Abstract—The traffic assignment problem (TAP) plays a key role in the context of efficient urban mobility. The TAP can be approached from various perspectives. One of the fundamental models to solve the TAP is the so-called User Equilibrium (UE), which assumes that drivers behave rationally aiming at minimising their own travel costs. However, this is a complex optimisation problem. To this regard, in this paper we propose the use of the GRASP metaheuristic to provide approximate solutions to the UE. The path re-linking mechanism is also used to increase the coverage of the solutions space. We advance the state-of-the-art by proposing a novel modelling for the TAP, through which one can adjust the granularity of the search space, thus making a more efficient, directed local search. We also devise an efficient assignment evaluation scheme that avoids redundant computations during the local search process. Additionally, we develop a novel greedy procedure for generating enhanced initial solutions for the GRASP algorithm. Based on experiments, we demonstrate that our approach outperforms classical algorithms, providing solutions that are significantly closer to the UE. Moreover, our empirical results show that the stability and fairness levels achieved by our approach are considerably better than those achieved by other methods.

I. INTRODUCTION

Urban mobility plays a key role in modern society. Dealing with the declining efficiency of existing traffic infrastructures, however, has been a major challenge. Over the years, expanding the available capacity has been a common practice. Conversely, such a practice does not necessarily enhance the traffic performance. Indeed, it can even deteriorate the traffic outcome, a phenomenon known as Braess’ paradox [1]. Thus, conceiving ways of efficiently exploiting the existing infrastructure is imperative.

The traffic assignment problem (TAP) is an important step towards managing traffic systems. Specifically, the TAP addresses how to efficiently connecting the traffic infrastructure (supply) and the vehicles1 (demand) that are going to use it. However, such a task is far from trivial. Whereas considering the shortest routes may seem appealing at first, it may lead to huge congestions. Thus, the key point of the TAP is how to spread the traffic into the roads. Precisely, solving the TAP means distributing the vehicles into the road network considering their origins and destinations (a.k.a. OD pairs) to minimise some associated cost (such as travel time or money expenses). Therefore, the TAP is essentially an optimization problem.

1Hereinafter, we use the terms vehicle and driver alternately.

One of the fundamental models to solve the TAP is the so-called user equilibrium (UE). The UE model assumes that rational drivers choose their routes aiming at minimising their own travel costs. Specifically, under UE, each vehicle is assigned to one of the least cost routes of its OD pair. Thus, users have no reason to deviate from the routes to which they were assigned. It is worth noting that, regardless the model being employed, the search space of the TAP is enormous and finding an optimal assignment may be infeasible. To this regard, the use of metaheuristics to approximate the UE has shown a promising alternative.

In a previous paper [2], we developed a metaheuristic to find the UE. In that work, we devised a compact solution representation and a novel performance metric that promotes fairer and stabler assignments. However, the algorithmic apparatus was not efficient enough to address realistic scenarios. Specifically, the problem stemmed from the increased number of redundant evaluations during the local search process.

Against this background, in this paper, we build upon our previous work to deliver a far more efficient approach for the TAP. We employ the Greedy Randomised Adaptive Search Procedure (GRASP) algorithm with the Path Relinking (PR) mechanism. GRASP is a metaheuristic that builds greedy solutions and improves them with local search. The PR mechanism promotes a higher coverage of the search space by keeping track of the most diverse solutions. The solutions are represented and evaluated as in our previous work [2]. However, we advance previous achievements by controlling the granularity of the search space and avoiding redundant computations during the local search process. Based on experiments, we show that our approach provides reasonably stable and fair assignments even in a more realistic setting, outperforming other methods in the literature.

In more detail, our main contributions are:

- We describe a novel greedy generation procedure based on the classical all-or-nothing assignment (defined in Section II). The complexity of this method is a factor d lower than in our previous work [2], with d representing the number of vehicles.
- We propose a more efficient way of evaluating solutions during the local search. Specifically, we show that the value of a neighbouring solution can be obtained by calculating only the differences to the original solution.

The final publication is available at [http://doi.org/10.1109/CEC.2016.7743966](http://doi.org/10.1109/CEC.2016.7743966).
The rest of this paper is structured as follows. Section II formalises the TAP and related concepts. Sections III and IV describe related work and the GRASP+PR algorithm, respectively. We detail our approach in Section V and evaluate it in Section VI. Finally, Section VII presents the concluding remarks and future work directions.

II. TRAFFIC ASSIGNMENT PROBLEM

In this section, we present the basic concepts underlying traffic assignment. We refer the reader to [3] and [4] for a more comprehensive overview.

Let \( G = (N, L) \) denote a road network, where the set \( N \) of nodes represents the intersections, and the set \( L \) of directed links represents the roads between intersections. Each link \( k \in L \) has a cost \( c_k \) associated with crossing it. The demand for trips generates a flow of vehicles. Let \( T_{ij} \) be the demand for trips between origin \( i \in N \) and destination \( j \in N \). The set of all such demands is represented by an origin-destination (OD) matrix \( T = \{ T_{ij} \mid \forall i, j \in N, i \neq j, T_{ij} \geq 0 \} \). The total demand is denoted \( d = \sum_{T_{ij} \in T} T_{ij} \). A trip is made by means of a route \( R \subseteq L \), which is a sequence of links connecting an origin to a destination. The cost of a route \( R \) is given by \( C_R = \sum_{k \in R} c_k \).

