An advanced GA-VNS combination for multicriteria route planning in public transit networks

Omar Dib, Laurent Moalic, M Manier, A Caminada

An advanced GA-VNS combination for multicriteria route planning in public transit networks

O. Dib[1], L. Moalic[2], M. Manier[3] and A. Caminada[4]

1. IRT SystemX, 8 Avenue de la Vauve, 91120 Palaiseau France
omar.dib@irt-systemx.fr

2. OPERA – UTBM, Belfort France
{omar.dib ; laurent.moalic ; marie-ange.manier; alexandre.caminada}@utbm.fr

Abstract: Nowadays, passengers in urban public transport systems do not only seek a short-time travel, but they also ask for optimizing other criteria such as cost and effort. Therefore, an efficient routing system should incsorporate a multiobjective analysis into its search process. Several algorithms have been proposed to optimally compute the set of nondominated journeys while going from one place to another such as the generalisation of the algorithm of Dijkstra. However, such approaches become less performant or even inapplicable when the size of the network becomes very large or when the number of criteria considered is very important. Therefore, we propose in this paper an advanced heuristic approach whereby a Genetic Algorithm (GA) is combined with a Variable Neighbourhood Search (VNS) to solve the Multicriteria Shortest Path Problem (MSPP) in multimodal networks. As transportation modes, we focus on railway, bus, tram and pedestrian. As optimization criteria, we consider travel time, monetary cost, number of transfers and the total walking time. The proposed approach is compared with the exact algorithm of Dijkstra, as well as, with a standard GA and a pure VNS. Experimental results have been assessed by solving real life itinerary problems defined on the transport network of the city of Paris and its suburbs. Results indicate that the proposed combination GA-VNS represents the best approach in terms of computational time and solutions quality for a real world routing system.

Keywords: Multimodal networks; multicriteria analysis, genetic algorithms; variable neighbourhood search; hybrid metaheuristic; modelling & solving; real-world application.

1 Introduction

With the dawn of history, people had the "walking" as their only transport mode. Hence, the transportation system consisted of networks and routes only. However, with people discovering the possibility to use animals as a mean to travel or transfer goods from one place to another, a new mode of transport has been added to the whole system. Later on, different types of vehicles were added simultaneously. Thus, we started talking about a "Multimodal Transportation System (MTS)" in which nodes and terminals have been added to the components of the initial transportation system.

The transport network always comprises a set of routes where a route represents a single path between two points. Nodes and terminals are the contact or exchange points where it is possible for people to change from one mode to another.

In fact, a MTS, also called combined transport system, can be also seen as a combination of all traveller modes and kinds of transportation systems operated through various systems (SteadieSeifi et al. 2014). In another term, it is a set of choices of transport modes that travellers can use simultaneously according to their needs and preferences in order to reach their destinations.

The MTS originally gained its importance from the fact that combining several modes of transport into one large system would maximize users' profits. Since each mode has both its pros and cons, the system tends to combine the advantages of several modes in an attempt to overcome the drawbacks of each mode. For instance, a bike's capacity in terms of travelled distance remains limited due to the limited speed and the associated physical effort. However, it is cheap, environmentally friendly, and can take its rider even through narrow roads. On the other hand, public transport such as a bus or a metro can be used for long-distance journeys, but at the same time, its direction is scheduled and its stations might be somehow distant.

Furthermore, multimodal transport can save passengers' time. For example, to go from one place to another, a passenger can only take a bus but the journey's travel time will be one hour. However, by combining several transportation modes, the passenger can take one bus to the subway station, then a metro to reach his/her destination; the trip will finally take half an hour.

Nowadays, according to (Liu 2011) the human mobility within urban areas always happens in a multimodal transportation network. People are more prone to use more than one mode of transport to reach their destinations. However, the transport system has become more and more complex; the number of passengers is increasing and new modes of transportation and infrastructures enter the system day after day.

As a result, the multi-modal transportation system may not always be user-friendly. Users usually find themselves more confused with having several possibilities to go from one place to another. Consequently, for the sake of helping people to efficiently find their best routes through the complex transportation scheme, route planning in multimodal transport system is gaining more and more importance.

As in real life, commuters do not only seek short time travels. However, they tend to consider other elements into their journeys such as monetary cost, comfort (quality of mode) and effort (walking distance, number of transfers, waiting time…). Therefore, there is a real need to develop a seamless routing application that provides passengers with efficient itineraries according to their needs and preferences.

Routing applications whether they arise in transportation area or other domains such as communication networks refer for solving Shortest Path Problems (SPPs). While solving some routing problems can be done in a straightforward manner, computing shortest paths under certain circumstances is not always an easy task. For instance, solving the one-to-one SPP in static networks can be easily accomplished by applying the well-known algorithm of Dijkstra. On the other side, computing multicriteria shortest paths appears to be more difficult especially in large-scale dynamic networks.

Computing itineraries with respect to (w.r.t) several criteria refers to the Multiobjective Shortest Path Problem (MSPP), a fundamental problem in the field of multiobjective optimization. Solving the emerging problem consists of finding the set of non-dominated journeys from which the user chooses his/her most preferred one. As the Pareto dominance concept, given two journeys j_1 and j_2, it is said that j_1

dominates j_2 if there is at least one criterion for which j_1 has a better value than j_2 and there is no criterion for which j_2 has a better value than j_1. A journey j is then called Pareto-optimal if it is not dominated by any other journey.

The main difficulty in multiobjective contexts stems from the fact that, in many optimization problems, determining the entire set of nondominated solutions is a tedious task since one problem may have a huge number of nondominated solutions (even in case of two objectives). Additionally, and in contrast to single criteria search, one cannot abort the search after finding a first optimal solution. Indeed, even after finding all Pareto-optima, search algorithms require a substantial amount of time to guarantee that no further solutions exist. Therefore, in many optimization problems, especially those requiring real time answers, the main focus is not on finding the optimal Pareto solution set. Rather, most approaches lies in using heuristics methods whereby near optimal solutions are computed in reasonable computational time.

The main aim of this paper is to propose a new approach for solving the MSPP. A new combination process is introduced in which a Variable Neighbourhood Search is coupled with a Genetic Algorithm in order to solve the MSSP in multimodal networks. As transportation modes, the following networks are used: railway, metro, tram, bus and pedestrian mean. As criteria, the total travel time, monetary cost, number of transfers, and walking time are considered. Assessing the proposed approach has been done by solving a wide range of real life itinerary planning problems based on the real data of the French region Il-de-France that includes the city of Paris and its suburbs. The introduced approach has been also compared with an exact approach based on the algorithm of Dijkstra, which computes the whole set of Pareto solutions. Comparison is performed regarding the quality of solutions found and the computational time.

