may last a few hours, and in which, as Sheffi (1985) points out, traffic often exhibits a steady-state behavior. We assume that the arcs of the road network are characterized by a capacity representing the maximum number of vehicles per time unit at which vehicles can enter the arc and experience no slowdown due to congestion effects. The proposed approach will assign paths to drivers so as to minimize congestion while not increasing their experienced travel inconvenience too much. A maximum level of travel inconvenience is ensured and a certain level of fairness is maintained by limiting the set of considered paths for each Origin-Destination pair to those whose relative difference with respect to the shortest (least-duration) path, called travel inconvenience, is below a given threshold.

An important feature of the approach, and a critical design choice, is that only linear programs are solved. To have any potential practical value, a route guidance approach has to be computationally efficient. Computational efficiency has prompted us to restrict ourselves to the use of linear programming models (even if that would mean sacrificing some accuracy in our modeling choices).

The distribution of traffic is evaluated by two measures: the minimum maximum arc utilization in the network (a system perspective) and the weighted average travel inconvenience (a user perspective). Arc utilization, i.e., the ratio of the number of vehicles entering an arc per time unit and its capacity, is used as an arc congestion measure. The weighted average travel inconvenience averages the experienced travel inconvenience over all possible paths weighted by the number of vehicles that enter the path per time unit. The weighted average travel inconvenience is minimized under the constraint that the minimum maximum arc utilization does not exceed a given limit: the minimum maximum arc utilization achievable if it is greater than one, or one if it is smaller than one (by definition no slowdown due to congestion effects occurs when the minimum maximum arc utilization is one).

The linear structure of the models allows us to derive theoretical properties. We will show in particular that for the problem of minimizing the maximum arc utilization, results analogous to those well known for the maximum flow problem, such as the max flow-min cut theorem, hold.

An extensive and comprehensive computational study demonstrates that in many settings relatively small values of the maximum allowed travel inconvenience lead to a minimum maximum arc utilization less than or equal to one, i.e., avoidance of congestion effects. The computational tests are carried out using randomly generated benchmark instances that represent characteristic features of the most common circular-shaped city road networks. As mentioned before, our assumption is that (self-driving) vehicles follow their assigned path, i.e., a 100% compliance rate. Our investigation shows, among others, that compliance is critical; it becomes much more difficult to reduce or avoid congestion when the recommended paths are not followed.

Finding a system-optimal traffic distribution that ensures a certain level of fairness is also considered by Jahn et al. (2005). They also limit the set of paths for an OD pair to limit the travel inconvenience. However, they model the arc travel time as a function of the number of vehicles on that arc (using a widely adopted non-linear increasing function). The model assigns paths to users with the objective of minimizing the total user travel time. The model is non-linear and a column generation solution method is proposed and tested on real road networks. The approach we propose is substantially different as it seeks to avoid congestion by minimizing the maximum arc utilization. We find the minimum congestion level first and then minimize the user travel inconvenience.

A few other related papers have been published. In Sen et al. (2001), a static multi-objective approach seeking to minimize the mean travel time cost and the travel time variance is presented, while in Lujak et al. (2015) a centralized path assignment model is proposed in which the objective is to minimize the geometric path duration mean for all drivers in the system. Centrally controlled traffic systems are often encountered in large warehouses where automated guided vehicles have to be routed (see Bartlett et al. (2014)). In Kaspi and Tanchoco (1990), an integer programming model is proposed which seeks to minimize the weighted path length of a set of automated guided vehicles.

The remainder of the paper is organized as follows. In Section 2.2, the proactive route guidance setting considered is formally introduced, the path-based linear programming models comprising our proactive route guidance approach are given, as well as an algorithm to generate the sets of eligible paths, and supporting theoretical results are presented. In Section 2.3, we discuss the generation of road networks on which the approach is tested, discuss detailed results for a specific instance, and average results for the complete set of instances. Finally, some conclusions are drawn in Section 2.4.

2.2 Proactive route guidance

The basic idea of the system-optimal approach we propose is to assign paths other than the shortest (least-duration) path to vehicles in order to reduce, and possibly avoid, congestion in the road network, but to do so in a way that minimizes the inconvenience experienced by the drivers, in part by only considering alternative paths whose relative increase with respect to the shortest path is within a pre-defined limit. That is, we impose a limit on user travel inconvenience, in terms of the maximum allowed increase relative to the shortest path, which we denote by γ and refer to as the *maximum allowed travel inconvenience.*

More specifically, for each OD pair, we generate a set of eligible paths, which are those Origin-Destination paths with a travel inconvenience that is no more than the specified limit γ. Furthermore, we assume that, for the time period of interest, the demand associated with an OD pair is specified in terms of the number of vehicles entering the network at the origin per time unit. The basic premise of the approach is that the congestion of the road network depends on the utilization of its arcs, where the utilization of an arc is the ratio of the number of vehicles entering an arc per time unit and the maximum number of vehicles per time unit at which vehicles can enter the arc and experience no slowdown due to congestion effects. When the flow per time unit on an arc exceeds its capacity, i.e., the arc utilization is greater than 1, the arc is said to be *congested.* We say that the road network is congested if at least one of its arcs is congested.

Given a maximum allowed travel inconvenience, and, thus, a set of eligible paths for each OD pair, the approach obtains a system-optimal distribution of traffic using a hierarchical approach. First, a linear programming model is used to minimize the maximum arc utilization, i.e., the level of congestion. If the minimum maximum arc utilization exceeds one, then congestion in the road network is unavoidable for the imposed maximum allowed travel inconvenience; to reduce the level of congestion even further, longer, less convenient paths need to be allowed. On the other hand, when the minimum maximum arc utilization is less than or equal to one, congestion can be avoided with the imposed maximum allowed travel inconvenience. In fact, there is no reason to seek a minimum maximum arc utilization below one, because it can only come at the expense of unnecessary user travel inconvenience. Second, for a given minimum maximum arc utilization, another linear programming model is used to minimize the weighted average travel inconvenience subject to the constraint that the maximum arc utilization does not exceed the minimum possible value computed using the previous model (or one if this value is less than one).

Our approach focuses on a system that exhibits a steady-state behavior and seeks to determine the minimum maximum allowed travel inconvenience that allows elimination of congestion in the system by proactive route guidance, if such a maximum allowed travel inconvenience exists. It is possible, of course, that the optimization reveals that such a maximum allowed travel inconvenience does not exist, due to a combination of network infrastructure characteristics, e.g., a small number of bridges and/or tunnels connecting different parts of the city, and demand characteristics, e.g., extremely high demand. In such situations, more elaborate linear programming models are needed. In Section 2.4, we briefly discuss this possible extension.

Next, we present the two path-based linear programming models used in the approach, followed by the algorithm for generating eligible paths and supporting theoretical results.

2.2.1 Optimization models

We consider a directed network $G = (V, A)$, where V represents the set of vertices and $A \subseteq V \times V$ represents the set of arcs. Each arc $(i, j) \in A$ represents a road segment on which vehicles can travel. The length of arc $(i, j) \in A$, depending on the user' preferences, may represent the traveling time or the length in space, and it will be denoted by l_{ij}. The capacity of each arc $(i, j) \in A$ is denoted by u_{ij} and represents the maximum rate (number of vehicles per time unit) that can enter an arc (i, j) at the averagespeed. We consider a set C of OD pairs. An OD pair $c \in C$ is described by its origin $O_c \in V$, its destination $D_c \in V$ and a flow rate (number of vehicles per time unit) d_c, representing the rate of vehicles traveling from O_c to D_c. We let $D = \sum_{c\in C} d_c$ be the the total flow rate in the network.

Let γ be the maximum allowed travel inconvenience. A set of paths K_c^γ is associated with each OD pair c and contains all paths that are not longer than $(1 + \gamma)\, SP_c$, where SP_c is the shortest path from the origin O_c to the destination D_c. The travel inconvenience γ_c^k associated with path $k \in K_c^\gamma$, $c \in C$, is defined as the relative increase of path k with respect to the shortest (least-duration) path SP_c. Denoting by l_c^k the length of path k for OD pair c, we have $\gamma_c^k = \frac{l_c^k - SP_c}{SP_c}$.

The optimization problem formulations use an indicator a_{ij}^{kc}, which takes value 1 if path $k \in K_c^\gamma$ contains arc $(i, j) \in A$, and 0 otherwise. The critical parameter ρ_γ^* represents the minimum possible value of the maximum arc utilization. This value is computed by a linear programming model, referred to as the *congestion model* and defined below. The decision variables y_c^k represent the flow rate (number of vehicles per time unit) of OD pair $c \in C$ routed on path $k \in K_c^\gamma$. The auxiliary variables x_{ij} and ρ represent the total flow rate (number of vehicles per time unit) traveling on arc $(i, j) \in A$ and an upper bound on the maximum arc utilization, respectively. The *inconvenience model* is a linear programming model seeking to minimize the weighted average travel inconvenience over all OD pairs, defined as $\frac{1}{D} \sum_{c\in C} \sum_{k\in K_c^\gamma} \gamma_c^k y_c^k$. A summary of the sets, parameters and variables is given in Table 2.1.

The inconvenience model

$$
\begin{aligned}
\min \quad & \frac{1}{D}\sum_{c\in C}\sum_{k\in K_c^\gamma}\gamma_c^k y_c^k && \\
& \rho \le \max(1,\rho_\gamma^*) && && (2.1)\\
& \frac{x_{ij}}{u_{ij}} \le \rho && \forall (i,j)\in A && (2.2)\\
& x_{ij} = \sum_{c\in C}\sum_{k\in K_c^\gamma} a_{ij}^{kc} y_c^k && \forall (i,j)\in A && (2.3)\\
& d_c = \sum_{k\in K_c^\gamma} y_c^k && \forall c\in C && (2.4)\\
& \rho \ge 0 && && (2.5)\\
& x_{ij} \ge 0 && \forall (i,j)\in A && (2.6)\\
& y_c^k \ge 0 && \forall c\in C \quad \forall k\in K_c^\gamma. && (2.7)
\end{aligned}
$$

Constraint (2.1) guarantees that the maximum arc utilization is minimized in case congestion is unavoidable ($\rho_\gamma^* > 1$), and ensures that the flow rate on any arc is less than or equal to the arc's capacity, otherwise ($\rho_\gamma^* \le 1$). Constraints (2.2) bound the maximum arc utilization, i.e., $\max_{(i,j)\in A}\{x_{ij}/u_{ij}\} \le \rho$ and constraints (2.3) set the flow rate on arc (i,j). Constraints (2.4) ensure that the required flow rate d_c of an OD pair $c \in C$ is routed on (a subset of) its eligible paths. Finally, constraints (2.5) - (2.7) define the domains of the decision variables. Although the natural domain of variables y_c^k and x_{ij} is the set of non-negative integers, it is reasonable to relax it to the set of real numbers as long as the values d_c are large. Note that y_c^k/d_c can be interpreted as the fraction of the flow rate d_c that is routed on path $k \in K_c^\gamma$.

The minimum maximum arc utilization ρ_γ^* required in constraint (2.1) is computed with the following linear programming model.

The congestion model

$$
\begin{aligned}
\rho_\gamma^* \equiv min \quad & \rho \\
\text{s.t.} \quad & (2.2) - (2.7)
\end{aligned}
$$

The congestion model focuses purely on congestion, i.e., on minimizing the maximum arc utilization, without taking into account the impact on the drivers. On the other hand, the inconvenience model without constraints (2.1) - (2.2) focuses purely on user travel inconvenience without taking into account the impact on congestion. For this reason, a hierarchical approached is used. We first focus on congestion and then focus on user travel inconvenience.

The following remarks provide basic relations linking the maximum allowed travel inconvenience γ, the path set K_c^γ, and the minimum value of the maximum arc utilization ρ^*_γ.

Remark 1. *Let $K_c^{\gamma_1}$, $K_c^{\gamma_2}$ be the sets of eligible paths for OD pair $c \in C$ for γ_1 and γ_2, respectively. If $\gamma_1 \leq \gamma_2$, then $K_c^{\gamma_1} \subseteq K_c^{\gamma_2}$.*

Remark 2. *Let $K_c^{\gamma_1}$, $K_c^{\gamma_2}$ be the sets of eligible paths for OD pair $c \in C$ for γ_1 and γ_2, respectively. If $K_c^{\gamma_1} = K_c^{\gamma_2}$ $\forall c \in C$, then the minimum maximum arc utilization is the same for the two settings.*

Remark 3. *Let $\rho^*_{\gamma_1}$, $\rho^*_{\gamma_2}$ be the optimum values of the congestion model when the maximum allowed travel inconvenience is γ_1 and γ_2, respectively. If $\gamma_1 \leq \gamma_2$ then $\rho^*_{\gamma_1} \geq \rho^*_{\gamma_2}$.*

The models can be modified to take into account that not all vehicles may follow the recommended path. We define the *compliance rate* to be the fraction α of vehicles following the recommended path and assume that $1-\alpha$ vehicles, instead, choose the shortest (least-duration) path. For sake of simplicity, we assume that the compliance rate is identical for all OD pairs. The compliance rate is taken into account by adding to each model the constraints

$$y_c^{SP} \geq (1-\alpha)d_c \quad \forall c \in C. \tag{2.8}$$

These constraints impose that at least a fraction $1-\alpha$ of the demand is routed along its shortest path. This fraction represents those drivers acting "selfishly".

2.2.2 Generation of the eligible paths

To the best of our knowledge, whereas the problem of finding the first K shortest paths is well known (see Yen (1971) and Eppstein (1994)), no algorithm has been published for the computation of all shortest paths between two vertices whose length does not increase the shortest path length by a given percentage. Thus, in order to compute the set K_c^γ, an ad hoc algorithm has been developed. The algorithm relies on depth-first search and consists of a main body *ConstructingEligiblePaths* (sketched in Algorithm 1) and a recursive routine *ScanVertex* (sketched in Algorithm 2).

Inputs for algorithm *ConstructingEligiblePaths* are the network (V, A), the set of OD pairs C, and the maximum allowed travel inconvenience γ. The algorithm first computes the

Problem 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 allowed travel inconvenience γ

Parameters

u_{ij}	capacity 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
SP_c	length of the shortest path for OD pair $c \in C$
l_c^k	length of path $k \in K_c^\gamma$
γ_c^k	relative length increment of path $k \in K_c^\gamma$, with respect to SP_c: $\gamma_c^k = \frac{l_c^k - SP_c}{SP_c}$
d_c	flow rate for OD pair $c \in C$
D	total flow rate over all OD pairs: $D = \sum\limits_{c \in C} d_c$
α	compliance rate

Decision variables

y_c^k	flow rate of OD pair $c \in C$ routed on path $k \in K_c^\gamma$

Auxiliary variables

x_{ij}	total flow rate entering arc $(i,j) \in A$: $x_{ij} = \sum\limits_{c \in C} \sum\limits_{k \in K_c^\gamma} a_{ij}^{kc} y_c^k$
ρ	upper bound on arc utilization : $\rho \geq \frac{x_{ij}}{u_{ij}} \quad \forall (i,j) \in A$
f_{ij}^c	flow rate of the OD pair c traversing arc $(i,j) \in A$

Table 2.1. Proactive route guidance approach notation

Algorithm 1: ConstructingEligiblePaths

```
input  : $V$, $A$, $C$, $\gamma$
$s \leftarrow ComputeShortestPathMatrix(V, A)$
for $i \in V$ do
    $status(i) \leftarrow INACTIVE$
for $c \in C$ do
    $Path \leftarrow \emptyset$
    $cPath \leftarrow 0$
    $lPath \leftarrow 0$
    $ScanVertex(O_c)$
```

Algorithm 2: ScanVertex

```
input  : $i$
global variables: $s, \gamma, status, Path, cPath, lPath.$
$status(i) \leftarrow ACTIVE$
$Path(cPath) \leftarrow i$
if $i = D_c$ then
    $Record(Path)$
else
    for $j \in \delta^+(i)$ do
        // j is the endpoint of an arc exiting from i
        if $status(j) = INACTIVE$ and $lPath + l_{ij} + s(j, D_c) \leqslant (1 + \gamma)\, SP_c$ then
            $cPath \leftarrow cPath + 1$
            $lPath \leftarrow lPath + l_{ij}$
            $ScanVertex(j)$
            $lPath \leftarrow lPath - l_{ij}$
            $cPath \leftarrow cPath - 1$
$status(i) \leftarrow INACTIVE$
```

shortest path matrix (the shortest path for each pair of vertices) and labels all vertices as inactive. Labels will change from inactive to active and back during computation. Next, for each OD pair $c \in C$, initializations are made and a depth-first search to find the set K_c^γ is performed by means of the recursive routine *ScanVertex*.

Routine *ScanVertex* operates as follows. The input vertex i is labeled as active (already visited) and inserted in the growing path at the current position. If the input vertex is the destination vertex the procedure records the path, labels the vertex as inactive and exits. Otherwise, the routine is recursively called on each vertex reached by any outgoing arc from the input vertex provided that the new vertex is not labeled as active (which would imply a cycle) and the length of the growing path is within the fixed threshold. The vertex count and the total length are increased and decreased before and after recursion.

The algorithm has time complexity $\mathcal{O}\left(q^{|V|+1}\right)$, where q is an upper bound on the number of outgoing arcs from any vertex. The exponential complexity is due to the problem nature as explained in the following example. Consider the network, consisting of $n = 3h+1$ vertices and with $q = 3$, shown in Figure 2.1. Consider vertex 1 as O_c and vertex $h+1$ as D_c and let $\epsilon \leq \gamma$. The h arcs connecting the $h+1$ central vertices with length 1 define the shortest path from O_c to D_c, and, thus, $SP_c = h$. The remaining arcs have length $\frac{1}{2} + \frac{\epsilon}{2h}$. All possible paths in this network from O_c to D_c belong to the set K_c^γ. The number of the possible paths is $3^h = 3^{\frac{n-1}{3}}$.

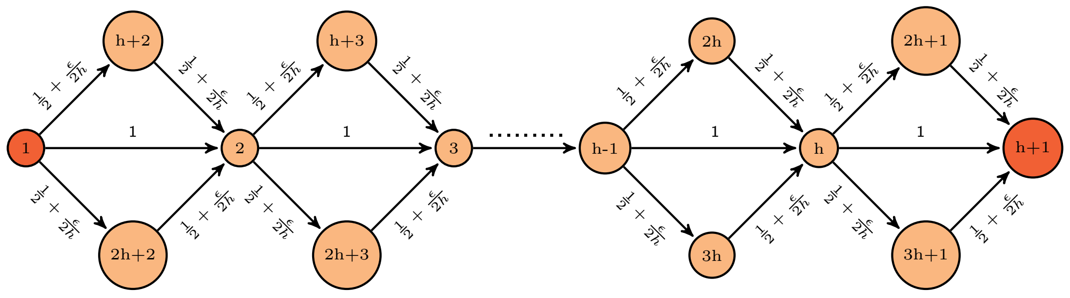


Figure 2.1. An instance demonstrating the exponential complexity of the algorithm for generating the sets of eligible paths for the OD pairs

2.2.3 Lower bounds on the minimum maximum arc utilization

In this section we propose an arc-based formulation for the congestion model where, for each OD pair c, all possible paths from O_c to D_c are implicitly considered. The model, which we refer to as the *unconstrained congestion model*, is equivalent to the congestion model in which the maximum allowed travel inconvenience is set to $+\infty$. Allowing all paths for all the OD pairs means that the result of the unconstrained congestion model is a lower bound for the congestion model for any value of γ.

The unconstrained congestion model

$$\rho^*_\infty \equiv min \quad \rho$$

$$\sum_{j\in V} f^c_{ij} - \sum_{j\in V} f^c_{ji} = 0 \quad \forall c \in C \quad \forall i \in V, i \neq O_c, i \neq D_c \tag{2.9}$$

$$\sum_{j\in V} f^c_{ij} - \sum_{j\in V} f^c_{ji} = d_c \quad \forall c \in C \quad i = O_c \tag{2.10}$$

$$\sum_{j\in V} f^c_{ij} - \sum_{j\in V} f^c_{ji} = -d_c \quad \forall c \in C \quad i = D_c \tag{2.11}$$

$$\rho \geq \frac{\sum_{c\in C} f^c_{ij}}{u_{ij}} \quad \forall c \in C \quad \forall (i,j) \in A \tag{2.12}$$

$$f^c_{ij} \geq 0 \quad \forall c \in C \quad \forall (i,j) \in A \tag{2.13}$$

$$\rho \geq 0. \tag{2.14}$$

Constraints (2.9) - (2.11) guarantee the flow rate conservation. The objective function together with Constraints (2.12) set the minimum maximum arc utilization to the minimum of the maximum value, over all arcs, of the ratio between the total flow rate and the arc capacity. Finally, Constraints (2.13) and (2.14) define the domains of the decision variables.

Remark 4. *The value ρ^*_∞ of the unconstrained congestion model is a lower bound for ρ^*_γ, for any value of the maximum allowed travel inconvenience γ, i.e.,*

$$\rho^*_\gamma \geq \rho^*_\infty \quad \forall \gamma.$$

Thus, the value ρ^*_∞ is the minimum level of congestion that can be achieved when vehicles can be sent along any possible path (regardless of the experienced travel inconvenience). This value may be greater than one due to the structure of the network and high levels of the traffic. In the following, we derive lower bounds on ρ^*_∞ that can be computed from the network parameters. The results we derive recall results that are well known for the maximum flow problem (see Ford and Fulkerson (1956)).

Definition 1. *A cut-set for a set of OD pairs $\hat{C} \subseteq C$ is a minimal arc set $A_{\hat{C}}$ such that in the graph $G' = (V, A \setminus A_{\hat{C}})$ no path connecting O_c to D_c exists for any OD pair in $\hat{C}$. The capacity of a cut-set $A_{\hat{C}}$ is $u_{\hat{C}} = \sum_{(ij)\in A_{\hat{C}}} u_{ij}$.*

Definition 2. *A minimum capacity cut-set for $\hat{C}$ is a cut-set $A^*_{\hat{C}}$ with minimum capacity. We denote the capacity of $A^*_{\hat{C}}$ by $u^*_{\hat{C}}$.*

Theorem 1. *Let $A^*_{\hat{C}}$ be a minimum cut-set for any set of OD pairs $\hat{C} \subseteq C$, and let $u^*_{\hat{C}}$ be its capacity. Then,*

$$\rho^*_\infty \geq \max_{\hat{C} \subseteq C} \frac{\sum_{c \in \hat{C}} d_c}{u^*_{\hat{C}}}.$$

Proof. Let $\bar{G}$ be a network in which the capacities $\hat{u}_{ij}$ are the capacities u_{ij} of the original network G multiplied by the minimum maximum arc utilization ρ^*_∞, i.e. $\hat{u}_{ij} = \rho^*_\infty u_{ij}$. Let $\hat{C} \subseteq C$ be a set of OD pairs and $A^*_{\hat{C}}$ a minimum cut-set for $\hat{C}$. In $\bar{G}$ it is possible to route the demand of the OD pairs in $\hat{C}$ without violating the capacities $\hat{u}_{ij}$. Hence, $\sum_{c \in \hat{C}} d_c \leq \sum_{(ij) \in A^*_{\hat{C}}} \hat{u}_{ij} = \sum_{(ij) \in A^*_{\hat{C}}} \rho^*_\infty u_{ij} = \rho^*_\infty \sum_{(ij) \in A^*_{\hat{C}}} u_{ij} = \rho^*_\infty u^*_{\hat{C}}$. Hence, $\frac{\sum_{c \in \hat{C}} d_c}{u^*_{\hat{C}}} \leq \rho^*_\infty$. As this inequality holds for any subset $\hat{C} \in C$, the claim follows.

□

In the following, we show that in the case of a single OD pair equality holds, but that examples can be found with strict inequality for the case of multiple OD pairs.

Theorem 2. *Let $|C| = 1$ and let us denote by c the OD pair. Let A^*_c be a minimum cut-set for c with capacity u^*_c. Then, $\rho^*_\infty = \frac{d_c}{u^*_c}$.*

Proof. See Appendix .1. □

Remark 5. *Equality $\rho^*_\infty = \max_{\hat{C} \subseteq C} \frac{\sum_{c \in \hat{C}} d_c}{u^*_{\hat{C}}}$ does not hold when there are two or more OD pairs in the network, that is when $|C| \geq 2$.*

This example shows that the equality does not hold in the case of two OD pairs. In Figure 2.2 all arcs have capacity u equal to 2 and each OD pair has demand equal to 2. Deriving a lower bound on the minimum maximum arc utilization means exploring all the possible subsets of OD pairs in C. The number of possible subsets in C is given by the cardinality of the power set of C, $P(C)$. The cardinality of the power set depends exponentially on the number of OD pairs contained in C, i.e. $|P(C)| = 2^n$, where n is the number of OD pairs in C.

In Figure 2.3 the optimal flows for each OD pair are shown. It is easy to see that the demand is completely routed. The minimum maximum arc utilization value is equal to $\rho^*_\infty = \frac{3}{2}$. Considering each subset $\hat{C} \subseteq C$ and calculating $\rho^{\hat{C}}_\infty = \frac{\sum_{c \in \hat{C}} d_c}{u^*_{\hat{C}}}$ for each subset, we obtain that the maximum of all these values is $\max_{\hat{C} \subseteq C} \rho^{\hat{C}}_\infty = \max_{\hat{C} \subseteq C} \frac{\sum_{c \in \hat{C}} d_c}{u^*_{\hat{C}}} = 1 < \rho^*_\infty$.

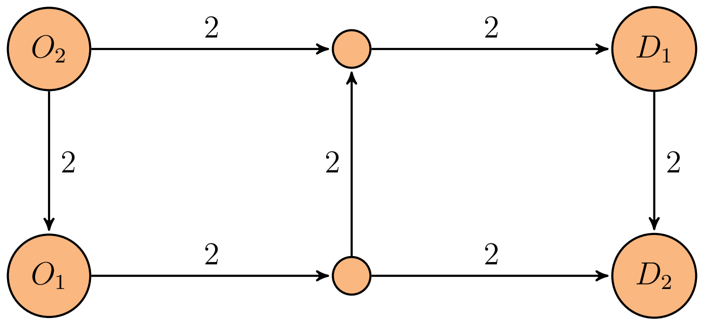


Figure 2.2. Proactive route guidance: example for two OD pairs

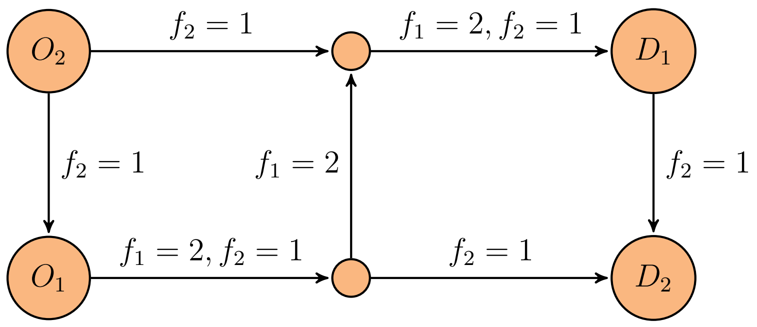


Figure 2.3. Proactive route guidance: optimal flows on counterexample graph

2.3 Computational results

The instance generator was implemented in Java, and the optimization models were solved by CPLEX 12.6.0. The generation of the instances and the experiments were run on a Windows 64-bit computer with Intel Xeon processor E5-1650, 3.50 GHz, and 16 GB Ram. Instance generation depends on a few controls, which are described in more detail in Chapter 6. Considering all possible combinations of these controls, the number of different types of networks is 16. Instance generation also depends on some random factors which are not included into the control set. For example, the coordinates of the nodes in a network is randomly perturbed from their original position. For each set of values for the controls, we generate 5 random instances. The total number of generated instances is therefore 80. The instances are available on `http://or-brescia.unibs.it/instances`. The statistics collected for each instance are described in Section 2.3.1. Detailed results and insights for a single, specific instance are presented and discussed in Section 2.3.2. Section 2.3.3 is devoted to summary results for all instances.

Network congestion	
ρ^*_γ	The minimum maximum arc utilization ρ^*_γ obtained from the congestion model.
x_{ij}/u_{ij} classes	The 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 ($1.5 < x_{ij}/u_{ij}$).
No-congestion γ	The no-congestion travel inconvenience, i.e. the minimum value of maximum allowed travel inconvenience needed to avoid congestion.
User experience	
Experienced inconvenience	The weighted average travel inconvenience, i.e.the optimal value of the inconvenience model.
Inconvenience classes	The percentage of the demand experiencing different levels of travel inconvenience: • level A: from 0% to 5% • level B: from 5% to 10% • level C: from 10% to 15% • level D: from 15% to 20% • level E: from 20% to 25% • level F: from 25% to 30% • level G: from 30% to 35%.
OD paths	
Generated paths	Average number of generated paths per OD pair.
Selected paths	Selected paths in the optimal solution of the inconvenience model: • average number of selected paths per OD pair • maximum number of selected paths per OD pair.

Table 2.2. Proactive route guidance approach statistics

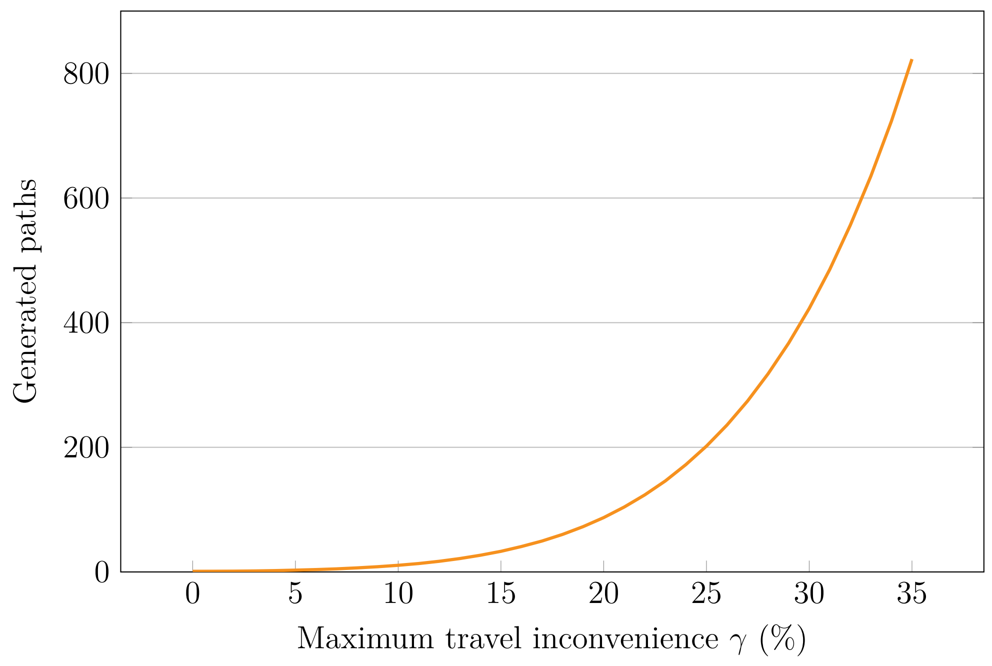


Figure 2.4. Proactive route guidance: average number of generated paths

2.3.1 Statistics

In our experiments, we vary the value of the maximum allowed travel inconvenience from $\gamma = 0\%$ (all vehicles follow the shortest origin-destination path) to $\gamma = 35\%$ (and the values of the compliance rate from 100% to 60%). We collect and compute the statistics shown in Table 2.2. To present large amounts of information in an easy-to-interpret and concise form, we mostly rely on graphs.

2.3.2 Detailed results and insights for a specific instance

In this section, we analyze the traffic patterns produced by the proactive route guidance approach for a single instance. We chose an instance with a `large`, `circular`, and `oligo-centric` city network with `high` in-city traffic and `in-peak` traffic density. In order to provide a detailed understanding of the potential benefits of adopting proactive route guidance, we solved the model with γ ranging from 0% to 35% in increments of 1%.

In Figure 2.4, we show the average number of generated paths per OD pair. As expected, the average number of generated paths grows exponentially. Clearly, at higher values of the maximum allowed travel inconvenience, there are more paths to choose from and it should be possible to reduce congestion more.

In Figure 2.5, we show the minimum maximum arc utilization ρ^*_γ for the different values

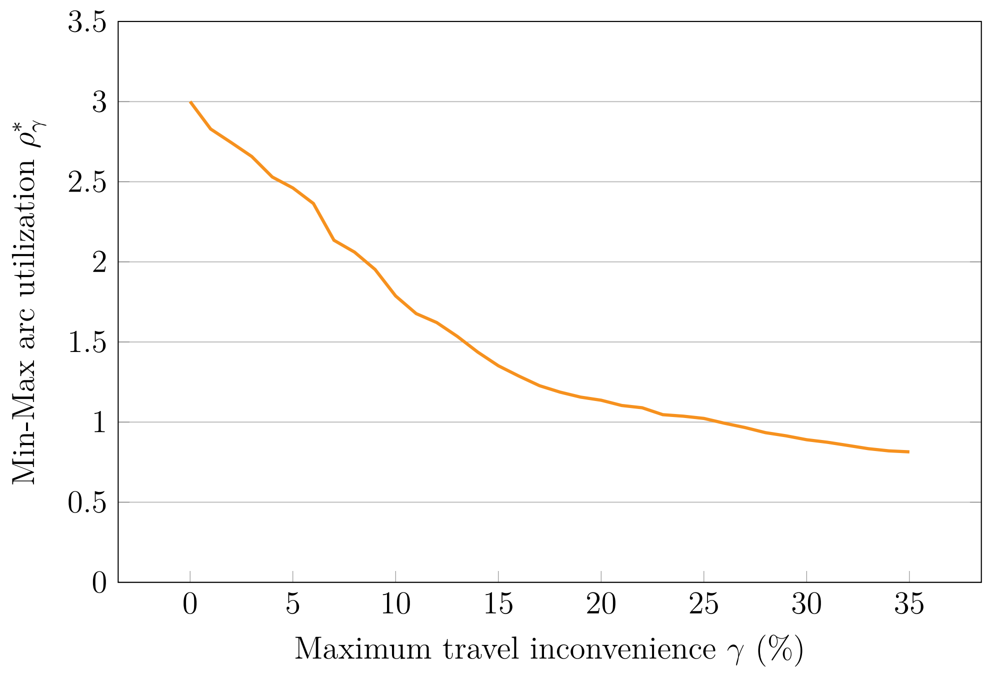


Figure 2.5. Proactive route guidance: minimum maximum (Min-Max) arc utilization

of the maximum allowed travel inconvenience γ. As expected, ρ^*_γ is a non-increasing function of γ. We note that for $\gamma = 26\%$ the value of ρ^*_γ drops below 1, which implies that there is no congestion in the system. Consequently, for values of the maximum allowed travel inconvenience greater than or equal to 26%, the proactive route guidance approach minimizes the weighted average travel inconvenience while ensuring that the minimum maximum arc utilization does not exceed one. (We note that the experienced travel inconvenience value is realistic only when a path has no arcs with an arc utilization that exceeds one. Otherwise, the experienced travel inconvenience value represents an underestimate, as congestion will be encountered along the path.)

In Figure 2.6, we show the arc utilization distribution for the different values of the maximum allowed travel inconvenience. We see that the fraction of unused arcs in the road network remains almost steady. When γ increases from 0% to 13%, the fraction of heavily congested arcs decreases, but, at the same time, the fraction of lightly congested arcs increases and the fraction of the non-congested arcs slightly decreases. We have no heavily congested arc for $\gamma \geq 14\%$, and, as previously observed, the road network becomes free of congestion when $\gamma \geq 26\%$. The two seemingly abrupt transitions from $\gamma = 13\%$ to $\gamma = 14\%$ when the road network looses heavily congested arcs, and from $\gamma = 25\%$ to $\gamma = 26\%$ when the road network looses all congested arcs, are smoother than they may appear. Indeed, for $\gamma = 13\%$ we have $\rho^*_\gamma = 1.54$ and all the 8.3% heavily congested arcs are very close to be classified as lightly congested. For $\gamma = 25\%$ we have $\rho^*_\gamma = 1.02$ and all the congested arcs are very close to be classified as non-congested.

In Figure 2.7, we show the weighted average travel inconvenience for maximum allowed travel inconvenience values greater than or equal to 26%, i.e., values for which congestion

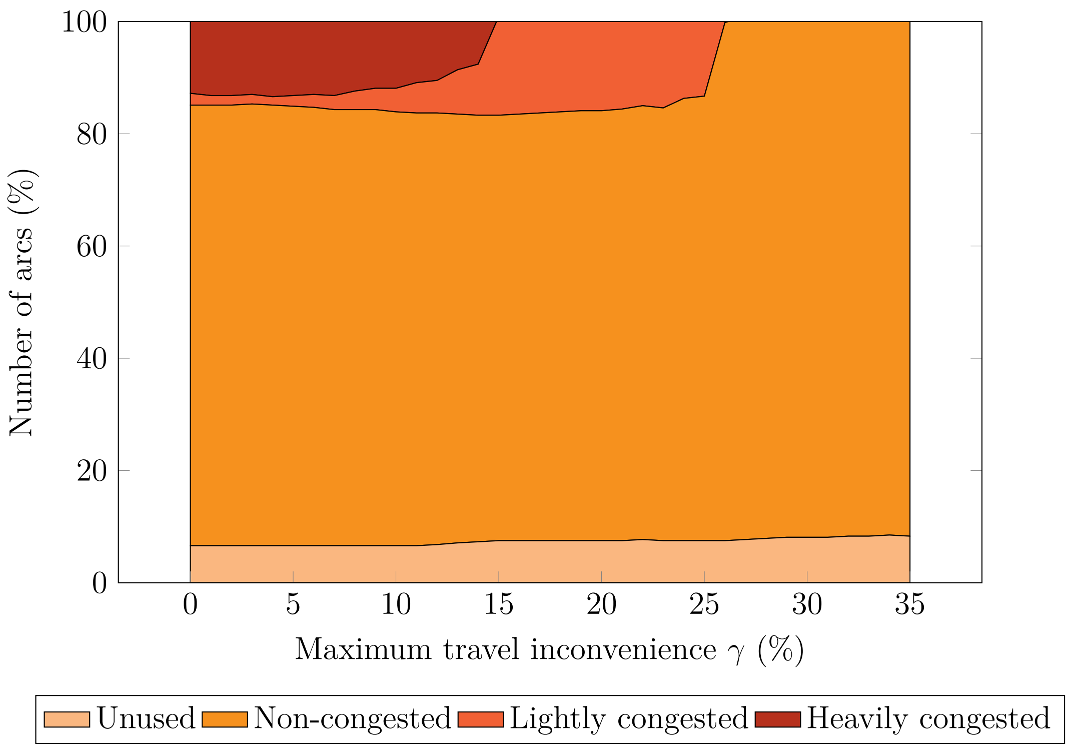


Figure 2.6. Proactive route guidance: arc utilization distribution

can be eliminated. As expected, we see that the weighted average travel inconvenience decreases, because additional paths are exploited to reduce the weighted average travel inconvenience rather than to reduce the maximum arc utilization.

In Figure 2.8, we show, for $\gamma = 26\%$, the number of OD pairs experiencing a certain level of travel inconvenience. Importantly, we see that for almost half of the OD pairs travel still occurs along the shortest path and that only for a relatively small number of OD pairs travel inconvenience exceeds 15%.

The pattern shown in Figure 2.8 is also seen in Figure 2.9, where we show the fraction of OD pairs that is experiencing a certain travel inconvenience level (A through G) for values of $\gamma \geqslant 26\%$. We see that most vehicles are sent along paths with little or no travel inconvenience (level A) and that only a small number of vehicles are sent along paths with a high travel inconvenience (levels E through G).

In Figure 2.10, we show the average and the maximum number of selected paths for an OD pair. We observe that the maximum number of paths selected for an OD pair is surprisingly small. For all but one of the maximum allowed travel inconvenience values, the number of selected paths is less than or equal to three. This implies that vehicles with the same origin and destination are assigned to only a small number of different paths. (This will facilitate developing schemes that assign paths in a fair way over time, e.g., a round-robin assignment scheme.) We also observe that for all maximum allowed travel inconvenience values, for most of the OD pairs, the vehicles are routed along a single path, as the average number of paths selected for an OD pair is very close to 1.