The cost of a link is typically modelled using an abstraction called volume-delay function (VDF). A VDF takes as input the flow of vehicles within a link and, based on its attributes (such as length and capacity), returns the cost (travel time) of such link. A simple VDF is presented in Equation (1), with \( t_k \) and \( f_k \) denoting, respectively, the free flow travel time (i.e., minimum travel time, when the link is not congested), and the flow on link \( k \in L \). This particular VDF represents the travel time on link \( k \), which is increased by 0.02 for each vehicle/hour of flow.

\[
c_k(f_k) = t_k + 0.02 \times f_k \tag{1}
\]

The TAP can be solved by means of two fundamental models: System Optimal (SO) and User Equilibrium (UE). A SO assignment describes the system as its best operation. However, its drawback is that some users may have no incentive to actually follow the routes to which they were assigned. On the other hand, the UE represents a traffic condition as stated in the Wardrop’s first principle: “under equilibrium conditions traffic arranges itself in congested networks such that all used routes between an OD pair have equal and minimum costs while all unused routes have greater or equal costs” [5]. Thus, the UE model is a more realistic way of assigning vehicles. Observe that finding the UE is a complex task. Thus, finding approximate solutions has shown a promising direction. Before advancing to this topic, however, we introduce some traditional methods.

One of the simplest traffic assignment methods is known as all-or-nothing. This method generates an assignment by selecting the shortest route for each OD pair, and then assigning the entire corresponding flow to that route. This approach, however, is quite naïve, since assigning all vehicles to the same routes can lead to congestions. Nonetheless, this method has shown useful to developing more advanced methods (as our greedy procedure, to be presented in Section V-B).

Methods that account for congestion effects present better performance. The so-called incremental assignment method, iteratively loads fractions \( p_n \) (with \( \sum_n p_n = 1 \)) of the demand into the lowest cost routes (one for each OD pair). After each iteration, a new set of lowest cost routes is created (using the all-or-nothing assignment) considering the flows accumulated on the previous ones. This method, however, does not have convergence guarantees. A similar approach is the method of successive averages (MSA). At each iteration of the MSA, a fraction \( \psi \) of the current assignment is changed towards the all-or-nothing one (based on current flows). This process is repeated until a given stop criterion is achieved. The MSA can be formalised as in Algorithm 1. It has been shown [6] that the MSA produces solutions convergent to the UE when \( \psi = 1/n \) (with \( n \to \infty \)). However, this method may take many iterations, especially in more complex scenarios, thus being very inefficient.

Algorithm 1: Pseudocode of the MSA (adapted from [3])

1: select an initial set of current link costs (usually free-flow travel times); initialise the flow of every link \( k: f_k = 0 \); set \( n = 0 \);
2: create an auxiliary all-or-nothing assignment based on current link costs, obtaining a set of auxiliary flows \( F^* \);
3: calculate the current flow of every link \( k \) as: \( f_k^n = (1-\psi)f_k^{n-1} + \psi F_k^* \), with \( \psi \in [0,1] \);
4: for every link \( k \), update the cost \( c_k \) based on the flow \( f_k^n \); if the predefined stop criterion was achieved, then stop; otherwise proceed to step 2;

The TAP has been approached from many perspectives. A genetic algorithm is presented in [7] to approximate the SO. Each position of the chromosome corresponds to a vehicle and its value to the route used by the vehicle. The results achieved are reasonable, but their approach concerns only the SO. Moreover, the size of the neighbourhood resulting from the proposed modelling is excessively large.

The TAP is indirectly addressed by means of the toll booth problem (TBP) in [8]. In the TBP, one must select links to charge tolls so that drivers have a monetary incentive to spread over the network. In their approach, the objective is to induce drivers to act on behalf of the social welfare, thus bridging the gap between the UE and the SO. A biased random-key genetic algorithm is used to accomplish such a task. However, their approach depends on a high number of tolls to achieve good results, which is neither practical nor popular.
Varia et al. [9] employ a genetic algorithm to optimise traffic signal settings. Basically, on each iteration of the algorithm, the evolved traffic signal settings are used as input for a modified version of the MSA. The aim is to jointly optimize the traffic signal settings and the TAP. Their approach, however, is more focused on the former problem.

The TAP is approached from a game theoretic point of view in [10] and [11]. The former models the problem as a population game, whereas the latter uses the framework of minority games. In both cases, the strategies are used to infer the best route for each player. The former has shown that the UE may be achieved under certain conditions. However, both approaches were validated in simple scenarios.

A fuzzy-based route choice model was proposed by Henn [12] to tackle the uncertainties regarding the drivers’ decision making. Basically, a fuzzy set is used to maintain predictions on routes costs, considering traffic uncertainties. Based on such fuzzy set, a degree of attractiveness is associated with each route. Their work, however, puts a major focus on tackling the uncertainties than on the assignment optimality.

Dias et al. [13] present an ant colony optimization algorithm for minimising the SO in a decentralised way. In their work, vehicles deposit pheromone into the roads they travel, which repels rather than attracts other vehicles. However, a central authority is required to keep the pheromone information, which is a quite strong requirement for a decentralised system. Moreover, their work does not provide any statement on the optimality of the solutions.