The main motivation behind using metaheuristics instead of other methods stems from their ability to provide near optimal solutions in reasonable computational time. Meta-heuristics also offer high performance and flexibilities to support several categories of optimization (single criteria, multi-criteria) whether in static, dynamic or even stochastic graphs.

Among several metaheuristics GAs have been chosen since they are adaptive heuristic search algorithms, which are premised on the evolutionary ideas in natural selection and gene types. Additionally, they were essentially designed to simulate process in natural system necessary for evolution, specifically those that follow the principles of survival of the fittest. Therefore, and in contrast to traditional search techniques, GAs are capable of avoiding randomness search by intelligently exploiting historical information that guide the search process towards regions of better performance within the solution space.

Since GAs are primarily based on blind operators (crossover and mutation), the search process may converge prematurely. Therefore, to enhance the efficiency of the method, we have chosen to combine it with another metaheuristic. That is, the drawbacks of the traditional GAs are overcomed by applying another metaheuristic inside its search procedure. To do so, the single solution metaheuristic VNS is chosen.

Selecting VNS originated from the fact that it can explore distant neighborhoods of the current incumbent solution. VNS can also move from one solution to another by repeatedly applying local search improvements in order to reach local optima. Once a local optimum is detected, VNS is able to jump out of it and eventually find better solutions by dynamically changing the neighborhood‘s structures. Therefore, VNS is more likely to prevent the optimization process from rapidly falling into local optima.

Unlike traditional single-search based meta-heuristics such as Tabu Search (TS) and Simulated Annealing (SA), VNS algorithm’s structure is very simple and does not require tuning many parameters. For instance, in TS, one have to tune several parameters that will largely affect the performance of the method such as the tabu object , the tabu list, tabu length and candidate solution, aspiration criterion, tabu frequency, stopping Criterion. Moreover, in SA, many parameters have also to be tuned in order to achieve the best performance such as the state generated function, the state accepted function, the temperature update function, the inner loop termination criterion, the outer loop termination criterion, the initial temperature etc. In contrast to such approaches, VNS does not require many parameters. The neighboring structures and the moving strategy would be enough to efficiently use the method. Besides VNS is more general and the freedom is large which can be designed in various forms for particular problems. Its idea is also simple and easy to implement, and the algorithm structure is independent of the problem, so VNS is suitable for all kinds of optimization problems. What is more is that VNS can be easily embedded into other approaches such as population-based metaheuristics.

The remaining of this paper is structured as follows: in next section, some related works are presented. In Section 3, the way the multimodal network has been represented is explained. Section 4 is devoted to introduce the proposed GA-VNS. Experimental results are presented in Section 5. Finally, Section 6 gives some comments and outlines future works.

2 Literature Review

Routing is a widely studied topic in transport systems, mainly because of its relevance to real world applications in a wide range of fields such as energy, military and communication networks. The major research effort on this problem relates to two things: modelling a transport network and solving routing issues. While the former consists of defining how to adequately represent a transport system, the latter deals with developing efficient strategies to support routing issues faced by passengers and transport operators.

In terms of modelling, (Pyrga et al. 2007; Delling et al. 2009; Dib et al. 2015) have done extensive works to incorporate the multimodality aspect into their models. (Liu et al. 2009) proposed a switch point approach to model multimodal transport networks. (Van Nes 2002) conducted several researches for efficiently designing multimodal transport networks. (Ayed and Khadraoui 2008) proposed also a transfer graph approach for multimodal transport problems. (Zhang et al. 2011) introduced a generic method to construct a multimodal transport network representation by using transfer links, which is inspired by the so-called super-network concept. (Pyrga et al. 2004) has also done relevant works to generalize a time-expanded model that deals with realistic transfers. (Bast et al. 2010) also handled multimodal networks by incorporating predefined transfer arcs between nearby stations.

When it comes to routing algorithms, several approaches have been proposed for solving basic and advanced routing problems. For instance, (Pajor 2009) adapted the algorithm of Dijkstra to take into account the time dependency and the multimodality aspect of the transport system. (Zografos et al. 2009) described an algorithm for itinerary planning based on dynamic programming. (Wang 2008) did a study on handling times and fares in a routing algorithm for public transport. (Pyrga et al. 2008) solved the earliest arrival problem on the time-expanded model in a straightforward manner by using a modified version of the algorithm of Dijkstra.

Computing Multicriteria Shortest Paths has been also studied recently. For instance, (Hamacher et al. 2006) proposed a backward label-setting algorithm for identifying important solutions for the all to one multiple criteria time-dependent shortest path. (Modesti et al.1998) also used a linear utility function that incorporates travel time, ticket cost, and "inconvenience" of transfers. Moreover, other label setting algorithms such as (Martins 1984; Corley and Moon 1985) and label correcting algorithms such as (Skriver and Andersen, 2000) have been modified for solving the MSPP.

Although the above-mentioned works on handling multicriteria using straightforward approaches are very significant, however, they have some drawbacks. From one side, they usually handle the multicriteria problem by transforming it to a simple single criterion problem. Thus, the decision maker will surely loose some interesting solutions. From another side, such approaches may cause exponential running time during the resolution phase of the problem. That is, classical approaches may suffer from a high computational time, which make them unusable within real world routing system where passengers seek real time answers.

To overcome such drawbacks, several works have focused on applying heuristic approaches such as metaheuristics to provide high quality solutions within reasonable computational time. Metaheuristics usage is not only limited to the transportation field but it can also be found in various areas such as networking, scheduling and logistics.