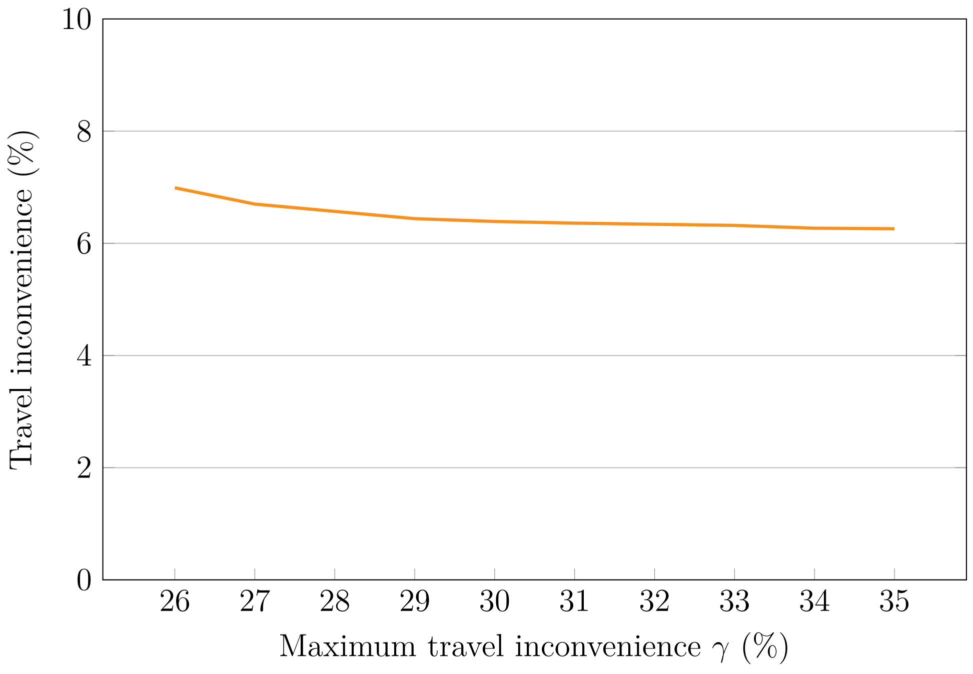


Figure 2.7. Proactive route guidance: weighted average experienced travel inconvenience

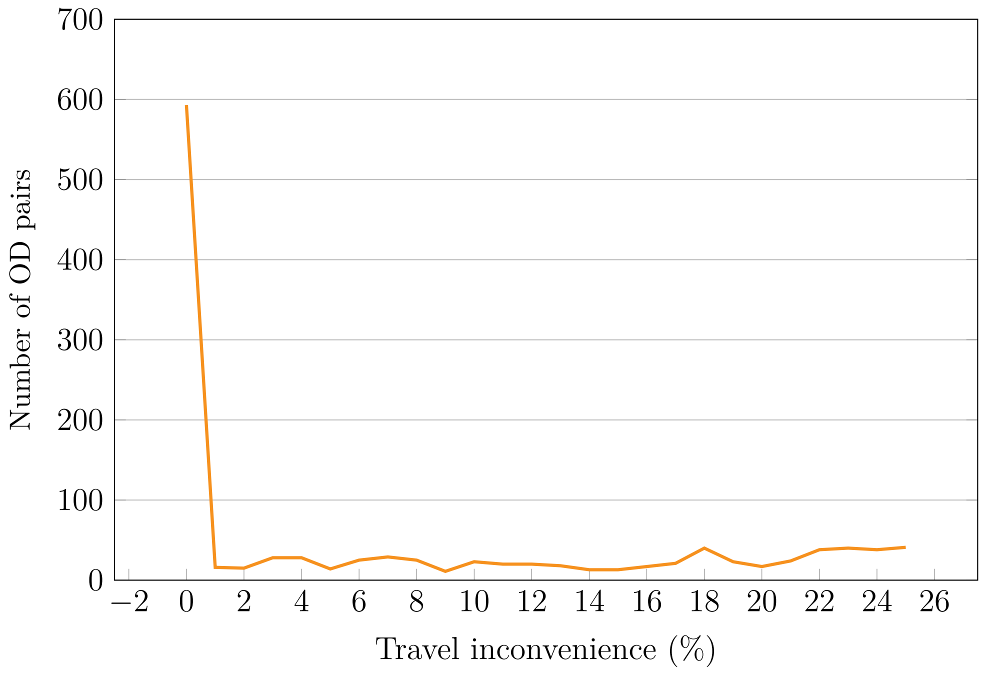


Figure 2.8. Proactive route guidance: experienced travel inconvenience at $\gamma = 26\%$

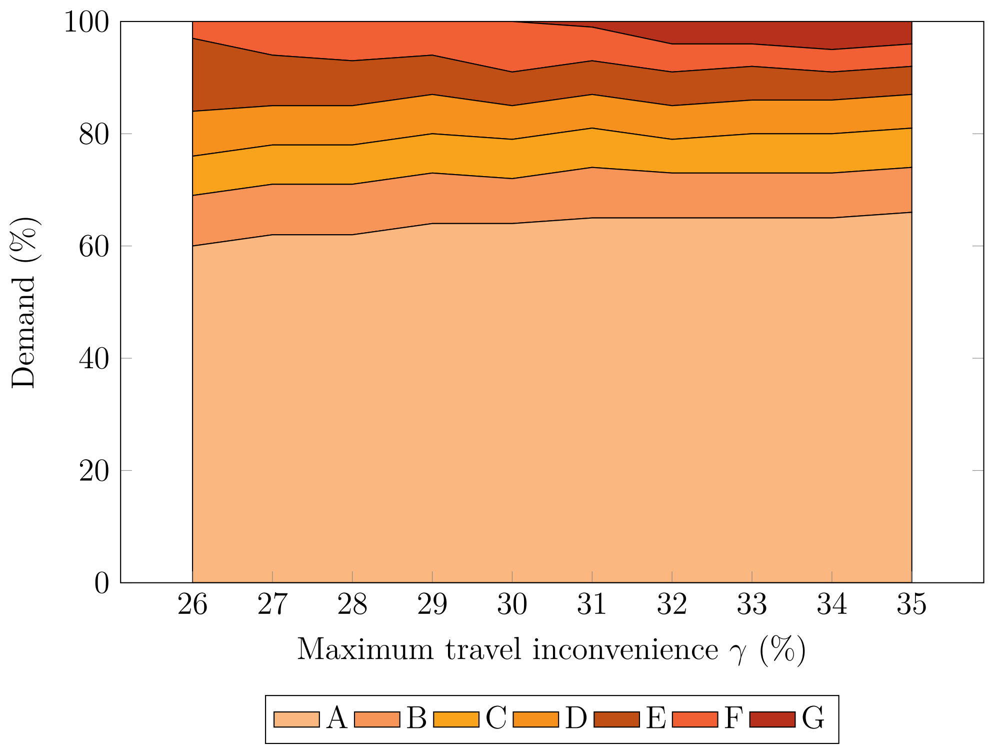


Figure 2.9. Proactive route guidance: experienced travel inconvenience distribution for various levels of maximum travel inconvenience

However, we note that when a single path is selected for an OD pair, this path does not necessarily have to be the shortest path and that, in fact, in many cases it is not.

Even though we are focusing on an environment in which a specific origin-destination path can be assigned to each vehicle and in which that vehicle will follow the assigned path, it is informative to examine what happens when the 100% compliance assumption does not hold. Figure 2.11 shows the minimum maximum arc utilization ρ^*_γ that can be achieved as a function of maximum allowed travel inconvenience for different compliance rates. We see that for $\alpha = 90\%$ congestion can still be avoided as the minimum maximum arc utilization falls below 1 at $\gamma = 30\%$, but that for lower levels of compliance, this is no longer possible.

2.3.3 Summary results for all instances

In this section, we present summary statistics for groups of instances. In Figures 2.12 -2.14, we group instances that require the same maximum allowed travel inconvenience to be able to reach the no-congestion state. In the final two charts, i.e., Figures 2.15-2.16, we look at all instances, but for specific maximum allowed travel inconvenience values.

In Figure 2.12, we show for each value of the maximum allowed travel inconvenience the

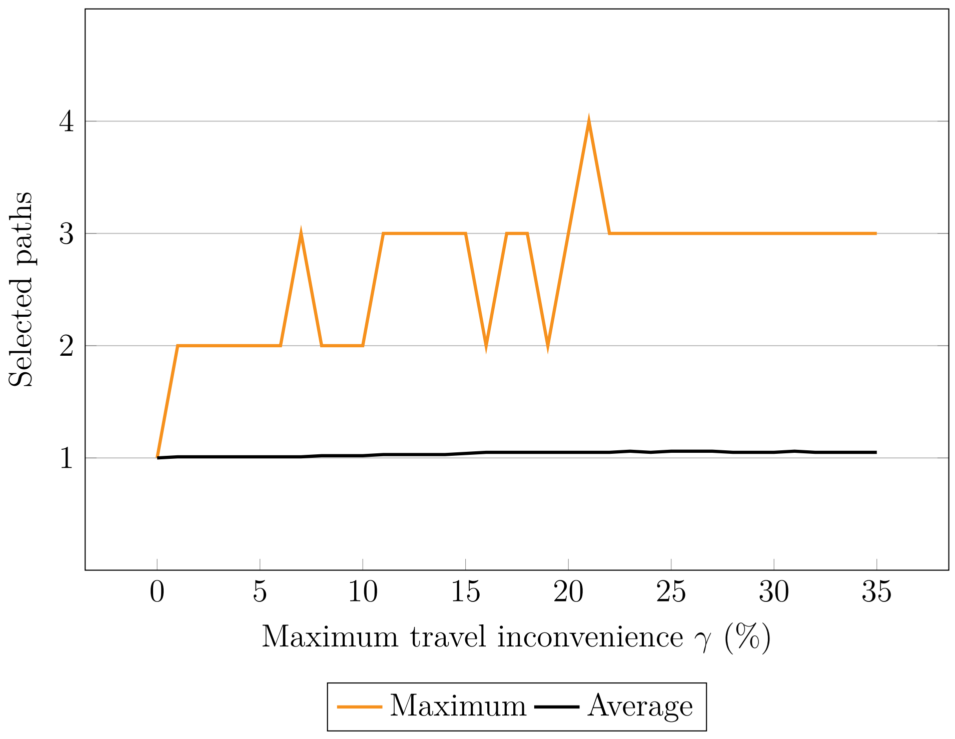


Figure 2.10. Proactive route guidance: selected paths

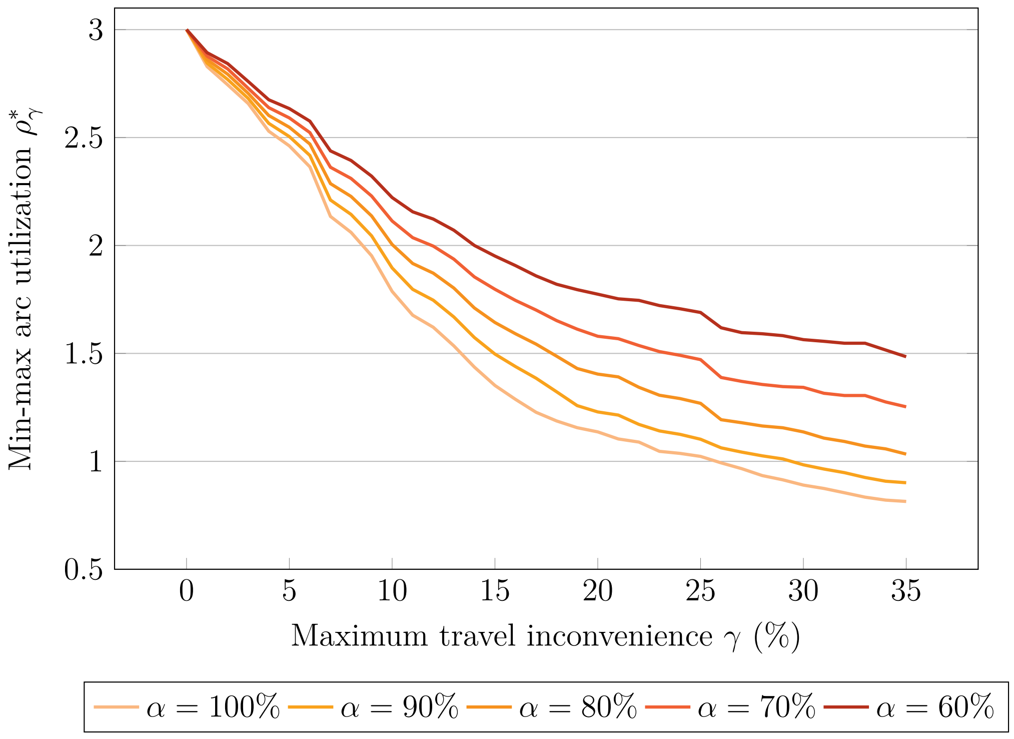


Figure 2.11. Proactive route guidance: impact of the compliance rate on the minimum maximum (min-max) arc utilization

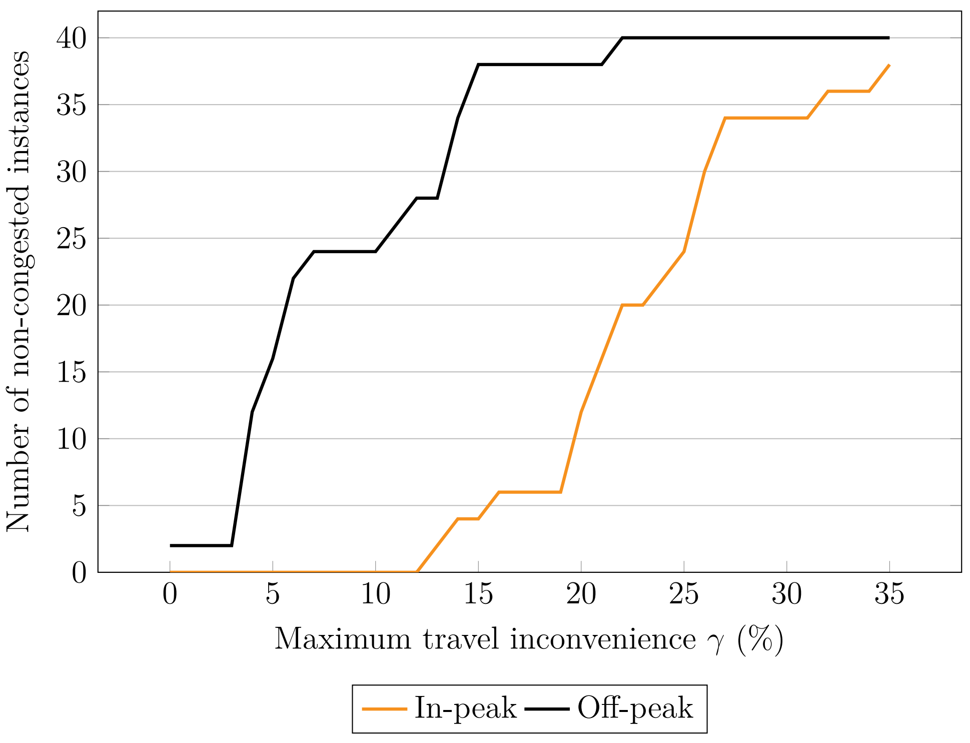


Figure 2.12. Proactive route guidance:
number of instances in a no-congestion state for different values of γ (%)

number of instances for which it is possible to reach a no-congestion state (distinguishing in-peak and off-peak traffic densities). We observe that with off-peak traffic density at $\gamma = 22\%$ it is possible to reach a no-congestion state for all 40 instances, but that with in-peak traffic density even at $\gamma = 35\%$ it is not possible to reach the no-congestion state for two instances. However, the value of minimum maximum arc utilization for these two instance is very close to one, and a no-congestion state is attained at $\gamma = 36\%$.

In Figures 2.13 and 2.14, we consider particular statistics and average this statistic over all instances that reach the no-congestion state at a particular maximum travel inconvenience value. In Figure 2.13, we show the average and the maximum number of selected paths. We observe that the results suggest that when a higher maximum travel inconvenience is required to reach the no-congestion state, more OD pairs will be assigned more than one path as both the average of the maximum number of selected paths and the average of the average number of selected paths are (slightly) higher. In Figure 2.14, we show the weighted average travel inconvenience. We observe that, as expected, the weighted average travel inconvenience is higher when it is more difficult to reach a no-congestion state, but also that the weighted average travel inconvenience remains relatively small even for instances where it is challenging to reach a no-congestion state.

In Figure 2.15, we show the average minimum maximum arc utilization over all instances with in-peak and off-peak traffic densities. As expected, we see that the minimum maximum arc utilization is (significantly) smaller when the traffic density is smaller, as fewer vehicles have to be accommodated in the same road network. We see that with in-peak

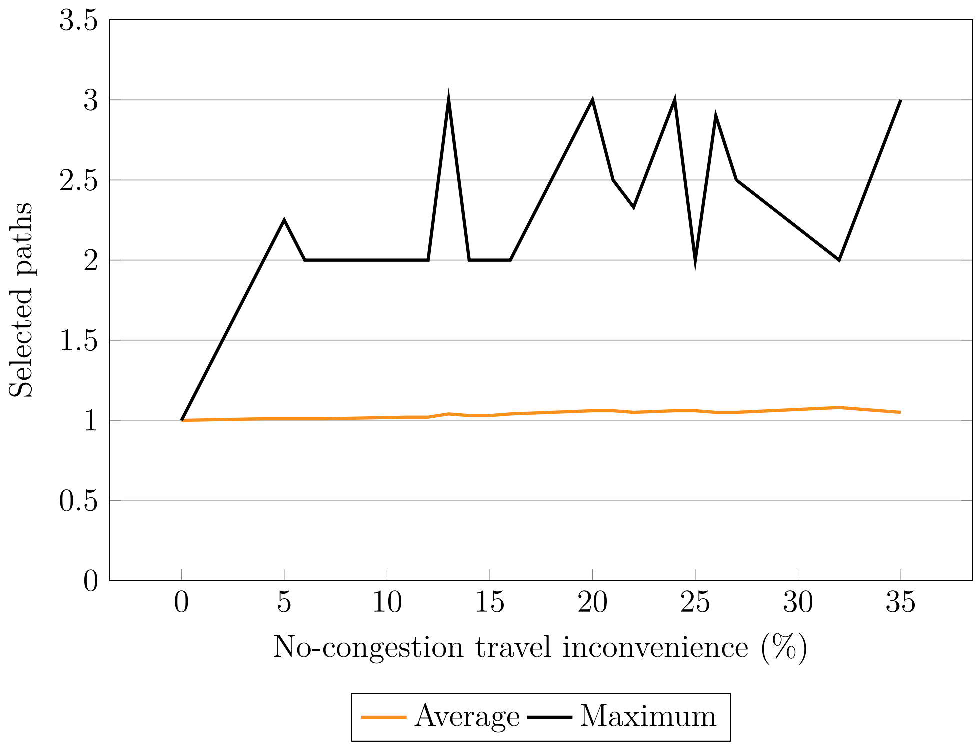


Figure 2.13. Proactive route guidance: number of paths chosen at the no-congestion travel inconvenience γ (%)

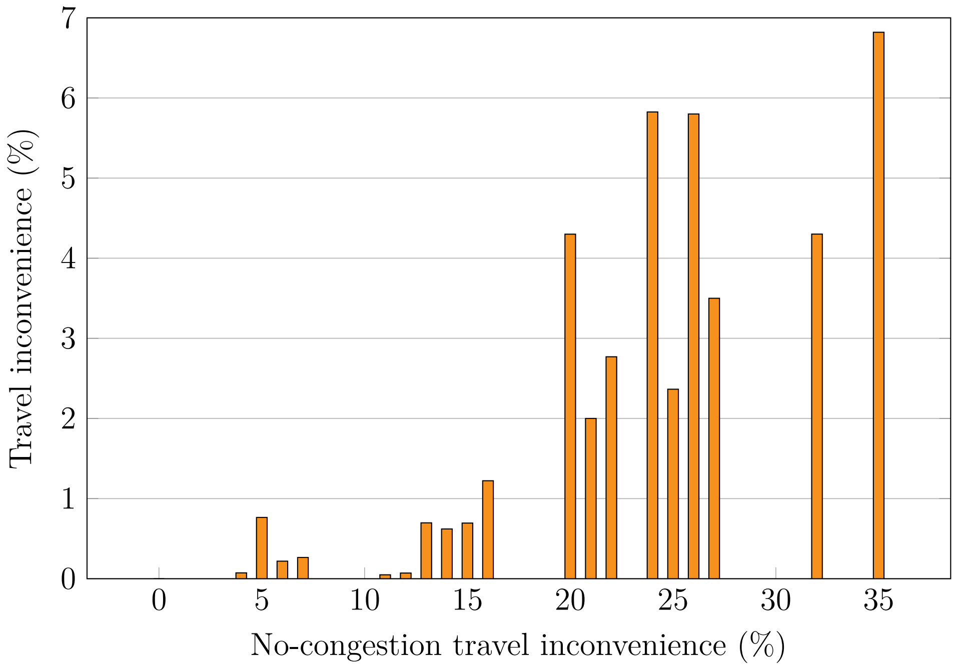


Figure 2.14. Proactive route guidance: weighted average experienced travel inconvenience at the no-congestion travel inconvenience value (%)

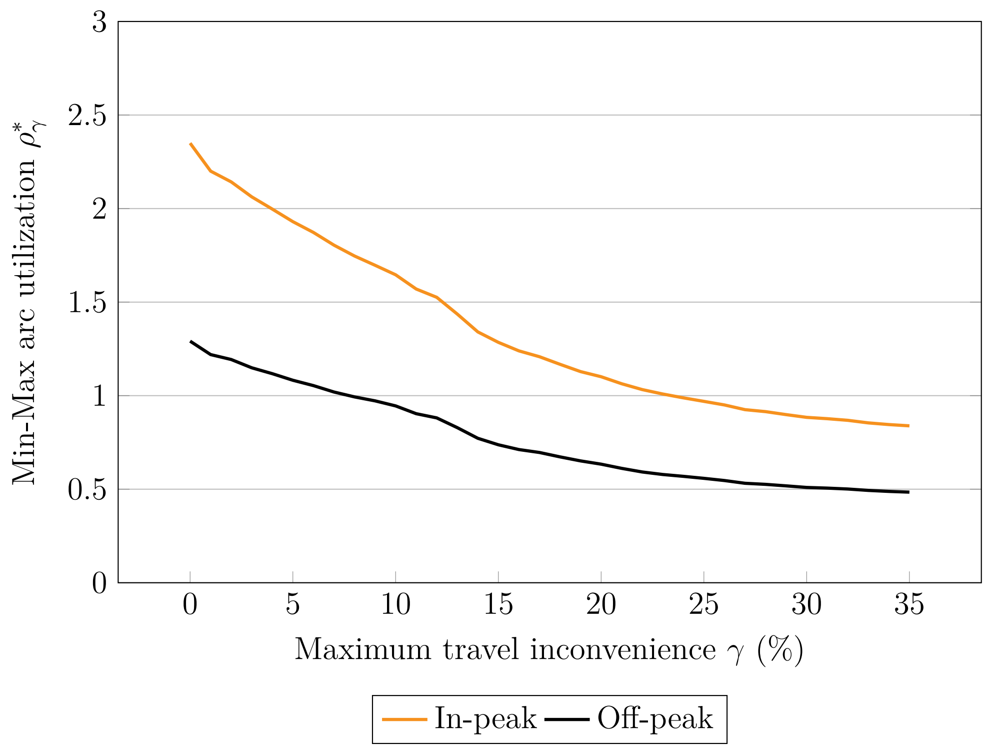


Figure 2.15. Proactive route guidance: minimum maximum (Min-Max) arc utilization

traffic density it is possible (on average) to reach a no-congestion state at $\gamma = 23\%$. On the other hand, with off-peak traffic density, a no-congestion state is reached (on average) at $\gamma = 8\%$.

In Figure 2.16, we show the average arc utilization distribution. The average number of unused arcs remains almost the same while the average number of non-congested arcs steadily increases with increasing values of maximum allowed travel inconvenience. At $\gamma = 35\%$, the proactive route guidance approach is able to eliminate heavily congested arcs in all instances and reduce the average number of lightly congested arcs to less than 0.5%. This congestion “residue” is due to two instances that remain lightly congested at $\gamma = 35\%$.

2.4 Final remarks and future research

In this paper, we have conducted a computational study of the potential of proactive route guidance to reduce or avoid congestion in an environment in which a specific origin-destination path can be assigned to each vehicle in the system and in which that vehicle will follow the assigned path. We believe that the advent of autonomous (or self-driving) vehicles makes this a (much more) realistic assumption.

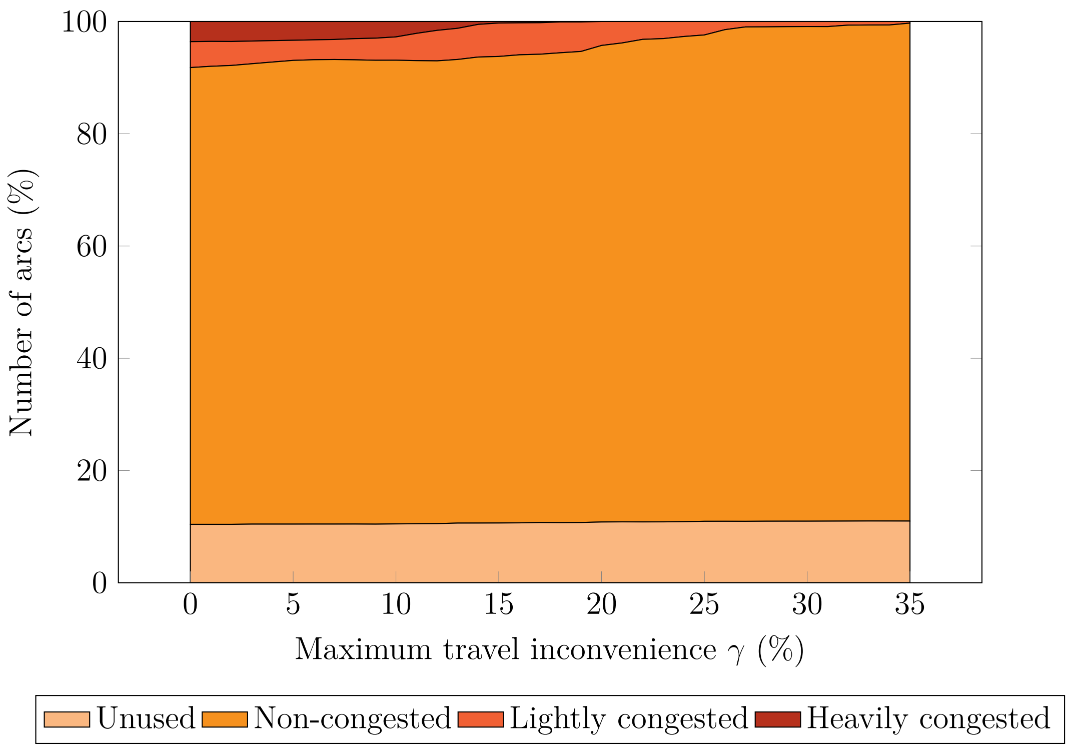


Figure 2.16. Proactive route guidance: arc utilization distribution

Our proposed approach has an important (computational) advantage over previously proposed approaches: it relies only on linear programming. As such, it is more likely to scale and be usable in real-world settings. However, much needs to happen to make that a reality. One important step in that direction, and one that we are currently pursuing, is the dynamic generation of paths (as opposed to generating the set of eligible paths upfront). Given the exponential growth of the sets of eligible paths when the maximum allowed travel inconvenience increases, this will be critical.

Another critically important extension is incorporating traffic-dependent arc travel times, which can be done efficiently by employing piecewise linear approximations of these functions. This will allow more accurate modeling of travel times along paths that contain one or more congested arcs, i.e., arcs with a utilization greater than one, which is especially important in situations where it is not possible to reach a no-congestion state.

A final extension, but one that is more involved, considers varying demand rates over time (instead of assuming steady-state behavior), which will allow more accurate modeling of the traffic dynamics.

3. Heuristic path generation for the proactive route guidance problem

Abstract

The benefits in reducing traffic congestion of system optimum with respect to user equilibrium traffic assignments are well known. Recently a linear programming based approach was proposed that aims at achieving a compromise between the system perspective, namely eliminating congestion, and the user perspective, that is minimizing individual travel times. The approach, called proactive route guidance, assigns to users paths that increase the travel times by at most a given percentage, called maximum allowed travel inconvenience. The approach requires the enumeration of all feasible paths that may be memory and time consuming, especially when large networks and/or high values of the maximum allowed travel inconvenience are considered. In this paper a heuristic is presented to generate a subset of all feasible paths that is based on the iterative search of improving paths. Computational experiments show that the number of paths generated by the heuristic is smaller with respect to the complete set by one or two orders of magnitude on small instances and by higher orders of magnitude when the size of the instances increases. On instances with 150 nodes, where the complete enumeration takes an acceptable computational time, the results show that the quality of the heuristic solutions is very close to that of the optimal ones.

Keywords. Traffic congestion; Proactive route guidance; Heuristic path generation.

3.1 Introduction

Traffic is one of the most pressing problems in modern cities. The benefits in reducing traffic congestion of system optimum with respect to user equilibrium traffic assignments are well known (see, for example, Mahmassani and Peeta (1993) and Roughgarden and

Tardos (2002)). While system optimum approaches are used to coordinate traffic and possibly eliminate congestion when automated guided vehicles are involved (see Kaspi and Tanchoco (1990) and Bartlett et al. (2014)), the application of a system optimum approach to road networks has been considered to be not realistic due to fact that users take their own decisions. Nowadays connected vehicles and technological advances, including driverless vehicles, make the possibility of coordinating traffic more realistic. However, a still relevant drawback of a system optimum approach is that it does not consider fairness towards users.

The idea of integrating a system optimum approach with user fairness considerations has been studied in different papers. User fairness was first considered in Jahn et al. (2005), where the arc travel times were fixed to the travel times experienced under user equilibrium assignment and only the paths with duration smaller than the least duration obtained under user equilibrium were considered as feasible paths. A non linear optimization model was presented, a column generation solution method proposed and computational results were carried out on real road networks. User fairness was also considered in Lujak et al. (2015), where a mathematical programming model was proposed based on Nash welfare optimization. A linear programming based approach to control the trade-off between system and user perspectives, called proactive route guidance, was recently proposed in Angelelli et al. (2016a). The approach assigns paths to users with the goal of minimizing congestion while guaranteeing a maximum level of inconvenience to users. This is achieved by limiting the set of paths considered for an origin-destination pair to those that have a relative difference in length with respect to the shortest path below a given threshold. The approach hierarchically minimizes the maximum arc utilization, which measures the congestion and captures the system perspective, and the average travel inconvenience, that accounts for the user perspective. The fact that the approach is based on linear programming models makes it computationally valuable. However, the enumeration of all feasible paths is requested for every origin-destination pair and this step may be memory and time consuming, especially when big networks and/or high values of maximum allowed travel inconvenience are considered.

In this paper a heuristic is presented to generate the feasible paths for the proactive route guidance approach. Methods for the generation of paths in a road network have been widely studied in literature. According to Prato (2009) most of the deterministic path generation methods are based on repeated shortest path searches over the network. These methods are popular because of the efficiency of shortest path algorithms. The well known k-shortest path algorithm proposed by Yen (1971) is an example of deterministic generation methods based on the shortest path calculation. Several variants of the k-shortest path algorithm have been developed. An interesting variant involves using more than one objective function (see Ramming (2001) and Van der Zijpp and Catalano (2005)). In Ramming (2001) a literature review on the main path generation methods is provided. Deterministic path generation is carried out also using heuristics as the link penalty and the link elimination methods. The idea underlying link penalty methods is to iteratively penalize the links belonging to the shortest paths as proposed in de la Barra et al. (1993). In link penalty methods the weight of the links belonging to the shortest path is increased by a percentage and then a new shortest path search on the modified network is performed. Other variants are presented in Park and Rilett (1997), Bekhor and Prato (2006) and Bekhor et al. (2006). The link elimination method consists in iteratively removing one or more links from the network and in performing a new shortest path search on the modified network. This latter method was first presented in Azevedo et al. (1993) and variants can be found in Prato and Bekhor (2006), Prato and Bekhor (2007) and Frejinger and Bierlaire (2007).

The heuristic presented in this paper can be seen as a link elimination method applied to the congested arcs of the network. The heuristic set of paths is generated through an iterative process. The algorithm starts with a set of paths that contains the shortest path for each origin-destination (OD) pair only. At each iteration, new paths that improve the current solution are identified and added to the restricted set until a stop condition is satisfied. Computational experiments show that the number of paths generated by the heuristic is smaller with respect to the complete set by one or two orders of magnitude on small instances and by higher orders of magnitude when the size of the instances increases. On instances with 150 nodes the complete enumeration takes an acceptable computational time and the solution obtained with the heuristic set of paths can be compared with the optimum obtained with the complete set. The results show that the quality of the heuristic solutions is very close to that of the optimal ones. In particular, if the maximum allowed travel inconvenience is 5%, the congestion (the maximum arc utilization) obtained with the heuristic set of paths coincides with the minimum congestion obtained with the complete enumeration. If the maximum allowed travel inconvenience is 10%, the heuristic congestion coincides with the optimum on 38 over 40 instances. For higher values of the maximum allowed travel inconvenience the heuristic congestion increases on average by at most 0.79% (when maximum allowed travel inconvenience is 30%). The average travel inconvenience with the heuristic set of paths increases on average at most by 0.50 when maximum allowed travel inconvenience is 20%.

The paper is organized as follows. In Section 3.2 the proactive route guidance approach is recalled. In Section 3.3 the heuristic for the path generation is described. In Section 3.4 the results of a thorough computational analysis are presented and the benefits of the heuristic versus the complete enumeration are shown. Finally, some conclusions are drawn in Section 3.5.

3.2 The proactive route guidance approach

In order to make the paper self-contained, we recall in this section the proactive route guidance approach of Angelelli et al. (2016a). The problem is defined on a graph $G = (V, A)$ representing a road network, where vertices V represent road intersections and arcs A represent allowed directed links between intersections. For each arc $(i, j) \in A$, the practical capacity u_{ij} and its length l_{ij} are provided. The former represents the maximum rate of vehicles that can enter and traverse the arc in free-flow conditions, whereas the latter is proportional to the arc traversal time in free-flow conditions. A set C of origin-destination (OD) pairs is also given where each OD pair $c \in C$ is defined by three parameters: $O_c \in V$, $D_c \in V$ and $d_c > 0$, the origin, the destination and the rate of vehicles entering the network in O_c with destination D_c, respectively. The demand d_c of any OD pair $c \in C$ has to be routed through one or more paths in G from origin O_c to destination D_c. A path is measured on the basis of the so called *path travel inconvenience*, computed as the relative increment $\gamma_{c,p}$ of the length of a path p with respect to the shortest path from O_c to D_c. A parameter γ, called maximum allowed travel inconvenience, defines the set of feasible paths P_c for each OD pair $c \in C$ as the set of paths p such that $\gamma_{c,p} \leq \gamma$. Traffic assignment is described by variables $y_{c,p}$, $c \in C$, $p \in P_c$, where variable $y_{c,p}$ represents the amount of demand routed on path $p \in P_c$. Parameters $a^{ij}_{c,p}$ connect feasible paths to arcs. More precisely, $a^{ij}_{c,p} = 1$ if arc (i, j) is traversed by path $p \in P_c$, 0 otherwise. The total rate of vehicles traversing an arc (i, j) is, thus, $x_{ij} = \sum\limits_{c \in C} \sum\limits_{p \in P_c} a^{ij}_{c,p} y_{c,p}$, and the *arc congestion level* is defined as the ratio x_{ij}/u_{ij}. If the arc congestion level exceeds 1, then the arc is said to be *congested*, while it is said *uncongested* otherwise. Analogously, we define the *path congestion level* as the maximum congestion level of its arcs, and the *OD pair congestion level* as the maximum congestion level of its feasible paths. Finally, the *network congestion level* is the maximum arc congestion level (equivalently, the maximum path or OD pair congestion level). The network is said to be *congested* if its congestion level exceeds 1, and *uncongested* otherwise.

The proactive route guidance approach assigns the demand d_c of each OD pair $c \in C$ to a set of paths in P_c with the objective to minimize the so called weighted average travel inconvenience, that is the sum of the weighted travel inconvenience over all paths where the weight of a path is the ratio between the demand assigned to the path and the total demand. Assignment of demand to paths has to leave the network uncongested, if possible, or at the minimum congestion level otherwise.

The proactive route guidance approach consists in two hierarchical linear programming models: the *congestion model* and the *inconvenience model.* The congestion model finds the minimum network congestion level ρ^*. If $\rho^* \leq 1$, the inconvenience model is solved to minimize the weighted average travel inconvenience while keeping the network uncongested. Otherwise, the inconvenience model is solved imposing ρ^* as an upper bound on the network congestion level. In Table 3.1 we summarize the used notation.

The inconvenience model

$$
\begin{aligned}
\bar{\gamma}^* \equiv \min \quad & \frac{1}{\sum\limits_{c \in C} d_c} \sum_{c \in C} \sum_{p \in P_c} \gamma_{c,p} y_{c,p} && && \\
& \rho \leq \max(1, \rho^*) && && (3.1) \\
& \frac{x_{ij}}{u_{ij}} \leq \rho && \forall (i,j) \in A && (3.2) \\
& x_{ij} = \sum_{c \in C} \sum_{p \in P_c} a^{ij}_{c,p} y_{c,p} && \forall (i,j) \in A && (3.3) \\
& d_c = \sum_{p \in P_c} y_{c,p} && \forall c \in C && (3.4) \\
& y_{c,p} \geq 0 && \forall c \in C \quad \forall p \in P_c. && (3.5)
\end{aligned}
$$

Constraints (3.1) and (3.2) ensure that the network remains uncongested if $\rho^* \leq 1$, or that its congestion level is minimized when $\rho^* > 1$. Constraints (3.3) set the flow rate on arc (i, j) equal to sum of flows on paths containing the arc (i, j). Constraints (3.4) ensure that the demand of an OD pair $c \in C$ is completely routed on its feasible paths. Constraints (3.5) bound variables $y_{c,p}$ to be non-negative.

The minimum network congestion level ρ^* used in constraint (3.1) is the optimal value of the following congestion model.

The congestion model

$$
\begin{aligned}
\rho^* \equiv min \quad & \rho \\
& \text{s.t.} \quad (3.2) - (3.5)
\end{aligned}
$$

The congestion model shares with the inconvenience models the same decision variables and operative constraints. However, the congestion model focuses on the network congestion level only, while the inconvenience model without constraints (3.1) - (3.2) would take into account only user travel inconvenience.

In Angelelli et al. (2016a) all the feasible paths from origin to destination for each OD pair are generated. The set of feasible paths obviously depends on the value of the maximum allowed travel inconvenience γ and the number of paths increases when γ increases. In the rest of the paper, we refer to this method as the *complete enumeration* (CE). It has been shown that, with the CE, not only the number of generated paths grows with γ, but that the total number of feasible paths, given γ, is, in the worst case, exponential in the instance size. Since each path is associated with a variable in the models, the number of variables in the models grows, in the worst case, exponentially. For example, for a benchmark instance with 150 vertices and $\gamma = 15\%$, 70125 paths had to be generated. For an instance with 300 vertices, the number of paths was 9649118. In Angelelli et al. (2016a) it was observed that in the optimal solution of the proactive route guidance

Proactive route guidance approach notation	
G	road network
V	set of vertices
A	set of arcs
u_{ij}	practical capacity of arc $(i,j) \in A$
l_{ij}	length of arc $(i,j) \in A$
C	set of OD pairs
d_c	flow rate for OD pair $c \in C$
O_c	origin vertex of OD pair $c \in C$
D_c	destination vertex of OD pair $c \in C$
SP_c	length of the shortest path for OD pair $c \in C$
$l_{c,p}$	length of path p from O_c to D_c
$\gamma_{c,p}$	inconvenience of path p from O_c to D_c : $\gamma_{c,p} = \frac{l_{c,p} - SP_c}{SP_c}$
γ	maximum allowed travel inconvenience
P_c	set of feasible paths for OD pair $c \in C$: $P_c = \{$ path p from O_c to $D_c \mid \gamma_{c,p} \leq \gamma\}$
$a^{ij}_{c,p}$	is 1 if path $p \in P_c$ contains arc (i,j), 0 otherwise
$y_{c,p}$	flow rate of OD pair $c \in C$ routed on path $p \in P_c$
x_{ij}	total flow rate entering arc $(i,j) \in A$: $x_{ij} = \sum\limits_{c \in C} \sum\limits_{p \in P_c} a^{ij}_{c,p} y_{c,p}$

Table 3.1. Proactive route guidance approach notation

approach only few paths were used for each OD pair. This a characteristic of the solution that allows us to hope that a well designed heuristic can achieve high quality solutions.

3.3 The heuristic path generation algorithm

The *Heuristic Path Generation* (HPG) algorithm aims at generating, with respect to the CE, a substantially smaller set of paths that contains most of the paths used in the optimal solution of the inconvenience model and, thus, such that the solution of the inconvenience model, computed using the heuristic set of paths, is close to the optimum.

The heuristic set of paths is generated through an iterative process. The algorithm starts with a set of paths, that we call restricted set, that contains the shortest path for each OD pair only. At each iteration, new paths are identified and added to the restricted set until a stop condition is satisfied. More precisely, at each iteration an estimate of ρ^* is computed by solving the congestion model using the paths currently available in the restricted set. Then, an improving set of paths, that can guarantee a reduction of the current congestion level of the network, is searched for. These paths are selected, for each OD pair, among the shortest ones that comply with the maximum allowed travel inconvenience γ. If an improving set of paths is found, then it is added to the restricted set, and the process continues. Otherwise, the algorithm stops and the current restricted set is the final heuristic set of paths.

The heuristic assignment of demand to the paths is found by solving the inconvenience model using the final restricted set of paths generated by the HPG algorithm and using as congestion value the final estimate of ρ^*.

The HPG algorithm is sketched in Algorithm 3. Variable P^{Res} represents the restricted set of paths and variable P^{Imp} represents the improving set. They are initialized with the empty set and the set of shortest paths for each OD pair, respectively. At each iteration the improving set is added to the restricted set, and the congestion model is solved on the new restricted set. The resulting optimal value ρ' and optimal solution y' are used in the next step to search for a new set of improving paths by means of the *SearchForImprovingPaths* routine that will be described in detail in Section 3.3.1. If the search is successful ($P^{Imp} \neq \emptyset$), the restricted set is updated and the congestion model is solved again. When routine *SearchForImprovingPaths* fails ($P^{Imp} = \emptyset$), the path generation ends. Then, the inconvenience model is solved using the restricted set of paths P^{Res} and as ρ^* the current estimate ρ'. The heuristic value of the objective function of the inconvenience model is denoted by $\bar{\gamma}$.

Algorithm 3: Heuristic Path Generation (HPG) algorithm

```
input  : Network $G$ and set of OD pairs $C$
output: Approximate value of the minimum weighted average travel inconvenience $\bar{\gamma}$
global : $G, C$
– $P^{Res} := \emptyset$;
– $P^{Imp} :=$ set of shortest paths from origin $O_c$ to destination $D_c$ for each $c \in C$;
while $P^{Imp} \neq \emptyset$ do
    – $P^{Res} := P^{Res} \bigcup P^{Imp}$;
    – solve congestion model on the path set $P^{Res}$:
        - $\rho' :=$ optimal value;
        - $y' :=$ optimal solution;
    /* searches for a set of paths able to improve the solution */
    – $P^{Imp} :=$ SearchForImprovingPaths($P^{Res}, y', \rho'$);
– solve the inconvenience model on the path set $P^{Res}$ and $\rho^* = \rho'$:
    - $\bar{\gamma} :=$ optimal value;
return $\bar{\gamma}$
```

3.3.1 Searching for an improving set of paths

At each iteration of the HPG algorithm the congestion model provides a heuristic solution. The objective of the routine *SearchForImprovingPaths* is to find an improving set of paths, that is a set of paths such that, when added to the restricted set, the congestion model produces a new (sub-)optimal solution with a smaller network congestion. In order to provide details on how the routine *SearchForImprovingPaths* works we need to introduce a few definitions and state some properties of a feasible solution of the congestion model.

Let variables $\widehat{y}$ indicate a feasible solution of the congestion model, which we call a feasible flow assignment, and let $\widehat{\rho}$ be the corresponding network congestion level. For all $c \in C$, let $\widehat{P}_c \subseteq P_c$ be the set of paths with a positive flow ($\widehat{y}_{c,p} > 0$) and $\widehat{P} = \bigcup_{c \in C} \widehat{P}_c$. Finally, an arc is said to be critical if its congestion level equals $\widehat{\rho}$. Analogously, an OD pair c and a path $p \in P_c$ are said to be critical if their congestion levels equal $\widehat{\rho}$, respectively.

Proposition 1. *Let $\widehat{y}$ be a feasible flow assignment with network congestion level $\widehat{\rho}$ and let c be a critical OD pair. If a path $p \in P_c$ exists with congestion level strictly less than $\widehat{\rho}$, then a feasible flow assignment exists such that:*

1. *No arc increases its congestion level to $\widehat{\rho}$ or above.*
2. *Congestion level of path p is strictly less than $\widehat{\rho}$.*
3. *Congestion level of c is strictly less than $\widehat{\rho}$.*
4. *No other OD pair increases its congestion level to $\widehat{\rho}$.*
5. *Network congestion level does not increase.*

Proof. Let us move an arbitrarily small flow $\epsilon > 0$ from every critical path in $\widehat{P}_c$ to path p.

1. The only arcs that may increase their congestion level are those in p as they receive a positive flow ϵ from other paths. Provided ϵ is small enough to keep the path congestion level of p strictly lower than $\widehat{\rho}$, no arc increases its congestion level to $\widehat{\rho}$ or above.
2. It follows directly from Proposition 1.1.
3. All paths in $\widehat{P}_c$ with congestion level $\widehat{\rho}$ yield a positive flow ϵ to p. Consequently, their congestion level must decrease by a positive quantity. Proposition 1.1 completes the argument.

4. The only OD pairs that can increase their congestion level are those with paths sharing arcs with p, but from Proposition 1.2 we know that all these arcs maintain a congestion strictly less than $\widehat{\rho}$.