A number of agent-based approaches model reinforcement learning algorithms in the route choice process. Some examples include using learning automata [14] and Q-learning [15]. In these approaches, the drivers create their own set of routes and learn which of them to take. However, such methods do not provide any guarantee on the solutions quality. Furthermore, these approaches were not validated in more realistic scenarios.

As can be seen, most of the approaches presented here do not fully address the UE. Moreover, even those concerning the UE are suitable for small scenarios only. An exception is the work of Buriol et al. [8], which, on the downside, has a strong dependence on a high number of tools. To this regard, we recently devised a compact representation of the TAP to approximate the UE with metaheuristics [2]. We further developed a novel performance evaluation metric able to promote fairer and stabler assignments. Despite of promising results, the search complexity was high. In order to handle this issue, in the present paper, we propose an efficient way of exploring the search space and evaluating the solutions, which leads to even better assignments.

IV. THE GRASP+PR ALGORITHM

A. Greedy Randomised Adaptive Search Procedure

The GRASP metaheuristic was devised by Feo and Resende in [16]. GRASP can be regarded as a multistart algorithm, i.e., iterations are independent among themselves so that the knowledge from previous iterations is not used in subsequent ones. Broadly speaking, at each iteration of GRASP, a solution is greedily built and locally improved. The best solution found along all iterations is returned. In this section, we present the main concepts underlying this method.

In the context of greedy algorithms, a problem is defined as follows. Let \( E = \{e_1, \ldots, e_n\} \) be the ground set, \( F \subseteq 2^E \) be the set of feasible solutions, and \( \phi : 2^E \to \mathbb{R} \) be an objective function. A solution \( S \in F \) can be represented as a vector over the ground set \( E \). As a minimisation problem, the objective here is to find an optimal solution \( S^* = \arg\min_{S \in F} \phi(S) \). Note that the definitions of \( E, \phi, \) and \( F \) are domain-specific.

In the case of the TAP, a possible representation could be \( E \) as the set of drivers, \( \phi \) as the average travel cost, and \( F \) as the set of possible allocations of all drivers into routes.

In greedy algorithms, solutions are iteratively built from scratch. At each iteration, an element of the ground set is moved to the solution, until a feasible\(^2\) solution is obtained. Typically, a greedy algorithm chooses the element that increases the solution’s cost the least. However, such a criterion can be relaxed to better explore the search space. A common strategy here is known as \( \alpha \)-greedy, in which one of the \( \alpha \% \) best elements is chosen uniformly at random.

The greedily generated solutions are not necessarily optimal. To this regard, the GRASP algorithm employs a local search procedure to improve them. Let a single modification in a solution be called a movement. A solution \( S' \) resulting from a movement in \( S \) is called its neighbour. The set of \( S \)'s neighbours \( N(S) \subseteq F \) is called its neighbourhood. We refer to solution \( S \) as a local optimum if \( \phi(S) \leq \min_{S' \in N(S)} \phi(S') \). Observe that a local optimum can be, but not necessarily is a global optimum. Along these lines, a local search mechanism works by successively applying single, small changes in the solution until a (local) optimum is found. A common strategy to guide a local search is the so-called best improvement, in which a better neighbour \( B(S) = \{S' \in N(S) \mid \phi(S') < \phi(S)\} \) is selected uniformly at random.

B. Path Relinking

The path relinking (PR) mechanism was firstly proposed by Glover [17] within the context of tabu search. In general terms, PR explores the directed neighbourhood from an initial to a target solution, returning the best solution found along the trajectory.

Some additional notation is necessary. Consider an initial solution \( S_s \) and a target one \( S_t \). Let \( \Delta(S_s, S_t) \) be the symmetric difference between two solutions, i.e., the number of movements required to transform one into the other. The PR mechanism works by exploring the directed neighbourhood \( D(S_s) = \{S' \in N(S_s) \mid \Delta(S', S_s) < \Delta(S_s, S_t)\} \) of the initial solution (and the successive ones), typically using a best improvement strategy, until the target solution is reached. In this sense, the search takes at most \( \Delta(S_s, S_t) \) iterations to

\(^2\)Infeasible solutions may also be produced. Our approach, however, prevent them to occur (as shown in Section V). Thus, we omit such concept to simplify the notation.
reach \( S_i \) from \( S_n \), and the best solution found along such a traverse is returned.

The use of PR within GRASP was introduced by Laguna and Martí [18]. In GRASP+PR, a set \( P \) of reference solutions is maintained to store the best and most diverse solutions found during GRASP iterations. This can be seen as a kind of memory, where information acquired in previous iterations may lead to an improvement in the subsequent ones.

The PR can be used in two ways [19]: as an intensification step of incumbent solutions or as a post-optimisation mechanism. In the former, PR is applied between the incumbent solution \( S \) (after the local search) and a reference solution \( S' \in P \). Usually, a reference solution is chosen at random with probability proportional to \( \Delta(S, S') \). This strategy gives priority to longer traverses, thus promoting a higher coverage of the search space. The PR can also be used as a post-optimisation step, after GRASP iterations have finished. In this case, PR is applied to every pair of reference solutions, followed by a local search. This additional step tends to improve the algorithm’s outcome, as only the best and most diverse solutions are explored.