For instance, (Baswana et al. 2009; Doerr et al. 2011) worked with evolutionary algorithms to compute single source shortest paths using single-objective fitness. (Dib et al. 2015) introduced an advanced hybrid metaheuristic for route planning in road networks. (Rakesh et al. 2012) also proposed a novel GA to find shortest paths in computer networks. (Behzadi et al. 2008; Kumar el al. 2010) also worked with GAs to find shortest paths in data networks. (Gen et al. 1997) proposed a priority-based encoding method to represent all possible paths in a graph. (Delavar et al. 2001) proposed a GA with a part of an arterial road regarded as a virus to select route to a given destination on an actual map under a static environment. Moreover, (Davies and Lingras 2003) presented a GA based strategy to find the shortest path in a dynamic

network, which adapted to the changing network information by rerouting during the course of its execution. (Chakraborty 2004, 2005) proposed a GA based algorithm with a novel fitness function for simultaneous multiple routes searching for car navigation to avoid overlap. In the work of (Huang et al. 2004), a GA was introduced for determining the weights of different criteria, which eventually achieve a series value of each criterion and sum the up as the final cost. (Hochmair 2008) used GA for Pareto Optimal route set searching in order to reduce the number of route selection criteria. The GA based solution for multimodal shortest path problem presented by (Abbaspour and Samadzadegan 2009; Dib et al. 2015) showed the robustness of this approach through empirical studies and concluded that GA based approaches can efficiently explore the search space in order to find very good multimodal paths.

Other advanced Evolutionary algorithms have been proposed such as (Deb et al. 2002). In the latter's work, the so-called NSGAII (Nondominated Sorting Genetic algorithm II) has been introduced. The main motivation behind introducing this algorithm is to alleviate the difficulties of the traditional Multiobjective Evolutionary Algorithms (EAs) that use nondominating sorting and sharing. Such traditional approaches were suffering from three main drawbacks: 1) High computational complexity of nondominated sorting 2) Lack of elitism 3) Need for specifying the sharing parameter σshare. To overcome such drawbacks, NSGAII integrated a 1) fast-nondominated sorting approach 2) a fast crowded distance estimation procedure and 3) a simple crowded comparison operator. Experimental results have found that NSGAII outperforms most traditional approaches when solving several kind of optimization problems. There are several differences between the proposed algorithm and the standard scheme of NSGAII. Firstly, in the proposed algorithm the fast-non-dominated-sort that computes the set of all non-dominated fronts is not performed. Although, this operation maintains best non-dominated solutions, however, it may be costly for some applications that require very rapid answers such as the problem we are dealing with. The second difference lies in the selection operator. While in NSGAII, the tournament selection is used to select individuals for recombination, the proposed algorithm uses roulette wheel selection. Furthermore, the selection in NSGAII is based on a crowding-comparison operator, which requires having the nondomination rank and the crowding distance of individuals. In the proposed algorithm, the Average Weighted Rank is used as a selection criterion. Finally, ensuring the diversity among solutions in NSGAII is done using the selection operator itself since it integrates the crowding distance into the comparison between solutions. In the proposed algorithm, the diversity is maintained by the selection operator that may give chances with different probabilities to all individuals to pass their genes to the next generation, as well as with the mutation operator that is based on VNS approach.

The application of metaheuristics is not only limited to EAs; other methods were also used such as VNS, Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO). Examples of works done using such approaches can be found here (Hansen et al. 2014; Civicioglu et al. 2013; Böhm and Katinka 2016).

Several researches have focused in recent years on combining several metaheuristic to solve one single optimization problem. The main motivation for the hybridization process is to achieve better performance by exploiting and combining advantages of the individual pure strategies (Talbi 2013). We introduce in this paper a new hybridization approach in which the local search procedure VNS is used inside the population-based meta-heuristic GA. The hybridization process done between single solution and population-based meta-heuristics refers nowadays to the term memetic algorithm (MA) (Cotta and Carlos 2012). The term (MA) is inspired by both Darwinian principles of natural evolution and Dawkins' notion of a meme. MAs were first invented to reflect the fact that coupling genetic algorithms (GAs) with individual learning procedure may perform local refinements. Nowadays, MAs refer to a hybridization process done between population-based meta-heuristics such as GAs and single-point search or local search procedures such as Simulated Annealing (SA) (Aarts et al. 2014) and Tabu Search (TS) (Glover and Manuel 2013). MAs have proved to be effective in solving a wide range of optimization problems such as graph coloring, scheduling etc.

After giving an introduction about the studied problem, as well as, expressing the primal motivation behind the work and the idea behind the combination between two metaheuristics GA and VNS, the modeling approach adopted to represent a public transit network is introduced in the next section.

3 Modeling Approach

This section considers modelling a multimodal transportation network. It should be clarified that the term multimodal is used in the sense of multiple fixed scheduled transport services. A key difference to static networks is that public transit networks are inherently time-dependent, since certain segments of the network can only be traversed at specific, discrete points in time. As such, the first challenge concerns appropriately modelling the timetable in order to enable the computation of journeys.

Roughly speaking, a timetable consists of a set of stops (such as bus or train platforms), a set of routes (such as bus or train lines), and a set of trips. Trips correspond to individual vehicles that visit the stops along a certain route at a specific time of the day. Trips can be further subdivided into sequences of elementary connections, each given as a pair of (origin/destination) stops and (departure/arrival) times between which the vehicle travels without stopping.

The key modelling in this approach is to represent each transportation mode as a separate directed graph. An additional work is then done to integrate all sub-graphs into one larger graph. As a first step of modelling, three types of nodes are introduced; nodes correspond to stations, platforms and departure events.

Although in real life, a station may have several access points, it is assumed in this paper that each station has one and only one entrance area. A station also comprises a set of platforms where passengers wait for vehicles. An edge is then inserted between a platform and its parent station; its weight represents the minimal time required to access that platform from the entrance point of the station.

It is worth mentioning that most of representations in the literature disregard platforms. Instead, they only focus on vehicles. However, platforms are essential since transfers inside stations are made between platforms. Moreover, in some routing issues such as evacuations, platforms play an essential role since they give ideas about the capacity of vehicles or even the saturation of the transport system. As a result, we decided to integrate platforms into the proposed modelling scheme.

A platform in the proposed representation cannot belong to more than one station; however, a station can contain one or several platforms. Each platform has also a type (Bus, railway, tram...) to differentiate between modes. This information can be used by routing algorithms that deal with the preferences of users (i.e. a user may prefer to only take the bus along his/her journey).