5. It follows directly from Proposition 1.1.

□

Corollary 1. *If the new flow assignment $\widetilde{y}$ produced under Proposition 1 assumptions has network congestion level $\widetilde{\rho} = \widehat{\rho}$, then:*

1. *Critical OD pairs in $\widetilde{y}$ are a proper subset of critical OD pairs in $\widehat{y}$. In particular, OD pair c is not critical in $\widetilde{y}$.*

2. *Critical arcs in $\widetilde{y}$ are a proper subset of critical arcs in $\widehat{y}$. In particular, critical arcs in critical paths $p \in \widehat{P}_c$ are not critical in $\widetilde{y}$.*

Proof. It follows directly from Proposition 1. □

Proposition 2. *Let $\widehat{y}$ be a feasible flow assignment with network congestion level $\widehat{\rho}$ and let h be a critical arc in a critical path $p \in \widehat{P}_c$ for some critical OD pair $c \in C$. Let us also assume that a path $p' \in P_c$ exists such that p' may contain some critical arcs from p, but not h and no critical arc from any other path.*

Then, a feasible flow assignment exists such that:

1. *No arc increases its congestion level to $\widehat{\rho}$ or above.*

2. *Congestion level of h is strictly less than $\widehat{\rho}$.*

3. *Congestion level of p' is at most $\widehat{\rho}$.*

4. *No other OD pair increases its congestion level to $\widehat{\rho}$.*

5. *Network congestion level is at most $\widehat{\rho}$.*

Proof. Let us move an arbitrarily positive flow $\epsilon > 0$ from p to p' small enough to keep the path congestion level of p' stricly lower than $\widehat{\rho}$.

1. Let us consider four different types of arcs:
   - Arcs shared by paths p and p' maintain their congestion level as the flow moved from p to p' converges again on these arcs. In particular, shared critical arcs maintain their congestion level at $\widehat{\rho}$, while other shared arcs maintain their congestion strictly below $\widehat{\rho}$.
   - Arcs in p but not in p' decrease their congestion level as the total flow through them is decreased by ϵ. This is true in particular for arc h.
   - Arcs in p' but not in p increase their congestion level; however, provided that ϵ is small enough, their congestion levels remain strictly smaller than $\widehat{\rho}$.
   - Arcs neither in p nor in p' remain unchanged.
2. It follows directly from argument for Proposition 2.1.
3. It follows directly from argument for Proposition 2.1.
4. The only OD pairs that can increase their congestion level are those with paths sharing arcs with p, but from Proposition 2.2 we know that all these arcs maintain a congestion strictly less than $\widehat{\rho}$.
5. It follows directly from argument for Proposition 2.1.

□

Corollary 2. *If the new flow assignment $\widetilde{y}$ produced under Proposition 2 assumptions has network congestion level $\widetilde{\rho} = \widehat{\rho}$, then critical arcs in $\widetilde{y}$ are a proper subset of critical arcs in $\widehat{y}$. In particular, arc h is not critical in $\widetilde{y}$.*

Proof. It follows directly from Proposition 2. □

Remark 6. *If assumptions of Proposition 1 are satisfied, assumptions of Propositions 2 are satisfied for all critical arcs in $\widehat{P}_c$. In this sense Proposition 1 is a particular case of Proposition 2.*

Proposition 3. *Let a feasible flow assignment $\widehat{y}$ on which Proposition 2 can be applied be given and let $\widehat{\rho}$ be its network congestion level. Let $\widetilde{y}$ the feasible flow assignment resulting from the Proposition 2 application and let $\widetilde{\rho}$ be its network congestion level. Then there are three possible cases:*

- *$\widetilde{\rho}=\widehat{\rho}$ and Proposition 2 can be applied on the new flow assignment $\widetilde{y}$;*
- *$\widetilde{\rho}=\widehat{\rho}$ and Proposition 2 cannot be applied because there exists no path satisfying the assumptions of the Proposition 2;*
- *$\widetilde{\rho}<\widehat{\rho}$, i.e. there are no more critical OD pair.*

Proof. The new flow assignment $\widetilde{y}$ can have network congestion level equal or smaller than $\widehat{\rho}$. If $\widetilde{\rho}<\widehat{\rho}$ than there are no more critical OD pairs since there is no arc with arc congestion level greater or equal to $\widehat{\rho}$. According to Corollary 2, if network congestion level $\widetilde{\rho}$, corresponding to the new feasible flow assignment $\widetilde{y}$ found by Proposition 2, equals $\widehat{\rho}$, the number of arcs with congestion level $\widehat{\rho}$ is reduced with respect to $\widehat{y}$. Thus, either Proposition 2 can be applied on $\widetilde{y}$ and, hence, the new flow assignment satisfy one of the three possible cases or the Proposition 2 cannot be applied because there exists no path satisfying the assumptions. □

Remark 7. *Let $\widetilde{y}$ be the new flow assignment obtained after applying the transformation described in Proposition 1 or Proposition 2. If $\widetilde{\rho}=\widehat{\rho}$ and a critical OD pair $\widetilde{c}$ for $\widetilde{y}$ has a path $\widetilde{p} \in \widetilde{P}_{\widetilde{c}}$ with no critical arcs, then Proposition 1 can be recursively applied on feasible flow $\widetilde{y}$, OD pair $\widetilde{c}$ and path $\widetilde{p}$.*

The basic step of the *SearchForImprovingPaths* routine is to search for a path $p' \in P$ that satisfies the assumptions of Proposition 2 for some critical arc h of a path $p \in \widehat{P}_c$ for some critical OD pair c. According to Corollary 2, such path is able to relieve arc h and, according to Remark 7 and Corollary 1, may also relieve a critical path, and, recursively, even more critical arcs, and critical paths. The recursion stops when the relieved arc is not able to relieve a whole path. We use the word *relieve* with respect to a critical element (arc, path, OD pair) in the sense that a new traffic assignment can be found such that the congestion level of the relieved critical element is strictly less than $\widehat{\rho}$ while the network congestion level does not get worse. If, according to Proposition 3, we manage to iterate the basic step until all critical OD pairs are relieved, we finally improve the network congestion. Notice that according to Remark 6 we do not need to check explicitly for assumptions of Proposition 1.

Routine *SearchForImprovingPaths* is sketched in Algorithm 4. The initialization phase exploits the current optimal solution of the congestion model on $\widehat{P}$, $\widehat{y}$ and $\widehat{\rho}$. The set $ImprovingPaths$ is first initialized to the empty set. Set A^{crit} is initialized with all critical arcs, set C^{crit} is initialized with all critical OD pairs, and sets $A^{crit}_{c,p}$ for all $c \in C^{crit}$ and $p \in \widehat{P}_c$ are initialized with critical arcs in path $p \in \widehat{P}_c$.

During the main loop execution, for each critical OD pair $c \in C^{crit}$, for each critical path $p \in \widehat{P}_c$ and for each critical arc $h \in A^{crit}_{c,p}$, routine *SearchForNewPath* is used to search for a path p' from O_c to D_c such that the assumptions of Proposition 2 are satisfied. If the search is successful, the new path is stored in $ImprovingPaths$ and *RelieveCriticalElements* routine is run to find the relieved critical elements in a potential flow assignment on $\widehat{P} \bigcup ImprovingPaths$ and, consequently, update the critical sets $C^{crit}, A^{crit}, A^{crit}_{c,p}$. Details on this task will be given in Section 3.3.3. When C^{crit} has been emptied, all critical OD pairs are relieved and the algorithm terminates with all the paths stored in $ImprovingPaths$. This set guarantees that there exists a feasible flow assignment on $\widehat{P} \bigcup ImprovingPath$ that allows a congestion network smaller than $\widehat{\rho}$. If, at some iteration, the search of a new path p' fails on all triples (c, p, h), then the algorithm is not able to relieve some of the critical OD pairs and returns an empty set.

Algorithm 4: *SearchForImprovingPaths*

```
input  : $\widehat{P}, \widehat{y}, \widehat{\rho}$
output: A list of paths $ImprovingPaths$
global : $G, C, C^{crit}, A^{crit}, A^{crit}_{c,p}$
– $ImprovingPaths := \emptyset$;
– Using solution $\widehat{P}$, $\widehat{y}$, $\widehat{\rho}$, initialize:
    – $A^{crit}$ := set of critical arcs;
    – $C^{crit}$ := set of critical OD pairs;
    – $A^{crit}_{c,p}$ := set of critical arcs $\forall c \in C^{crit}, p \in \widehat{P}_c$;
while $C^{crit} \neq \emptyset$ do
    – $success := false$;
    for $c \in C^{crit}$ do
        for $p \in \widehat{P}_c$ do
            for $h \in A^{crit}_{c,p}$ do
                – $p' := SearchForNewPath(c, p, h)$;
                if $p'$ is not null then
                    – $success := true$;
                    – add $p'$ to $ImprovingPaths$;
                    – Execute $RelieveCriticalElements(h)$ to update $C^{crit}, A^{crit}, A^{crit}_{c,p}$;
    if $success = false$ then
        – return $\emptyset$;
– return $ImprovingPaths$;
```

3.3.2 Searching for the new path

Given an OD pair $c \in C^{crit}$, a critical path $p \in \widehat{P}_c$ and a critical arc $h \in A^{crit}_{c,p}$, the *SearchForNewPath* routine, sketched in Algorithm 5, is devoted to finding a feasible path p' from O_c to D_c able to relieve arc h. In order to do that, a subgraph of G is created: all critical arcs A^{crit} are first removed from the graph, then all arcs in $A^{crit}_{c,p}$ but h are reinserted. In other words, the only critical arcs allowed in the new path are those that have already been relieved and the not-yet-relieved ones in p but h. The search for a path could address any path in P_c, here we search for the shortest one in view of the fact that this path will eventually concur to define a solution for the inconvenience model.

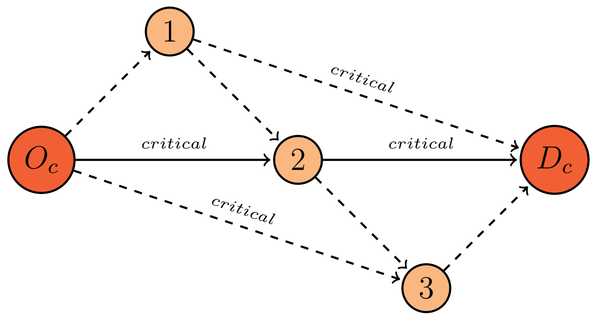
Algorithm 5: *SearchForNewPath*

```
input  : $c, p, h$
output: $p'$
global : $G, A^{crit}, A^{crit}_{c,p}$
– $A' := (A \setminus A^{crit}) \bigcup (A^{crit}_{c,p} \setminus \{h\})$;
– $G' := (A', V)$;
– Search in $G'$ the shortest feasible path $p'$ from $O_c$ to $D_c$;
if $p'$ exists then
    – return $p'$
else
    – return null
```

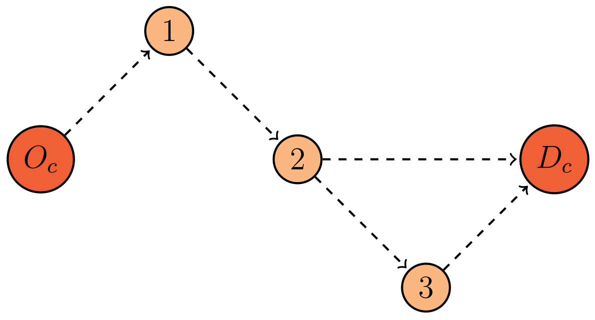
In Figures 3.1 and 3.2 we show an example of how *SearchForNewPath* routine operates in two different runs. In Figure 3.1(a) we have a critical OD pair c with a single path $p1 =< O_c - 2 - D_c >$ in $\widehat{P}$ emphasized with solid lines. Other arcs available in the graph are depicted with dashed lines. Critical arcs are appropriately labeled. For ease of exposition, suppose that all arcs have length 1 and that maximum allowed length is 3. Notice that path $< O_c - 1 - 2 - 3 - D_c >$, which could relieve both arcs $(O_c, 2)$ and $(2, D_c)$, is not feasible having length 4.

Suppose now that the first run is on path $p = p1$ and arc $h = (O_c, 2)$. Figure 3.1(b) shows the subgraph obtained after removing critical arcs not in p $\{(1 - D_c), (O_c, 3)\}$ and arc h; Figure 3.1(c) shows the new path $p2 =< O_c - 1 - 2 - D_c >$ found and able to relieve arc h.

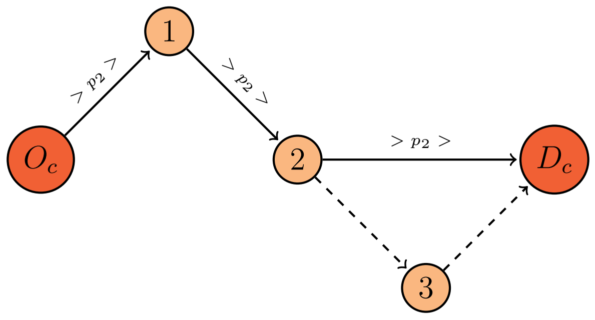
In Figure 3.2(a) we see the generated subgraph in a second run of *SearchForNewPath* routine where $p = p1$ and $h = (2, D_c)$: critical arcs not in p ($(1, D_c)$ and $(O_c, 3)$) and arc h are removed. Figure 3.2(b) shows the new path $p3 =< O_c - 2 - 3 - D_c >$ found and able to relieve arc h.



(a) Critical arcs based on $\widehat{P}, \widehat{y}, \widehat{\rho}$

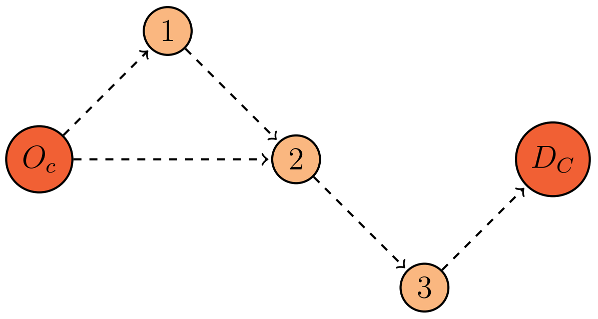


(b) Subgraph G' with $p_1 =< O_c - 2 - D_c >$, $h = (O_c, 2)$

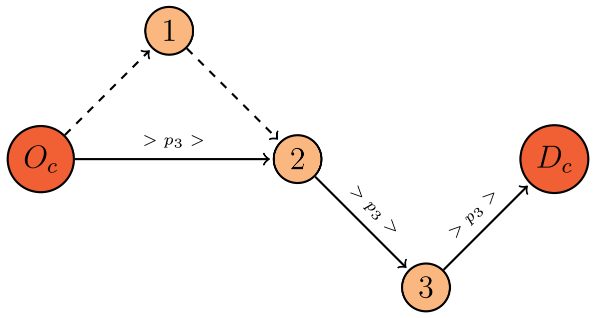


(c) New path relieving arc $(O_c, 2)$

Figure 3.1. Heuristic Path Generation: an example of *SearchForNewPath* routine for a path containing two critical arcs (continues in Figure 3.2)



(a) Subgraph G' with $p =< O_c - 2 - D_c >$, $h = (2, D_c)$



(b) New path relieving arc $(2, D_c)$

Figure 3.2. Heuristic Path Generation: an example of *SearchForNewPath* routine for a path containing two critical arcs

3.3.3 Relieving critical elements

When a new path p' is found by *SearchForNewPath* routine, at least one arc h can be relieved. As a consequence of this fact, the network congestion level cannot increase and the number of arcs with congestion level $\widehat{\rho}$ decreases as well (see Corollary 2). For this reason we remove the relieved arc from A^{crit} and all $A^{crit}_{c,p}$ where present. By doing this it could happen that a set $A^{crit}_{c,p}$ is emptied for some $c \in C^{crit}$ and $p \in \widehat{P}_c$. In this lucky case, according to Remark 7 and Proposition 2, we know that all critical arcs on p are relieved. Thus, all arcs still in $A^{crit}_{c,p}$ for $p \in \widehat{P}_c$ are automatically relieved and can be recursively treated by *RelieveCriticalElements* without adding any new path to $\widehat{P}_c$. Moreover, OD pair c is relieved and we remove it from C^{crit}.

This is what *RelieveCriticalElements* routine does. Details are provided in Algorithm 6. In particular, recursion is implemented in iterative form by adding relieved arcs to a queue.

In Figure 3.3 an example is shown of how the *RelieveCriticalElements* routine works. In Figure 3.3(a) we see the optimal solution produced by the congestion model at the first iteration of the HPG algorithm. Set $\widehat{P}$ contains paths $p_1 =< O_1 - 2 - D >$ and $p_2 =< O_2 - 2 - D >$.

Algorithm 6: *RelieveCriticalElements*

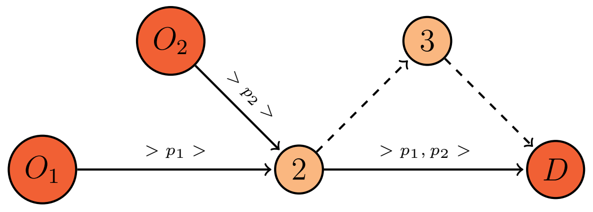
```
input  : $h$
global : $\widehat{P}, C^{crit}, A^{crit}, A^{crit}_{c,p}$
– $R := \{h\}$;
while $R \neq \emptyset$ do
    – Remove one arc $a$ from $R$;
    – Remove $a$ from set $A^{crit}$;
    for all $c \in C^{crit}$ and $p \in \widehat{P}_c$ do
        – Remove $a$ from $A^{crit}_{c,p}$, if present;
    for all $c \in C^{crit}$ do
        if $p \in \widehat{P}_c$ exists such that $A^{crit}_{c,p} = \emptyset$ then
            – Remove $c$ from $C^{crit}$;
            for all $p \in \widehat{P}_c$ do
                – Add $A^{crit}_{c,p}$ to $R$; // avoid duplicates
– return;
```

In a first scenario let us assume that only arc $(2, D)$ is critical. *SearchForImproving-Paths* routine would initialize $A^{crit} = \{(2, D)\}, A^{crit}_{1,p_1} = \{(2, D)\}, A^{crit}_{2,p_2} = \{(2, D)\}$ and $C^{crit} = \{1, 2\}$. When *SearchForImprovingPaths* routine executes *SearchForNewPath* on $c = 1, p = p_1$, and $h = (2, D)$ a path $p' =< O_1 - 2 - 3 - D >$ is found (see Figure 3.3(b)). Now arc $(2, D)$ can be relieved: routine *RelieveCriticalElements* removes arc $(2, D)$ from $A^{crit}, A^{crit}_{1,p_1}$ and A^{crit}_{2,p_2}; since A^{crit}_{1,p_1} and A^{crit}_{2,p_2} are empty, OD pairs 1 and 2 are removed from C^{crit}. Since C^{crit} is now empty, *SearchForImprovingPaths* return path p' which is sufficient to improve current solution.

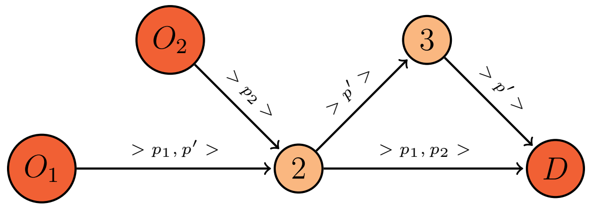
In another scenario, let us assume that in Figure 3.3(a) arcs $(O_1, 2)$ and $(2, D)$ are critical. *SearchForImprovingPaths* routine would initialize $A^{crit} = \{(O_1, 2), (2, D)\}, A^{crit}_{1,p_1} = \{(O_1, 2), (2, D)\}, A^{crit}_{2,p_2} = \{(2, D)\}$ and $C^{crit} = \{1, 2\}$. When *SearchForImprovingPaths* routine executes *SearchForNewPath* on $c = 1, p = p_1$, and $h = (2, D)$ a path $p' =< O_1 - 2 - 3 - D >$ is found (see Figure 3.3(b)). Now arc $(2, D)$ can be relieved: routine *RelieveCriticalElements* removes arc $(2, D)$ from $A^{crit}, A^{crit}_{1,p_1}$ and A^{crit}_{2,p_2}; since A^{crit}_{2,p_2} only is empty, OD pair and 2 is removed from C^{crit}. Routine *SearchForImprovingPaths* would try another iteration to relieve OD pair 1.

3.3.4 HPG may miss to find some optimal paths

The HPG algorithm aims at generating a good solution for the inconvenience model by iteratively solving restricted versions of the congestion model. At each iteration a set of paths to be added to a restricted set are searched for in such a way that the congestion level can be reduced. When the *SearchForImprovingPaths* routine ends with a set of paths, we are sure that the current solution can be improved, but the converse is not true. Here, we provide two examples depicted in Figures 3.4 and 3.5 to illustrate this



(a) Solution of the congestion model on the first restricted set



(b) A new path p' for OD pair 1

Figure 3.3. Heuristic Path Generation: an example of *RelieveCriticalElements* routine

point.

Figure 3.4 shows an instance in which the network congestion level obtained with the HPG algorithm is suboptimal. Let the network be as in Figure 3.4(a) and let the maximum allowed travel inconvenience be $\gamma = 30\%$. All arcs have capacity u equal to 1, all the four OD pairs $C = \{1, 2, 3, 4\}$ have demand equal to 1 and the arc duration is equal to 1 for each arc. In Figure 3.4(b) the optimal flows for the congestion and for the inconvenience models are shown: labels f_c on arcs represent the flow of OD pair c. It is easy to see that the demand is completely routed and the network congestion level is $\frac{5}{6}$. However, the HPG algorithm stops with a network congestion level of 1. That happens because the algorithm misses some paths used in the optimal solution of the congestion model. In the first iteration of *SearchForImprovingPaths* the shortest paths only are considered (Figure 3.4(c)). All OD pairs are critical and critical arcs are shown in Figure 3.4(d). The *SearchForNewPath* routine finds a new path only for OD pair 3 $SP3_2 \equiv < O_3 - D_4 - D_3 >$ (Figure 3.4(e)), the critical arc set is reduced by the *RelieveCriticalElements* and the critical OD pairs set becomes $C^{crit} = \{1, 2, 4\}$ (Figure 3.4(f)). Now *SearchForNewPath* is not able to relieve any other arc and fails. The same does *SearchForImprovingPaths* routine. Thus, the HPG algorithm stops at the value $\rho' = 1$. However, in this case the HPG algorithm finds the optimal solution of the inconvenience model because the shortest paths are sufficient to keep the network uncongested.

In Figure 3.5 an example where the HPG algorithm finds the minimum network congestion level and fails to find the optimal solution of the inconvenience model is provided. Let us consider a network with 2 OD pairs $C = \{1, 2\}$, $\gamma = 30\%$, and demand $d_1 = 8$ and $d_2 = 7$. Capacities and arc durations are as shown in Figure 3.5(a). In Figure 3.5(b) the optimal flows for the congestion and the inconvenience model are shown: labels f_c on arcs represent the flow of OD pair c on the arc. It is easy to see that the demand is completely routed, the network congestion level is equal to 1 and the weighted average travel inconvenience is around 2.5%. The paths considered in the first iteration of

SearchForImprovingPaths are shown in Figure 3.5(c), critical OD pairs are $\{1, 2\}$; there is only one critical arc with congestion level 1.5 shown in Figure 3.5(d). First iteration of routine *SearchForNewPath* finds a path $SP1_2$ represented in Figure 3.5(e) that relieves the only critical arc and OD pairs 1 and 2 altogether. Thus, *SearchForImprovingPaths* is successful and returns only one path $SP1_2$. The new obtained network congestion level is 1 with only one critical arc shown in Figure 3.5(f). At this point, during the second iteration of *SearchForImprovingPaths*, routine *SearchForNewPath* fails and the same does *SearchForImprovingPaths*. The set of paths for the inconvenience model is suboptimal, although the congestion network level is optimal, and the resulting weighted average travel inconvenience is $8.\bar{3}\%$.

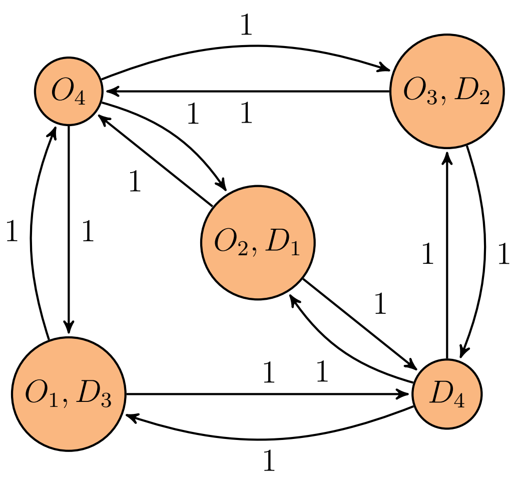
3.4 Computational results

The HPG algorithm and the CE 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 16 GB Ram. Experiments are organized in two parts. The first part is mainly devoted to comparing the set of paths generated with the CE and the set generated with the HPG algorithm on instances of increasing size. Experiments were carried out using 8 benchmark instances, with up to 330 nodes, and values of γ ranging from $\gamma = 0\%$ to $\gamma = 25\%$ with step 5%. The second part is devoted to comparing the optimal solutions of the congestion and inconvenience models (with paths generated through the CE) and the heuristic solutions of the models when only the paths generated with the HPG algorithm are used. Here experiments were carried out on 40 benchmark instances with 150 nodes and values for γ ranging from $\gamma = 0\%$ to $\gamma = 35\%$ with step 5%. The instances were generated taking into account different demand patterns and point attractiveness distributions as explained thoroughly in Chapter 6 and are available at `http://or-brescia.unibs.it/instances`. The statistics collected for each instance are described in Section 3.4.1. Results for the former set of experiments are presented and discussed in Section 3.4.2. Section 3.4.3 is devoted to summarize results for the latter set of experiments.

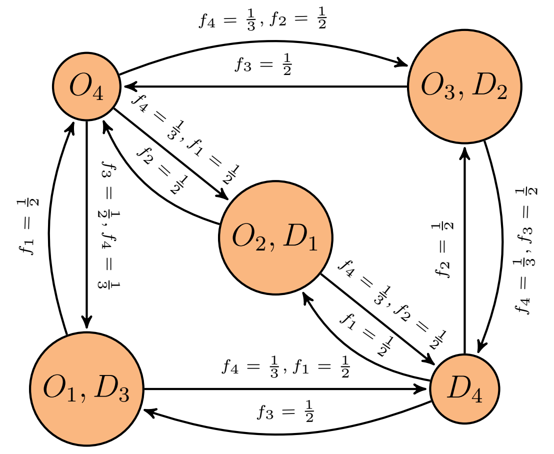
3.4.1 Statistics

A number of statistics on the CE and the HPG algorithm are collected or computed on all the tested instances.

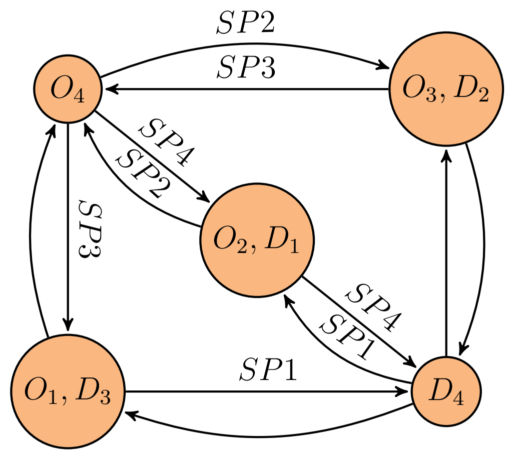
- Computational time. The computational time is computed considering the total computational time. For the CE this is the time to generate the paths and to solve the congestion and the inconvenience models. For the HPG algorithm it is the time required by the HPG algorithm.



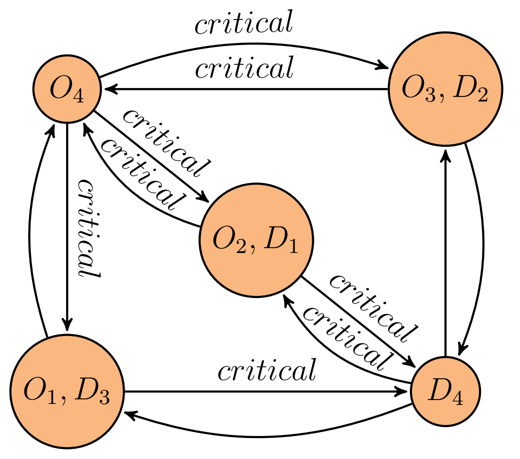
(a) Initial graph with arc durations


O_4
O_3, D_2
O_2, D_1
O_1, D_3
D_4
$f_4 = \frac{1}{3}, f_2 = \frac{1}{2}$
$f_3 = \frac{1}{2}$
$f_4 = \frac{1}{3}, f_1 = \frac{1}{2}$
$f_2 = \frac{1}{2}$
$f_1 = \frac{1}{2}$
$f_3 = \frac{1}{2}, f_4 = \frac{1}{3}$
$f_2 = \frac{1}{2}$
$f_4 = \frac{1}{3}, f_3 = \frac{1}{2}$
$f_4 = \frac{1}{3}, f_2 = \frac{1}{2}$
$f_1 = \frac{1}{2}$
$f_4 = \frac{1}{3}, f_1 = \frac{1}{2}$
$f_3 = \frac{1}{2}$

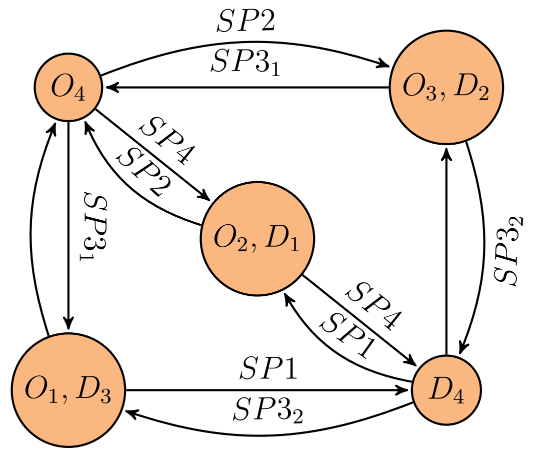
(b) Optimal flows for the congestion and the inconvenience models



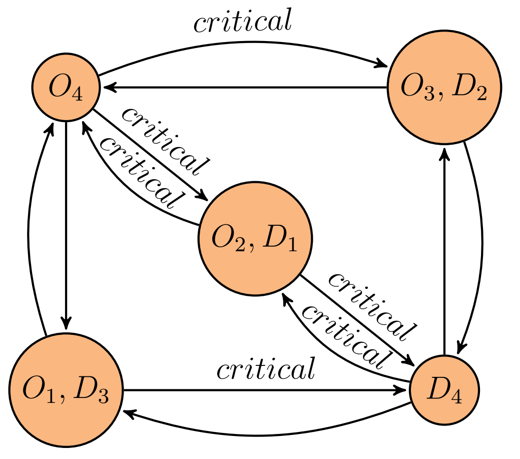
(c) Initial paths for the HPG algorithm: shortest paths



(d) Critical arcs with the shortest paths

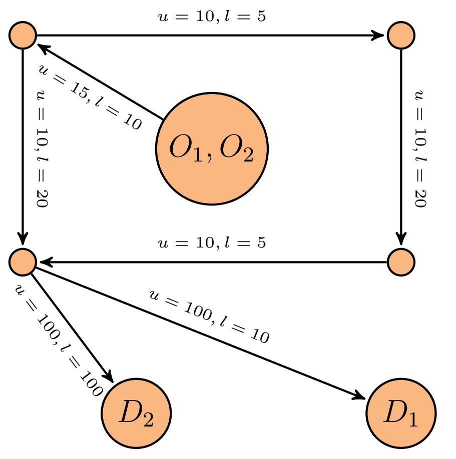


(e) Augmented path set after the first iteration of *SearchForNewPath*

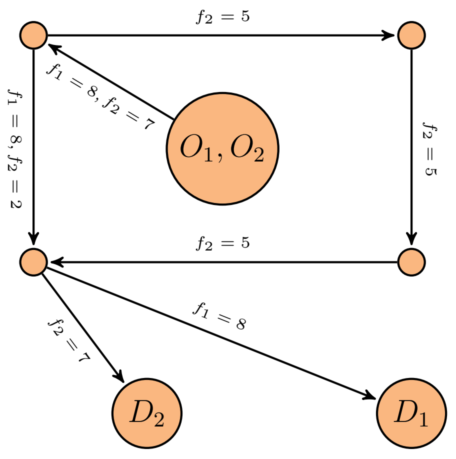


(f) Critical arcs after the first iteration of *SearchForNewPath*

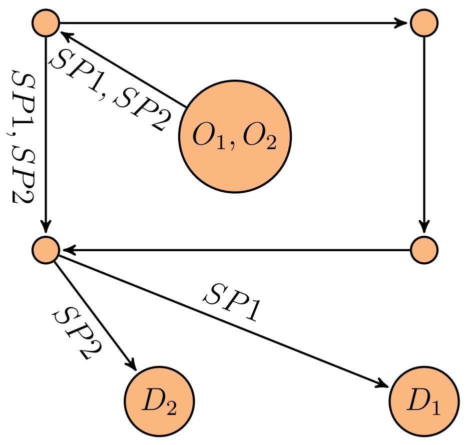
Figure 3.4. Heuristic Path Generation: failure of *SearchForImprovingPaths* routine at the first iteration



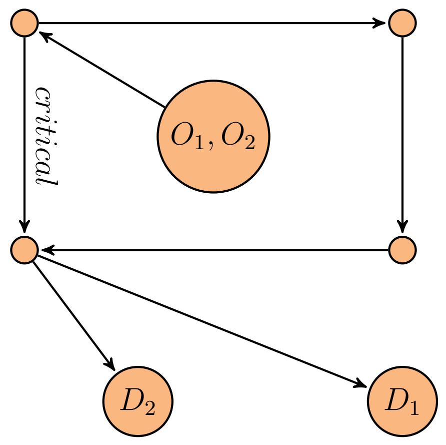
(a) Initial graph with arc capacities u and durations l



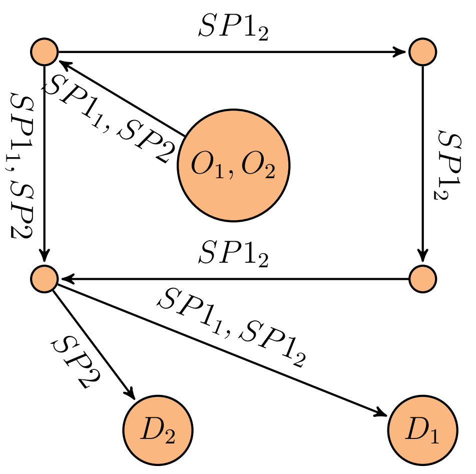
(b) Optimal flows for the congestion and the inconvenience models



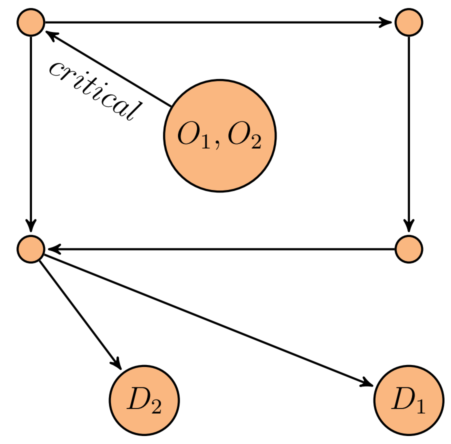
(c) Initial paths for the HPG algorithm: shortest paths



(d) Critical arcs with the shortest paths



(e) New path set after the first iteration of *SearchForImprovingPaths*



(f) Critical arc set during the second iteration of *SearchForImprovingPaths*

Figure 3.5. Heuristic Path Generation: failure of *SearchForImprovingPaths* routine at the second iteration

- Network Congestion.
  - network congestion level: the network congestion level corresponding to the generated solution.
  - the 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 ($1.5 < x_{ij}/u_{ij}$)

    For this statistic results are reported in a graphical form.
  - relative congestion gap $\epsilon_\gamma = \frac{\rho_{HPG} - \rho_{CE}}{\rho_{CE}}$, where ρ_{CE} is the optimum of the congestion model (with paths generated through CE) while ρ_{HPG} is the value of the congestion model when paths are generated with the HPG algorithm.
- User Experience.
  - weighted average travel inconvenience corresponding to the generated solutions.
  - absolute inconvenience gap, $\kappa_\gamma = \bar{\gamma}_{HPG} - \bar{\gamma}_{CE}$, where $\bar{\gamma}_{CE}$ is the optimum of the inconvenience model (with paths generated through CE), while $\bar{\gamma}_{HPG}$ is the value of the inconvenience model when paths are generated with the HPG algorithm. Usually gaps are computed in terms of relative error but the weighted average travel inconvenience is a percentage itself and, using a percentage of a percentage as quality measure can be misleading because if the weighted average travel inconvenience is very small, then the relative error will turn out to be very high, even if the difference between the two solutions is small.
- Memory usage. The number of generated paths.
- Iterations. The number of iterations performed by the HPG algorithm.

3.4.2 Comparison of the heuristic set of paths with the complete set

As already mentioned, experiments were carried out considering 8 networks, with a number of nodes that ranges from 120 to 330 and with γ values ranging from $\gamma = 0\%$ to $\gamma = 25\%$ with step 5%. In Tables 3.2 and 3.3 the statistics are presented. At each step the number of nodes in the network is increased by 30 nodes. Note that, for instances with a number of nodes greater than 150 and for some values of γ, the statistics for the CE are not shown. This is because the solver was not able to produce a solution because it ran out of memory. For these cases we report the number of paths generated by the

CE with the computational time required to generate them (values are marked with *). We observe how slowly the number of paths generated by the HPG algorithm grows with respect to the CE as the number of nodes increases. With $\gamma = 25\%$, for the 120 nodes instance this number is approximately 3% of the number generated by the CE while for the 180 nodes instance it is approximately 0.4%. The percentage with $\gamma = 25\%$ and a higher number of nodes is not available because the solver is not able to handle the number of variables needed for the CE. However, considering $\gamma = 20\%$, for the 210 nodes instance the percentage is approximately 0.3%. Considering 240, 270 and 300 nodes instances, the percentage experienced with $\gamma = 15\%$ is 0.4%, 0.2% and 0.06%, respectively. For the 330 nodes instance, the value of γ for which we have the statistic is 10% and the percentage of paths generated with the HPG algorithm with respect to the CE is 0.1%. These statistics indicate that the HPG algorithm requires a memory that may be 3 orders of magnitude less than the CE. From the point of view of the computational time, it is possible to observe the same positive behaviour of the HPG algorithm. The percentage of the computational time required by the HPG algorithm with respect to the CE is around 50% considering $\gamma = 25\%$ and 120 nodes, whereas considering 150 nodes this percentage decreases to 2.7%. For a higher number of nodes we do not have results for $\gamma = 25\%$. However, with $\gamma = 20\%$ on the 180 nodes instance the HPG algorithm takes only 2.8% of the computational time taken by the CE. The best result in terms of computational time happens with the 300 nodes instance where the largest value of γ for which we have the CE solution is 10%. In this case the HPG algorithm takes 0.28% of the time spent by the CE.

After the analysis of the memory and time saving of the HPG algorithm on these instances, we consider its effectiveness in generating high quality solutions. The relative congestion gap ϵ_γ is very low. The maximum experienced value of ϵ_γ is around 4.4% with 150 nodes and with $\gamma = 25\%$. From the point of view of the weighted average travel inconvenience, the κ_γ value has a tendency to increase with γ, but is always below 1.1%. This means that a user routed using the HPG algorithm can experience, on average, a travel inconvenience that is 1.1% higher than with the CE. Furthermore, this value appears with high values of γ. With $\gamma = 25\%$ the weighted average travel inconvenience is on average 7%. It means that with the HPG algorithm users will experience on average 8%, in the worst case.

3.4.3 Effectiveness of the heuristic set of paths

In this section we summarize the results obtained on 40 benchmark instances with 150 nodes and values of γ ranging from $\gamma = 0\%$ to $\gamma = 35\%$ with step 5% In Table 3.4 the computational times required by the CE and by the HPG algorithm are shown. The HPG algorithm confirms to be widely less time consuming than the CE, especially when high values of γ are considered. In fact, with $\gamma = 35\%$ the CE, on average, spends more than 1000 seconds per instance while the HPG algorithm spends 90 seconds per instance. In this case the use of the HPG algorithm with γ up to 10% is not convenient because it takes an amount of time similar to the CE. However, as shown in Table 3.3, the HPG

		γ					
Number of nodes	Stats	0%	5%	10%	15%	20%	25%
120	Paths CE	1170	2335	6766	17448	39528	79935
	Paths HPG	1170	1242	1549	2034	2367	2676
	Time CE (sec)	1	2	4	9	18	35
	Time HPG (sec)	0	1	4	7	10	18
	Congestion gap ϵ_γ (%)	0	0	0	0	0	0
	Inconvenience gap κ_γ (%)	0	0.0001	0.001	0.002	0.005	0.008
150	Paths CE	1170	4157	18849	70215	216528	575120
	Paths HPG	1170	1343	2460	3758	4626	5305
	Time CE (sec)	1	4	13	42	285	1609
	Time HPG (sec)	0	3	15	29	41	45
	Congestion gap ϵ_γ (%)	0	0	0	0	0	4.4
	Inconvenience gap κ_γ (%)	0	0.04	0.4	0.7	1.1	1
180	Paths CE	1170	6758	45828	230508	916527	3068987
	Paths HPG	1170	1286	2876	4123	10397	12508
	Time CE (sec)	1	7	39	376	4848	1089(*)
	Time HPG (sec)	0	2	20	31	137	169
	Congestion gap ϵ_γ (%)	0	0	0	0	1.8	–
	Inconvenience gap κ_γ (%)	0	0	0.4	0.6	0.2	–
210	Paths CE	1170	10496	98330	640963	3212485	–
	Paths HPG	1170	1532	2832	5821	9443	14555
	Time CE (sec)	2	11	95	2688	1472(*)	–
	Time HPG (sec)	0	4	19	61	111	261
	Congestion gap ϵ_γ (%)	0	0	0	0	–	–
	Inconvenience gap κ_γ (%)	0	0.1	0.4	0.5	–	–

Table 3.2. Heuristic Path Generation: results for different number of nodes in the network

Number of nodes	Stats	γ					
		0%	5%	10%	15%	20%	25%
240	Paths CE	1170	19084	257488	2201324	–	–
	Paths HPG	1170	2025	2653	8363	13929	16770
	Time CE (sec)	2	21	771	1408(*)	–	–
	Time HPG (sec)	0	15	16	109	275	374
	Congestion gap ϵ_γ (%)	0	0	0	–	–	–
	Inconvenience gap κ_γ (%)	0	0.2	0.4	–	–	–
270	Paths CE	1170	26820	449250	4588806	–	–
	Paths HPG	1170	1728	2093	8479	10119	9507
	Time CE (sec)	3	37	1892	3121(*)	–	–
	Time HPG (sec)	0	8	13	119	182	167
	Congestion gap ϵ_γ (%)	0	0	0	–	–	–
	Inconvenience gap κ_γ (%)	0	0.1	0.3	–	–	–
300	Paths CE	1170	37475	777428	9649118	–	–
	Paths HPG	1170	1554	2060	6215	9855	8967
	Time CE (sec)	3	60	5106	7476(*)	–	–
	Time HPG (sec)	0	9	15	92	194	160
	Congestion gap ϵ_γ (%)	0	0	0	–	–	–
	Inconvenience gap κ_γ (%)	0	0.1	0.3	–	–	–
330	Paths CE	1170	54599	1481404	–	–	–
	Paths HPG	1170	1386	2291	5260	13306	13136
	Time CE (sec)	4	93	1640(*)	–	–	–
	Time HPG (sec)	0	4	16	65	350	340
	Congestion gap ϵ_γ (%)	0	0	–	–	–	–
	Inconvenience gap κ_γ (%)	0	0.1	–	–	–	–

Table 3.3. Heuristic Path Generation: results for different number of nodes in the network

algorithm becomes faster than the CE when the number of nodes increases, even for small values of γ.