V. TRAFFIC ASSIGNMENT WITH GRASP+PR

In this section, we present the main aspects of our approach. A general overview of our GRASP+PR approach is presented in Algorithm 2.

A. Solution Encoding and Evaluation

A solution to the TAP corresponds to an assignment of every vehicle into a route from its origin to its destination. Recall that \( T_{ij} \subseteq T \) represents the demand for trips between origin \( i \) and destination \( j \), \( d = \sum_{T_{ij} \in T} T_{ij} \) is the total demand (number of vehicles), and \( T \) is called the OD matrix. Each trip is made through a route \( R \subseteq L \). Note that an assignment does not necessarily uses all possible routes. In fact, limiting the set of routes provides a reasonable approximation for the problem [20]. To this regard, we limit the number of routes per OD pair to the \( k \) shortest ones. Let \( R_{ij} = \{ R_{ij}^1, \ldots, R_{ij}^k \} \) be the set of \( k \) shortest routes between an origin \( i \) and a destination \( j \). These sets are computed considering free flow travel times, which can be performed by the KSP algorithm [21].

We represent a solution \( S \) as a matrix of size \( |T| \times k \), where rows represent the OD pairs, and columns represent the routes available for these OD pairs. To be precise, the value stored on position \( S[ij][r] \) represents the amount of vehicles from \( T_{ij} \) assigned to route \( R_{ij}^r \). We introduced such a compact representation in [2], and it is much more efficient than representing each vehicle individually. Moreover, this representation avoids redundant movements (as shown ahead, in Section V-C).

The solution evaluation is defined in terms of the UE. To this end, we employ the metric defined in [2], which measures the number of vehicles assigned to non-least cost routes. We remark that, under UE, all vehicles are assigned to their least cost routes, thus having no incentive to deviate from such assignment. In this sense, this metric measures how many drivers have an incentive to deviate from the assignment. Precisely, the value \( \phi : S \rightarrow \mathbb{N} \) of solution \( S \) is given by Equation (2), where \( R_{ij}^* = \min_{R_{ij} \in R_{ij}} C(R) \) represents one3 of the least cost routes of the OD pair \( ij \) under assignment \( S \). In this function, an UE assignment has value 0 and the farther the solution is from the UE, the higher its value. As shown in [2], minimising \( \phi \) lead to fairer assignments. Thus, our objective is to minimise \( \phi \).

\[
\phi(S) = \sum_{T_{ij} \in T} \sum_{R_{ij} \in R_{ij}} \begin{cases} S[ij][r] & \text{if } C(R_{ij}^r) > C(R_{ij}^*) \\ 0 & \text{otherwise} \end{cases}
\]

(2)

In terms of complexity, the evaluation of a given solution involves the computation of (i) the flow on each link, (ii) the cost of each link, (iii) the cost of each route, and (iv) the value of \( \phi \). This process can be performed in \( O(|T| \times k \times l) \), and is repeated for every solution. When a solution is being built from scratch, this process is unavoidable. However, when a solution is being generated from another (e.g., during the local search, as shown ahead), this process can be simplified by calculating only the corresponding differences. Specifically, we can recalculate only the portions corresponding to the changed flows in \( O(k \times (|T| + l)) \) time, which is much more efficient. For the sake of comparison, considering the Sioux Falls instance (to be presented in Section VI), recalculating the value from the differences is two orders of magnitude faster than calculating the value from scratch. We omit the proofs due to the lack of space.

**Proposition 1:** The solution evaluation process can be performed in \( O(|T| \times k \times l) \), if the solution was built from scratch, or in \( O(k \times (|T| + l)) \), if the solution was generated from another, by computing only their differences.

between the shortest routes between an origin \( s \) and the best solution found along such a
traverse is returned.

The use of PR within GRASP was introduced by Laguna and Martí [18]. In GRASP+PR, a set \( P \) of reference solutions is maintained to store the best and most diverse solutions found during GRASP iterations. This can be seen as a kind of memory, where information acquired in previous iterations may lead to an improvement in the subsequent ones.

The PR can be used in two ways [19]: as an intensification step of incumbent solutions or as a post-optimisation mechanism. In the former, PR is applied between the incumbent solution \( S \) (after the local search) and a reference solution \( S' \in P \). Usually, a reference solution is chosen at random with probability proportional to \( \Delta(S, S') \). This strategy gives priority to longer traverses, thus promoting a higher coverage of the search space. The PR can also be used as a post-optimisation step, after GRASP iterations have finished. In this case, PR is applied to every pair of reference solutions, followed by a local search. This additional step tends to improve the algorithm’s outcome, as only the best and most diverse solutions are explored.

V. TRAFFIC ASSIGNMENT WITH GRASP+PR

In this section, we present the main aspects of our approach. A general overview of our GRASP+PR approach is presented in Algorithm 2.