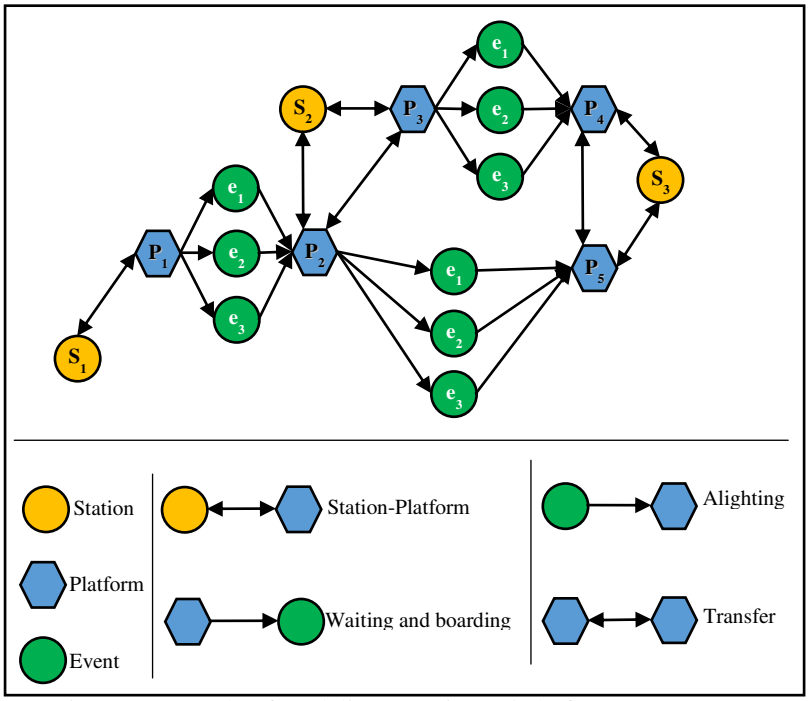

Figure1: Example of modeling 3 stations, 5 platforms, 9 events

Since a timetable consists of time-dependent events (e.g., a vehicle departing at a stop) that happen at discrete points in time, a space-time graph has been built to unroll time. Roughly speaking, a vertex is created for every event in the timetable that consists of vehicle departing from a platform *p* at *dt* (departure time) and arrives to another platform *q* at *at* (arrival time). Timestamps are inserted into event nodes to account for the departure and arrival times. Event nodes are ordered in the way that a higher-level node refers to an earlier event.

In addition, waiting, boarding and alighting edges are inserted between event nodes and platforms. Additionally, to account for transfers between and inside stations, transfer edges associated with transfer times have been inserted. It is worth mentioning that most of the representations in the literature assume that one station has a fixed transfer time. Thereby, transferring between any two platforms inside a station takes the same time. However, such assumption usually yield incorrect results when applying the routing algorithm. Therefore, in this work, it is considered that each station has a variable transfer time. We present in Figure 1, the scheme of a small multimodal network, which consists of 3 station, 5 platforms and 9 departure events.

It is worth to clarify that this model can also be elaborated further to handle additional information such as the capacity of vehicles. For instance, an event node can store an information about the maximum and current capacity of its current vehicle. This information is crucial when dealing with evacuation situations that require rerouting of passengers from one vehicle/mode to another.

4 Proposed approach: Hybrid metaheuristic GA-VNS

As aforementioned, the main contribution of this paper is to adapt and apply a heuristic method, which is based on a collaboration between two meta-heuristics, for solving the MSPP. The meta-heuristics used are GA that belongs to the population-based algorithms and VNS that belongs to the single point search algorithm.

The proposed method proceeds with a population of solutions as standards GAs works. However, in the proposed approach, initial solutions are generated using a double search algorithm that is able to provide a set of feasible paths between any two nodes. The details of this algorithm is described later in this article.

Once a population of solutions (feasible paths) are successfully generated, an enhancement operation is accomplished over the first population. That is, a VNS is applied over individuals belonging to the first population. It is decided to improve initial solutions since their quality would possibly help the algorithm in approaching from the optimal region within the search space. More details about the VNS adaptation to the problem will be discussed later. Once the VNS is performed, individuals are sent to an evaluation process. To evaluate an individual, a fitness function is computed using the Weighted Average Ranking (WAR) technique.

After the improving phase, several genetic operations are performed (*selection*, *crossover* and *mutation*) in the goal of increasing the algorithm's chance to find better solutions. To begin with, a roulette wheel selection has been used as a selection operator. A selection probability is computed according to the WAR associated with each individual. After that, the crossover operation is performed. In this operation, two individual are chosen according to the selection operator in order to form new individuals (offsprings). By doing so, a new population is produced having twice the size of the current population. The best half individuals are then selected for the next generation and the rest are ignored. Duplicated individuals are replaced with newly generated chromosomes to avoid reprocessing the same individual. Single point crossover technique has been used in order to produce offsprings. An intersection node between two individuals is selected to be the crossover point. Current individuals exchange then part of genes with each other before or after the crossover point to generate offsprings.

After that phase, a new population of solutions is created. Thus, new paths between the origin and the destination point have been probably detected. Therefore, re-applying the VNS procedure to the new population will possibly enhance the paths' qualities. That is, the algorithm performs a special mutation operation based on VNS method over each individual (path) in the population. Thanks to this technique, the algorithm will have more chances to exploit and explore new regions of the search space. The whole process

is repeated until the algorithm reaches the predefined stopping criteria. The following scheme represents the algorithms' steps that will discussed in more details in next paragraphs.