	γ							
Time (sec)	0%	5%	10%	15%	20%	25%	30%	35%
HPG	0	3	11	19	27	42	62	90
CE	1	3	11	31	85	207	456	1084

Table 3.4. Heuristic Path Generation: computational time (sec)

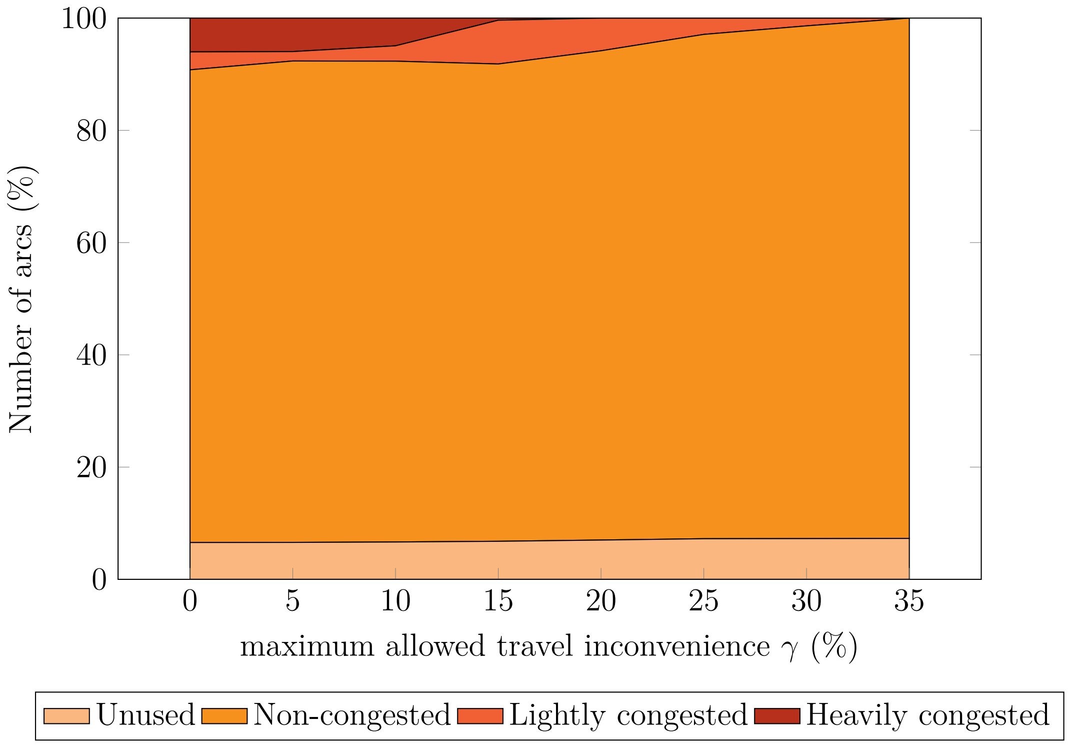
In Tables 3.5 and 3.6 statistics on the congestion gap ϵ_γ are shown. In Table 3.5 the average and the maximum ϵ_γ over all instances and for different γ values are shown. Note that, for $\gamma = 5\%$, ϵ_γ is 0 as all instances are optimally solved with respect to the congestion model. With $\gamma = 10\%$ two instances present positive, although negligible, values of ϵ_γ. With higher values of γ, the average ϵ_γ increases. The worst case takes place when the γ value is 30% where the average ϵ_γ value is equal to 0.79%. The maximum value of ϵ_γ follows the behaviour of the average ϵ_γ with no congestion gap with $\gamma = 5\%$ and a very small value, 0.02%, with $\gamma = 10\%$. As in the average case, the worst case appears with $\gamma = 15\%$ for which the maximum value is 3.26%.

	γ							
ϵ_γ (%)	0%	5%	10%	15%	20%	25%	30%	35%
Average	0	0	0	0.51	0.41	0.75	0.79	0.17
Maximum	0	0	0.02	3.26	1.87	3.05	2.94	0.84

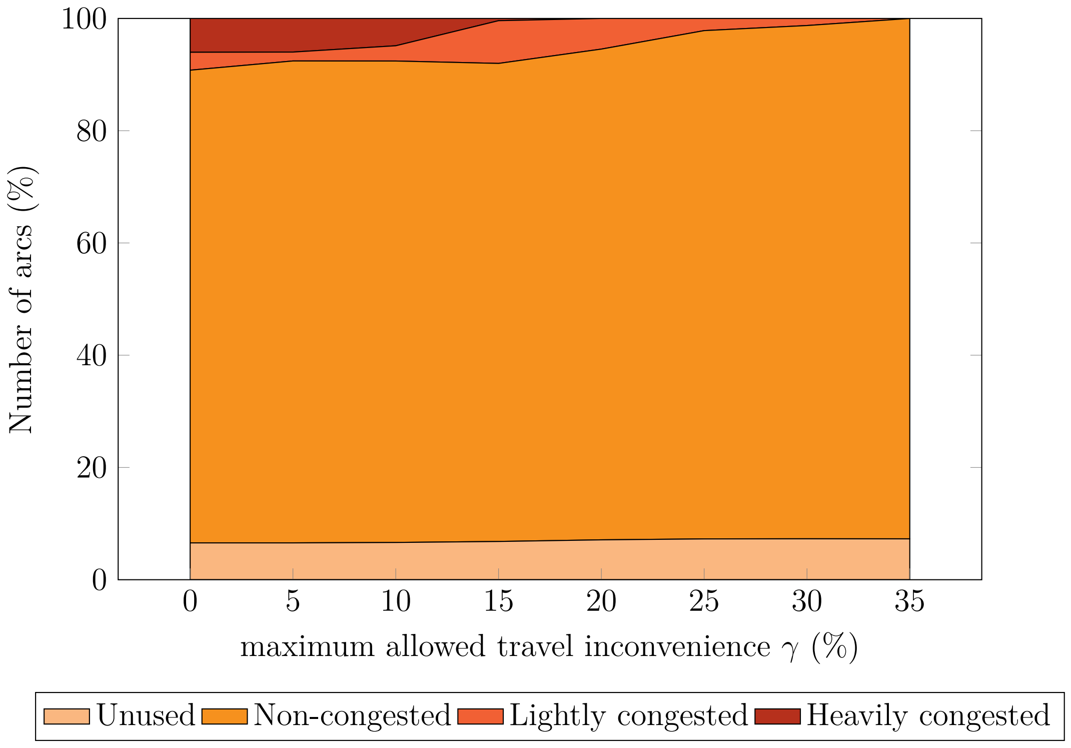
Table 3.5. Heuristic Path Generation: average and maximum relative congestion gap

Table 3.6 reports the number of instances reporting ϵ_γ in different classes of values. Almost all the instances are experiencing an ϵ_γ value under 3%. One exception is with $\gamma = 15\%$ where 5% of the instances (2 over 40 instances) are experiencing a value of ϵ_γ between 3% and 3.26%, which is the maximum ϵ_γ over all the instances with $\gamma = 15\%$. The other exception is when $\gamma = 25\%$ is considered and 5% of the instances have a congestion gap between 3% and 3.05% (maximum congestion gap over all instances with $\gamma = 25\%$). With $\gamma = 5\%$ the congestion gap is always 0 and with $\gamma = 10\%$ the congestion gap is very low (under 0.5%). When $\gamma = 10\%$, the HPG algorithm has strictly positive ϵ_γ only on 5% of the instances, that is on 2 over 40. The HPG algorithm produces $\epsilon_\gamma = 0$ with $\gamma = 15\%$ on 80% of the instances and the rest of the instances are experiencing ϵ_γ values distributed between 1% and 3%. For higher values of γ the percentage of instances experiencing a congestion gap equal to 0 decreases. For $\gamma = 35\%$, 55% of the instances present no congestion gap and all the instances are affected by congestion gaps that are not greater than 1%.

In Figures 3.6(a) and 3.6(b) the distribution of arc congestion level on different congestion classes is shown. Note that this value is averaged over all instances. The two graphics present almost the same behaviour. This means that, on average, the traffic distribution over all arcs is well approximated by the HPG algorithm.



(a) Arc congestion level distribution with the HPG algorithm



(b) Arc congestion level distribution with the CE

Figure 3.6. Heuristic Path Generation: arc congestion level distribution for the two methods

	γ							
ϵ_γ classes (%)	0%	5%	10%	15%	20%	25%	30%	35%
ϵ_γ =0%	100	100	95	80	70	40	45	55
0%< ϵ_γ ≤0.5%	0	0	5	0	5	15	10	35
0.5%<ϵ_γ ≤1%	0	0	0	0	5	15	10	10
1%<ϵ_γ ≤2%	0	0	0	5	20	15	25	0
2%<ϵ_γ ≤3%	0	0	0	10	0	10	10	0
3%<ϵ_γ ≤4%	0	0	0	5	0	5	0	0

Table 3.6. Heuristic Path Generation: instance experiencing ϵ_γ classes

Table 3.7 shows the behaviour of the absolute inconvenience gap κ_γ of the weighted average travel inconvenience. In the worst case, $\gamma = 20\%$, the value of κ_γ is on average less than 0.5%. This means that users are diverted to a route that is, on average, 0.5% longer than the one assigned by the CE. In terms of maximum, κ_γ is, in the worst case, less than 2.5%. In Table 3.8 the distribution of κ_γ among different classes is shown. Note that, considering $\gamma = 5\%$ and $\gamma = 10\%$, all the instances present κ_γ values greater than 0 but always lower than 0.5%. Regarding $\gamma = 10\%$, only 10% of the instances have $\kappa_\gamma = 0$, but almost all of the remaining instances are experiencing κ_γ values under 1%. For higher values of γ there are no instances experiencing $\kappa_\gamma = 0$, but very few instances are experiencing high values of κ_γ. In fact, in the worst case, $\gamma = 25\%$ and $\gamma = 30\%$, the percentage of instances experiencing κ_γ values greater than 1.5% is only 10%.

	γ							
κ_γ (%)	0%	5%	10%	15%	20%	25%	30%	35%
Average	0	0.04	0.14	0.26	0.50	0.49	0.44	0.44
Maximum	0	0.1	0.43	1.08	1.34	1.93	2.44	2.48

Table 3.7. Heuristic Path Generation: average and maximum absolute inconvenience gap on $\bar{\gamma}$

	γ							
κ_γ classes (%)	0%	5%	10%	15%	20%	25%	30%	35%
κ_γ =0%	100	0	0	10	0	0	0	0
0%< κ_γ ≤0.5%	0	100	100	55	55	55	70	70
0.5%<κ_γ ≤1%	0	0	0	30	10	25	20	15
1%<κ_γ ≤1.5%	0	0	0	5	35	10	0	10
1.5%<κ_γ ≤2%	0	0	0	0	0	10	5	0
2%<κ_γ ≤2.5%	0	0	0	0	0	0	5	5

Table 3.8. Heuristic Path Generation: instance experiencing κ_γ classes

In order to compare the memory usage of the HPG algorithm with respect to the CE, we report the number of paths generated by the two methods in Table 3.9. The number of paths generated with the CE rapidly grows with increasing values of γ. Notice that with $\gamma = 35\%$ less than 0.3% of paths are generated by the HPG algorithm with respect to the CE.

	γ							
Approach	0%	5%	10%	15%	20%	25%	30%	35%
HPG	1170	1449	2212	3129	4090	5397	6881	8375
CE	1170	4055	18536	69449	214936	572058	1355029	2944161
% HPG on CE	100	36	12	5	2	1	0.5	0.3
Iterations	1	149.35	366	607.8	782.5	928	1071	1202.6

Table 3.9. Heuristic Path Generation: number of paths and iterations of the HPG algorithm

In Table 3.9 also the number of iterations of algorithm *SearchForImprovingPaths* is shown. Consider that, at each iteration, a linear programming problem is solved on the restricted set and, on average with $\gamma = 35\%$, more than 1200 iterations are performed.

3.5 Conclusions

In this paper we have presented a heuristic for the generation of paths for the proactive route guidance approach, a linear programming based approach that aims at finding a system optimum traffic assignment that takes into account fairness for users by using only paths with limited inconvenience. The computational complexity of the approach is determined 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 the heuristic reduces by orders of magnitude the number of generated paths and the amount of memory usage. The results also show that the quality of the solutions of the proactive route guidance approach obtained using only the heuristic set of paths is very close to that of the optimal ones. Although the heuristic has been designed to generate a set of paths that allow us to find high quality solutions of the proactive route guidance approach, it may be useful for the generation of paths required by other approaches and models for traffic assignment.

4. System optimal routing of traffic flows with user constraints using linear programming

Abstract

For static traffic assignment problems, it is well-known that (1) the total travel time in a user-equilibrium solution can be substantially higher than the total travel time in a system-optimum solution, and (2) the user-experienced travel time in a system-optimum solution can be substantially higher that the user-experienced travel time in a user-equilibrium solution. By seeking system optimal traffic flows subject to user constraints, a comprise solution can be obtained that balances system and user objectives. We propose a linear programming based approach to efficiently obtain a solution that effectively balances system and user objectives. A computational study reveals that solutions with near-optimal total travel times can be found in which most users experience travel times that are better than user-equilibrium travel times and few users experience travel times that are slightly worse than user-equilibrium travel times.

Keywords Traffic assignment, congestion, latency function, constrained system optimum, linear programming, piecewise linear approximation

4.1 Introduction

Road congestion is one of the most pressing problems in urban areas, as it has negative economic, environmental, and health impacts. Since in most situations, road network infrastructure expansion is not feasible, because of a lack of available space or the huge costs, alternative solutions have to be explored. Route guidance is natural and popular choice. Many vehicles, nowadays, are equipped with navigational devices that can compute a shortest route (based on user preferences) from an origin to a destination. The latest versions of these devices can also display current traffic conditions and use current traffic data on shortest route computations. Unfortunately, these devices do not consider the potential impact of the directions provided to a driver on the traffic system. As a consequence, the same route may be suggested to many users, which, if the recommended routes are followed by the users, may simply result in congestion occurring in other parts of the road network rather than a reduction in system-wide congestion or in avoiding an increase in system-wide congestion. Coordinating route guidance across the users of the system, and adherence to the suggested routes by the users, is needed to reduce congestion or, even better, to avoid congestion. In the (near) future, with the introduction of self-driving vehicles, this may become a reality. Our research focuses on optimization models to support these developments.

Route choice, or traffic assignment, concerns the selection of routes (alternatively called paths) between origins and destinations in transportation networks. It is the fourth step in the conventional transportation forecasting model, following trip generation, trip distribution, and mode choice. Traffic assignment problems are defined on a road network with an origin-destination (OD) matrix specifying demand, i.e, the number of vehicles per time unit that is expected to travel from each origin to each destination. Details on forecasting demand on road networks can be found in Sheffi (1985), Ben-Akiva and Lerman (1985), Florian and Hearn (1999) and de Dios Ortuzar and Willumsen (2011).

A distinction is made between static and dynamic traffic assignment problems. In a dynamic traffic assignment problem, origin-destination demand varies over time. Early studies on dynamic traffic assignment date back to Merchant and Nemhauser (1978). However, this class of traffic assignment problems is very challenging Jahn et al. (2005) and there is no generally accepted model or methodology Papageorgiou (1990); Mahmassani and Peeta (1995); Peeta and Ziliaskopoulos (2001); Ben-Akiva et al. (2012). In a static traffic assignment problem, on the other hand, origin-destination demand is assumed to be constant over time. Sheffi (1985) points out that this assumption is realistic during a rush hour period when traffic exhibits a steady-state behavior. This class of problems has been widely studied (see Sheffi (1985) and Patriksson (2015) for references) and is also the focus of our research.

In his seminal work, Wardrop (1952) introduced two, now famous, principles that describe the characteristics of the so-called user and system equilibrium. In the user equilibrium, the journey times along all routes used from an origin to a destination are equal and less than the journey time that would be experienced by a single user on any unused route, i.e., no user can lower his transportation cost through unilateral action. In the system equilibrium the average journey time is minimum, i.e., the sum of the journey times experienced by all users is as small as possible.

To compute journey times, it is critical to be able to compute the experienced travel time on an arc, which, of course, depends on the traffic flow on the arc. To do so, a *latency function* $t_a(x)$ is used, which gives the experienced travel time on arc a given traffic flow x. The most popular latency functions are the Davidson's function $t_a(x) = t_a^{FF}(1 + \frac{\alpha x_a}{u'_a - x_a})$, where t_a^{FF} is the arc free-flow travel time and u'_a and α are tuning parameters, and the U.S. Bureau of Public Road function $t_a(x) = t_a^{FF}[1 + 0.15(\frac{x_a}{u_a})^4]$, where t_a^{FF} represents the arc free-flow travel time and u_a represents the maximum rate of vehicles that can enter an arc without experiencing substantial delays due to congestion.

Differences between user equilibrium and system optimum traffic assignments are extensively studied in literature (see, for example, Mahmassani and Peeta (1993)). The user equilibrium ensures fairness for users traveling between the same origin and destination. On the other hand, in a system optimum traffic assignment, some users may be assigned to paths that are much longer, in terms of distance or travel time, than paths assigned to other users traveling from the same origin to the same destination.

Controlling the unfairness that is inherent in a system optimum traffic assignment is critical when seeking an efficient and implementable traffic assignment. This has lead several researchers to explore system optimal routing of traffic flows with user constraints, in which only paths that ensure a certain level of fairness among users with the same origin and destination are considered. In order to generate a set of paths that is likely to result in an acceptable level of fairness, the concept of *normal length* of an arc is introduced, an a priori estimate of the traversal time of an arc. Natural choices for the normal length of an arc are its distance, its free-flow traversal time, or its traversal time in the user equilibrium. The distance and free-flow traversal time are normal length measures that do not depend on the demand in the network, while the traversal time in the user equilibrium does depend on the demand in the network. The set of paths generated for an origin-destination pair contains all paths that have a travel time that is within a certain percentage of a shortest travel time path evaluated using the normal length.

To the best of our knowledge, the first attempt at computing a constrained system optimum is due to Jahn et al. (2000). Their model seeks to minimize the total (experienced) travel time, uses the Davidson latency function, and restricts the set of paths considered by using the geographic distance as the normal length of an arc. The solution method is based on the Frank-Wolfe algorithm. In a subsequent paper, the latency function is replaced by the one provided by the U.S. Bureau of Public Road and the traversal time in the user equilibrium is used as the normal length of an arc Jahn et al. (2005). The computational results, on data for seven real road networks, show that total travel time close to the system optimum total travel time can be achieved with travel times experienced by users close to the travel times experienced by users in the user equilibrium traffic assignment. A more theoretical assessment of the efficiency and fairness of constrained system optimum traffic assignments is presented in Schulz and Stier-Moses (2006). Recently, Lujak et al. (2015) investigated a model in which the weighted geometric mean of the experienced travel times of the used paths is minimized. A generic latency function is used and the set of paths considered is restricted by using the free-flow traversal time as the normal length of an arc. The solution method is a multi-agent negotiation model. Correa et al. (2007) study the problem of minimizing the maximum experienced travel time on any used path in road networks with congestion, i.e., road networks with flow-dependent arc traversal times. They compare the resulting traffic assignment with the system optimum and user equilibrium traffic assignments in terms of the average experienced travel time on any used path (which is equivalent to the total experienced travel time) and the unfairness, i.e., the maximum ratio of the largest experienced travel time and the smallest experience travel time for an origin-destination pair. Note that in this setting there are no restrictions on the set of paths considered. Angelelli et al. (2016a) present a linear programming model in which user inconvenience is minimized while keeping the network non-congested, if possible, or at its minimum congestion level, otherwise. They assume a constant, flow-independent, latency function and measure user inconvenience as the ratio of the travel time of the user's path and the fastest travel time from the user's origin to the user's destination. The network is considered non-congested if the flow on every arc is less than or equal to the maximum rate of vehicles that can enter the arc without experiencing substantial delays. The computed user inconvenience is accurate only when the network can be kept non-congested. The set of paths considered is restricted by using the free-flow traversal time as the normal length of an arc. The advantage of their proposed model is that it can be solved with any (commercial) linear programming solver. The disadvantage is that it assumes constant, flow-independent, arc traversal times.

In this paper, a linear programming model is presented, thus retaining the desirable property that it can be solved efficiently by any (commercial) linear programming solver, that accommodates a flow-dependent latency function, thereby removing the unwanted limitation that user inconvenience is only computed accurately when the system is non-congested. The model seeks to minimize the total (experienced) travel time, using the latency function provided by the U.S. Bureau of Public Road, and restricts the set of paths considered by using the free-flow traversal time as the normal length of an arc. A piecewise linear approximation of the flow-dependent latency function ensures the linearity of the model. Thus, the model assigns paths to users so as to minimize the total travel time experienced, but uses only paths that are expected to have a limited inconvenience. An extensive computational study demonstrates that solutions with near-optimal total travel times can be found in which most users experience travel times that are better than user-equilibrium travel times and few users experience travel times that are slightly worse than user-equilibrium travel times.

The remainder of the paper is organized as follows. In Section 4.2, we introduce and discuss the relevant traffic assignment models. In Section 4.3, we introduce the solution methods developed to solve these traffic assignments models. In Section 4.4, we discuss the results of an extensive study of the constrained system optimum model with a piecewise linear approximation of the latency function. Finally, in Section 4.5, we present some concluding remarks.

4.2 Traffic assignment models

In this section, we present the mathematical models that form the basis of this study. We consider a directed network $G = (V, A)$, where V represents the set of vertices and $A \subseteq V \times V$ represents the set of arcs. Each arc $(i, j) \in A$ represents a road segment on which vehicles can travel. A latency function $t_{ij}(x)$, is defined for each arc $(i,j) \in A$ representing the arc traversal time when the rate of vehicles entering the arc is x. The free-flow traversal time of an arc $(i,j) \in A$ is denoted by t_{ij}^{FF} and represents the traversal time experienced when no other vehicles enter the arc $t_{ij}^{FF} = t_{ij}(0)$. In addition, there is a set $C \subseteq V \times V$ of commodities, i.e., represented by origin-destination (OD) pairs, where each $c \in C$ has an origin $O_c \in V$, a destination $D_c \in V$, and a demand rate d_c, representing the number of vehicles per time unit that travel from O_c to D_c. The set of all the possible paths between origin O_c and destination D_c is denoted by K_c. The indicator a_{ij}^{kc} takes value 1 if path $k \in K_c$ contains arc $(i,j) \in A$ and takes value 0 otherwise. The decision variables y_c^k represent the flow of commodity $c \in C$ routed on path $k \in K_c$. The variables x_{ij} represent the (total) flow on arc $(i,j) \in A$.

4.2.1 The user equilibrium

The user equilibrium model (UE MODEL), as formulated in Beckmann et al. (1956), is the following:

$$\min \quad \sum_{(i,j)\in A} \int_0^{x_{ij}} t_{ij}(\omega)\, d\omega$$

$$x_{ij} = \sum_{c\in C} \sum_{k\in K_c} a_{ij}^{kc} y_{ck} \qquad \forall (i,j) \in A \tag{4.1}$$

$$d_c = \sum_{k\in K_c} y_{ck} \qquad \forall c \in C \tag{4.2}$$

$$x_{ij} \geq 0 \qquad \forall (i,j) \in A \tag{4.3}$$

$$y_{ck} \geq 0 \qquad \forall c \in C \quad \forall k \in K_c. \tag{4.4}$$

Constraints (4.1) set the flow on an arc as the sum of the flow on each path passing through the arc. Constraints (4.2) ensure that the demand d_c of commodity $c \in C$ is routed on paths in K_c. Finally, constraints (4.3) - (4.4) define the domains of the decision variables. As pointed out in Sheffi (1985), the objective function has no direct interpretation but is constructed in such a way that the solution to the model is equivalent to the user equilibrium.

4.2.2 The system optimum

The system optimum model (Path-based SO model), as formulated in Beckmann et al. (1956), is the following:

$$\min \sum_{(i,j)\in A} t_{ij}(x_{ij})x_{ij}$$
$$(4.1) - (4.4).$$

The objective function minimizes the total travel time experienced by the users while the constraints are the same as in the user equilibrium model formulation. The equivalent arc-based formulation (Arc-based SO model) is the following:

$$\min \sum_{(i,j)\in A} t_{ij}(x_{ij})x_{ij}$$
$$x_{ij} = \sum_{c\in C} x_{ij}^c \qquad \forall (i,j)\in A \tag{4.5}$$
$$\sum_{j\in V} x_{ij}^c - \sum_{j\in V} x_{ji}^c = 0 \qquad \forall c\in C \quad \forall i\in V, i\neq O_c, i\neq D_c \tag{4.6}$$
$$\sum_{j\in V} x_{ij}^c - \sum_{j\in V} x_{ji}^c = d_c \qquad \forall c\in C \quad i = O_c \tag{4.7}$$
$$\sum_{j\in V} x_{ij}^c - \sum_{j\in V} x_{ji}^c = -d_c \qquad \forall c\in C \quad i = D_c \tag{4.8}$$
$$x_{ij}^c \geq 0 \quad \forall c\in C \qquad \forall (i,j)\in A. \tag{4.9}$$

Variables x_{ij}^c represent the flow of commodity c on arc (i,j), Constraints (4.5) impose that the total flow on arc (i,j) is the sum of the flows of all commodities $c \in C$ passing through it, and constraints (4.6) - (4.8) guarantee flow conservation and that demand is completely routed. Finally, constraints (4.9) define the domain of the variables x_{ij}^c.

Note that the user equilibrium is a feasible solution to the system optimum model, and, therefore, the total travel time experienced in a system optimum is less than or equal to the total travel time experienced in a user equilibrium. The difference between the total travel times of the two traffic assignments is commonly referred to as the *price of anarchy* and reflects how selfish routing (user equilibrium) affects the total travel time when compared to a system-wide coordinated traffic assignment (system optimum).

4.2.3 The constrained system optimum

Like the system optimum model, the constrained system optimum model (C-SO MODEL) assigns paths to commodities so as to minimize the total travel time experienced by users. However, the set of paths considered is restricted as follows. We define the path *inconvenience* to be the relative percentage difference between the normal length of the path and the shortest normal length path from the same origin to the same destination. For a given percentage γ, the set of paths K_c^γ for commodity $c \in C$ contains all the paths with an inconvenience not larger than γ. We refer to γ as the *maximum inconvenience.* Recall that the normal length of an arc (and thus a path) can be defined in a number of different ways. The C-SO MODEL formulation is the following:

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

$$x_{ij} = \sum_{c\in C}\sum_{k\in K_c^\gamma} a_{ij}^{kc} y_{ck} \qquad \forall (i,j) \in A \tag{4.10}$$

$$d_c = \sum_{k\in K_c^\gamma} y_{ck} \qquad \forall c \in C \tag{4.11}$$

$$x_{ij} \geq 0 \qquad \forall (i,j) \in A \tag{4.12}$$

$$y_{ck} \geq 0 \qquad \forall c \in C \quad \forall k \in K_c^\gamma. \tag{4.13}$$

The only difference with PATH-BASED SO MODEL is that the set of path K_c^γ for commodity $c \in C$ is a subset of K_c.

4.3 Solution methods

In this section, we present the solution methods for the models that form the basis of this study.

4.3.1 Computing the user equilibrium

There are several ways to compute the UE on a network. In Sheffi (1985) and Patriksson (2015) heuristic methods, such as the capacity restraint method and the incremental assignment method, as well as exact methods are proposed. We implemented the following convex combination method to obtain the UE:

Step 0: Perform an "all-or-nothing" assignment based on the free-flow traversal times

$t^0_{ij} = t^{FF}_{ij}$, $(i,j) \in A$, i.e., route the entire demand of a commodity on the fastest path with respect to t^{FF}_{ij}. The resulting flows are represented by $\{x^1_{ij}\}$. Set $n = 1$.

Step 1: Update the current traversal times based on the flows $\{x^n_{ij}\}$ and the latency function, i.e., $t^n_{ij} = t_{ij}(x^n_{ij})$, $(i,j) \in A$.

Step 2: Perform an "all-or-nothing" assignment based on the current traversal times t^n_{ij}. The resulting flows are denoted by $\{y^n_{ij}\}$. This gives a descent direction.

Step 3: Perform a line search, i.e., find the δ_n that solves

$$\begin{aligned} \min \quad & \sum_{(ij)\in A} \int_0^{x^n_{ij}+\delta_n(y^n_{ij}-x^n_{ij})} t_{ij}(\omega)\, d\omega \\ & 0 \le \delta_n \le 1 \\ & \delta_n \ge 0. \end{aligned}$$

Since we have assumed that the arc latency function is $t_{ij}(x_{ij}) = t^{FF}_{ij}[1+0.15(\frac{x_{ij}}{u_{ij}})^4]$, the resulting one variable optimization problem is

$$\begin{aligned} \min \quad & \sum_{(ij)\in A} \{t^{FF}_{ij}[x^n_{ij} + \delta_n(y^n_{ij} - x^n_{ij})] + \frac{0.15}{5u^4_{ij}}[x^n_{ij} + \delta_n(y^n_{ij} - x^n_{ij})]^5\} \\ & 0 \le \delta_n \le 1 \\ & \delta_n \ge 0, \end{aligned}$$

which can be solved easily (note that t^{FF}_{ij}, x^n_{ij} and y^n_{ij} are fixed).

Step 4: Compute flows $\{x^{n+1}_{ij}\}$, i.e., $x^{n+1}_{ij} = x^n_{ij} + \delta_n(y^n_{ij} - x^n_{ij})$, $(i,j) \in A$.

Step 5: Convergence test. Let l^n_c be the travel time for commodity c at the n-th iteration. If $\sum_{c\in C} \frac{|l^n_c - l^{n-1}_c|}{l^n_c} \le \kappa$, then STOP, otherwise set $n = n+1$ and return to Step 1.

4.3.2 Piecewise linear approximation

Since we want to take advantage of the enormous power of modern (commercial) linear programming solvers, we will replace the objective function in the system optimum models, when it is nonlinear, with a piecewise linear approximation. More specifically, we replace the terms $F_{ij}(x_{ij}) = t_{ij}(x_{ij})x_{ij}$ in the objective function of Arc-based SO model and C-SO model by a piecewise linear convex function on $[0, U_{ij}]$, where U_{ij} is a natural upper bound on the admissible flow x_{ij} through arc (i,j).

Let

$$B = \{b^0_{ij} = 0, b^1_{ij}, ..., b^{n-1}_{ij}, b^n_{ij} = U_{ij}\}$$

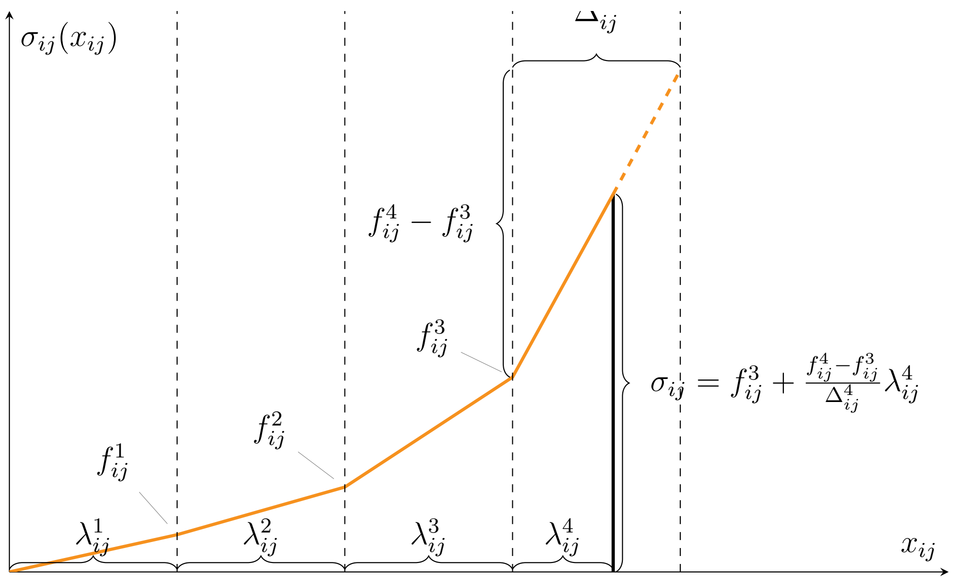


Figure 4.1. Linear constrained system optimum: $\sigma_{ij}(x_{ij})$ construction

be a set of breakpoints that partition the domain $[0, U_{ij}]$ of function F_{ij} into n intervals, and let

$$F = \{f^0_{ij} = F_{ij}(0) = 0, f^1_{ij} = F_{ij}(b^1_{ij}), ..., f^n_{ij} = F_{ij}(U_{ij})\}$$

be the set of corresponding function values. Furthermore, let $\Delta^h_{ij} = b^h_{ij} - b^{h-1}_{ij}$ for $h = 1, \dots, n$ be the length of the h-th interval. The piecewise linear approximation $\sigma_{ij}(x)$ of $F_{ij}(x)$ is given by (see Figure 4.1)

$$F_{ij}(x) \approx \sigma_{ij}(x) = \begin{cases} f^0_{ij} + \frac{f^1_{ij}-f^0_{ij}}{\Delta^1_{ij}}(x - b^0_{ij}) & x \in [b^0_{ij}, b^1_{ij}] \\ f^1_{ij} + \frac{f^2_{ij}-f^1_{ij}}{\Delta^2_{ij}}(x - b^1_{ij}) & x \in (b^1_{ij}, b^2_{ij}] \\ \ldots\ldots & \ldots. \\ f^{n-1}_{ij} + \frac{f^n_{ij}-f^{n-1}_{ij}}{\Delta^n_{ij}}(x - b^{n-1}_{ij}) & x \in (b^{n-1}_{ij}, b^n_{ij}]. \end{cases}$$

When the latency functions are convex, as is the case with the Davidson and the US Bureau of Public Road latency functions, then $F_{ij}(x)$ and its approximation $\sigma_{ij}(x)$ are convex as well. The convexity of $\sigma_{ij}(x)$ functions allows them to be easily expressed in a linear programming model.

This results in the following LIN-SO MODEL:

$$
\begin{aligned}
\min \quad & \sum_{(ij)\in A} \sigma_{ij} \\
& x_{ij} = \sum_{h=1}^{n} \lambda_{ij}^{h} \qquad \forall (i,j) \in A && (4.14) \\
& \sigma_{ij} = \sum_{h=1}^{n} \frac{f_{ij}^{h} - f_{ij}^{h-1}}{\Delta_{ij}^{h}} \lambda_{ij}^{h} \qquad \forall (i,j) \in A && (4.15) \\
& 0 \leq \lambda_{ij}^{h} \leq \Delta_{ij}^{h} \qquad \forall (i,j) \in A \quad \forall h = 1, ..., n && (4.16) \\
& x_{ij} = \sum_{c\in C} x_{ij}^{c} \qquad \forall (i,j) \in A && (4.17) \\
& \sum_{j\in V} x_{ij}^{c} - \sum_{j\in V} x_{ji}^{c} = 0 \qquad \forall c \in C \quad \forall i \in V, i \neq O_c, i \neq D_c && (4.18) \\
& \sum_{j\in V} x_{ij}^{c} - \sum_{j\in V} x_{ji}^{c} = d_c \qquad \forall c \in C \quad i = O_c && (4.19) \\
& \sum_{j\in V} x_{ij}^{c} - \sum_{j\in V} x_{ji}^{c} = -d_c \qquad \forall c \in C \quad i = D_c && (4.20) \\
& x_{ij}^{c} \geq 0 \quad \forall c \in C \qquad \forall (i,j) \in A \\
& x_{ij} \geq 0 \quad \forall (i,j) \in A.
\end{aligned}
$$

Variable x_{ij}^c represents the flow of commodity c on arc (i,j) and variable x_{ij} represents to total flow on arc (i,j). They are linked through constraints (4.14). Auxiliary variables λ_{ij}^h represent the fraction of flow x_{ij} assigned to interval $[b_{ij}^{h-1}, b_{ij}^{h}]$. Constraints (4.14) ensure that the entire flow x_{ij} is assigned. The convexity of $F_{ij}(x)$, the objective function, and constraints (4.15) guarantee that in an optimal solution the flow x_{ij} is assigned to intervals in order of increasing indices, i.e., $\lambda_{ij}^h > 0$ if and only if $\lambda_{ij}^{h-1} = \Delta_{ij}^{h-1}$ for $h = 2, \ldots, n$. Constraints (4.18) - (4.20) guarantee flow conservation and that the entire demand is routed.

The piecewise linear approximation can be also applied to the C-SO MODEL and results in the following Lin-C-SO model:

$$\begin{aligned}
\min \quad & \sum_{(ij)\in A} \sigma_{ij} && && \\
& x_{ij} = \sum_{h=1}^{n} \lambda_{ij}^{h} && \forall (i,j)\in A && (4.21)\\
& \sigma_{ij} = \sum_{h=1}^{n} \frac{f_{ij}^{h} - f_{ij}^{h-1}}{\Delta_{ij}^{h}} \lambda_{ij}^{h} && \forall (i,j)\in A && (4.22)\\
& 0 \le \lambda_{ij}^{h} \le \Delta_{ij}^{h} && \forall (i,j)\in A \quad \forall h = 1,...,n && (4.23)\\
& x_{ij} = \sum_{c\in C} \sum_{k\in K_c^{\gamma}} a_{ij}^{kc} y_{ck} && \forall (i,j)\in A && (4.24)\\
& d_c = \sum_{k\in K_c^{\gamma}} y_{ck} && \forall c\in C && (4.25)\\
& x_{ij} \ge 0 && \forall (i,j)\in A && \\
& y_{ck} \ge 0 && \forall c\in C \quad \forall k\in K_c^{\gamma}. &&
\end{aligned}$$

As before, variables x_{ij} and y_{ck} represent the flow on arc (i,j) and the flow on the k-th path of commodity $c \in C$, respectively, and constraints (4.24) and (4.24) ensure that the flow on arc (i,j) is set correctly and that the entire demand for a commodity is routed, respectively. Constraints (4.21)-(4.23) play the same role as the corresponding constraints in Lin-SO model. For convenience, the notation used in Lin-C-SO model is summarized in Table 4.1.

4.4 Computational results

A large and diverse set of instances has been used in a computational study to assess the performance of the Lin-C-SO model. The instances were generated taking into account different demand patterns, point attractiveness distributions and other parameters as explained thoroughly in Chapter 6. The instances are available at `http://or-brescia.unibs.it/instanc` For each instance, we generate a traffic assignment using a restricted set of paths for maximum inconvenience values, γ, ranging from 0% to 35% in increments of 1% (i.e., 36 traffic assignments). The linear programs are solved using CPLEX 12.6.0. The experiments were conducted on a Windows 64-bit computer with Intel Xeon processor E5-1650, 3.50 GHz, and 16 GB Ram. For all experiments, we have used the latency function proposed by the U.S. Bureau of Public Roads, i.e., $t_{ij} = t_{ij}^{FF}(1 + 0.15(\frac{x_{ij}}{u_{ij}})^4)$. To define a set of restricted paths, we use the free-flow travel time of an arc as its normal length.

Constrained system optimum notation

Sets

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

Parameters

u_{ij}	parameter representing the maximum rate of vehicles that can enter the arc $(i,j) \in A$ without experiencing substantial delays due to congestion
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
d_c	flow rate for OD pair $c \in C$
b_{ij}^h	h-th breakpoint related to the arc (i,j)
f_{ij}^h	value of the function $F(x_{ij})$ in breakpoint b_{ij}^h
n	number of intervals
Δ_{ij}^h	interval size

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 $(i,j) \in A$: $x_{ij} = \sum\limits_{c \in C} \sum\limits_{k \in K_c^\gamma} a_{ij}^{kc} y_{ck}$

Auxiliary decision variables

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

Table 4.1. Notation used in Lin-C-SO model

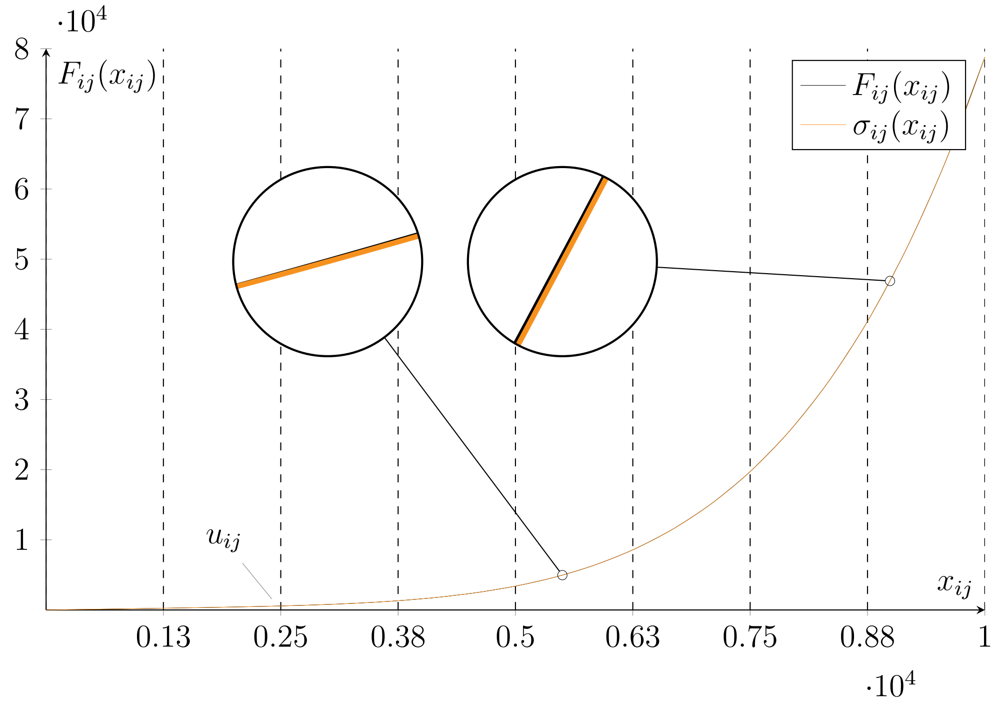


Figure 4.2. Linear constrained system optimum: $F_{ij}(x_{ij})$ and $\sigma_{ij}(x_{ij})$ curves

4.4.1 Piecewise linear approximation accuracy

In the computational experiments with LIN-SO MODEL and LIN-S-CO MODEL, we use a piecewise linear approximation of the objective function

$$F_{ij}(x_{ij}) = \sum_{(i,j)\in A} t_{ij}^{FF}(1 + 0.15(\frac{x_{ij}}{u_{ij}})^4)x_{ij}.$$

The piecewise linear approximation is obtained by dividing the domain $[0, U_{ij}]$ of the flow variable x_{ij} associated with arc (i, j) into $n = 1000$ equal length intervals. In Figure 4.2, we show $F_{ij}(x_{ij})$ and its piecewise linear approximation $\sigma_{ij}(x_{ij})$ for the latency function with parameters $t_{ij}^{FF} = 0.2$ hours and $u_{ij} = 2500$ vehicles/hour. We see that the two functions almost overlap - to enhance the visual effect, we have magnified parts of the figure using a 20x scaling factor. In fact, the maximum relative difference, i.e., $\frac{\sigma_{ij}(x_{ij})-F_{ij}(x_{ij})}{F_{ij}(x_{ij})}$, is always less than 0.5%. High-quality approximations can also be obtained with a smaller number of intervals, e.g., $n \geq 50$. However, as the solution times with the chosen number of intervals was acceptable, we did not explore the choice of the number of intervals in much detail.