A. Solution Encoding and Evaluation

A solution to the TAP corresponds to an assignment of every vehicle into a route from its origin to its destination. Recall that \( T_{ij} \subseteq T \) represents the demand for trips between origin \( i \) and destination \( j \), \( d = \sum_{T_{ij} \in T} T_{ij} \) is the total demand (number of vehicles), and \( T \) is called the OD matrix. Each trip is made through a route \( R \subseteq L \). Note that an assignment does not necessarily uses all possible routes. In fact, limiting the set of routes provides a reasonable approximation for the problem [20]. To this regard, we limit the number of routes per OD pair to the \( k \) shortest ones. Let \( R_{ij} = \{ R_{ij}^1, \ldots, R_{ij}^k \} \) be the set of \( k \) shortest routes between an origin \( i \) and a destination \( j \). These sets are computed considering free flow travel times, which can be performed by the KSP algorithm [21].

We represent a solution \( S \) as a matrix of size \( |T| \times k \), where rows represent the OD pairs, and columns represent the routes available for these OD pairs. To be precise, the value stored on position \( S[ij][r] \) represents the amount of vehicles from \( T_{ij} \) assigned to route \( R_{ij}^r \). We introduced such a compact representation in [2], and it is much more efficient than representing each vehicle individually. Moreover, this representation avoids redundant movements (as shown ahead, in Section V-C).

The solution evaluation is defined in terms of the UE. To this end, we employ the metric defined in [2], which measures the number of vehicles assigned to non-least cost routes. We remark that, under UE, all vehicles are assigned to their least cost routes, thus having no incentive to deviate from such assignment. In this sense, this metric measures how many drivers have an incentive to deviate from the assignment. Precisely, the value \( \phi : S \rightarrow \mathbb{N} \) of solution \( S \) is given by Equation (2), where \( R_{ij}^* = \min_{R_{ij} \in R_{ij}} C(R) \) represents one3 of the least cost routes of the OD pair \( ij \) under assignment \( S \). In this function, an UE assignment has value 0 and the farther the solution is from the UE, the higher its value. As shown in [2], minimising \( \phi \) leads to fairer assignments. Thus, our objective is to minimise \( \phi \).

\[
\phi(S) = \sum_{T_{ij} \in T} \sum_{R_{ij} \in R_{ij}} \begin{cases} S[ij][r] & \text{if } C(R_{ij}^r) > C(R_{ij}^*) \\ 0 & \text{otherwise} \end{cases}
\]

(2)

In terms of complexity, the evaluation of a given solution involves the computation of (i) the flow on each link, (ii) the cost of each link, (iii) the cost of each route, and (iv) the value of \( \phi \). This process can be performed in \( O(|T| \times k \times l) \), and is repeated for every solution. When a solution is being built from scratch, this process is unavoidable. However, when a solution is being generated from another (e.g., during the local search, as shown ahead), this process can be simplified by calculating only the corresponding differences. Specifically, we can recalculate only the portions corresponding to the changed flows in \( O(k \times (|T| + l)) \) time, which is much more efficient. For the sake of comparison, considering the Sioux Falls instance (to be presented in Section VI), recalculating the value from the differences is two orders of magnitude faster than calculating the value from scratch. We omit the proofs due to the lack of space.

**Proposition 1:** The solution evaluation process can be performed in \( O(|T| \times k \times l) \), if the solution was built from scratch, or in \( O(k \times (|T| + l)) \), if the solution was generated from another, by computing only their differences.

1Multiple routes may share the least cost. However, such a distinction is not relevant for the purpose of calculating Eq. (2).
B. Solution Generation

In GRASP, solutions are generated in a greedy, randomised way. However, to prevent any bias in the generation process, we also allow the generation of uniform solutions. These two procedures are used alternately, according to parameter $\gamma$: a solution is greedily generated with probability $(1 - \gamma)$, or uniformly generated with probability $\gamma$. Both procedures are described next.

The greedy procedure is inspired in the all-or-nothing assignment. Each element $E_{ij}$ in the ground set $E$ corresponds to an OD pair $ij$, and its cost corresponds to the amount that $\phi(S)$ would increase if the its whole flow $T_{ij}$ were assigned to $R_{ij}^* = \min_{R \in R_{ij}} C(R)$. The procedure starts with an empty solution $S$. On each iteration, the cost of every element in $E$ is updated to account for the flow allocated in previous iterations. Right after, an element $E_{ij}$ is selected using the $\alpha$-greedy strategy and its whole flow $T_{ij}$ is assigned to $R_{ij}^*$. This process is repeated until the ground set is empty, thus ensuring that every vehicle was assigned to exactly one route. This procedure can be performed in $O(|T|^2 \times k)$, which is much faster than the $O(|T|^2 \times k \times d)$ of the greedy procedure presented in [2] (proofs are omitted due to the lack of space).

The uniform generation procedure is more simple. The process also starts with an empty solution $S$. Consider an OD pair $ij$, its demand $T_{ij}$, and the corresponding set of routes $R_{ij}$. Let $U_{ij} = \{0,u_2,\ldots,u_k,T_{ij}\}$ be a sorted set, where $u_2,\ldots,u_k$ correspond to $k - 1$ natural numbers in the interval $[0,T_{ij}]$, generated uniformly at random. The solution is defined as $S_{ij}[r] = U_{ij}[r + 1] - U_{ij}[r]$ for every route $r$ of OD pair $ij$. This process is performed for every OD pair. The result is a feasible assignment, with demands distributed uniformly at random among the available routes.