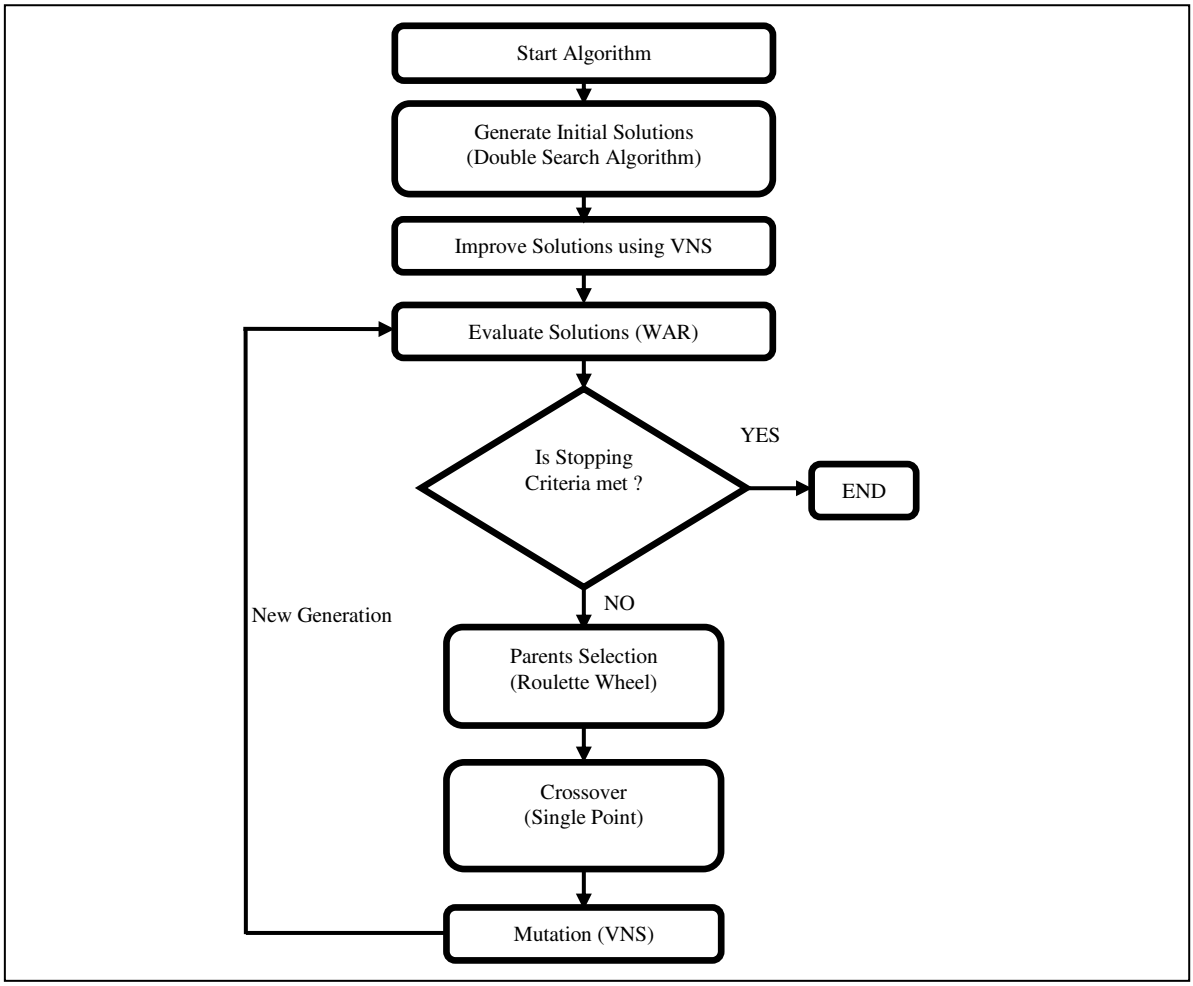

Figure 2. Steps of the proposed algorithm

4.1. Encoding

Encoding of individuals is a crucial step in GAs. It largely depends on the property of the problem, and will highly affect the design of genetic operations. Several encoding techniques have been proposed to encode solutions in GAs such as binary and permutation encoding. The former technique consists of computing and then concatenating the binary coding of each variable in a given individual. For instance, in the studied problem, a route consisting of four stations (2, 4, 9, 3) can be represented by the binary string 0010| 0100| 1001| 0011. In permutation encoding, variables are concatenating without any transformation. For instance, the previous route (2, 4, 9, 3) is encoded as 2| 4| 9| 3.Since the binary encoding, requires additional computation effort due to the transcoding (binary/integer), the permutation encoding is used in this work.

A solution for the studied problem is represented by a list of nodes where each node corresponds to a platform. The information stored in each node *n* are related to the identifier of the platform, its mode and the time at which *n* is traversed in addition to the value of the different objectives considered.

It is important to mention that the time at which each platform is traversed is computed by adding the traversed time of the predecessor platform to the exact travel time between the two platforms. The travel time between two platforms is computed by subtracting the arrival and departure times of the appropriate

departure event. This latter represents the first departing vehicle at the departure platform with respect to the arrival time at the departure platform.

To efficiently identified the first departure event at a platform w.r.t a departure time, event nodes are ordered w.r.t their departure times during the generation phase of the model. In addition, event nodes are stored in sub-categories according to predefined periods (e.g. the list of departing nodes from 8:00 to 9:00). Thanks to such implementation tricks, identifying the appropriate departure event would be very efficient and would also ensure the feasibility of encoding paths.

Since different solutions may represent different routes, the size of chromosomes is not static; it varies with the number of nodes constructing the route. Finally, a special node is inserted to deal with transferring from one mode to another. This special node plays an important role since it stores information related to the transfer (walking time, state of the transfer links ...). Thus, the computations of the different criteria especially those related to the number of transfers and walking time would be impacted by such transfer nodes. Furthermore, inserting a special node for transfers would help to efficiently identify crossover points as well as, to visualize the details of itineraries resulting from the algorithm.

We show in Figure 3 a solution for a routing request between the departure station *S* and the arrival station *T*. The encoded solution consists of using three modes: railway, metro and bus.

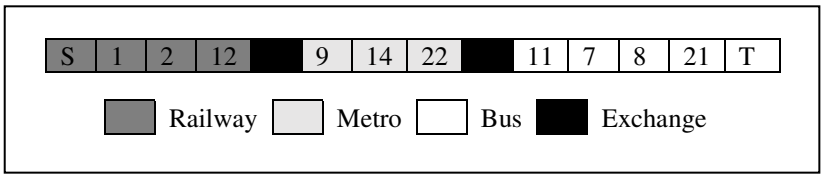

Figure 3. Example of encoding an individual

4.2. Evaluating an Individual

Several approaches can be used for computing the fitness function of a multiobjective optimization problem. The classical approach is to assign a weight w_i to each normalized objective function $o_i(x)$ so that the problem is converted to a single objective problem with a scalar objective function. While using this approach may largely enhance the computational time to solve the problem, it has the drawback of only providing one single solution. Therefore, if multiple solutions are desired, the problem should be solved multiple times with different weight combinations. In addition, selecting weight vector for each run remains a difficult issue for users. Furthermore, in such classic approach, there might be a good chance for losing some interesting solutions after solving the problem. Such solutions may be the best combination of many passengers. Therefore, such standard approaches are not used in in this work.

Evaluating individuals in this work has been done using the Weighted Average Ranking (WAR) technique. Unlike traditional techniques, which are based on a simple scalar value, WAR consists of assigning each individual (route) with a rank that depends on the value of objectives f_i *(route)*. This rank represents the average value of all ranks determined by sorting the current population in a separated manner according to each f_i *(route)*.