4.4.2 Statistics

We collect and compute the statistics shown in Table 4.2. Note that the total travel time of a traffic assignment is always evaluated using the non-linear latency function $t_{ij}^{FF}(1 + 0.15(\frac{x_{ij}}{u_{ij}})^4)$.

4.4.3 Results for the Lin-C-SO model

We analyze the traffic assignments produced by the Lin-S-CO model for maximum inconvenience values ranging from 0% to 35% in increments of 1% and averaged over the 80 instances. Recall that inconvenience and maximum inconvenience are defined in terms of the normal length of arcs and used only to define a restricted set of paths. The reported free-flow inconvenience and UE inconvenience are defined using the actual experienced travel times.

Figure 4.3 shows the (average) total travel time, $\Theta_{\text{Lin-C-SO}}$, as a function of the maximum inconvenience, γ. As a reference, we have included, in the form of two horizontal lines, the (average) total travel time of the user equilibrium assignment, Θ_{UE}, and the (average) total travel time of the system optimum traffic assignment, $\Theta_{\text{Lin-SO}}$. We see that when the maximum inconvenience is greater than 10%, then $\Theta_{\text{Lin-C-SO}}$ is almost equal to $\Theta_{\text{Lin-SO}}$. Furthermore, we see that only when the maximum inconvenience, γ, is very small. i.e., 0, 1, or 2%, we have $\Theta_{UE} \leq \Theta_{\text{Lin-C-SO}}$. Already with a maximum inconvenience of 3%, there are enough paths in the set of restricted paths used in the Lin-S-CO model to achieve a smaller (average) total travel time than in the UE. This is a manifestation of the "price of anarchy", i.e., individual decision making (UE equilibrium model) results in worse performance than coordinated decision making (SO and C-SO models).

Figure 4.4(a) shows the (average) free-flow inconvenience, I_{FF}, as a function of the maximum inconvenience, γ. The dashed line helps to identify those values of the maximum inconvenience for which the experienced free-flow inconvenience is larger than the maximum inconvenience. We see that that happens for maximum inconvenience values less than or equal to 4%. In addition, we see that when the maximum inconvenience is greater than 10%, I_{FF} is less than or equal to 3%. As the average free-flow inconvenience conveys only part of what is happening, in Figure 4.4(b), we show the maximum free-flow inconvenience over all instances as a function of the maximum inconvenience. As expected, it takes longer for the maximum experienced free-flow inconvenience to drop below the maximum inconvenience, i.e., only for maximum inconvenience values greater than or equal to 10%. We also see that maximum free-flow inconvenience stabilizes around 7% for maximum inconvenience values greater than or equal to 14%.

The free-flow inconvenience compares the experienced travel time to the shortest free-flow travel time. A more relevant and insightful comparison, however, may be the comparison with the experienced travel time in the user equilibrium traffic assignment, i.e., I_{UE}. Figures 4.5(a) and 4.5(b) show the average value of I_{UE} and the maximum value of I_{UE},

Travel time	
$\Theta_{\text{Lin-C-SO}}$	Experienced total travel time for the traffic assignment produced by the Lin-C-SO model.
$\Theta_{\text{Lin-SO}}$	Experienced total travel time for the traffic assignment produced by the Lin-SO model.
Θ_{UE}	Experienced total travel time for the traffic assignment produced by the UE algorithm.
User experience	
I_{FF}	Free-flow inconvenience, i.e., the average of the relative difference between the experienced travel time on the path and the travel time on the fastest path under free-flow conditions for the associated OD pair over all used paths weighted by the path flow.
I_{UE}	UE inconvenience, i.e., the average of the relative difference between the experienced travel time on the path and the travel time experienced in the user equilibrium for the associated OD pair over all used paths weighted by the path flow.
Arc utilization	
x_{ij}/u_{ij} classes	The fraction of arcs falling in each of the following six classes: • A ($0\% < x_{ij}/u_{ij} \leq 20\%$) • B ($20\% < x_{ij}/u_{ij} \leq 40\%$) • C ($40\% < x_{ij}/u_{ij} \leq 60\%$) • D ($60\% < x_{ij}/u_{ij} \leq 80\%$) • E ($80\% < x_{ij}/u_{ij} \leq 100\%$) • F ($x_{ij}/u_{ij} > 100\%$)
OD paths	
Selected paths	Selected paths in the optimal solution of the Lin-C-SO model: • average number of selected paths per OD pair, • maximum number of selected paths per OD pair.

Table 4.2. Linear constrained system optimum statistics

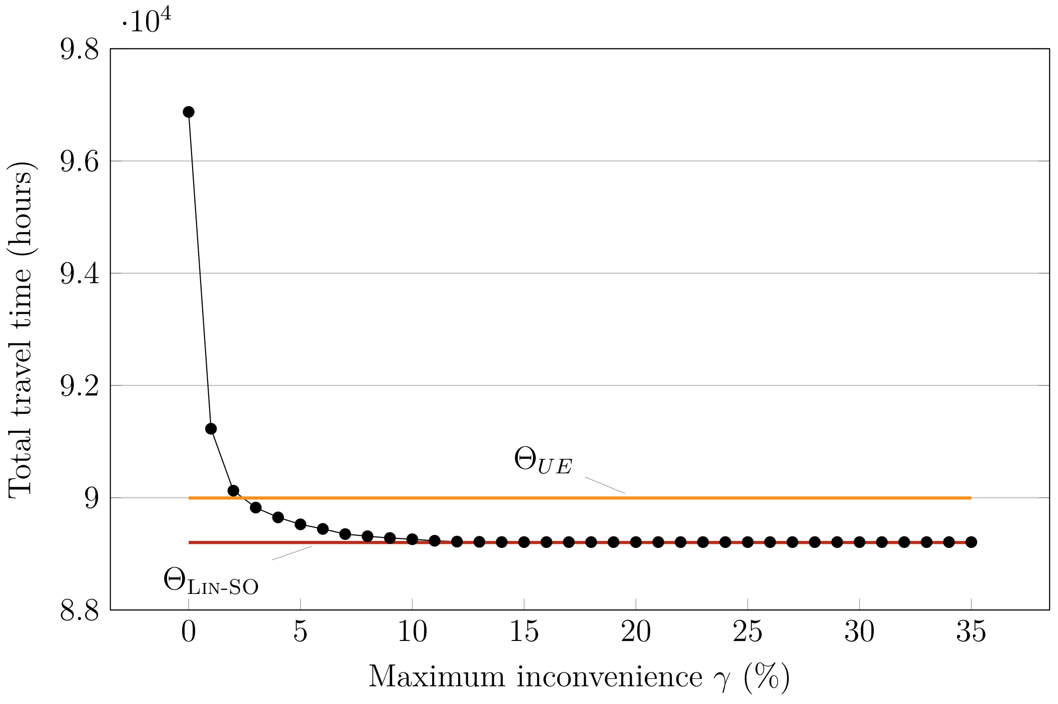
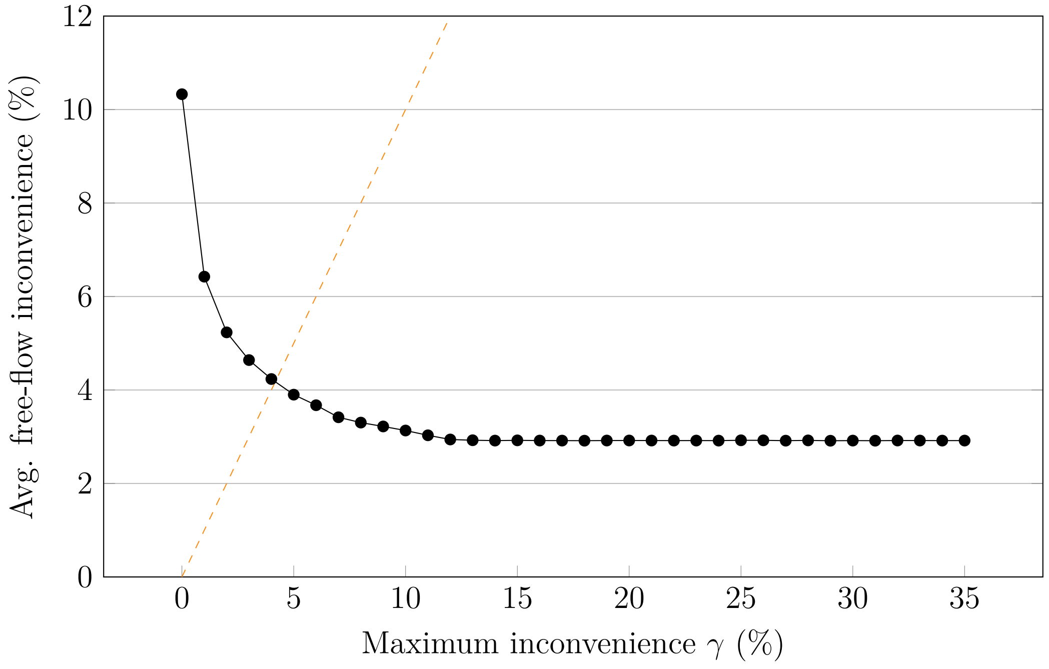


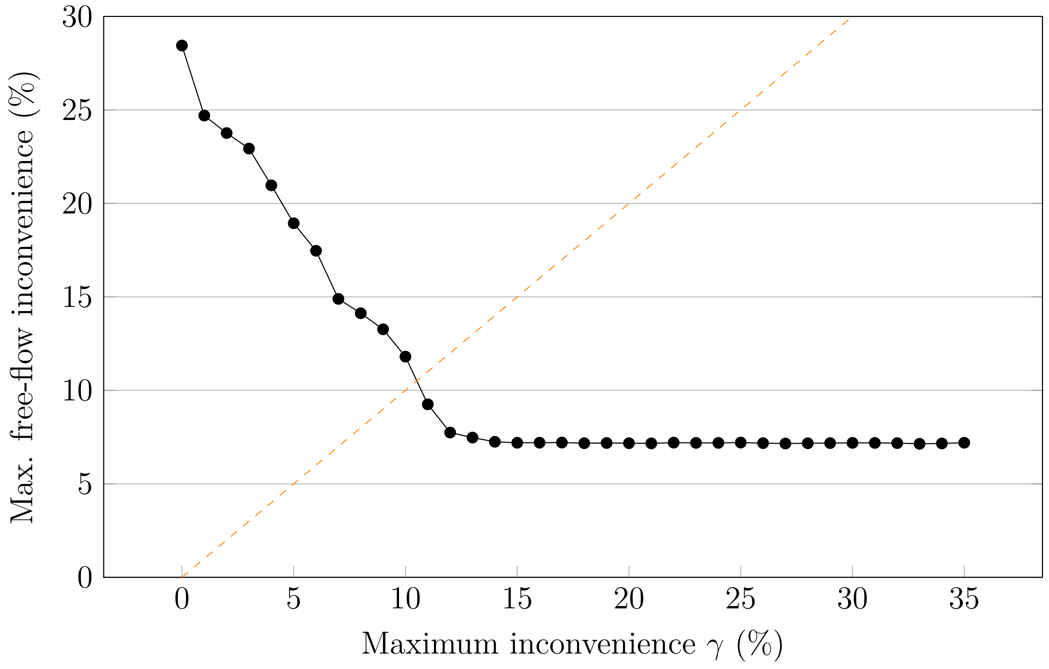
Figure 4.3. Linear constrained system optimum: total travel time, $\Theta_{\text{LIN-C-SO}}$ as a function of the maximum inconvenience, γ

respectively, as a function of the maximum inconvenience. We see that for a maximum inconvenience of 5%, the average I_{UE} becomes negative. This means that, on average, users are experiencing travel times that are shorter than the travel time experienced in the user equilibrium traffic assignment. In fact, for a maximum inconvenience of 5% the maximum I_{UE} is also low, about 7%. For higher maximum inconvenience values, we see even better performance, e.g., at 12% the average I_{UE} is -1% and the maximum I_{UE} is 0. That is, at 12%, we are able to reduce the average experienced travel time without increasing the maximum experience travel time (compared to the travel times experienced in the UE traffic assignment).

To provide more detail about the traffic assignment produced by the LIN C-SO MODEL, we present, in Figure 4.6, the average and maximum number of used paths for an OD pair, and, in Figure 4.7, information on the ratio of arc flow and arc capacity across the arcs in the network. Figure 4.6 shows that, on average, the number of used paths for an OD pair is close to one, which means that, on average, the demand of an OD pair is routed along a single path. However, it is also clear that in most settings the demand of a few OD pairs is routed along multiple paths, up to seven paths for $\gamma = 19\%$. Figure 4.7 shows the distribution of the ratio of arc flow and arc capacity across the arcs in the network. The arcs are divided into utilization classes A through F. Class A contains the least used arcs (with a ratio between 0 and 0.2). Intermediate classes contain arcs with moderate flows, in which any user experienced delays are still acceptable. Class F contains the most heavily used arcs with flows exceeding the practical capacity, which implies that on these arcs users will experience significant delays. As expected, we see that the fraction

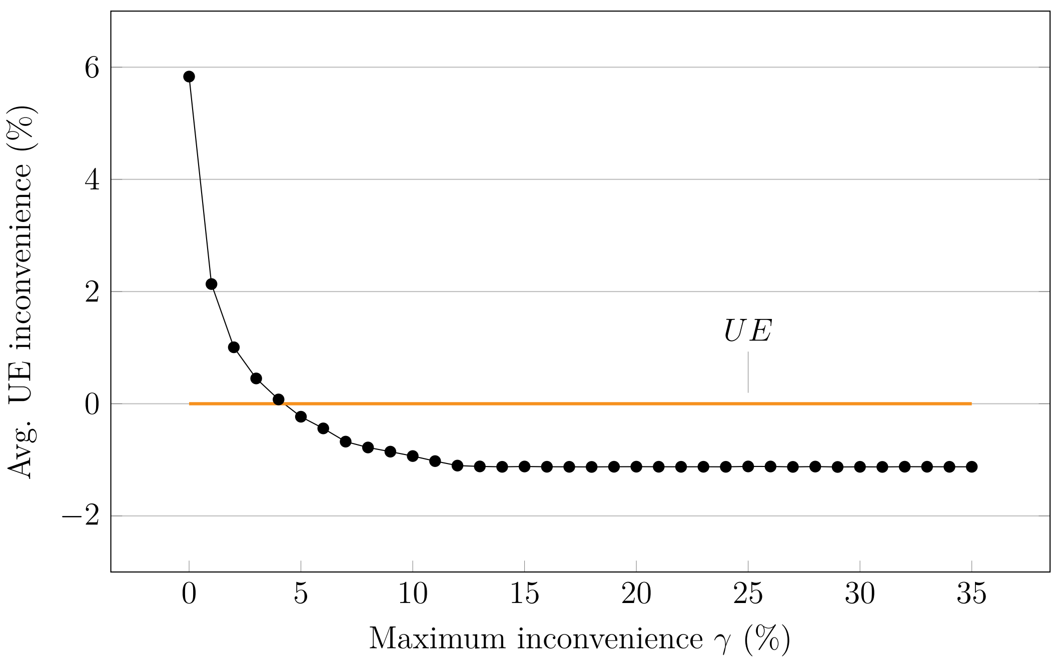


(a) Average I_{FF}

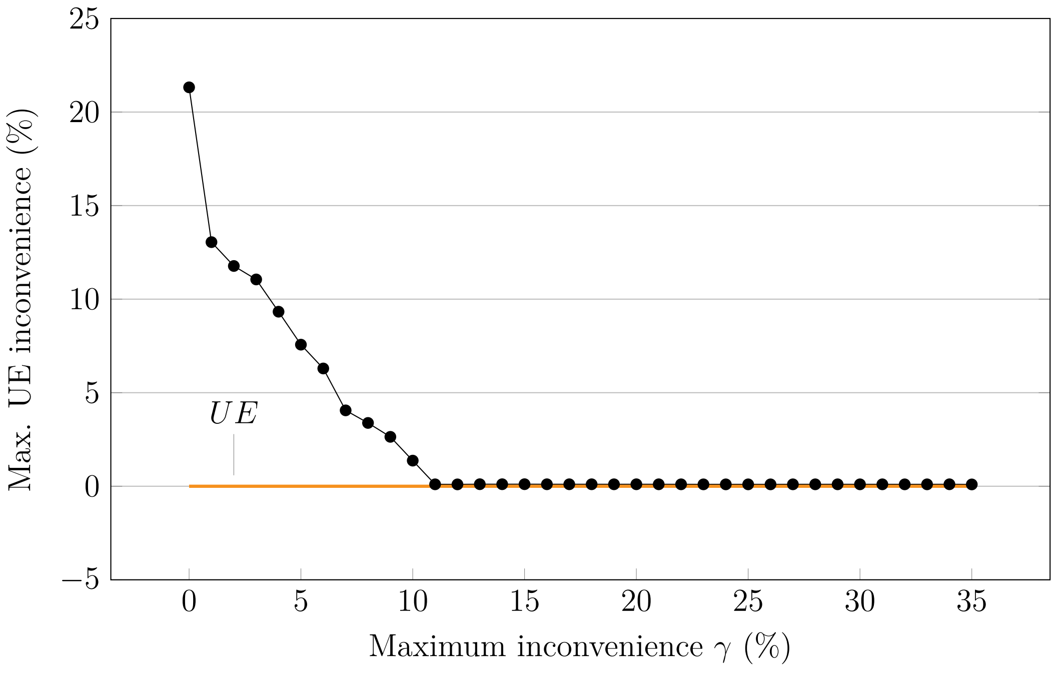


(b) Maximum I_{FF}

Figure 4.4. Linear constrained system optimum: free-flow inconvenience, I_{FF}



(a) Average I_{UE}



(b) Maximum I_{UE}

Figure 4.5. Linear constrained system optimum: UE inconvenience, I_{UE}

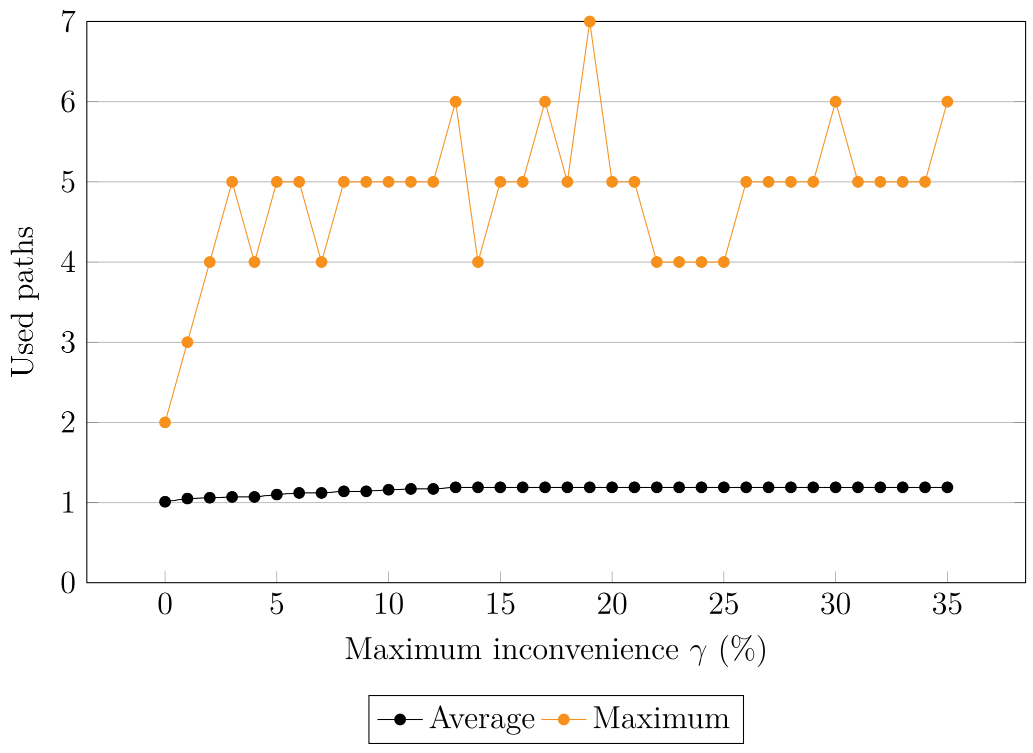


Figure 4.6. Linear constrained system optimum: average and maximum number of each OD pair used paths

of heavily used arcs decreases when the maximum inconvenience increases so as to avoid significant delays and minimize total experienced travel time.

4.5 Final remarks and future research

The LIN-C-SO MODEL has an important advantage over previously proposed constrained system optimum models: it is a linear programming model and, hence, it is more likely that even very large instances can be solved efficiently. As such, it has larger potential to be of use in practical settings. However, the upfront enumeration of the sets of restricted paths from an origin to a destination may prove to be computationally prohibitive for very large instances, when the number of paths could easily reach several millions. A natural extension, therefore, will be developing a column generation approach for the LIN-C-SO MODEL, possibly using heuristics to dynamically identify beneficial paths. Furthermore, as pointed out in the introduction, the steady-state assumption, which allows us to work with static traffic assignments, may be reasonable during rush hour periods, but will not be appropriate for a period in which the demand varies over time. To accommodate varying demand over time, a time-dependent variant of the LIN-C-SO MODEL can be investigated.

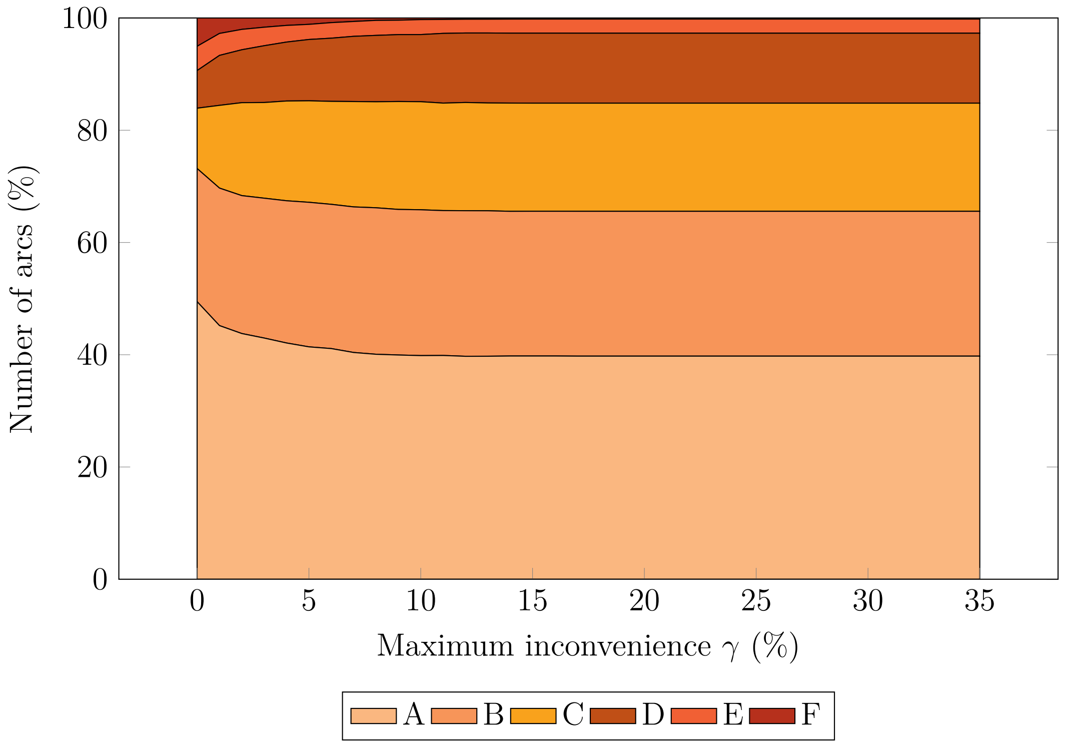


Figure 4.7. Linear constrained system optimum: arc utilization distribution

5. Heuristic path generation for the linear constrained system optimum model

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.

5.1 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 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 5.2 the linear constrained system optimum model is recalled. In Section 5.3 the heuristic iterative path generation algorithm for the linear constrained system optimum (HI-GEN) is described. In Section 5.4 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 5.5.

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 inconvenience γ. The relative difference between the normal length of path k and the shortest path for OD pair c is called *path 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 $\sigma_{ij}(x)$:

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

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

The Lin-C-SO(n) model

$$
\begin{aligned}
\min \quad & \sum_{(ij)\in A} \sigma_{ij} && && \\
& x_{ij} = \sum_{h=1}^{n} \lambda_{ij}^{h} && \forall (i,j) \in A && (5.1)\\
& \sigma_{ij} = \sum_{h=1}^{n} \frac{f_{ij}^{h} - f_{ij}^{h-1}}{\Delta_{ij}^{h}} \lambda_{ij}^{h} && \forall (i,j) \in A && (5.2)\\
& 0 \le \lambda_{ij}^{h} \le \Delta_{ij}^{h} && \forall (i,j) \in A \quad \forall h = 1, ..., n && (5.3)\\
& x_{ij} = \sum_{c\in C} \sum_{k\in K_c^{\gamma}} a_{ij}^{kc} y_{ck} && \forall (i,j) \in A && (5.4)\\
& d_c = \sum_{k\in K_c^{\gamma}} y_{ck} && \forall c \in C && (5.5)\\
& x_{ij} \ge 0 && \forall (i,j) \in A && (5.6)\\
& y_{ck} \ge 0 && \forall c \in C \quad \forall k \in K_c^{\gamma}. && (5.7)
\end{aligned}
$$

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 (5.3). The arc flow on arc (i,j) is obtained by summing over all λ_{ij}^{h} variables in constraints (5.1). Constraints (5.2) 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 (5.4) set the flow rate x_{ij} equal to the sum of flows on paths containing the arc (i,j). Constraints (5.5) guarantee that the demands d_c are completely routed on their feasible paths. Constraints (5.6)-(5.7) guarantee that variables x_{ij} and y_{ck} are non-negative. Table 5.1 summarizes the notation.

5.3 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

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 inconvenience γ

Parameters

γ	maximum 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 $[0, U_{ij}]$, $(h = 0, \dots, n))$
f_{ij}^h	value of the function $F(x_{ij})$ at breakpoint b_{ij}^h, $(h = 0, \dots, n)$
$\Delta_{ij}^h = b_{ij}^h - b_{ij}^{h-1}$	$h - th$ interval size, $(h = 1, \dots, n)$

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 $(i,j) \in A$: $x_{ij} = \sum\limits_{c \in C} \sum\limits_{k \in K_c^\gamma} a_{ij}^{kc} y_{ck}$

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 5.1. Notation related to the Lin-C-SO(n) model

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 7.

Algorithm 7: HI-GEN algorithm

```
input  : $G$ : graph of the road network,
         $C$ : set of OD pairs,
         $\gamma$ : maximum inconvenience,
         $n, l$ : approximation levels of the latency function
output: $x$ : heuristic solution of Lin-C-SO$(n)$
global : $G, C, \gamma, P^{Curr}$
– $P^{Curr} := \emptyset$;
– $L :=$ 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.

5.3.1 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 8.

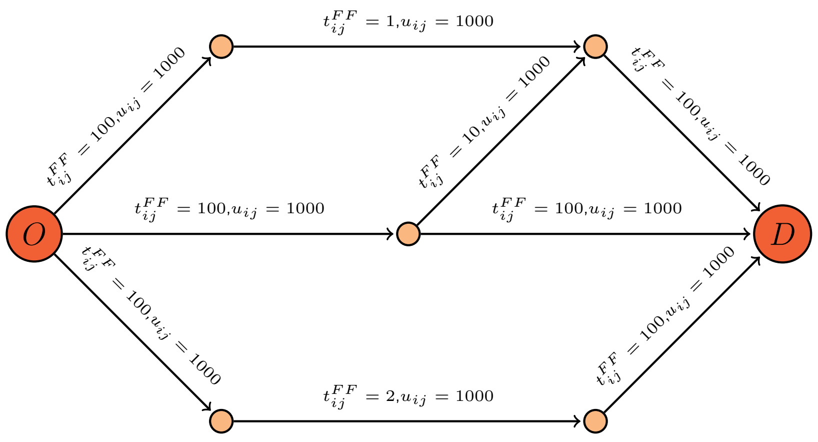
```
Algorithm 8: findCheaperPaths
input  : $x$ : traffic flow
output: $L$ improving path set
global : $G, C, \gamma, P^{Curr}$
– $L := \emptyset$;
for $c \in C$ do
    – $p$ := shortest path in $G$ from $O_c$ to $D_c$ with respect to arc lengths $t_{ij} = t_{ij}(x_{ij})$;
    – $nlp$ := length of $p$ 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.

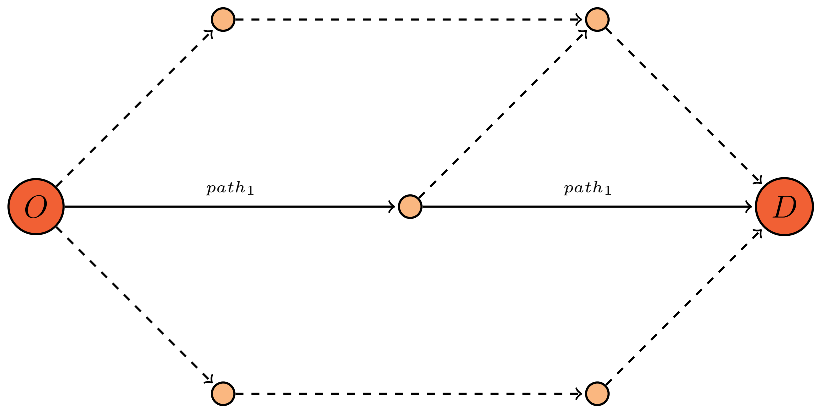
5.3.2 An example

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 5.1, 5.2 and 5.3 we provide an example instance with $l = 100$ and $\gamma = 30\%$. In Figure 5.1(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 5.1(b)). Figure 5.1(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 5.2(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 5.2(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 5.2(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 5.3(a). A new run of $findCheaperPaths$ function returns again $path_1$ (Figure 5.3(b)) and, hence, fails (no new path found as in Figure 5.3(b)), iterations stop and the R-LIN-C-SO(n)

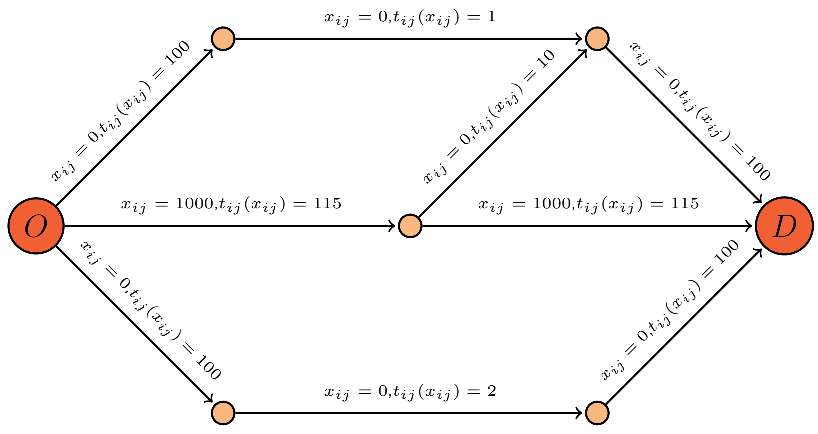
model is run in order to find the HI-GEN solution as in Figure 5.3(c) There are three paths in P^{Curr} and 350 units are assigned to $path_1$, 350 to $path_2$ and 300 to $path_3$. The total travel time is 201339.09 seconds.


$t_{ij}^{FF} = 1, u_{ij} = 1000$
$t_{ij}^{FF} = 100, u_{ij} = 1000$
$t_{ij}^{FF} = 10, u_{ij} = 1000$
$t_{ij}^{FF} = 100, u_{ij} = 1000$
$t_{ij}^{FF} = 100, u_{ij} = 1000$
$t_{ij}^{FF} = 100, u_{ij} = 1000$
O
D
$t_{ij}^{FF} = 100, u_{ij} = 1000$
$t_{ij}^{FF} = 100, u_{ij} = 1000$
$t_{ij}^{FF} = 2, u_{ij} = 1000$

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



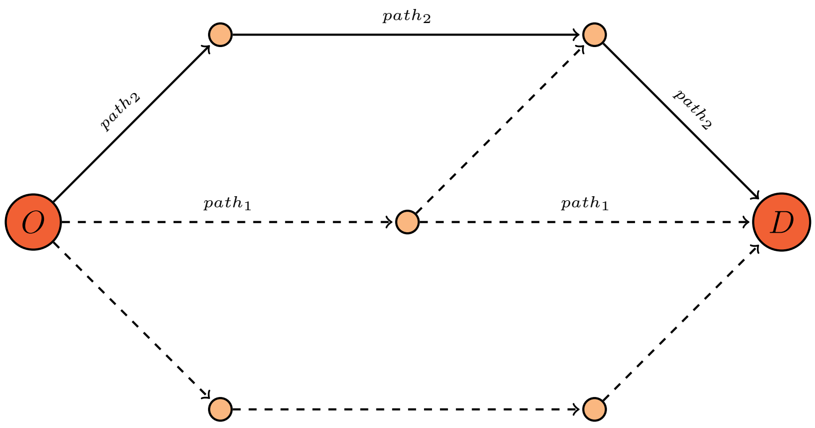
(b) The shortest path for OD pair O


$x_{ij} = 0, t_{ij}(x_{ij}) = 1$
$x_{ij} = 0, t_{ij}(x_{ij}) = 100$
$x_{ij} = 0, t_{ij}(x_{ij}) = 10$
$x_{ij} = 0, t_{ij}(x_{ij}) = 100$
$x_{ij} = 1000, t_{ij}(x_{ij}) = 115$
$x_{ij} = 1000, t_{ij}(x_{ij}) = 115$
O
D
$x_{ij} = 0, t_{ij}(x_{ij}) = 100$
$x_{ij} = 0, t_{ij}(x_{ij}) = 100$
$x_{ij} = 0, t_{ij}(x_{ij}) = 2$

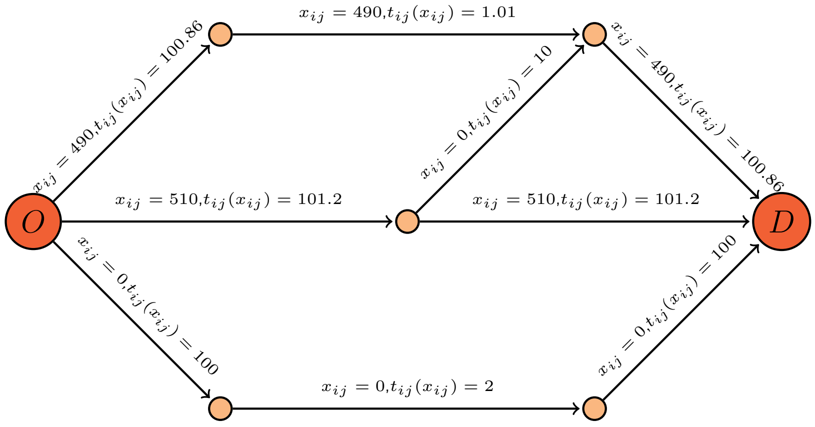
(c) Current solution and modified arc travel times

Figure 5.1. HI-GEN: an example with $l = 100$ (continues in Figure 5.2)

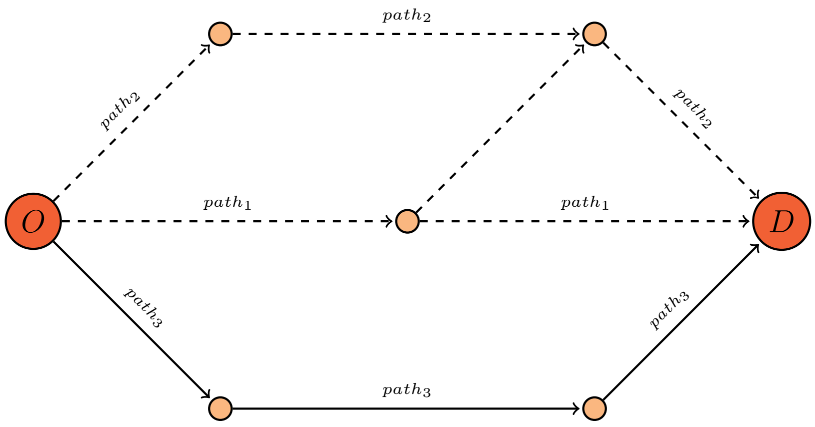
Finally, in Figure 5.4 the LIN-C-SO(n) solution is provided. The optimal solution assigns 374.5 units to $path_1$, 339.5 units to $path_2$ and 286 units to $path_3$. Total travel time,



(a) A cheaper path for O

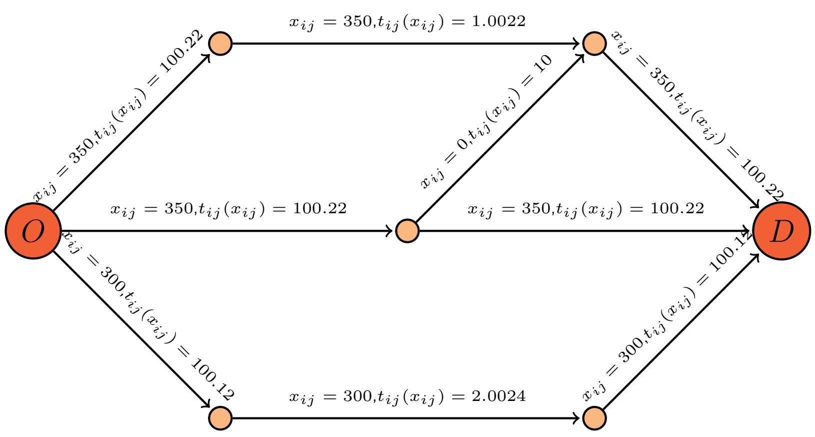

O
D
$x_{ij} = 490, t_{ij}(x_{ij}) = 1.01$
$x_{ij} = 490, t_{ij}(x_{ij}) = 100.86$
$x_{ij} = 490, t_{ij}(x_{ij}) = 100.86$
$x_{ij} = 0, t_{ij}(x_{ij}) = 10$
$x_{ij} = 510, t_{ij}(x_{ij}) = 101.2$
$x_{ij} = 510, t_{ij}(x_{ij}) = 101.2$
$x_{ij} = 0, t_{ij}(x_{ij}) = 100$
$x_{ij} = 0, t_{ij}(x_{ij}) = 100$
$x_{ij} = 0, t_{ij}(x_{ij}) = 2$

(b) Current solution and modified arc travel times

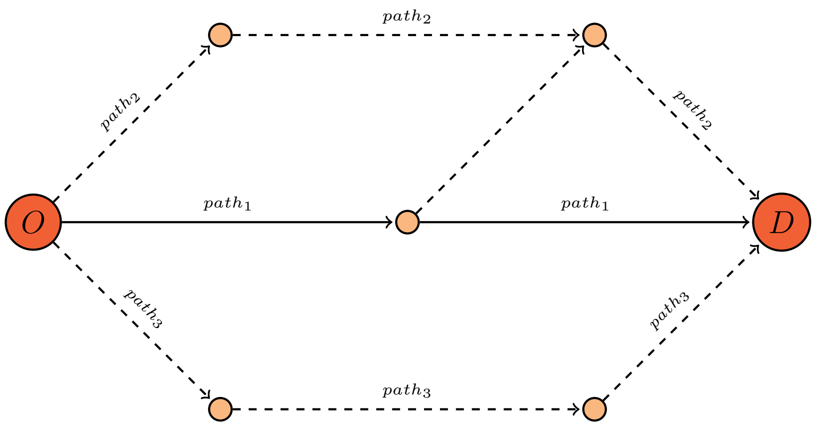


(c) Another cheaper path for O

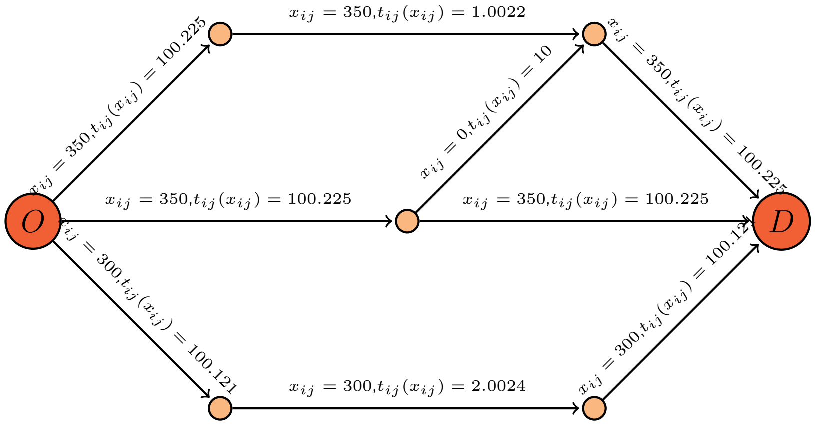
Figure 5.2. HI-GEN: an example with $l = 100$ (continues in Figure 5.3)


$x_{ij} = 350, t_{ij}(x_{ij}) = 1.0022$
$x_{ij} = 350, t_{ij}(x_{ij}) = 100.22$
$x_{ij} = 0, t_{ij}(x_{ij}) = 10$
$x_{ij} = 350, t_{ij}(x_{ij}) = 100.22$
O
$x_{ij} = 350, t_{ij}(x_{ij}) = 100.22$
$x_{ij} = 350, t_{ij}(x_{ij}) = 100.22$
D
$x_{ij} = 300, t_{ij}(x_{ij}) = 100.12$
$x_{ij} = 300, t_{ij}(x_{ij}) = 100.1$
$x_{ij} = 300, t_{ij}(x_{ij}) = 2.0024$

(a) Current solution and modified arc travel times



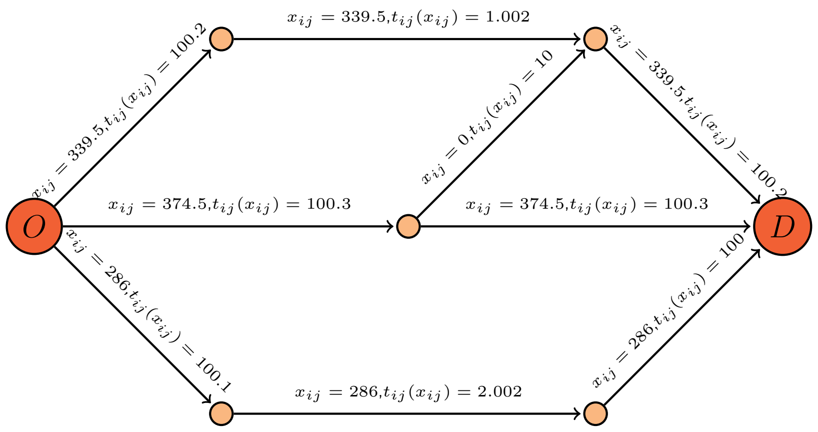
(b) No new cheaper paths can be found


$x_{ij} = 350, t_{ij}(x_{ij}) = 1.0022$
$x_{ij} = 350, t_{ij}(x_{ij}) = 100.225$
$x_{ij} = 0, t_{ij}(x_{ij}) = 10$
$x_{ij} = 350, t_{ij}(x_{ij}) = 100.225$
O
$x_{ij} = 350, t_{ij}(x_{ij}) = 100.225$
$x_{ij} = 350, t_{ij}(x_{ij}) = 100.225$
D
$x_{ij} = 300, t_{ij}(x_{ij}) = 100.121$
$x_{ij} = 300, t_{ij}(x_{ij}) = 100.12$
$x_{ij} = 300, t_{ij}(x_{ij}) = 2.0024$

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

Figure 5.3. HI-GEN: an example with $l = 100$

evaluated by means of the non linear latency function with the optimal solution as input, is 201331.49.