It is worth noting that both procedures generate only feasible solutions.

Proposition 2: The greedy and uniform procedures generate only feasible solutions.

Proof sketch: An assignment would be infeasible if and only if any vehicle were not assigned to a route between its OD pair, i.e., $\exists T_{ij} \in T : \sum_{r=1}^{k} S_{ij}[r] \neq T_{ij}$.

For the greedy procedure, each element in the ground set corresponds to an OD pair. Each element is selected exactly once (because, when selected, they are removed from the ground set and the algorithm continues until the ground set is empty), occasion in which the whole corresponding flow is assigned to a single route. This means that, if the ground set is empty, then the entire flow has been assigned.

For the uniform procedure, the flow of each OD pair is divided into $k$ portions, each assigned to a single route. Thus, no vehicle remains unassigned.

C. Neighbourhood

We now define the neighbourhood of a given solution $S$. Recall that a neighbour is obtained by performing a single movement in the solution. A movement here is characterised by moving a single vehicle from any non-least cost route $R_{ij}^c \in \{R_{ij}^c \in R_{ij} \mid C(R_{ij}^c) > C(R_{ij})\}$ to any least cost one $R_{ij}^*$. Recall that, by definition, a VDF does not distinguish vehicles in particular. Thus, when applying a movement, no distinction is made on which vehicle is being moved. This means that at most $(k/2)^2$ movements can be performed for each OD pair. Consequently, the number of neighbours is $O(|T|^2 \times k^2)$ (we omit the proof due to the space limitation).

The TAP instances usually have a high number of OD pairs. To this regard, searching the entire neighbourhood may be impractical. In order to overcome this problem, we reformulate the previous definition to accommodate a variable neighbourhood size. Specifically, a movement is characterised by moving a portion $p \in \mathbb{N}$ of the flow from any non-least cost route to any least cost one from the same OD pair. More formally, the $p$-neighbourhood of solution $S$ is defined as $N(S,p) = \{S' \in F \mid \exists i,j \in N, s \in [0,k), i \neq j, r \neq s, 0 < p \leq S_{ij}[r], S_{ij}[r] = S_{ij}[r] - p \land S_{ij}[s] = S_{ij}[s] + p\}$. Observe that $p$ must be at least 1, otherwise the neighbourhood would be empty. In this sense, we are able to control the granularity of the search space by a factor of $p$, thus making the search much faster. This is a generalisation of the previous definition, which could be considered the 1-neighbourhood.

Two interesting properties of the present modelling must be highlighted.

Proposition 3: The $p$-neighbourhood definition avoids infeasible solutions, i.e., $N(S,p) \subseteq F$ for $p > 0$.

Proof sketch: According to Proposition 2, only feasible solutions are generated by our approach. To be infeasible, a solution would have either a non-natural number or a total demand different from that of the OD matrix. However, by definition, a movement consists in moving a portion $p \in \mathbb{N}$ of the flow between two routes of the same OD pair. Thus the demand itself does not change. Moreover, by definition, routes with flow smaller than $p$ are not considered.

Proposition 4: The $p$-neighbourhood definition avoids redundant solutions, i.e., $\exists S', S'' \in N(S,p) : S' = S''$.

Proof sketch: Since $N(S,p)$ is a set, by definition it has no duplicate elements. Moreover, when building $N(S,p)$, each pair of least vs. non-least cost route is considered only once. Consequently, no equal elements can exist in $N(S,p)$.

The definition would otherwise allow redundant neighbours if the movement of each vehicle from route $R_{ij}^c$ to route $R_{ij}^*$ were considered in particular. In this case, such neighbours would be redundant because, from the solution definition perspective, they would be exactly the same solution.

D. Local Search

The local search procedure is applied to every incumbent solution to reach its local optimum. Basically, given a solution $S$, its best neighbourhood $B(S,p) = \{S' \in N(S,p) \mid \phi(S') < \phi(S)\}$ is successively explored using the best improvement strategy, until a local optimum is reached. As seen in previous subsection, the neighbourhood size is $O(|T|^2 \times k^2)$. Regarding the number of steps, in the worst case, all vehicles are assigned
to non-least cost routes. Thus, the number of steps of the local search is $O(|T| \times k^d \times |d|)$.

In this sense, we employ the concept of variable neighbourhood size to control the granularity of the search space, and so the complexity of the local search. We call this mechanism multistep local search (MLS). Let $M = \{x \in \mathbb{N} \mid x > 1\} \cup \{1\}$, with $M[i] > M[i+1]$, be the set of step sizes to be performed in the local search. Basically, given a solution $S$, the MLS applies a local search in the $M[0]$-neighbourhood of $S$ until a local optimum $S^+$ is found. At this point, the MLS starts a second local search, but in the $M[1]$-neighbourhood of $S^+$, until a local optimum is found. This process is repeated for each element in $M$.

To better understand how the MLS works, consider the following example. Suppose we must move a portion of 99 out of 100 vehicles from a route to another. Using the 1-neighbourhood, this process would take 99 steps to finish. On the other hand, using MLS with $M = \{10, 1\}$, we would need 9 steps in the 10-neighbourhood and 9 additional steps in the 1-neighbourhood, resulting in 18 steps. Note that, although the number of steps performed by each approach varies a lot, the local optimum is the same (or at least with the same value).