It is worth mentioning that by using this evaluation technique, the algorithm is prevented from normalizing all objectives. The normalization phase is not an easy task in the studied problem case since ranges of objectives are not known in advance. In addition, the normalization will surely consume additional computational time. Therefore, using WAR instead of traditional evaluation approaches represents an advantage for the proposed algorithm.

More precisely, the evaluation is performed as follows. The population is sorted Λ times according to each objective (where Λ is the number of objectives considered). At each sorting operation, each individual is assigned a rank value R_i according to its position in the current population. At the end of this operation, each individual will have Λ ranks. The global rank *GR* value will be then average of all ranks associated with an individual.

In Table 3 below, the above-explained technique is applied to evaluate seven individuals; two criteria are considered in a minimization problem. As can be seen from the table, the global average rank of individuals I_2 and I_5 is the smallest average (6/2). This property reflects very well that I_2 and I_5 are dominated by I_4. Since I_4 represents the individual that dominates the maximum number of individuals in the current populations, the WAR strategy results in a high GR for I_4.

Table 3. Example of evaluating individuals

Individual	*f1 score*	*f2 score*	*Rank according to f1*	*Rank according to f2*	*Global rank*
I_1	5	1	1	6	7/2
I_2	4	2	2	4	6/2
I_3	3.5	1.5	3	5	8/2
I_4	2	1.5	5	5	10/2
I_5	3	4	4	2	6/2
I_6	2	3	5	3	8/2
I_7	1	5	6	1	7/2

4.3. Generating Initial Solutions

Generating initial solutions for meta-heuristics is not always straightforward. Usually, good initial solutions might rapidly guide the search process towards important regions in the search space. Initial solutions in the proposed work are a set of feasible paths generated using a double search algorithm. A double search process that simultaneously run a forward search from the origin point *s* and a backward search from the destination point *t* has been used in order to get a set of feasible paths between a pair of nodes.

The algorithm works as follows:

- It initially defines two queues: one for the forward search *fq* and another for the backward *bq*. The source and target nodes *s* and *t* are then added to *fq* and *bq* respectively.
- After initialization, the algorithms starts repeatedly taking the element at the head of *fq*; adding its adjacent nodes to *fq*; removing it from *fq*. The same process is also done from the backward search.
- To keep a history of the search processes, nodes visited from the forward search are assigned an "F" flag and nodes visited from the backward search are assigned a "B" flag. Moreover, we store the incoming edges that allow the algorithm to reach each forward node and the adjacent edge of each backward node.
- Once the algorithm is about to add in *fq* a node that has already been visited from the backward search, it means that a path between *s* and *t* has been detected. Therefore, the procedure of constructing that path starts. The same process is also performed from the backward search.
- To reconstruct a path found, the historical information stored in each node object are used.
- The algorithm terminates if one of the queues *fq* and *bq* becomes empty. It has been remarked also that in some graph instances the proposed algorithm may provide too many initial paths. Therefore the maximal number of generated paths has been added as another termination criterion for this case. By doing so, the size of the first population is can be controlled.

It is worth to mention that as the algorithm performs, there is a chance that two identical paths are found between s and t. To tackle such problem, the algorithm keeps a history about the intersection edges found during the execution. It then ignores a path if the intersection edge has been already added to the list.

Finally, as can be easily noticed, the process of computing initial solutions mimics the search process used in bidirectional Dijkstra algorithm (Ikeda et al. 1994). However, using the bi-directional Dijkstra in its standard form is not possible since the exact arrival time at the destination node is not known a priori due to the time dependency property of the network. To overcome this issue, a search process is

launched from the destination node with ignoring the exact arrival at the destination node. As a result, a set of feasible paths are obtained with no guarantees about their qualities. Therefore, one can see the proposed algorithm to compute initial solutions as a modification of the bidirectional Dijkstra where the exact arrival time at the destination node is ignored and the stopping criterion to get many feasible solutions is adjusted.

4.4. Enhancing Initial Solutions Using VNS

Since the quality of initial solutions may affect the performance of meta-heuristics, we decide to enhance initial solutions. To do so, a VNS method is used as an enhancement operator. Indeed, a VNS method is applied over the individuals in the first population.

This technique will increase the diversity level of the initial population, as well as, establish a good repartition of solutions within the search space. Consequently, the algorithm's chance to find better solutions will also increase. In the following, the adaptation of the VNS method to deal with the studied problem is explained. Indeed, one of the major challenges addressed in VNS is the construction of the neighborhood structures. To deal with that issue, the proposed algorithm performs a preprocessing operation during the generation phase of the network.

The idea is to examine each edge in the graph and check if its starting and ending nodes share a common node. Two nodes *A* and *B* have a common node if and only if the end point of one adjacent edge of *A* is the same as the starting point of an incoming edge of *B*. After doing that, we will end up with two neighboring structures that we will use when applying VNS.

In the graph below, if we take the edge *(AC)*, it can be noticed that the node *B* is shared between the adjacent edge *(AB)* of *A* and the incoming edge *(BC)* of the node *C*. Hence, one can deduce that an alternative path can be found to reach the node *C* from *A*, which is in this case the path *(AB, BC)*.

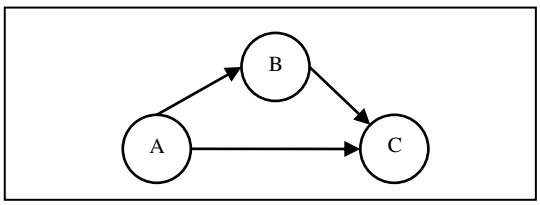

Figure 4. Constructing neighboring structures

Once such common nodes are found, constructing neighboring structures can be done as follows. If the path represented by the edge *(AC)* is dominated by the path *(AB, BC)*, thus, whenever the edge *(AC)* is met, it should be replaced by the path *(AB, BC)*. That case makes the first neighboring structure. That is, the first neighboring structure is a list containing replacement paths formed by two edges for each edge.

A second scenario may arise if the path *(AB, BC)* is dominated by the path *(AC)*, thus, the path *(AB, BC)* can be always replaced by the edge *(AC)* in any path. The second neighboring structure is then a list containing an edge that will replace a path formed by two edges.

In the case that *(AC)* and *(AB, BC)* are non-dominated, the replacement list of any of them should include the path itself and the other path. By doing so, the solution itself and a non-dominated solution belonging to the list of the neighborhood solutions are maintained.