$x_{ij} = 339.5, t_{ij}(x_{ij}) = 1.002$
$x_{ij} = 339.5, t_{ij}(x_{ij}) = 100.2$
$x_{ij} = 0, t_{ij}(x_{ij}) = 10$
$x_{ij} = 339.5, t_{ij}(x_{ij}) = 100.2$
O
$x_{ij} = 374.5, t_{ij}(x_{ij}) = 100.3$
$x_{ij} = 374.5, t_{ij}(x_{ij}) = 100.3$
D
$x_{ij} = 286, t_{ij}(x_{ij}) = 100.1$
$x_{ij} = 286, t_{ij}(x_{ij}) = 100$
$x_{ij} = 286, t_{ij}(x_{ij}) = 2.002$

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

Figure 5.4. HI-GEN: solution of the LIN-C-SO(n) CE

5.4 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 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 5.4.1. Results for the 150 nodes network graphs are presented and discussed in Section 5.4.2. Results for the increasing size of network graphs are presented and discussed in Section 5.4.3.

5.4.1 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.

5.4.2 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 5.2, 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.

In Tables 5.3 and 5.4 statistics on the optimality gap Θ_γ are shown. In Table 5.3, 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 5.4 the maximum Θ_γ over all instances and for different γ and l values is shown.

	HI-GEN algorithm					CE algorithm
	l					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 5.2. HI-GEN: computational time (sec)

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 5.3. HI-GEN: 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 5.4. HI-GEN: maximum optimality gap Θ_γ (%)

In Table 5.5, 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 5.5. HI-GEN: 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 5.6. 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 5.6. HI-GEN: number of generated paths by CE and HI-GEN algorithms

In Figure 5.7 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.

5.4.3 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 5.4.2, differences among different accuracy levels in generating paths are negligible. In Tables 5.8 and 5.9 the collected statistics are presented. At each step the number of nodes in the network is 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

		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 5.7. HI-GEN: arc utilization distribution (%)

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.

5.5 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

		γ				
	Stats	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 5.8. HI-GEN: 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			

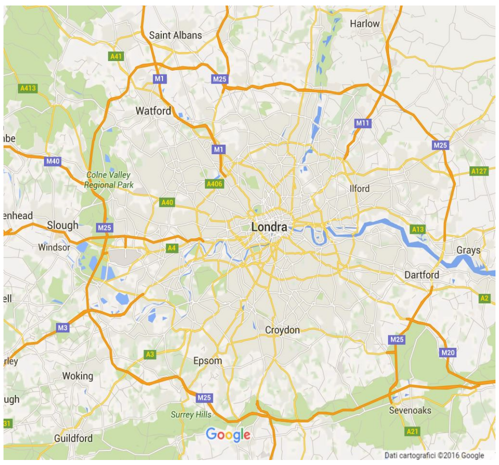
Table 5.9. HI-GEN: results for different number of nodes in the network

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.

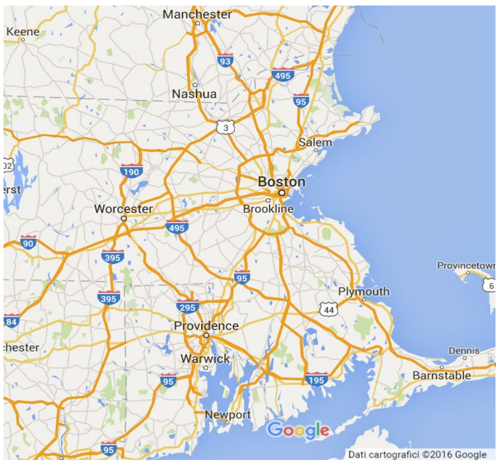
6. On the instance generation process

In this chapter the main guidelines used in generating road network instances for the proactive route guidance and for the linear constrained system optimum problem are presented. An instance for both problems is represented by a directed graph $G \equiv (V, A)$ where V is the set of vertices representing all junctions of the road network and A is the set of arcs representing the road segments, and by a set C of Origin-Destination pairs. The described problems and respective approaches require the following arc attributes: free-flow arc travel time, arc length and arc capacity. Free-flow arc travel time represents the travel time needed to traverse the arc when the arc is empty, arc length can be either the euclidean or the geographical arc length and the arc capacity represents the maximum rate of vehicle that can enter the arc without experiencing substantial delay due to congestion effects. Presented approaches require also a demand value associated to each OD pair $c \in C$, d_c, representing the rate of vehicles that intend to travel from the origin O_c to the destination D_c. In this chapter a graph generation algorithm able to create vertices, arcs and OD pairs in such a way the resulting instances will reflect the main features of some road network topologies/structure is presented. The generation algorithm models highways, bypasses and orbital roads with arc travel times and capacities depending on the road importance. Most circular cities are formed by an internal ring surrounding the downtown area. This internal ring is usually considered as a low-speed way allowing the access to the central area. Usually there exists an external ring where the heavy traffic coming from suburbs and surrounding towns converges. When the city is big enough, there usually exists a number of intermediate rings between the outer and the inner one. In addition, roads connecting rings and external highways or heavy traffic arterial roads are always present. We call circular a network topology of this kind (see Rome, London or Berlin as examples). In Figure 6.1 two examples of possible outputs of the generator and cities with similar networks are shown.

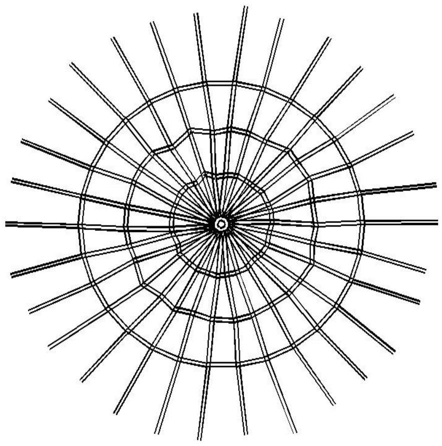
Circular networks are characterized by an angle of opening determining if the city network is semicircular, circular or with a specific angle. This opening angle is here called angular width and it reflects the degree of the circular sector forming the network. In order to generate a road network reflecting roads with different level of importance, some arc parameters have to be fixed as the free-flow speed, the number of lanes, the delay due



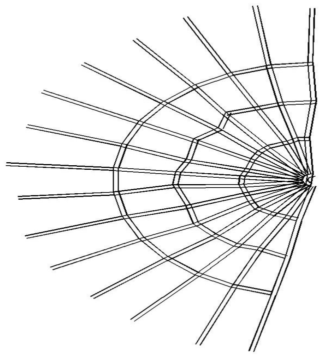
(a) Circular city



(b) Semicircular city



(c) Circular road network



(d) Semicircular road network

Figure 6.1. City with a circular and semicircular road network

to traffic regulations and the safety distance. The delay due to traffic regulations can be considered as an average delay due to traffic lights, greenswirls policies and one-way regulations that a user could experience on a certain arc and the safety distance is a parameter fixed by law. Length and the free-flow speed affects the free-flow travel time value, and the number of lanes and safety distance affect the arc capacity value. The key point in assigning these parameters to arcs is that a sort of continuity between adjacent arcs has to be guaranteed. For example, parameters are fixed in such a way arcs belonging to the same ring have the same free-flow speed and capacity, since they have more likely the same number of lanes, the same size and similar features. In order to simulate different network sizes, additional parameters have to be fixed as the internal and the external radius, i.e. the radius of the downtown area and of the metropolitan area respectively, the total number of rings, the angular width, the number of principal directions from which the morning commute traffic origins, a perturbation parameter that describes the regularity level in positioning vertices and the distance of the external origins from the city center. It is assumed that the traffic comes from the external zones to internal points of the city network simulating a morning commute scenario.

6.1 Instance generation

Instances are based on a directed graph G=(V,A) representing the road network in a metropolitan area including the city center and the surrounding towns. The generation of an instance uses a number of parameters that have to be fixed.
A summary of these parameters is shown in Table 6.1.

Generation is implemented through these two steps:

- Generation of the road network V
  - V is the set of vertices representing road intersections. We describe vertices $i \in V$ with Euclidean coordinates (x_i, y_i), but it can be substituted by geographical coordinates. The procedure for the vertex set generation is shown in Section 6.2
  - A is the set of arcs representing road segments. For each arc $(i, j) \in A$ the arc length l_{ij}, the free-flow arc travel time t_{ij}^{FF} and the arc capacity u_{ij} have to be generated. Length l_{ij} is computed as the euclidean arc length, $l_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$. Free-flow travel times t_{ij}^{FF} are computed using the following formula $t_{ij}^{FF} = \frac{l_{ij}}{v_{ij}^{FF}} + td$. The capacity u_{ij} is the rate of vehicles that can traverse an arc without experiencing slowdowns due to congestion and it computed using the formula $u_{ij} = \frac{l_{ij}}{sd} nl$, where sd is the safety distance and nl the number of lanes. The procedure for the arc set generation is shown in Section 6.3.
- Generation of the OD pairs with demands C.

Parameters for the vertex set V:	
npd	number of principal directions from the external zone to the city centre,
nr	number of rings,
α	angular width (for instance if we want a half circle city is 180, or if circular 360),
$IntRadius$	internal radius (a circumference around the downtown area),
$ExtRadius$	external radius (a circumference around the metropolitan area),
$Pert$	maximum rate of perturbation of the vertex coordinates, $Pert \leq 1$,
δ	percentage of distance of the external vertices from the city center with respect to the $ExtRadius$.
Parameters for the arc set A:	
v_{ij}^{FF}	free-flow speed,
td	delay due to traffic regulations,
sd	safety distance,
nl	number of lanes,
nh	number of highways from external centroid to the outer ring.
Parameters for the OD pair set C:	
$uppB$	upper bound percentage for demand,
$lowB$	lower bound percentage for demand,
it	percentage of in-city OD pairs on out-city OD pairs,
pa	point attractivity probability.

Table 6.1. Instance generation: vertex set parameters

– OD pairs $c \in C$ are distinguished only with respect to the pair of vertices (O_c, D_c) representing origin and destination, respectively. Procedure for the generation of the OD pair set is deeply explained in 6.4. For each of such OD pairs, we also generate a multiplicity or a demand representing the rate of vehicles that intend to travel from an origin O_c to a destination D_c. The demand value is randomly drawn from an interval depending on the considered level of traffic as explained in Section 6.4.

6.2 Generation of the vertex set

In order to generate the vertex set V, we have to generate all the points related to the rings, junctions between rings' points and links connecting different rings.

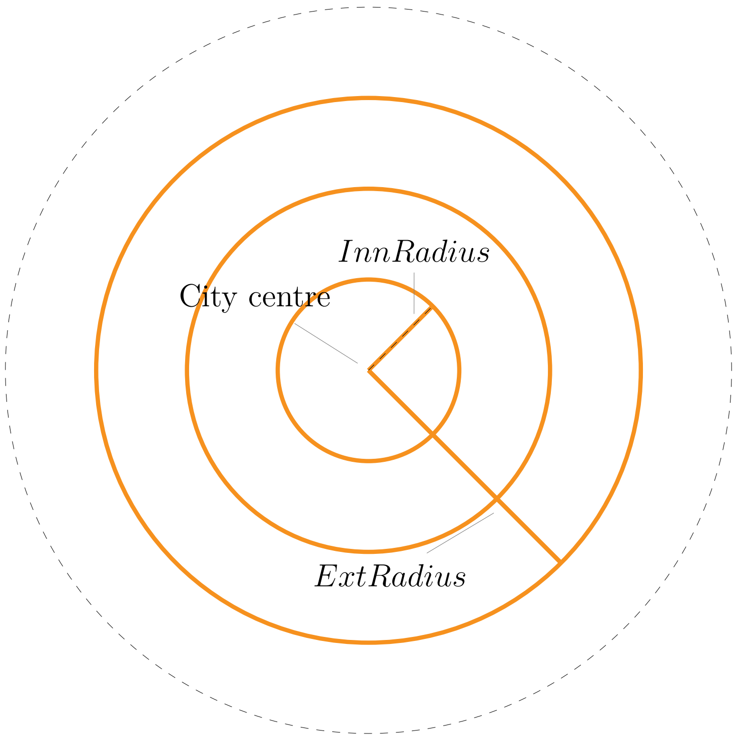
The vertex generator works through these steps:

- Generate the city centre point (Figure 6.2(a)),
- Create a circumference using the internal radius $IntRadius$ (Figure 6.2(a)),
- Create a circumference using the external radius $ExtRadius$ (Figure 6.2(a)),
- Create a number of equispaced intermediate rings in order to have nr rings globally (In Figures 6.2(a)-6.3(b) we have used $nr = 3$),
- Divide the plane into npd equivalent circular sectors (Figure 6.2(b)),
- Choose as vertex every intersection between the circumferences and the main directions (Figure 6.3(b)),
- Generate a vertex on each principal direction with distance from the center exceeding the $ExtRadius$ of a fixed percentage δ (Figure 6.3(b)). These vertices represent the points from which demand of the sorrounding towns originates. These vertices are called external centroids.

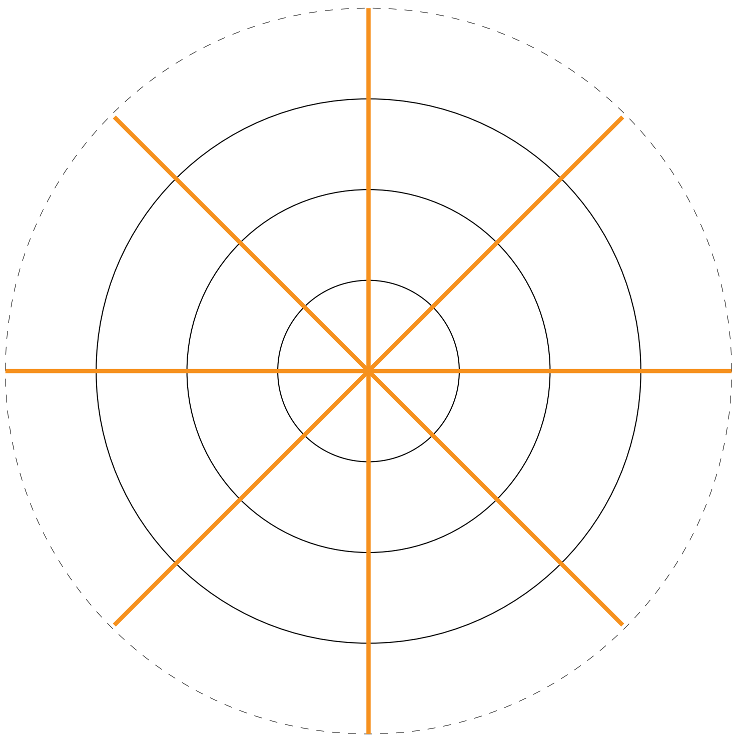
In fact, the generator is able to produce city networks with different angular widths using the parameter α. Since for our experiments we have considered only $\alpha = 360°$ in the following we will refer to the procedure with $\alpha = 360°$. However, considering other values of α, the procedure is very similar. In generating instances for the proposed problems also parameter δ is fixed equal to 50%.

In order to get a more realistic graph, the vertex generator can add some distortion to each point, as in Figure 6.4, through the following steps:

- Find the distance between the point and the nearest neighbour, $MinD$,
- $uB = MinD \cdot Pert$, where $Pert$ represents the maximum rate of perturbation of the vertex coordinates chosen,

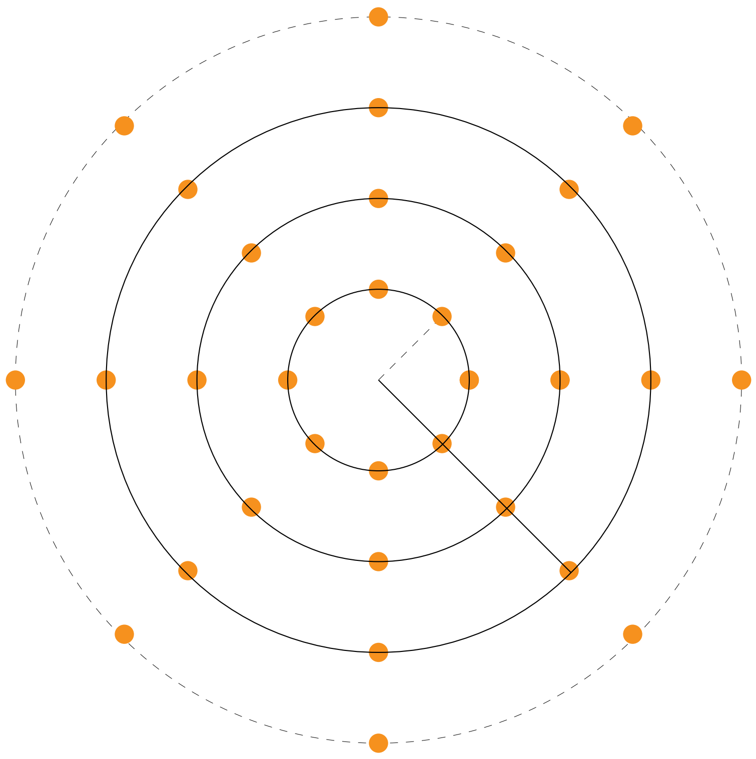


(a) Creation of the circumferences



(b) Creation of principal directions

Figure 6.2. Creation of the vertex set V


(a) Find the intersections

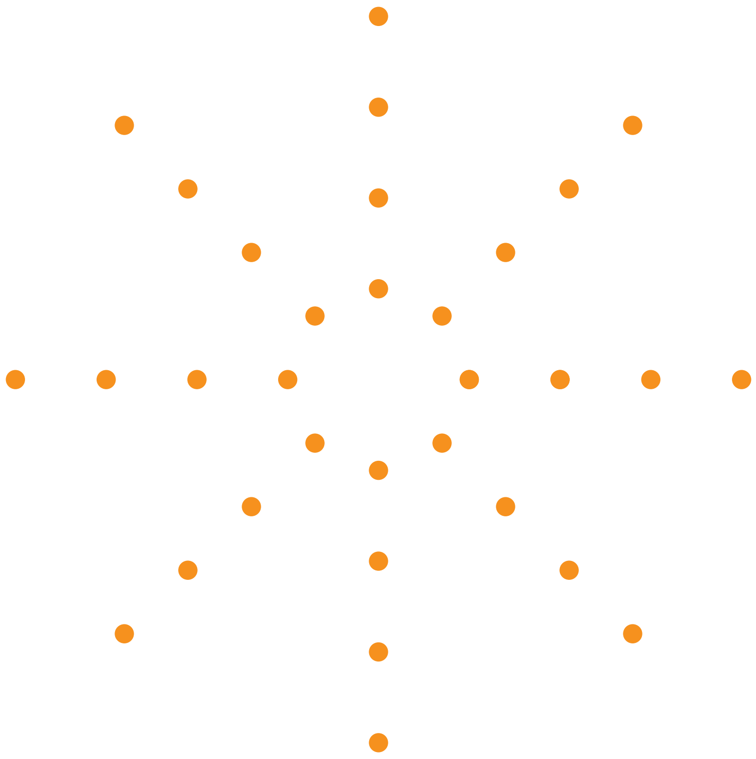

(b) Created vertices

Figure 6.3. Instance generation: creation of the vertex set V

- Generate a circular neighbourhood around the point we want to perturb with radius uB and choose a random point inside this neighbourhood. The generator uses this new point as vertex and deletes the old one.

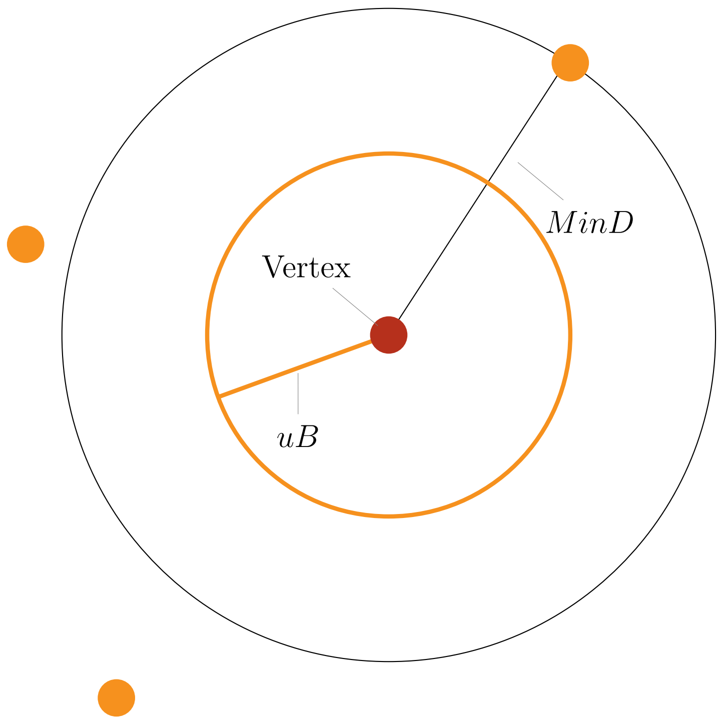


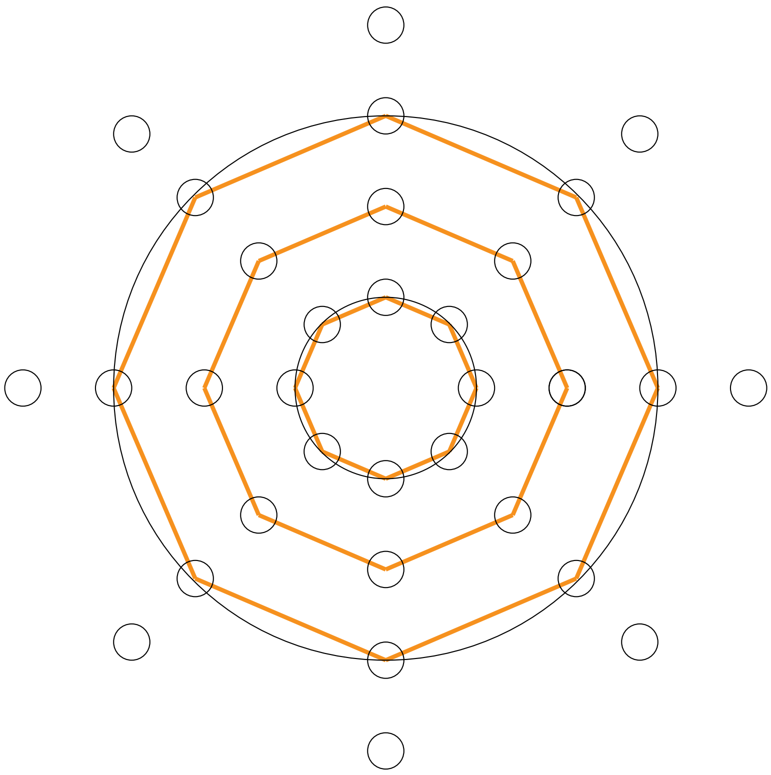
Figure 6.4. Instance generation: perturbation of the vertex coordinates

6.3 Generation of the arc set

The arc set A is generated connecting vertices on the rings, connecting rings to the next rings and connecting the external ring with the external centroids as in Figure 6.5. First, empty links are created and, then, parameters are generated. Empty links are created by connecting all the vertices for each ring and connecting each vertex of each link with the correspondent vertices on the closest rings. Then, the external ring vertices are connected with the correspondent external centroids. In order to simplify the choice of the initial parameters we have divided roads in four different classes with fixed parameters listed in Table 6.2.

In order to guarantee a sort of continuity between roads, each road is classified into the four classes as shown in Figure 6.6 and according to the following list:

- Internal ring: Internal roads,
- External Ring: Highways,


(a) Creating rings

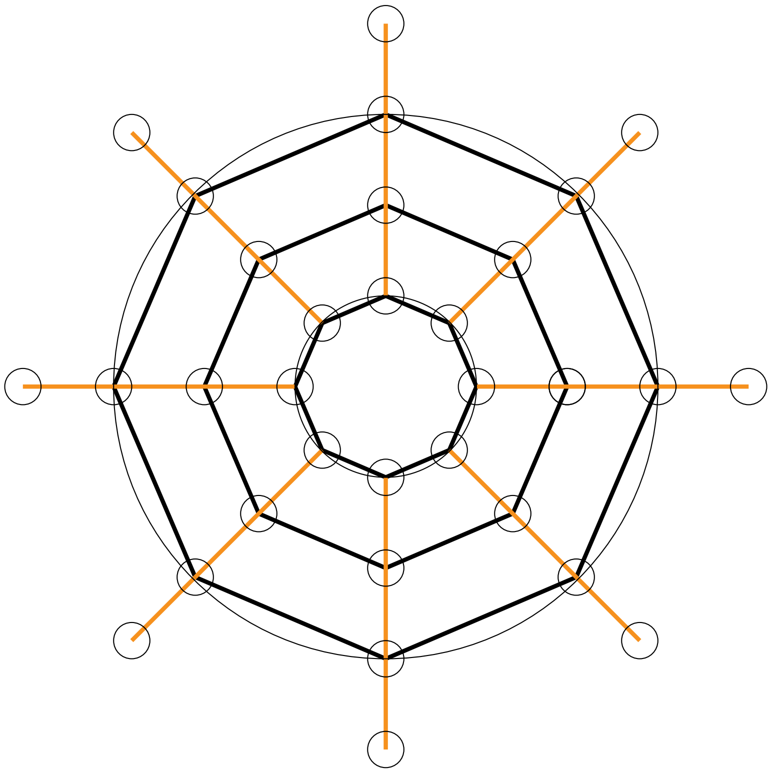

(b) Connecting rings and external centroids

Figure 6.5. Instance generation: connecting vertices

Road type	ln	v^{FF} (m)	sd (m/sec)	td (sec)
Internal Roads	2	13.89	10	150
Primary Roads	3	25	30	100
Linking Roads	2	25	0	150
Highways	4	30.55	40	100

Table 6.2. Instance generation: road dependent parameters

- Intermediate Rings: Primary Roads,
- Links between rings: Linking Roads,
- Roads from external centroids to the external ring: These roads could be either highways or primary roads. The generator chooses randomly nh of these roads and label them as highways.

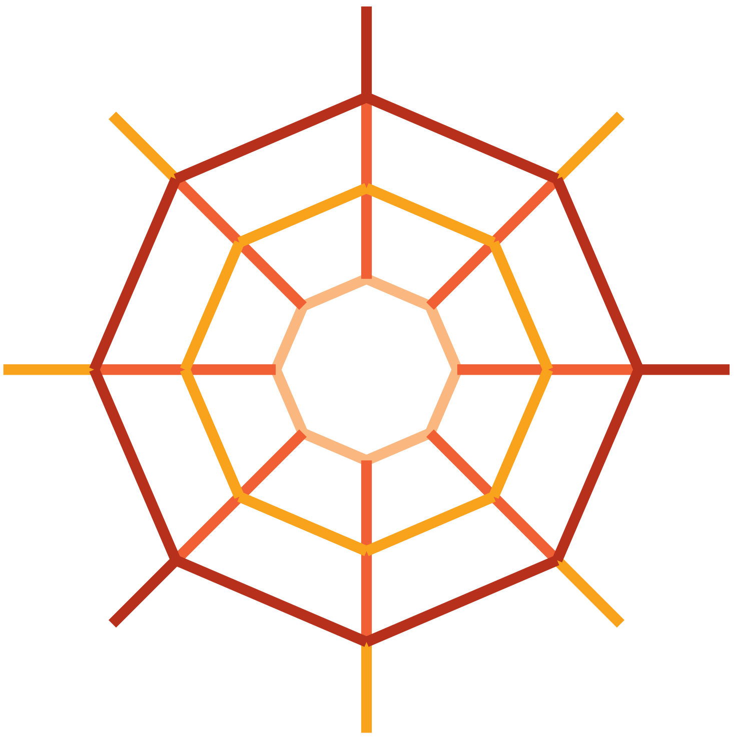


Figure 6.6. Instance generation: road classification in a three-rings instance

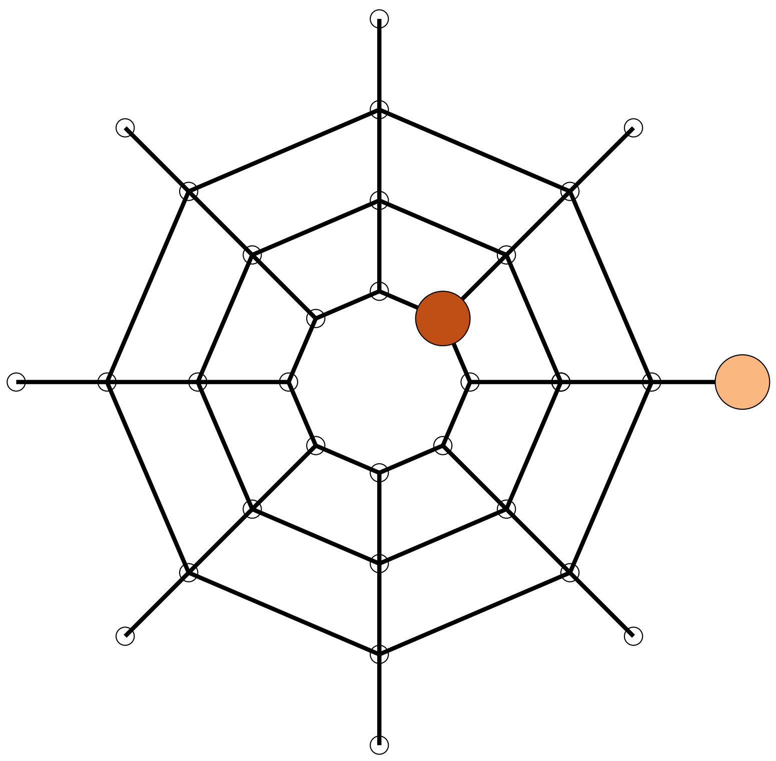
6.4 Generation of the OD pair set

Since we are generating instances for the morning commute it is reasonable to have most of the OD pairs with origin in the external centroids and destination inside or on the external ring. These OD pair are called out-city OD pair. However, a percentage of commuters, forming the so-called in-city OD pairs, can have origin on a vertex inside or on the external ring. For this reason we define a parameter it that represents the percentage of in-city OD pairs on out-city OD pairs number. We denote the set of origins as J. We choose as origin each external centroid and we collect them in J. A number of vertices inside or on the external ring equal to $it\%$ the number of external centroids is used as origin adn collected in J. An example of an in-city and an out-city OD pair is shown in Figure 6.7. For every origin we set a number of destinations $nDest = \lceil \frac{(nr+1)npd}{5} \rceil$. We assign to each vertex inside the external ring a boolean value determining if the vertex is allowed to be chosen as destination or not. A parameter pa, called point attractivity probability, represents the percentage of the vertices that are allowed to be chosen as destination. The OD pairs' destinations are drawn randomly among all the vertices that can be chosen as destination. Once the OD pairs are generated, the demand has to be fixed. Since usually roads entering the road networks are sized in such a way demand is serviced in most cases, it is reasonable to calculate demands proportional to the capacity of all the arcs exiting from the origin vertex. We call S the sum of all the arc capacities exiting from the origin vertex. Two parameters $uppB$ and $lowB$ related to the traffic density have to be provided to the generator and, then, the demand is randomly drawn from the interval $[lowB \cdot S; uppB \cdot S]$.

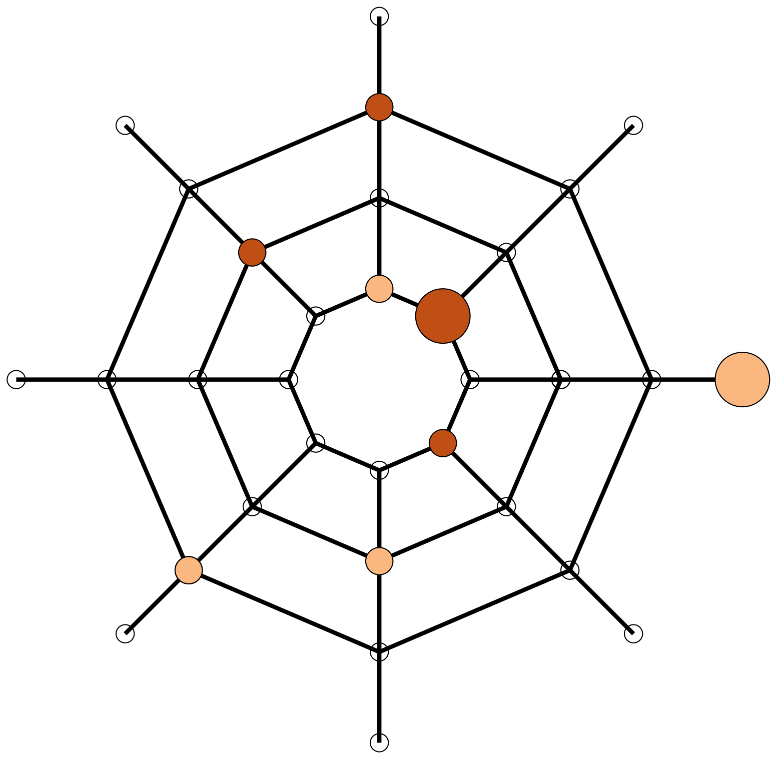
6.5 Controls on generation

The generator contains many parameters which allow changing the resulting network. Among these parameters there are some crucial and others that involve minor changes or technical details. The set of generated instances can not be very large so we have chosen, for each type of parameter, only a few values.

- **Size of the city**. The generator allows you to create small-medium or large city. The choice of this parameter is equivalent to the choice of different parameters simultaneously, such as the inner radius, the outer radius, the number of the outgoing main directions, the number of rings to create and the number of entering highways. The chosen values for this parameter are the followings:
  - **Small cities**
    - IntRadius: 2000 m
    - ExtRadius: 6000 m
    - nr: 2
    - npd: 15
    - nh: 2
  - **Big cities**
    - IntRadius: 1000 m
    - ExtRadius: 20000 m
    - nr: 4
    - npd: 30



(a) An in-city origin and an out-city origins



(b) Respective destinations

Figure 6.7. Instance generation: creating OD pairs

    * nh: 4

- **Point attractivity** Not all places are equally attractive for the city traffic. For this reason the generator implements a random procedure that identifies which points are desirable. To be able to choose these points, the generator requires a percentage that represents the number of attractive points on the total number of points. This percentage represents the point attractivity. The proposed values for this parameter are:
  - Oligo-centric cities. The point attractivity value is $pa = 40\%$. This means that only a few points on the graph are attractive.
  - Poli-centric cities. The point attractivity value is $pa = 80\%$. This means that a lot of points on the graph are attractive.
- **Internal traffic OD pairs**. Most of the traffic comes from surrounding cities and cross the city area or reach places inside the city limits. But there is a small component of traffic that is domestic, i.e. that is due to the people, living in the city, who decide to travel by car despite of the presence of public transportation. This parameter is a percentage of the whole amount of traffic that is in city traffic. The proposed values for this parameter are:
  - $it = 10\%$ in city traffic.
  - $it = 20\%$ in city traffic.
- **Traffic density**
  The flow for an OD pair is set randomly from some specified intervals. We distinguish `off-peak` and `in-peak` traffic densities for each network size:
  - Small-to-medium
    * `off-peak`: $lowB = 0.2$ and $uppB = 0.4$
    * `in-peak`: $lowB = 0.2$ and $uppB = 0.6$
  - Big
    * `off-peak`: $lowB = 0.1$ and $uppB = 0.3$
    * `in-peak`: $lowB = 0.3$ and $uppB = 0.6$

The set of instances is the result of the Cartesian product of all the control values.

Appendices

.1 The one OD pair case proof

Theorem 3. *Let $|C| = 1$ and let us denote by c the OD pair. Let A_c^* be a minimum cut-set for c with capacity u_c^*. Then, $\rho_\infty^* = \frac{d_c}{u_c^*}$.*

Proof. Let $\hat{\rho} = \frac{d_c}{u_c^*}$. We prove that $\rho_\infty^* = \hat{\rho}$ in two steps, showing that:

1. There exists a flow $\hat{f}_{ij}$, $\forall (i,j) \in A$, such that:
   - d_c flows from O_c to D_c,
   - $\hat{f}_{ij} \leq \hat{\rho}$, $\forall (i,j) \in A$,
   - $\hat{f}_{ij} = \hat{\rho}$ for some $(i,j) \in A$.
2. There does not exist a flow g_{ij}, $\forall (i,j) \in A$, such that:
   - d_c flows from O_c to D_c,
   - $\frac{g_{ij}}{u_{ij}} < \hat{\rho}$, $\forall (i,j) \in A$.

1. We solve the maximum flow problem on the graph G for the OD pair c. Let $\delta^+(O_c)$ be the set of the successors of O_c. From the max flow-min cut theorem, the maximum flow F_{max} is equal to the minimum cut-set capacity u_c^*. Flows f_{ij} in the optimal solution of the maximum flow problem are such that:

- $f_{ij} \leq u_{ij}$, $\forall (i,j) \in A$,
- $f_{ij} = u_{ij}$, $\forall (i,j)$ such that $(ij) \in A_c^*$.

We construct new flows $\hat{f}_{ij} = f_{ij}\hat{\rho}$, $\forall (i,j) \in A$, and obtain:

- $\sum_{j \in \delta^+(O_c)} \hat{f}_{O_c j} = \sum_{j \in \delta^+(O_c)} f_{O_c j}\hat{\rho} = \hat{\rho} \sum_{j \in \delta^+(O_c)} f_{O_c j} = \hat{\rho} F_{max} = \hat{\rho} u_c^* = d_c$. Thus, this flow satisfies all the demand,
- $\frac{\hat{f}_{ij}}{u_{ij}} = \frac{f_{ij}\hat{\rho}}{u_{ij}} \leq \hat{\rho}$, $\forall (i,j) \in A$,

- $\frac{\hat{f}_{ij}}{u_{ij}} = \frac{f_{ij}\hat{\rho}}{u_{ij}} = \hat{\rho}$, $\forall (i,j)$ such that $(i,j) \in A_c^*$ where $f_{ij} = u_{ij}$.

2. By contradiction. Suppose there exists a flow g_{ij} such that $\frac{g_{ij}}{u_{ij}} < \hat{\rho}$, $\forall (i,j) \in A$. Then, $\sum_{(i,j)\in A_c^*} g_{ij} < \sum_{(i,j)\in A_c^*} u_{ij}\hat{\rho} = \hat{\rho}\sum_{(i,j)\in A_c^*} u_{ij} = \hat{\rho}u_c^* = d_c$. As this flow is not able to satisfy all the demand, we have a contradiction.

$\square$

.2 Graphical representation of the proactive route guidance approach solution

.2.1 Proactive route guidance approach arc utilization

In Figures 8-13 the arc utilization produced by the proactive route guidance approach is displayed. Black arcs represents the unused arcs, arcs in green the uncongested arcs, arcs in orange the lightly congested arcs and arcs in red the heavily congested arcs.

.2.2 Proactive route guidance approach OD pair selected paths

In Figures 14-19 the paths assigned by the proactive route guidance approach to a single OD pair with increasing values of γ are highlighted in red.