E. Path Relinking

In this work, we use path relinking both as an intensification and as a post-optimisation step. For intensification, path relinking is applied to every incumbent solution after the local search procedure. To this end, a reference solution $S' \in P$ is drawn from the reference set with probability proportional to $\Delta(S, S')$. Once $S'$ is selected, the search is started from the best solution towards the worst one, using the best improvement strategy. The best solution found along the trajectory is locally improved and then it becomes the incumbent solution. After the GRASP iterations finish, the post-optimisation step is started. Basically, it applies path relinking between every pair of reference solutions, followed by a local search step. At the end, the best solution found is returned.

Concerning the reference set $P$, its size is limited to keep at most $\beta$ solutions. Initially, $P$ is populated with $\beta/2$ uniformly generated solutions. Next, at the end of each GRASP iteration, the incumbent solution (after path relinking) is considered to be included in $P$. Specifically, an incumbent solution $S$ replaces another in $P$ if its cost is better than the worst in $P$ (i.e., if $\phi(S) < \max_{S' \in P} \phi(S')$) and it is not closer to any other solution in $P$ than the closest pair of solutions in $P$ (i.e., $\min_{S' \in P} \Delta(S, S') > \min_{S' \in P, S'' \in P \setminus S} \Delta(S', S'')$). In this case, $S$ replaces the solution that is closest to him $S' = \arg \min_{S' \in P} \Delta(S, S')$. This mechanism prioritises the diversity of reference solutions, thus promoting a higher coverage of the search space.

VI. EMPIRICAL EVALUATION

A. Experimental Setup

In order to evaluate our approach, we employ two road network instances available in the literature: OW [3] and Sioux Falls [22]. The former is a small, synthetic network comprising 13 nodes, 48 links, and 1700 vehicles distributed among four OD pairs. In the OW instance, the cost of the links is calculated using Equation (1). The latter instance is an abstraction of the road network of Sioux Falls, SD, USA. This instance comprises 24 nodes, 76 links, and 360,600 vehicles distributed among 528 OD pairs. For the Sioux Falls instance, the cost of link $l_k$ is computed using Equation (3) [23], where $t_k$ is its free flow travel time, $f_k$ is the flow of vehicles on it, $X_k$ is its capacity, and $a$ and $b$ are parameters of the instance. All the values and parameters used in Equations (1) and (3) are provided with their corresponding instances.

$$c_k(f_k) = t_k \times \left(1 + \frac{a}{b} \frac{f_k}{q_k}\right)$$

(3)

The performance of solutions is measured by means of Equation (2). We also employ the performance metric called Average Excess Cost (AEC), as in Equation (4) (following the notation introduced in Section V-A), which accounts for the relative cost of travel times of non-least cost routes [3]. Hereafter, we refer to equations (2) and (4) as $\phi$ and AEC, respectively. Finally, the average travel time (avg-$tt$ hereafter, measured in minutes) is employed as a third performance metric. For the three metrics, the lower the resulting value, the better. Precisely, under UE, we have $\phi = \text{AEC} = 0$.

$$\delta(S) = \frac{\sum_{r_{ij} \in T} \sum_{r'_{ij} \in r_{ij}} S[|ij|][r] \times (C(r'_{ij}) - C(r_{ij}))}{\sum_{r_{ij} \in T} T_{ij}}$$

(4)

The experiments were performed as follows. The set of shortest routes is generated only once, using the KSP algorithm [21]. Regarding the algorithm’s parameters $k$, $\alpha$, $\beta$, and $\gamma$, we tested different combinations of values for them. For each such combination, 30 repetitions were performed, where a repetition corresponds to a single execution of the algorithm with 1000 iterations. The best configurations found were $k = 4$, $\alpha = 0.2$, $\beta = 20$, and $\gamma = 0.2$ for the OW instance, and $k = 4$, $\alpha = 0.01$, $\beta = 20$, and $\gamma = 0.2$ for the Sioux Falls instance. Observe that the value for $\alpha$ varies from one instance to another. This is because the $\alpha$-greedy strategy depends on the number of OD pairs. Thus, a lower value for $\alpha$ avoids an excessive relaxation when generating solutions for the Sioux Falls instance. Regarding the multistep local search, we defined $M = \{1\}$ for the OW instance because it is a small instance. For the Sioux Falls instance, we used $M = \{100, 10, 1\}$ because its demand is two orders of magnitude higher than in OW. The results of such configurations were selected for further analysis in the next subsection.

Our approach is compared against the MSA (introduced in Section II, with $\psi = 1/n$) and our previous work [2] (RB, hereinafter).

B. Results and Discussion

The performance of our approach throughout iterations is shown in Figure 1a, for the OW instance, and Figure 1b, for the Sioux Falls instance. In the plots, we present the average value of $\phi$ over 30 repetitions and the corresponding
standard deviation. Recall that an incumbent solution is the one generated in the current iteration of the algorithm, whereas the best solution so far is the one that shall be returned at the end of such algorithm. As can be seen, the average value of incumbent solutions varies considerably in both instances. This happens because of the independent, randomised nature of the solutions generation process. Nevertheless, it is clear that the average value of both incumbent and best so far solutions decrease with time. This is due to the path relinking mechanism, which uses the reference set as a kind of memory. To this regard, knowledge from previous iterations can be exploited to improve the subsequent incumbent solutions.