Figure 5 shows a graph with 13 edges and 8 nodes. Only one single criterion is used to facilitate the illustration. An example of the replacements included in the first neighboring structure is to substitute the edge *(SC)* with the 2-edges path *(SA, AC)*. Replacing the 2-edges path *(BD, DT)* by the edge *(BT)* is an example of an instance existed in the second replacement structure.

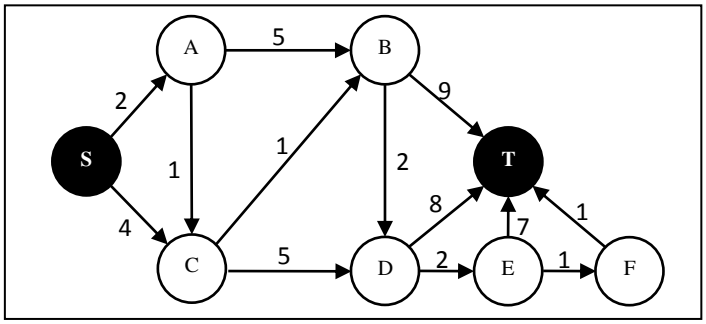

Figure 5. Performing VNS over an individual

After performing the preprocessing operation over the graph above, the algorithm ends up with two replacement lists that illustrate neighboring structures.

Table 1. Items in the first replacement list

Edge	Replacements
(SC)	(SA, AC)
(AB)	(AC, CB)
(CD)	(CB, BD)
(ET)	(EF, FT)
(DT)	(DE, EF, FT)

Table 2. Items from the second replacement list

Path	Replacements
(BD, DT)	(BT)
(DE, ET)	(DT)

After carefully examining the two replacement lists, it has been realized that the performance of our VNS can be improved by applying some enhancement operations. More specifically, we try to enhance the quality of solutions that can be found by applying the first structure using information from the two replacements list. For instance, the path *(DE, ET)* included in the second list can be replaced by the edge *(DT)*. However, if one carefully scrutinizes, it can be noticed that the edge *(ET)* is included in the first structure and the length of its replacement path added to the length of *(DE)* is shorter than the length of *(DT)*. Therefore, a new element can be added to the first structure and it will contain the substitution of *(DT)* by the path *(DE, EF, FT)*. This strategy is only applied in the case that it produces a new dominated solution.

The quality of elements in the first list can also be improved by using the list itself. For instance, let us imagine that the edge *(ET)* has a weight of 5. The first list will then contain a row containing the replacement of *(DT)* by *(DE, ET)*. However, *(ET)* is also in the list. Hence, *(DT)* can be replaced by *(DE, EF, FT)*.After the enhancement operations, the algorithm gets the chance to find better solutions and thereby its performance will increase. By taking the path *P= (SC, CD, DE, ET)* which is a solution to go from *s* to *t*, the neighbor solutions of *P* using the first neighborhood structure are:

P_{11}	SA	AC	CD	DE	ET
P_{12}	SC	CB	BD	DE	ET
P_{13}	SC	CD	DE	EF	FT

The neighbor solutions of *P* after applying the second neighborhood structure are:

P_{21}	SC	CD	DT

After defining the two neighboring structures, the mechanism that we adopted to move from one solution to another is explained. As shown in Figure 6, the VNS method takes as input a set of neighboring structures

and a single initial solution and it performs then some operations in order to enhance its initial solution.

Although the VNS approach is very popular for its easy adaptation, it turns out that its usage in a multicriteria environment is not an easy task. The difficulty steems from the fact that one solution may not only have one single better solution among all elements in the list of neighborhood solutions. However, a solution may have several dominated or non-dominated solutions in the list of neighborhood solutions. It is therefore very important for the efficiency of the method to define a strategy whereby the algorithm moves from one solution to a neighbor one during the search process.

To overcome all those difficulties, the VNS is adapted as follows. The VNS proceeds with the first neighboring structure. It examines then all the neighbors' solutions belonging to the current solution. The algorithm then selects the best neighbor among all neighbors according to the fitness.

To define which neighbor solution is the best one, all neighborhood solutions are evaluated according to a Weighted Average Ranking (WAR) that we will define in the next section. More the rank is higher, more the solution is better. VNS then compares the best-selected neighbor with the current solution; it moves to the best neighbor if its fitness is better than the current solution. If not, VNS changes the neighboring structure. The process is repeated while the current solution can be enhanced by using any of the neighboring structures.

```
Input:
Two neighboring structures nk, (k={1,2});
An individual (solution) x;
Iteration
While stopping rule is not satisfied do
        k=1;
        While k<=2 do
             Exploration of neighborhood: find the best neighbor x' of x (x'∈Nk (x))
             Move or Not:
            If f(x') < f(x) then
                x ← x', k ← 1;
            else
                k ← k+1;
            end if
         end while
end while
return the best solution
```

Figure 6. Variable Neighborhood Search

Assuming that the path x = (SC, CB, BD, DT) in Figure 5 is an initial solution to the VNS method with length 15. To apply VNS, the algorithm performs as follows:

It uses firstly the neighboring structure; it then calculates all the neighbors solutions; it then selects the best neighbor. Neighbor solutions of x using the first neighboring structure are:

x_{11}	SA	AC	CB	BD	DT	Fitness=14

x_{12}	SC	CB	BD	DE	EF	FT	Fitness=11

Neighbor solutions of x using the second neighboring structure are:

x_{21}	SC	CB	BT	Fitness=14

The best neighbor of x using the first neighborhood structure is x_{12}. The algorithm will then move to x_{12} since its fitness is better than x. The same process is repeated over x_{12}. The best neighbor of x_{12} using the first neighborhood structure is the path x^*= (SA, AC, CB, BD, DE, EF, FT) with length 10. The algorithm moves to x^* since its fitness is strictly better than x_{12}.The same process is repeated over x^*. However, x^* cannot be

enhanced either by using the first neighboring structure or the second one. Therefore, the algorithm stops and return x^*.

Finally, the VNS method is based on two-neighboring structures. At each time the proposed algorithm gets stuck at a local minimum, the structure of the neighborhood is changed. By following this technique, the algorithm will exploit and explore wide regions of the search space. Adding more than two neighboring structures will probably enhance the chance of the algorithm to find better solutions. However, that might increase the time to accomplish the preprocessing operation as well as the time to perform the VNS itself.