.3 Graphical representation of the linear constrained system optimum solution

.3.1 Linear constrained system optimum arc utilization

In Figures 20-23 the arc utilization produced by the linear constrained system optimum is displayed. Black arcs represents the unused arcs, arcs in green the uncongested arcs, arcs in orange the lightly congested arcs and arcs in red the heavily congested arcs. Figures

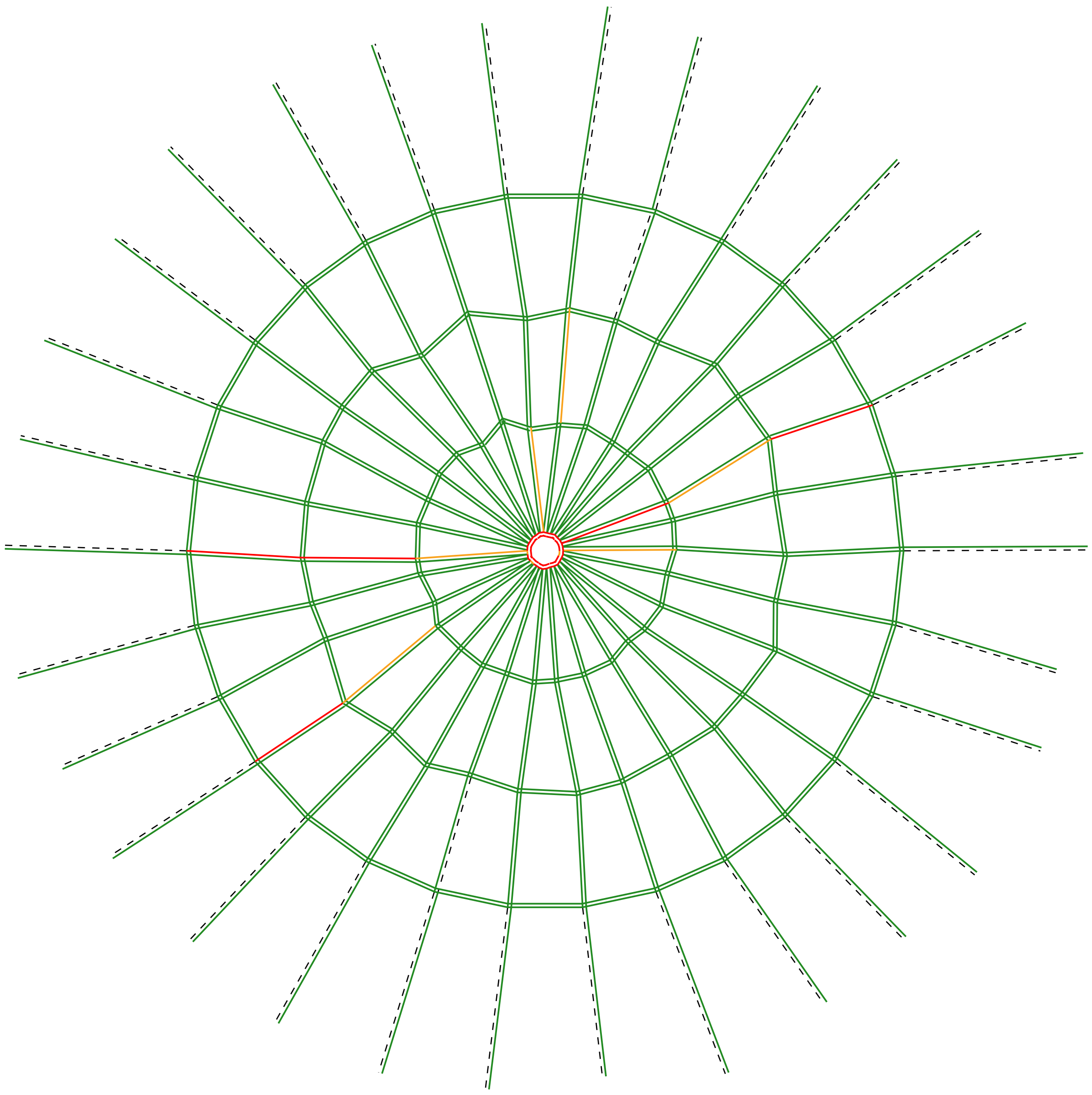


Figure 8. Arc utilization for $\gamma = 0\%$

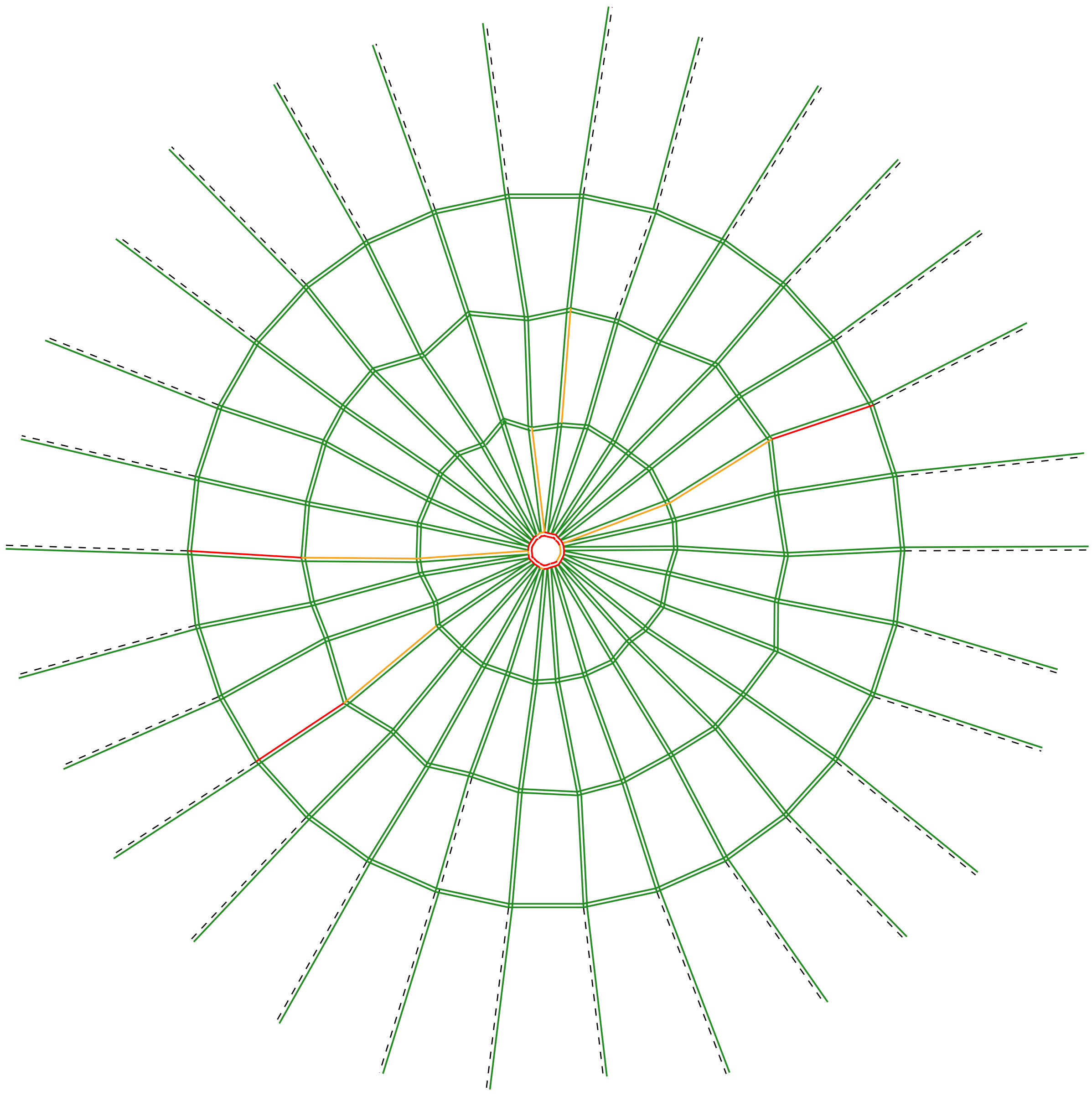


Figure 9. Arc utilization for $\gamma = 5\%$

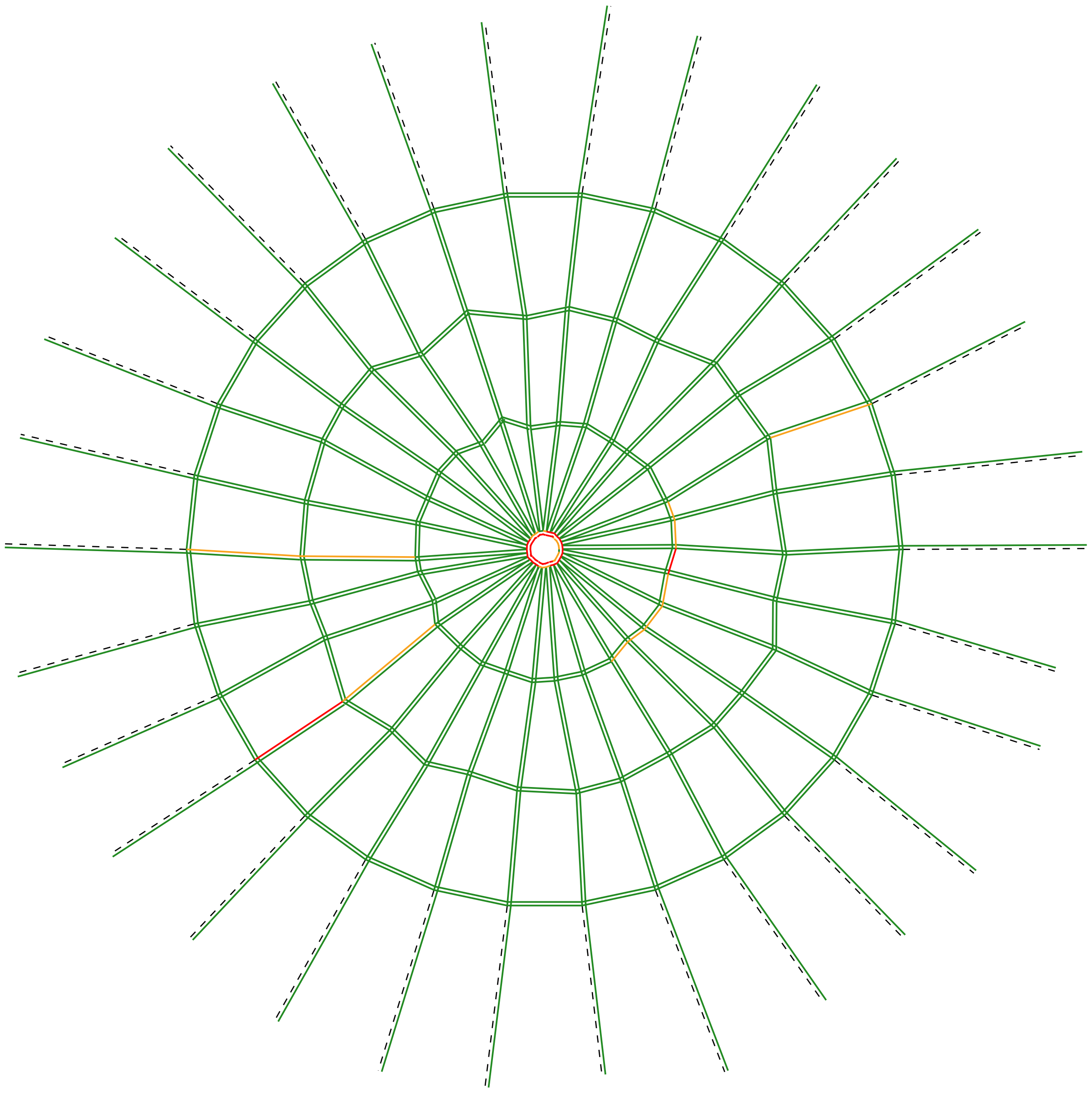


Figure 10. Arc utilization for $\gamma = 10\%$

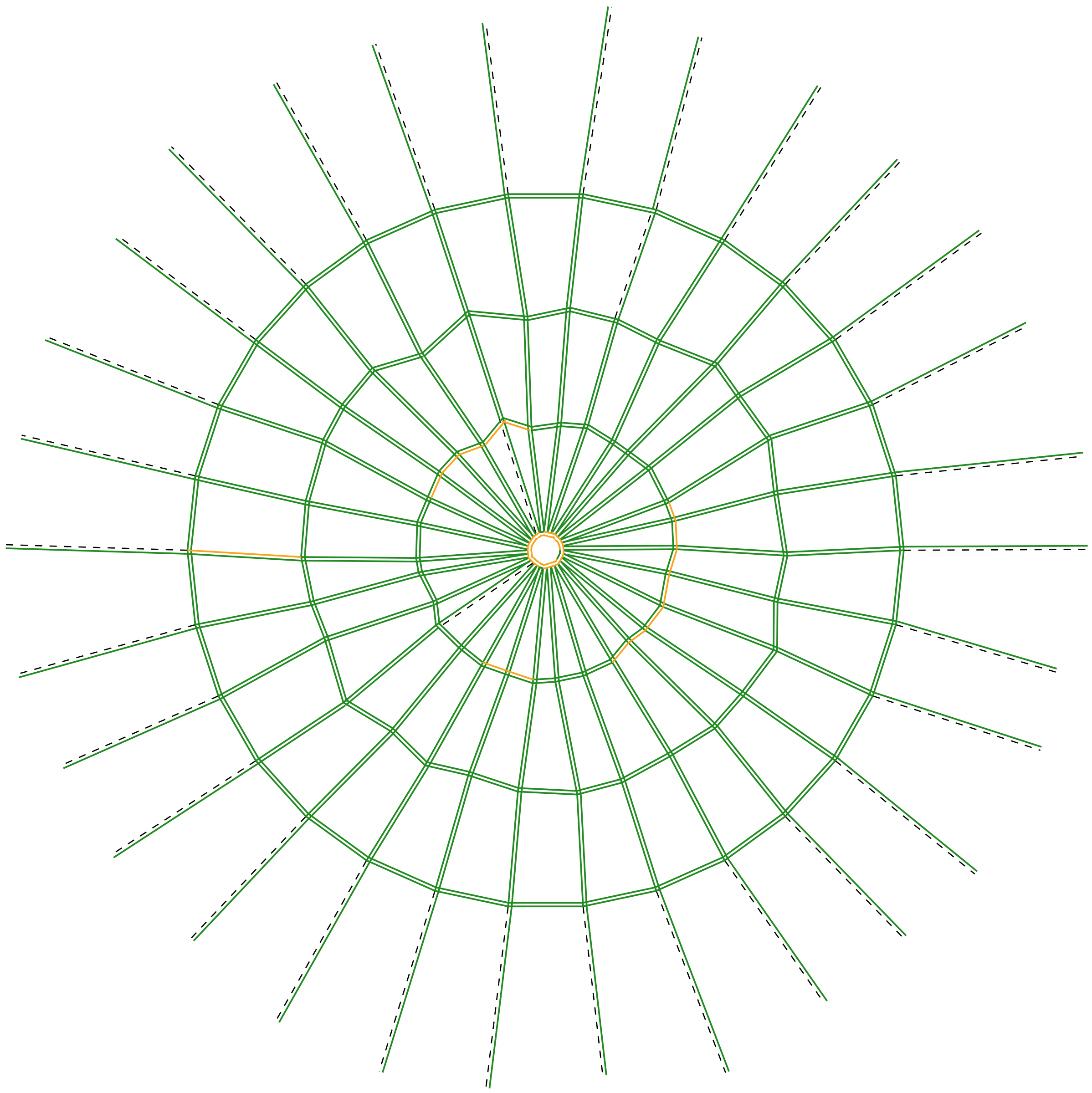


Figure 11. Arc utilization for $\gamma = 15\%$

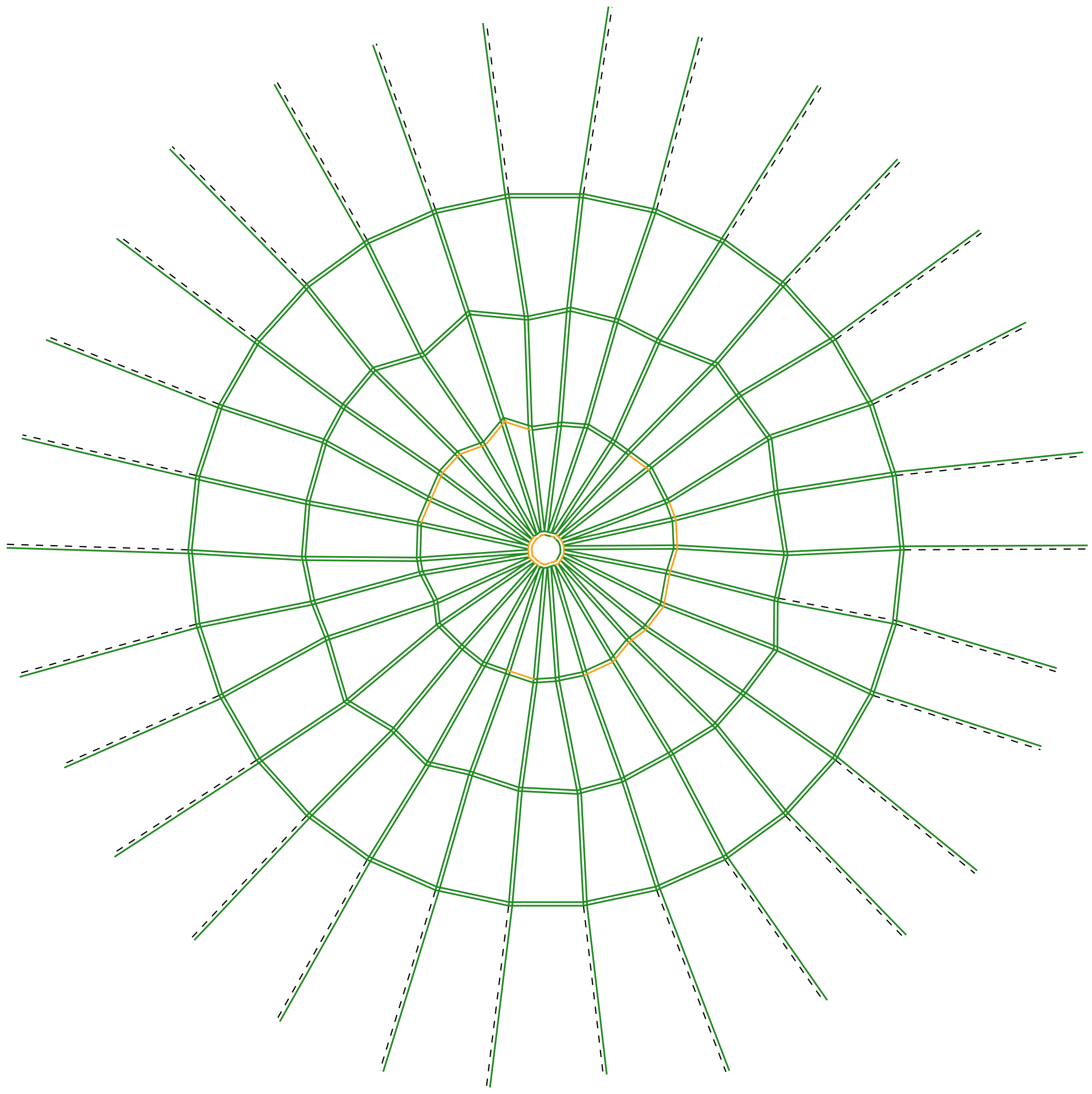


Figure 12. Arc utilization for $\gamma = 20\%$

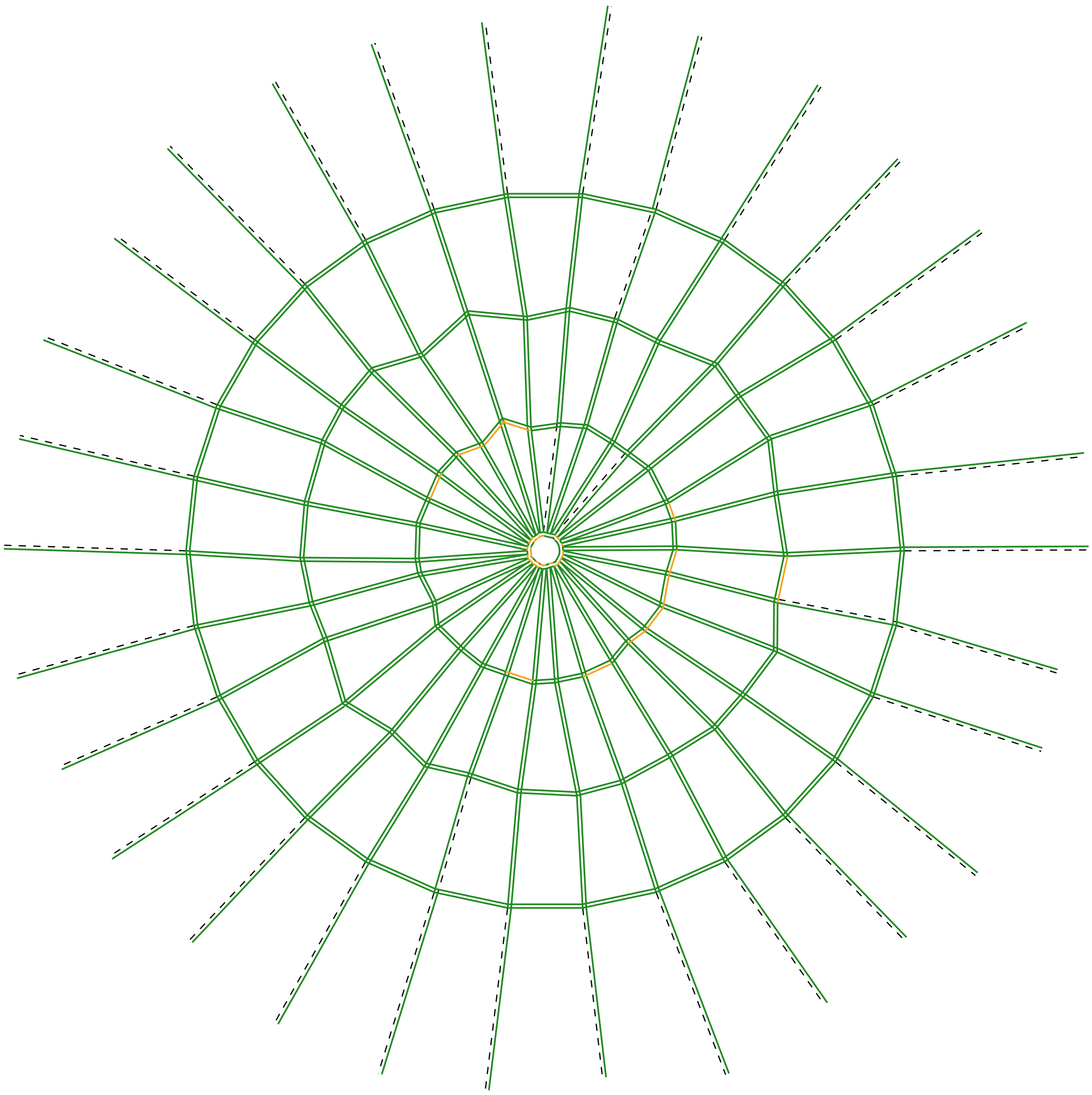


Figure 13. Arc utilization for $\gamma = 25\%$

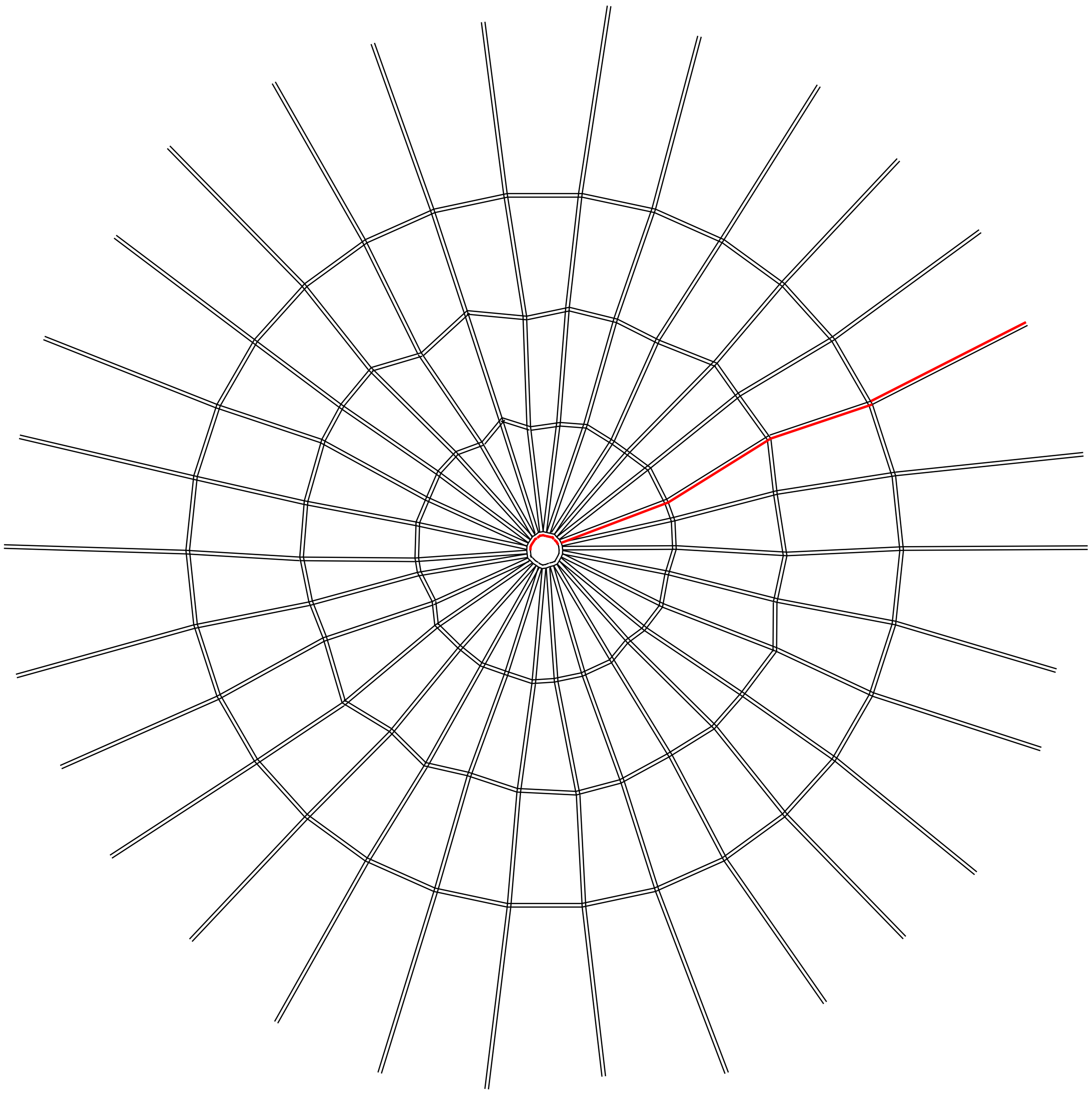


Figure 14. Selected paths for a single OD pair for $\gamma = 0\%$

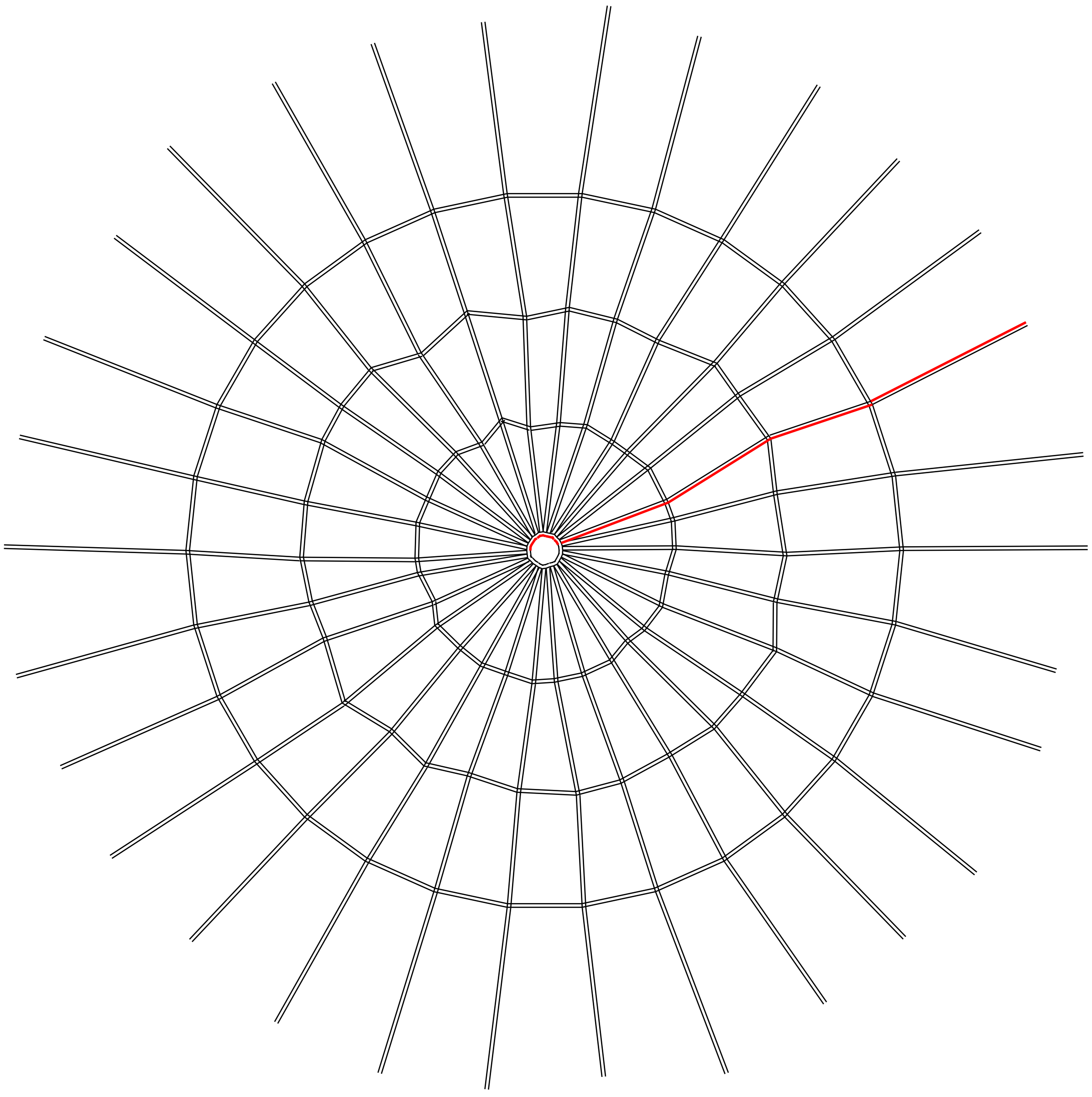


Figure 15. Selected paths for a single OD pair for $\gamma = 5\%$

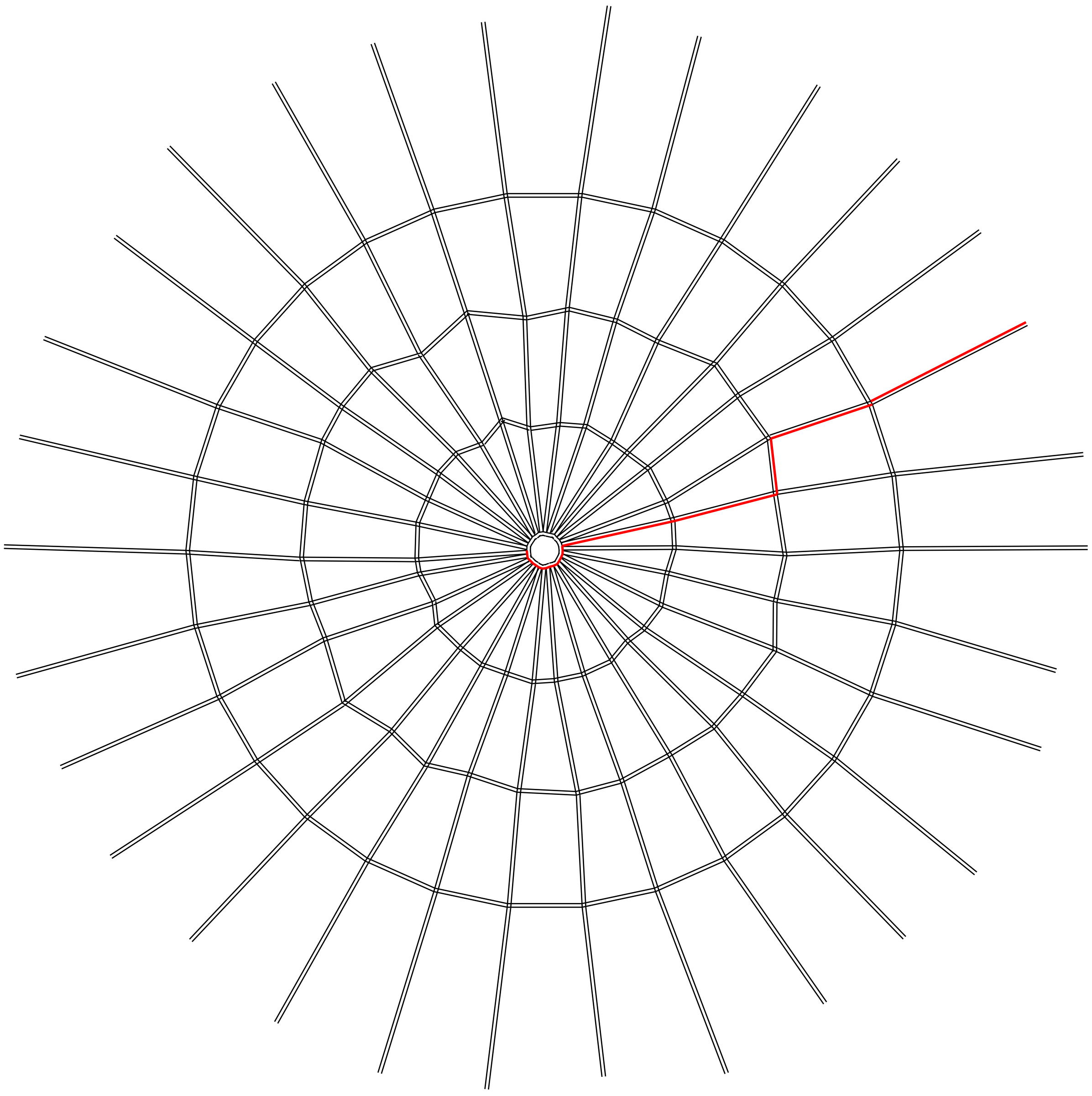


Figure 16. Selected paths for a single OD pair for $\gamma = 10\%$

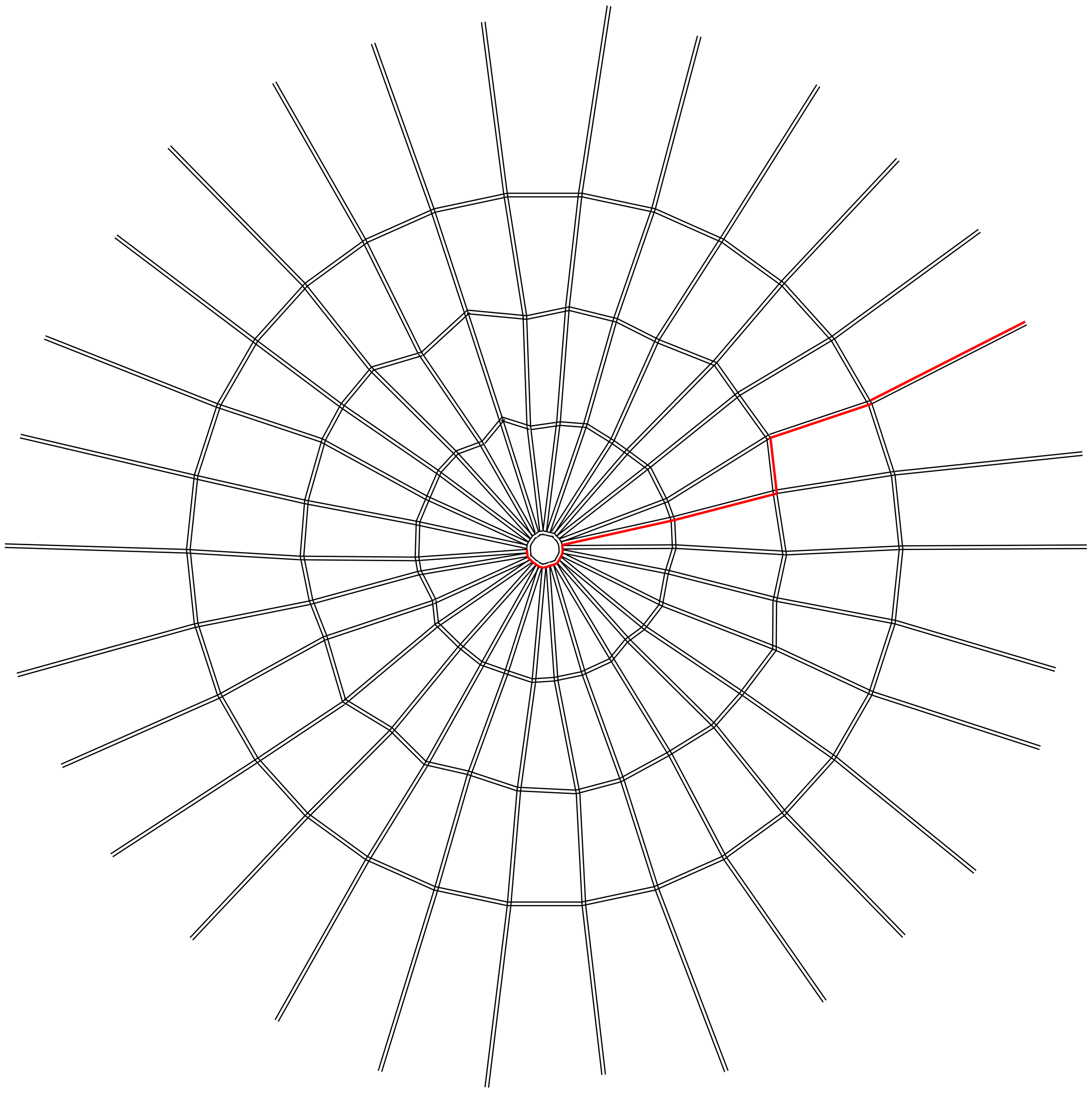


Figure 17. Selected paths for a single OD pair for $\gamma = 15\%$

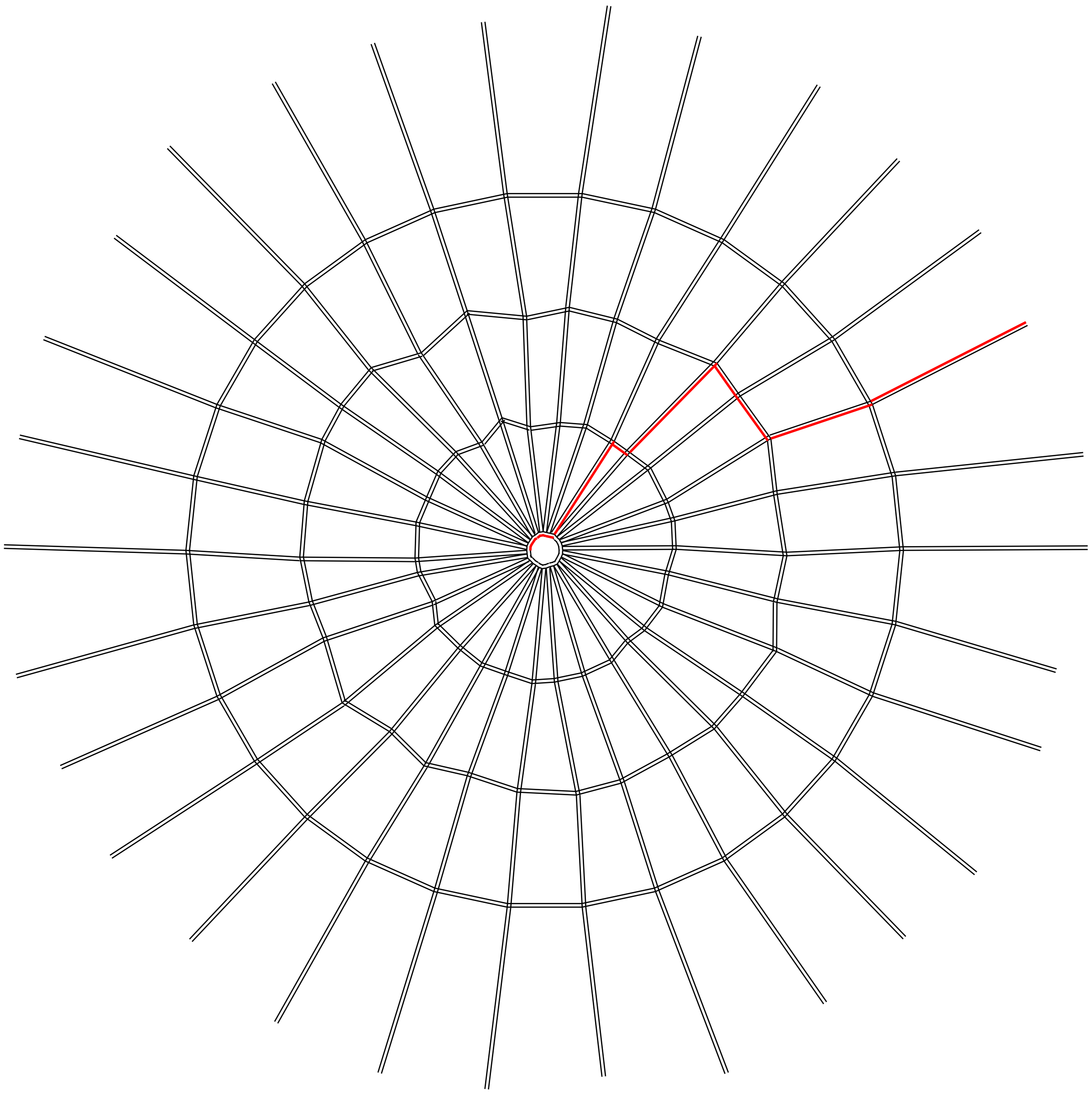


Figure 18. Selected paths for a single OD pair for $\gamma = 20\%$

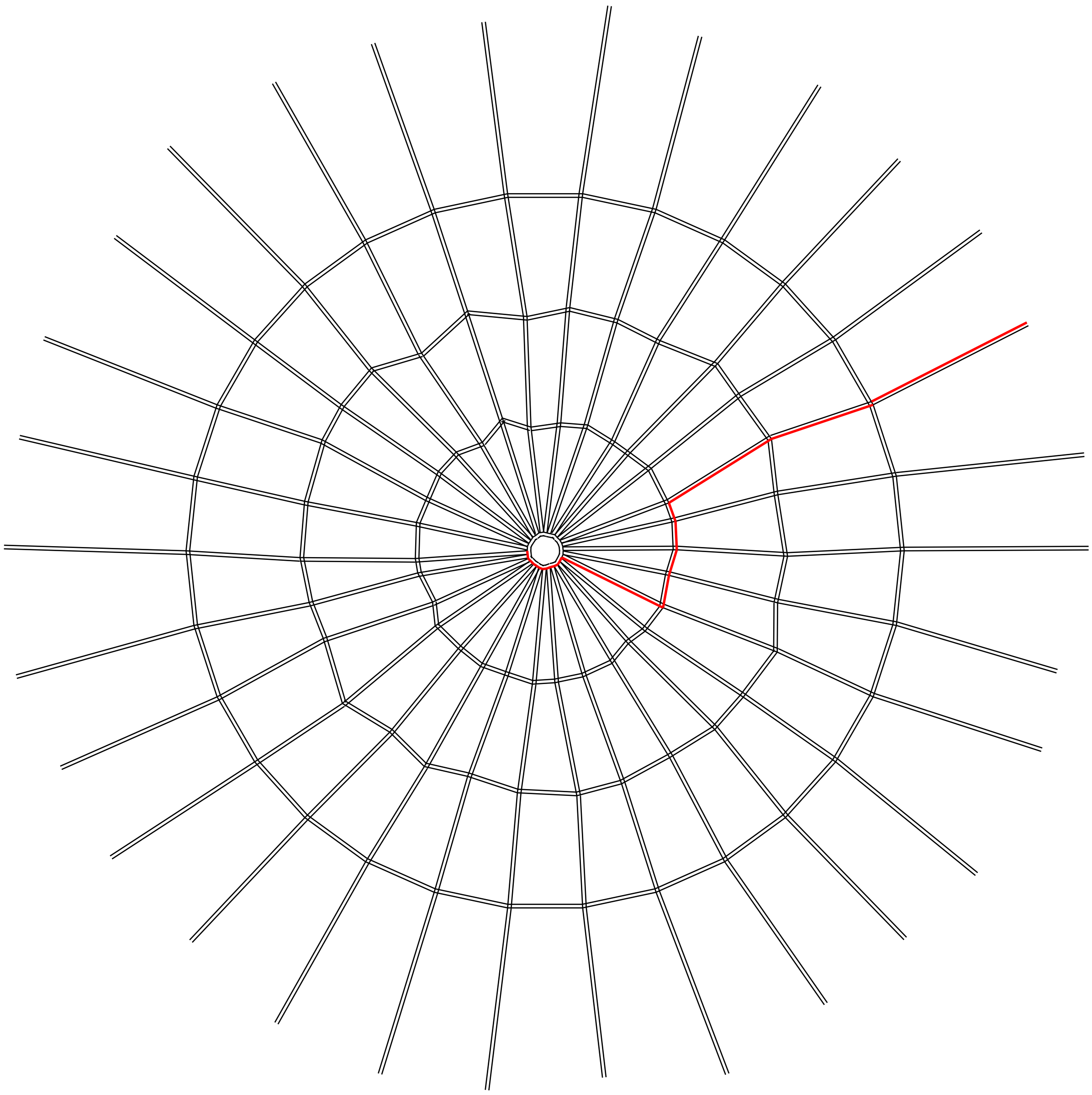


Figure 19. Selected paths for a single OD pair for $\gamma = 25\%$

for $\gamma = 20\%$ and $\gamma = 25\%$ are not shown since there is no difference when compared to the one with $\gamma = 15\%$.

.3.2 Linear constrained system optimum OD pair selected paths

In Figures 24-29 the used paths by a single OD pair with increasing values of γ are highlighted in red.