The results of our approach in comparison with MSA and RB [2] are presented in Table I. These results are presented for the OW and Sioux Falls instances, considering the metrics $\phi$, AEC, and avg-$tt$. Observe that RB runs out of memory in the very first iteration for the Sioux Falls instance. Except for the MSA, these results represent the average over 30 repetitions. Standard deviations are shown in parenthesis. We highlight the best results in bold to enhance the results presentation.

It can be observed that the solutions provided by our GRASP+PR approach are, on average, considerably closer to the UE than those provided by the other methods. In the case of the MSA, such a difference is a consequence of the problem formulation, which aims at minimising the number of vehicles assigned to non-least cost routes rather than their average travel times. In comparison with RB, however, such a difference can be explained by the greedy generation procedure and the path relinking mechanism. The greedy all-or-nothing-based generation procedure provides better initial solutions than those of RB. Consequently, it contributes to a more directed search. Regarding the path relinking mechanism, recall that, in this work, it is also used as a post-optimisation step. In the simulations, such a mechanism improved the solutions in 47.6%, on average, for the OW instance and 7.5% for the Sioux Falls one. Moreover, in this work, the reference set update mechanism was enhanced with a more selective update criterion, which increases the diversity of reference solutions.

To better understand the implication of these results, consider the average number of vehicles that would have a real incentive to deviate from the routes to which they were assigned. Using the methods proposed in this work, we achieve a fraction of only 1.2% of the drivers having such an incentive. In contrast, this number is of 5.1% (using RB) and 31.6% (using the MSA). Likewise, in the case of the Sioux Falls instance, our approach achieves a fraction of 3.2% of the drivers, which is still much better than the 8.4% achieved by the MSA. The direct consequence of these results is that our assignments are not only fairer than those provided by the other methods, but also more stable.

Regarding the avg-$tt$ metric, the MSA achieved better results than our approach. However, such a result was already expected, since the minimisation objective of the MSA is precisely the avg-$tt$. Moreover, observe that our approach also achieved good results. In fact, our results were less than 3% and 12% worse than the MSA in the OW and Sioux Falls instances, respectively. In contrast, regarding the $\phi$ metric, our results were 96.2% and 62% better than the MSA for the OW and Sioux Falls instances, respectively.

Finally, considering the AEC metric, it can be observed that our approach performed better than the MSA on the OW instance, whereas the MSA performed better in the Sioux Falls instance. Again, this is a consequence of the optimisation objective of the MSA. Recall that the AEC metric is also based on vehicles’ travel times. In fact, the advantage of the MSA when it comes to travel times can be explained by its structure: it works by iteratively changing the route

![Graph](image-url)

**Fig. 1.** Average performance of our approach throughout iterations for the OW and Sioux Falls instances. Shaded lines represent the standard deviation.
of a dwindling portion of the demand in order to reduce the differences among the vehicles’ travel times. However, from the UE definition, minimising travel time per se is not the main objective. In fact, this is the focus of the system optimal (SO) model. Notwithstanding, our approach performed much better than the MSA in the OW instance.

To summarise, the results achieved by our approach demonstrate its efficiency, fairness, and robustness to approximate the UE of the TAP. Despite the heuristic nature of our approach, it became evident that our solutions are fairer and closer to the UE than those provided by the MSA and RB. Naturally, our results are specific to the instances employed here and may not hold for specific road networks. However, this limitation does not invalidate the generality of our results, since the OW and Sioux Falls instances have the same characteristics of other instances present in the literature [22], such as a high demand, multiple OD pairs, routes overlapping, and congestions.

VII. CONCLUSIONS

Solving the traffic assignment problem (TAP) is an important task towards an efficient usage of the traffic infrastructure. In this paper, we presented the use of the GRASP+PR metaheuristic to approximate the user equilibrium (UE) of the TAP. We advanced the state-of-the-art by devising an efficient modelling of the problem, through which one can adjust the granularity of the search space and avoid redundant computations during the local search. We empirically show that our approach generates reasonably stable and fair assignments even in more realistic settings, outperforming other methods of the literature in such aspects.

Although satisfactory results were achieved, there is still space for improvements. In the experimental evaluation, we have presented three different metrics to assess the performance of our approach, but only one of them is used as our minimisation objective. To this regard, it would be interesting to employ a multi-objective method to optimise more than one of such functions at the same time. Additionally, it would be interesting to investigate some theoretical aspects of the generated assignments, like the price of anarchy [24]. Improving the process of generating the set of routes would also be useful. Some ideas in this direction include: allowing a variable number of routes per OD pair (or even per vehicle), and updating the set of routes periodically based on costs from previous iterations. Minor improvements include investigating other strategies for neighbourhood exploration, solutions construction, and path relinking.

VIII. ACKNOWLEDGMENTS

We thank the reviewers for their comments. The authors are partially supported by CAPES and CNPq grants.

REFERENCES