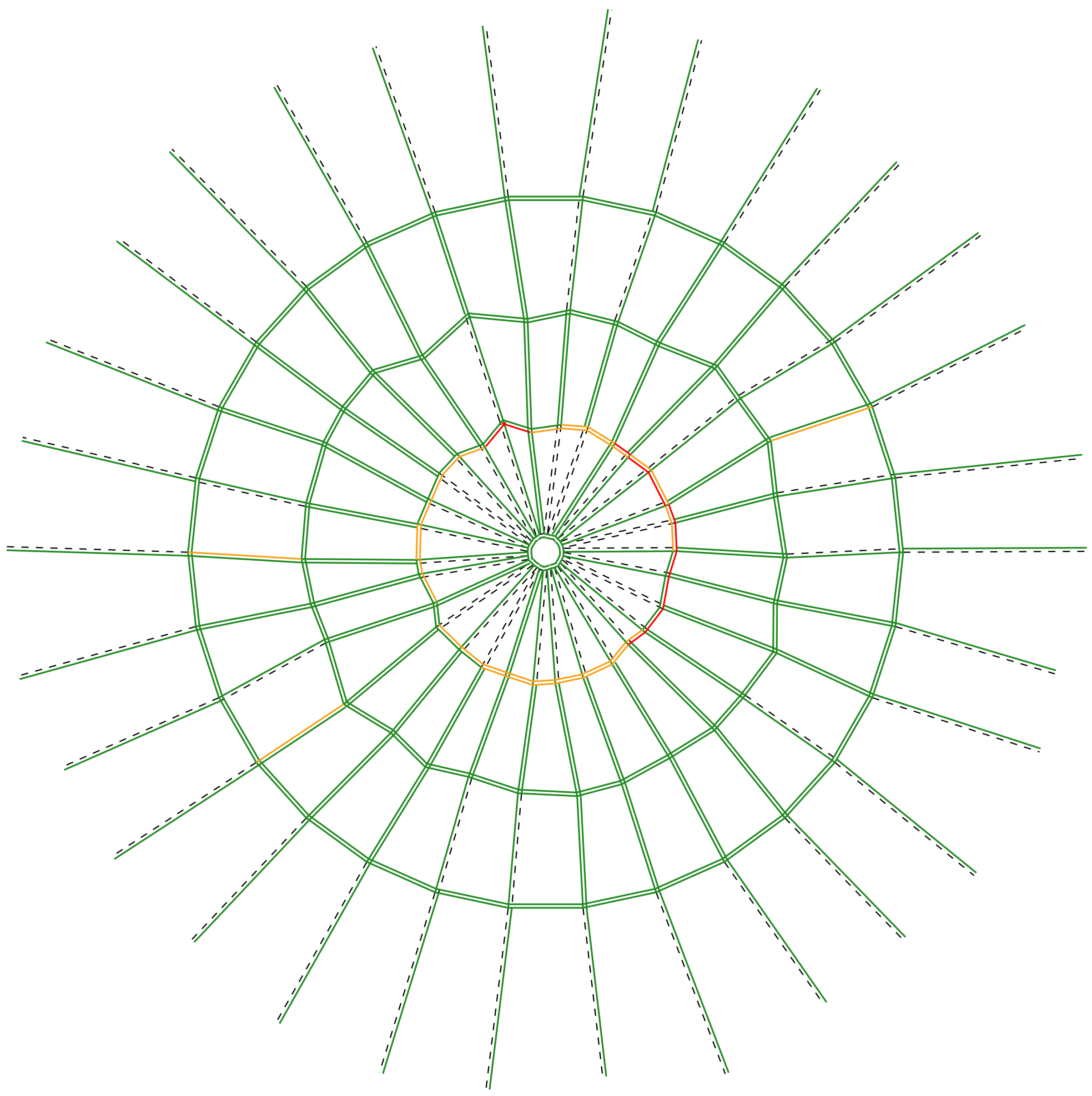


Figure 20. Arc utilization for $\gamma = 0\%$

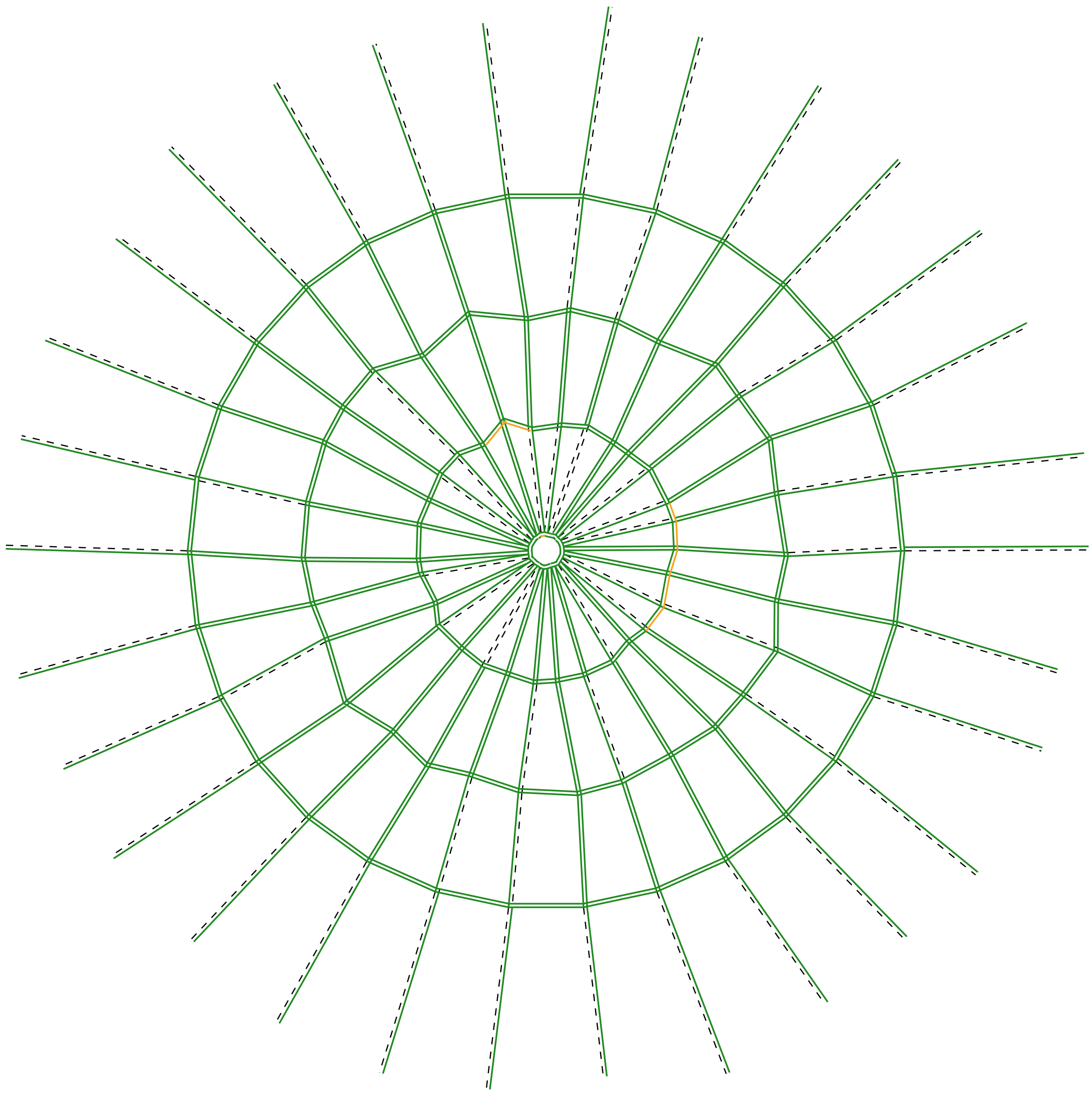


Figure 21. Arc utilization for $\gamma = 5\%$

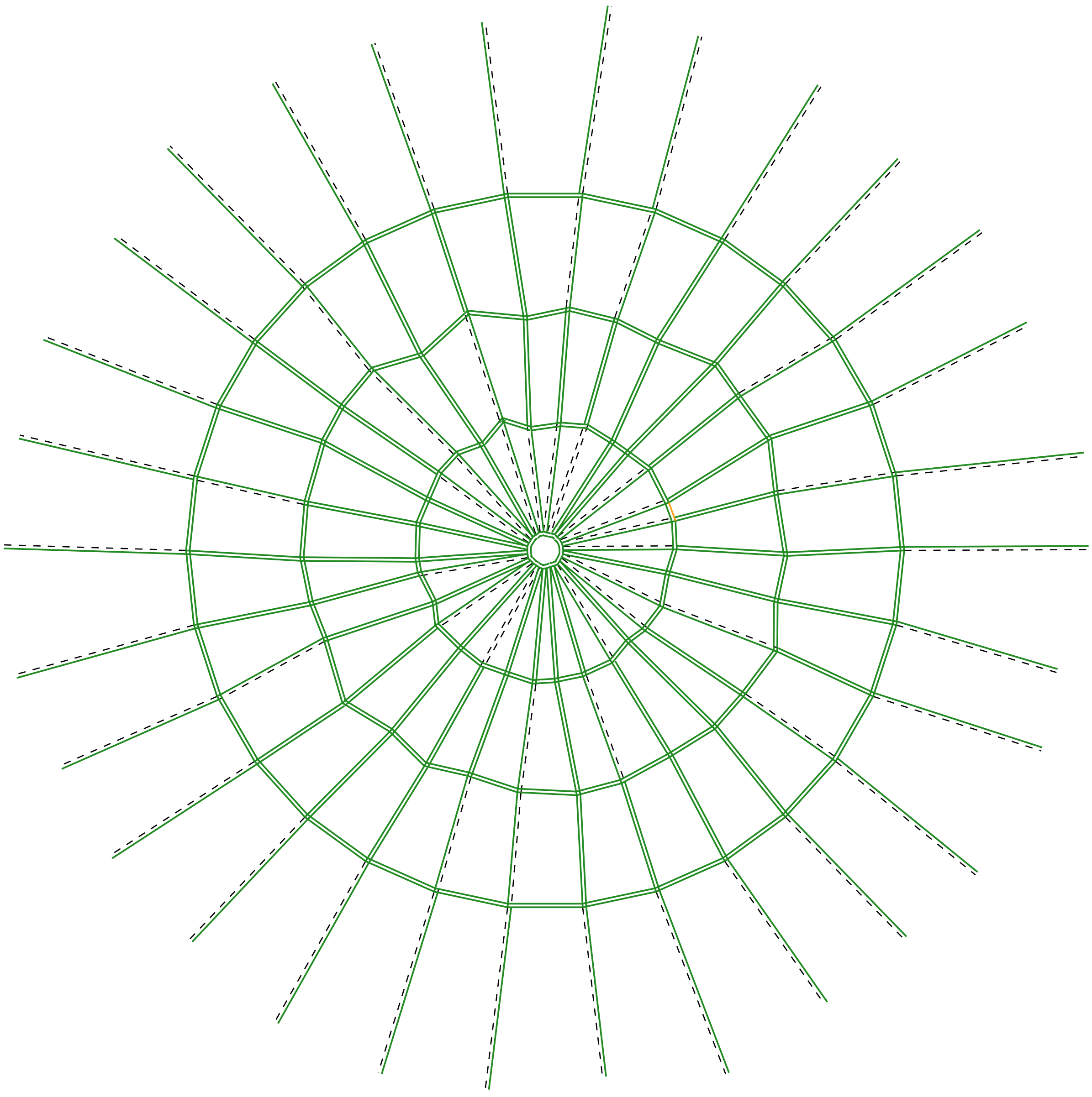


Figure 22. Arc utilization for $\gamma = 10\%$

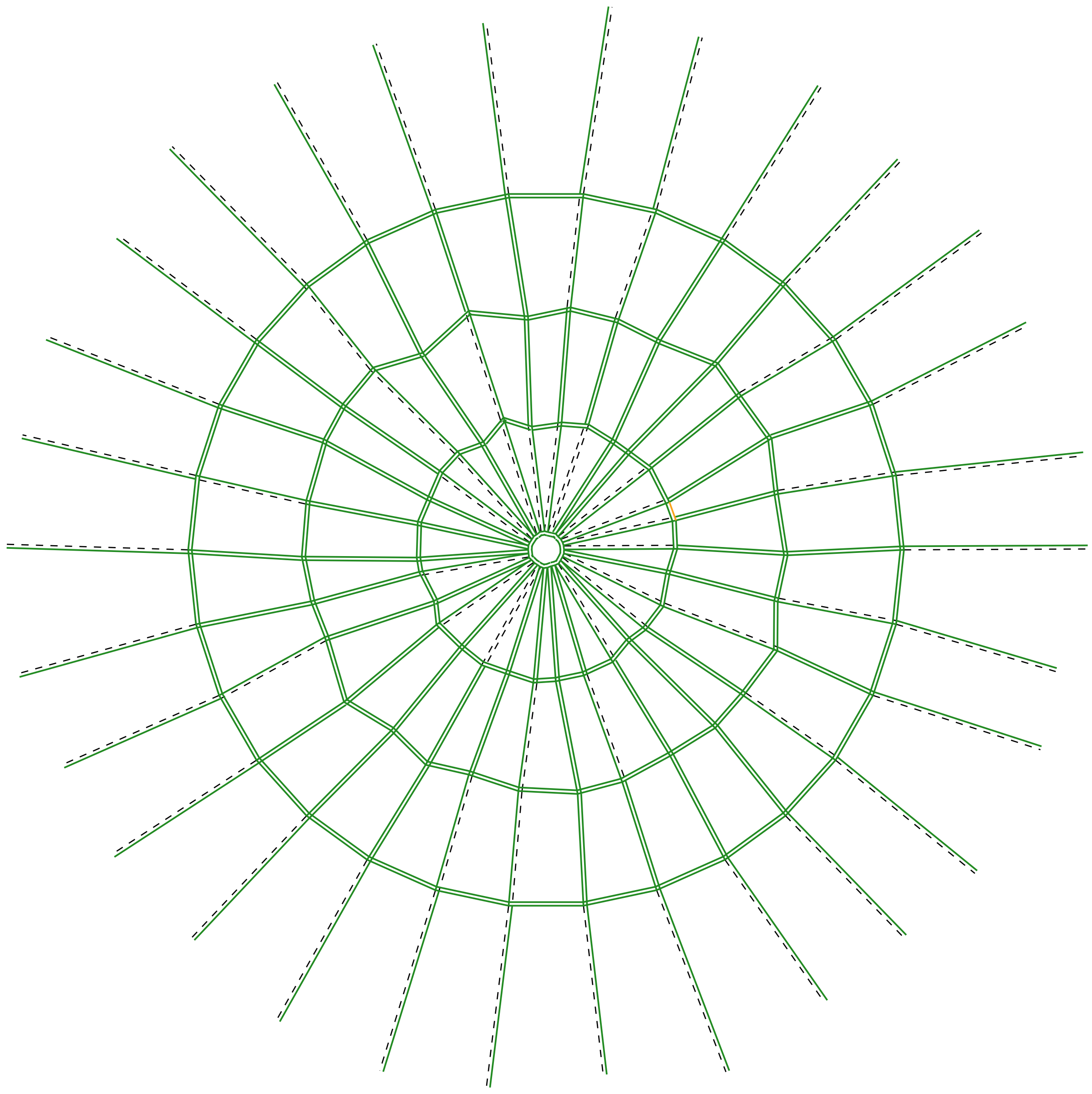


Figure 23. Arc utilization for $\gamma = 15\%$

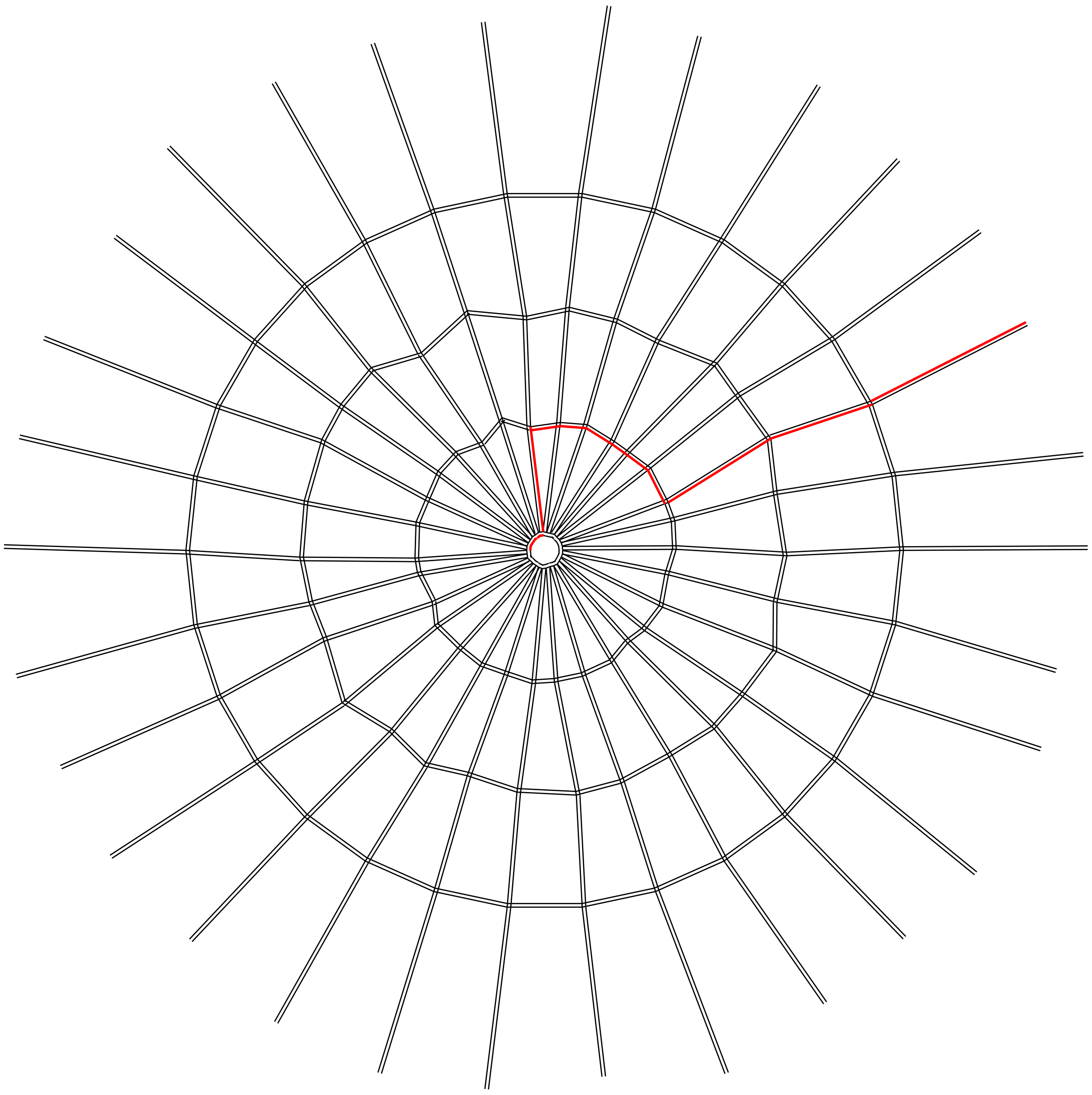


Figure 24. Selected paths for a single OD pair for $\gamma = 0\%$

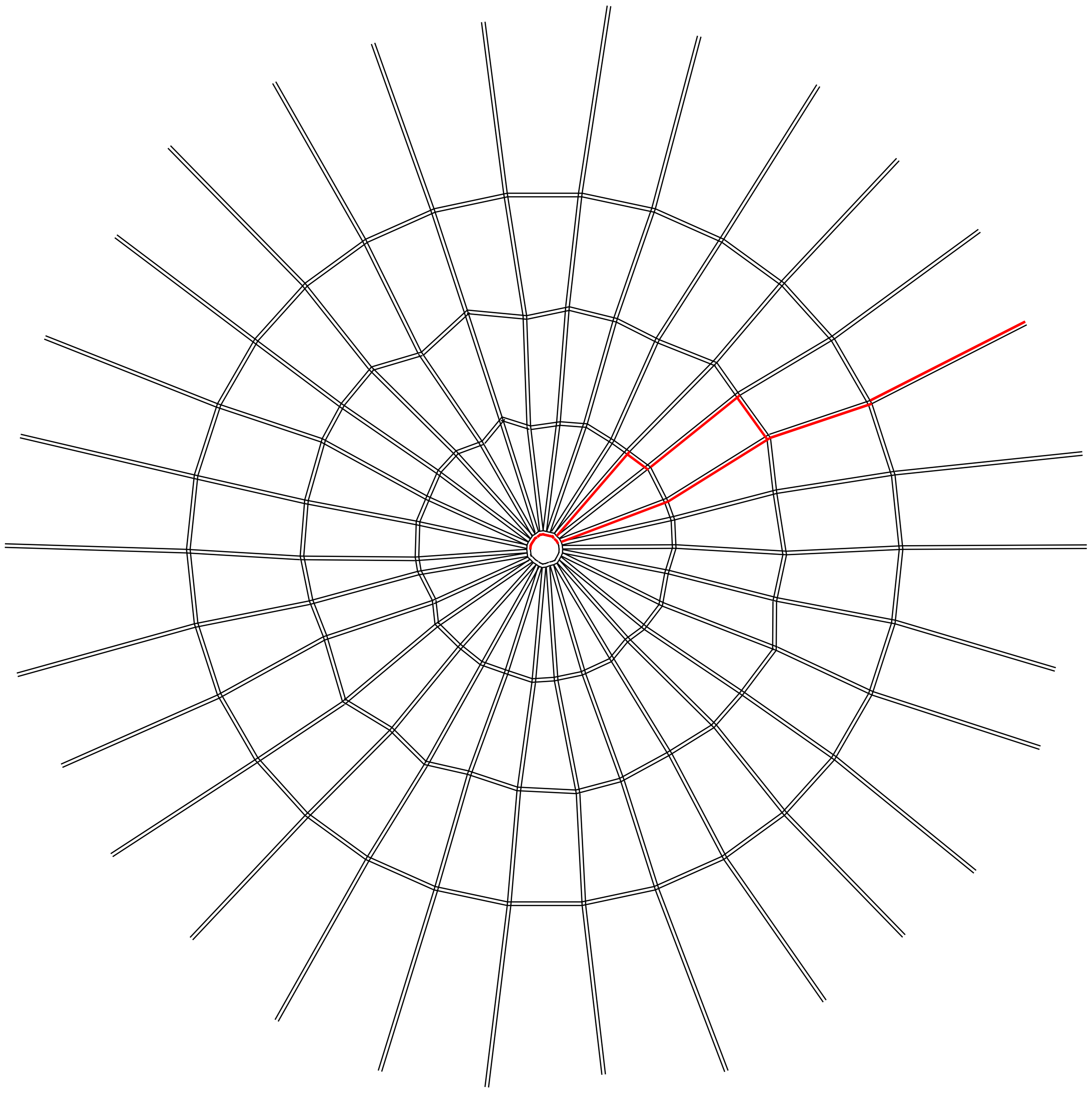


Figure 25. Selected paths for a single OD pair for $\gamma = 5\%$

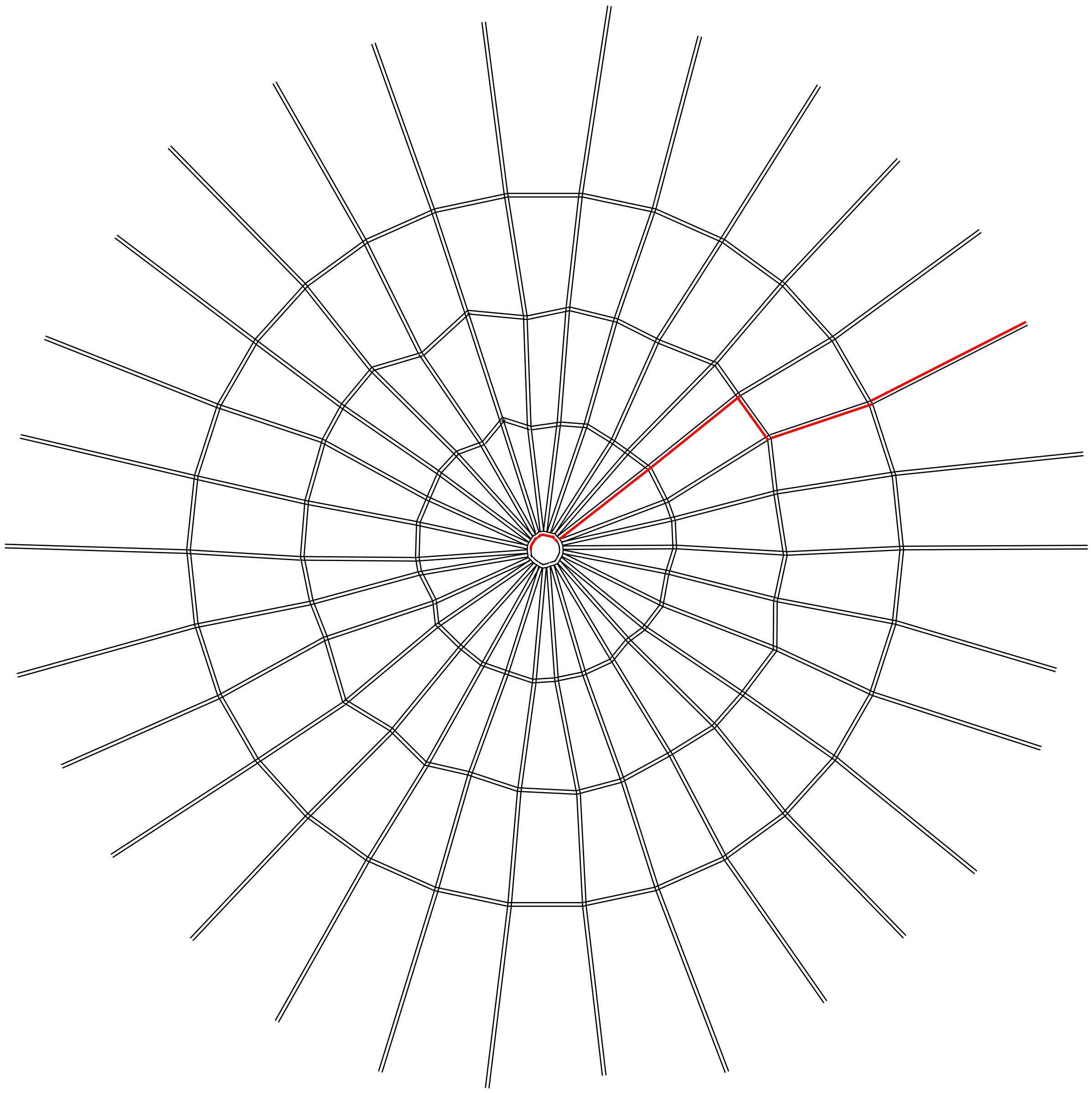


Figure 26. Selected paths for a single OD pair for $\gamma = 10\%$

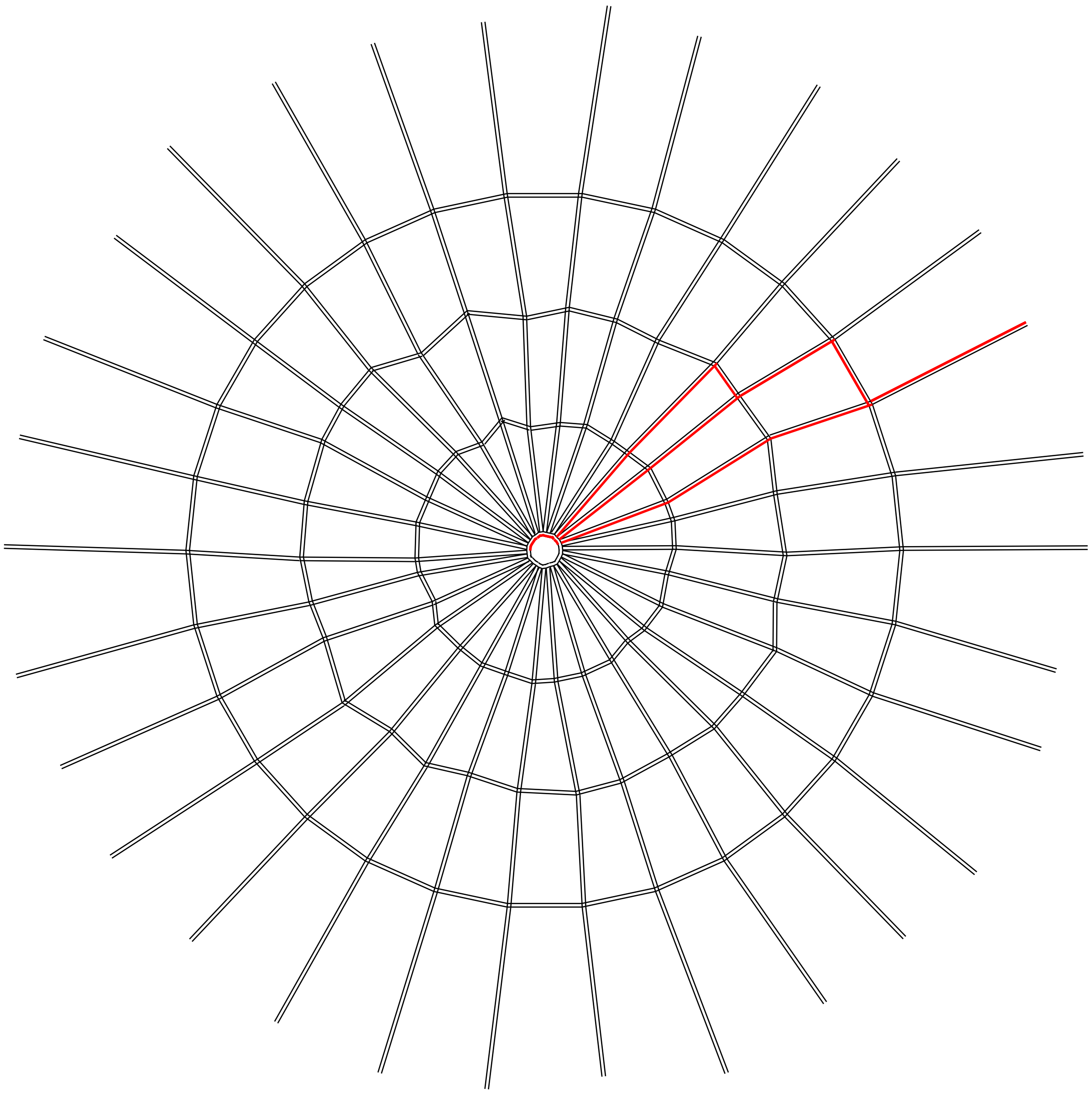


Figure 27. Selected paths for a single OD pair for $\gamma = 15\%$

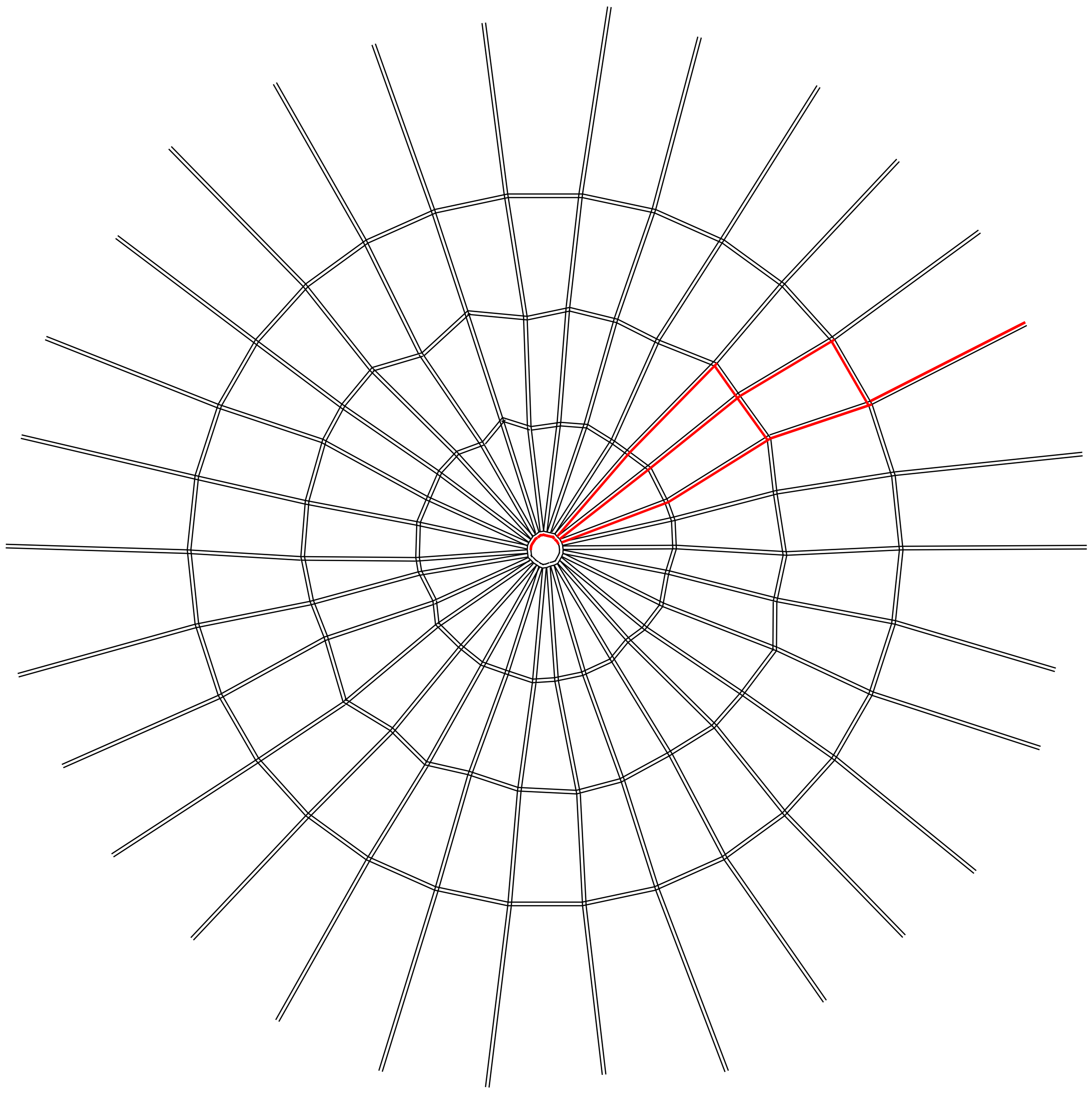


Figure 28. Selected paths for a single OD pair for $\gamma = 20\%$

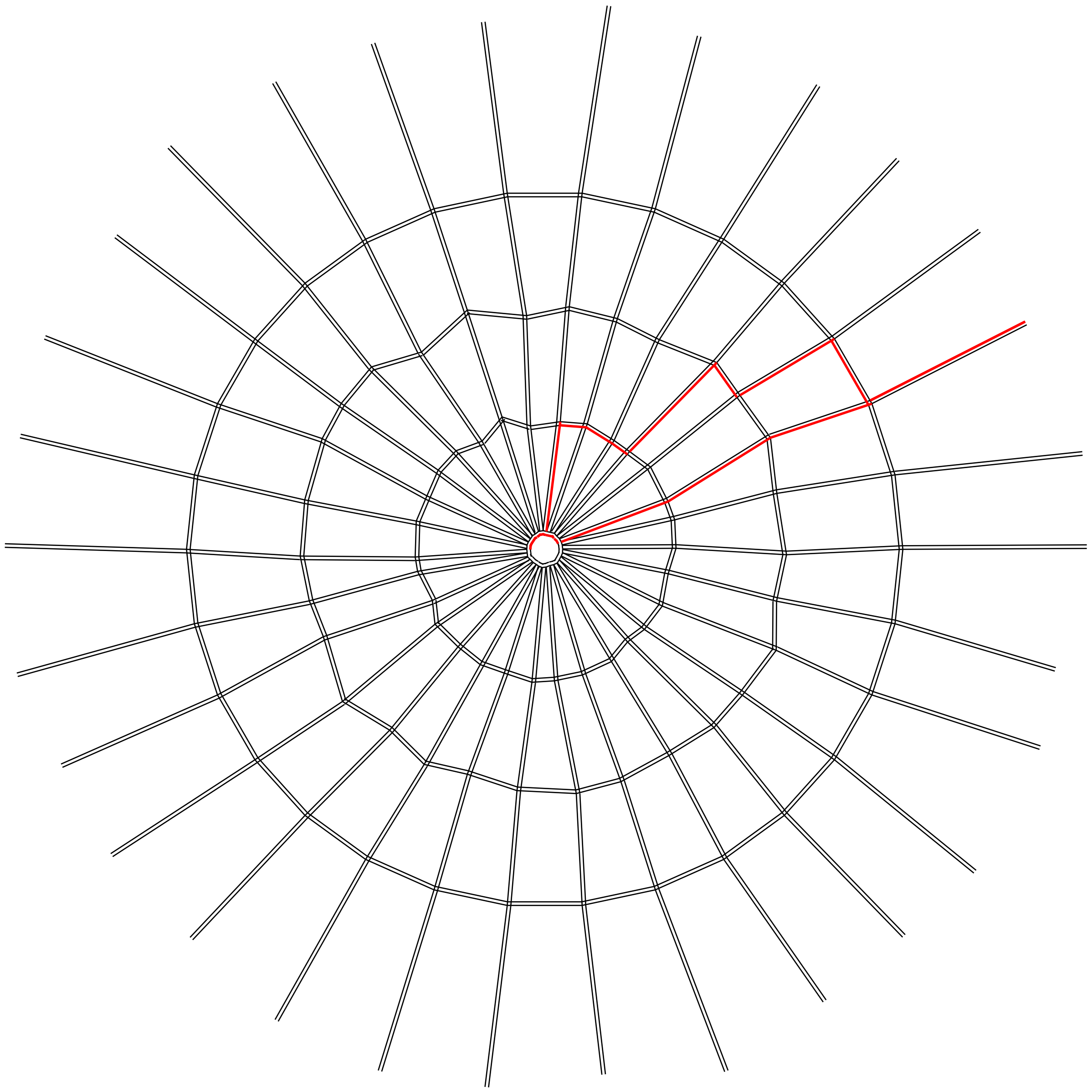


Figure 29. Selected paths for a single OD pair for $\gamma = 25\%$

List of Figures

List of Tables

List of Algorithms

Bibliography

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).

E. Angelelli, V. Morandi, and G. Speranza. Heuristic path generation for the linear constrained system optimum model. Technical Report 7-2016, University of Brescia, Department of Economics and Magagement, 2016d. URL `http://www.unibs.it/dipartimenti/economia-e-management/ricerca/working-paper`.

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.

K. Bartlett, J. Lee, S. Ahmed, G. Nemhauser, J. Sokol, and B. Na. Congestion-aware dynamic routing in automated material handling systems. *Computers & Industrial Engineering*, 70:176–182, 2014.

V. Bayram, B. . Tansel, and H. Yaman. Compromising system and user interests in shelter location and evacuation planning. *Transportation Research Part B: Methodological*, 72: 146 – 163, 2015.

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.

M. Bell and Y. Iida. *Transportation network analysis.* John Wiley & Sons, 1997.

M. E. Ben-Akiva and S. R. Lerman. *Discrete choice analysis: theory and application to travel demand.* MIT press, 1985.

M. E. Ben-Akiva, S. Gao, Z. Wei, and Y. Wen. A dynamic traffic assignment model for highly congested urban networks. *Transportation Research Part C: Emerging Technologies*, 24:62–82, 2012.

E. Ben-Elia, R. Di Pace, G. Bifulco, and Y. Shiftan. The impact of travel informations accuracy on route-choice. *Transportation Research Part C: Emerging Technologies*, 26, 2013.

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

M. E. Campbell. *Route selection and traffic assignment.* Highway Research Board, Washington, DC, 1950.

J. R. Correa, A. S. Schulz, and N. E. Stier-Moses. Fast, fair, and efficient flows in networks. *Operations Research*, 55:215–225, 2007.

C. Daganzo and Y. Sheffi. On stochastic models of traffic assignment. *Transportation science*, 1977.

J. de Dios Ortuzar and L. G. Willumsen. *Modelling transport.* John Wiley & Sons, 2011.

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.

A. de Palma and R. Lindsey. Traffic congestion pricing methodologies and technologies. *Transportation Research Part C: Emerging Technologies*, 19, 2011.

D. Eppstein. Finding the k shortest paths. In *Foundations of Computer Science, 1994 Proceedings, 35th Annual Symposium on*, pages 154–165. IEEE, 1994.

J. C. Falcocchio and H. S. Levinson. *Road Traffic Congestion: A Concise Guide.* Springer, 2015.

M. Florian and D. Hearn. Network equilibrium and pricing. In *Handbook of Transportation Science*, pages 361–393. Springer, 1999.

L. R. Ford and D. R. Fulkerson. Maximal flow through a network. *Canadian journal of Mathematics*, pages 399–404, 1956.

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

K. Holmberg and D. Yuan. A multicommodity network-flow problem with side constraints on paths solved by column generation. *INFORMS Journal on Computing*, 15:42–57, 2003.

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.

P. Kachroo and K. Özbay. *Feedback ramp metering in intelligent transportation systems.* Springer Science & Business Media, 2011.

M. Kaspi and J. Tanchoco. Optimal flow path design of unidirectional agv systems. *The International Journal Of Production Research*, 28:1023–1030, 1990.

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.

H. Lin, T. Roughgarden, Éva Tardos, and A. Walkover. Stronger bounds on braess's paradox and the maximum latency of selfish routing. *SIAM Journal on Discrete Mathematics*, 25:1667–1686, 2011.

M. Lujak, S. Giordani, and S. Ossowski. Route guidance: Bridging system and user optimization in traffic assignment. *Neurocomputing*, 151:449–460, 2015.

J. Luo and J.-P. Hubaux. A survey of inter-vehicle communication. Technical report, "Ecole politecnique Fédéral Lausanne", 2004.

H. Mahmassani and S. Peeta. *Network performance under system optimal and user equilibrium dynamic assignments: implications for advanced traveler information systems.* Transportation Research Board, 1993.

H. S. Mahmassani and S. Peeta. System optimal dynamic assignment for electronic route guidance in a congested traffic network. *Urban traffic networks: dynamic flow modeling and control*, 1995.

D. K. Merchant and G. L. Nemhauser. A model and an algorithm for the dynamic traffic assignment problems. *Transportation Science*, 12:183–199, 1978.

J. D. Murchland. Braess's paradox of traffic flow. *Transportation Research*, 4:391–394, 1970.

G. F. Newell. *Traffic flow on transportation networks.* MIT Press, 1980.

M. Papageorgiou. Dynamic modeling, assignment, and route guidance in traffic networks. *Transportation Research Part B: Methodological*, 24:471–495, 1990.

M. Papageorgiou and A. Kotsialos. Freeway ramp metering: An overview. In *Intelligent Transportation Systems, 2000. Proceedings. 2000 IEEE*, pages 228–239. IEEE, 2000.

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.

M. Patriksson. *The traffic assignment problem: models and methods.* Courier Dover Publications, 2015.

S. Peeta and A. Ziliaskopoulos. Foundations of dynamic traffic assignment: The past, the present and the future. *Networks and Spatial Economics*, 1, 2001.

S. Peeta, J. Ramos, and R. Pasupathy. Content of variable message signs and on-line driver behavior. *Transportation Research Record: Journal of the Transportation Research Board*, pages 102–108, 2000.

R. B. Potts and R. Oliver. *Flows in transportation networks.* Academic Press, 1972.

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.

T. Roughgarden and É. Tardos. How bad is selfish routing? *Journal of the ACM (JACM)*, 49, 2002.

P. Ryus, M. Vandehey, L. Elefteriadou, R. Dowling, and B. Ostrom. Highway capacity manual 2010. *TR News*, pages 45–48, 2011.

A. S. Schulz and N. E. Stier-Moses. Efficiency and fairness of system-optimal routing with user constraints. *Networks*, 48, 2006.

S. Sen, R. Pillai, S. Joshi, and A. Rathi. A mean-variance model for route guidance in advanced traveler information systems. *Transportation Science*, 35:37–49, 2001.

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

Y. Sheffi and W. Powell. A comparison of stochastic and deterministic traffic assignment over congested networks. *Transportation Research Part B: Methodological*, pages 53–64, 1981.

M. Sichitiu and M. Kihl. Inter-vehicle communication systems: a survey. *IEEE Communications Surveys & Tutorials*, pages 88–105, 2008.

P. Stopher. Reducing road congestion: a reality check. *Transport Policy*, 11:117–131, 2004.

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.