The strategy have adopted to move from one solution to another is the best enhancing neighbor. Other techniques can be used such as "The first/best/least enhancing neighbor". Such techniques may provide better results and extensive researches are conducting nowadays to decide which technique is the most performant. To study the impact of such techniques on the performance of the VNS approach, we measure in Section 5 the quality of solutions obtained and the rate of convergence after varying the strategy to select the next solution from the neighborhoods list of the VNS approach. In addition, in this work, the first structure neighborhood is applied before the second one. In fact, deciding which neighboring structure should be applied at which order has been always an issue for the VNS method. Further works will be done in near future to study if the order of neighboring structure will affect the performance of the proposed approach.

4.5. Parent Selection

Generally speaking, the surviving probability of an individual is related to its efficiency in comparison with the other individuals in the population. In evolutionary algorithms and like in the natural selection phenomenon, a stochastic character is introduced in the probability of selection. The most famous stochastic techniques to accomplish the selection process are roulette wheel and rank selection. In the former, a roulette wheel is used where are placed all chromosomes in the population, every has its place big accordingly to its fitness function. Therefore, the better the chromosomes are, the more chances to be selected they have. Rank selection consists however, of ranking the population and then every chromosome receives fitness from this ranking. The worst will have fitness 1, second worst 2 etc. and the best will have fitness N (number of chromosomes in population). The roulette wheel is used in this work as a selection operator. For each individual, a selection probability is computed by dividing the WAR of the individual over the sum of finesses of all the individuals in the population.

Considering the example below (Table 4, Figure 7), solutions have practically, equivalent portions except for the individual I_4. This latter has the high probability selection since it is the fittest individual in the population. Therefore, its chance to survive and pass its genes to the next generation will be higher than the other individuals.

Table 4. Example of evaluating individuals

I	*GR*	*P(xi)*
I_1	3,5	0,13
I_2	3	0,11
I_3	4	0,15
I_4	5	0,19
I_5	3	0,11
I_6	4	0,15
I_7	3,5	0,13

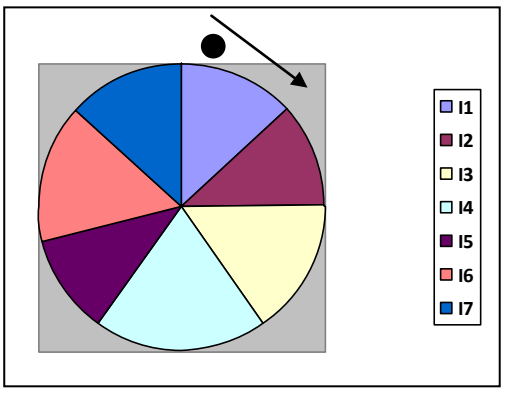

Figure 7. Roulette wheel selection

4.6. Crossover

After the selection and during the reproduction operation, individuals will be modified in order to generate new individuals (offspring) for the next generation. The two prominent operators to accomplish this operation are crossover and mutation.

The crossover consists of exchanging elements between two individuals initially selected in the goal of producing one or two new individuals. This operator is usually applied with certain probability (usually high) and may be accomplished in several ways depending on the structure of the problem itself.

When using the binary representation, the single and multiple point are the most used techniques to perform the crossover. In the former, one crossover point is selected, binary strings from the beginning of the chromosome to the crossover point are copied from one parent, and the rest is copied from the second parent. In the latter such as the two-point crossover, two crossover points are selected, binary strings from beginning of chromosome to the first crossover point is copied from one parent, the part from the first to the second crossover point are copied from the second parent and the rest is copied from the first parent.

The single point crossover has been used as a crossover operator. The crossover point is chosen to be any exchange node that is shared between parents. After selecting two individuals from the current population according to the selection operator, new individuals (offsprings) are produced w.r.t the crossover point. By doing so, a new population having twice the size of the current population is produced. The best half individuals are then selected for the next generation and the rest are ignored. The goodness of an individual is measured according to its weighted average rank. More the individual's WAR is high, more its chance to pass its genes to the next generation would be high. It is also worth to mention that the best solution at each generation is copied as it is to the next generation in order to avoid losing the elite solution. Since there is a chance that the same individual is duplicated in the population as the generations go on, duplicated individuals are therefore replaced with newly generated chromosomes. This process will undoubtedly increase the diversity within the population.

In the next figure, an example is given to illustrate the crossover operation. Two parents P_1 and P_2 have been selected according to their fitness. The two offsprings O_1 and O_2 have been generated after exchanging parts between P_1 and P_2 w.r.t the crossover point.

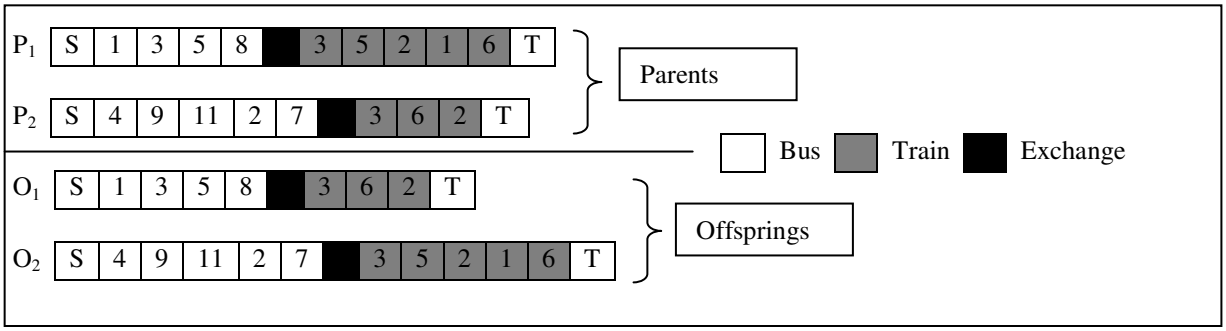

Figure 8. Single Point Crossover

One point worth mentioning is that after the crossover operations used in this method, the algorithm does not care about the feasibility of the new generated path. The algorithm will always end up with a feasible path from the source to the destination. Therefore, the algorithm does not lose time to check the validity of the offspring nor to perform some additional operations to repair the infeasible solutions.

Finally, experimental results show that using more than one crossover point might increase the diversity of the proposed approach. Therefore, the algorithm's chance to find better solutions will also increase. However, that may be at the expense of additional computational efforts. It is decided therefore to only use the simple crossover technique.

4.7. Mutation

Crossover operation may produce degenerate population. The algorithm may therefore get trapped at local minima. To overcome this issue, we perform the mutation operation.

The VNS has been used as a special mutation operator. That is, a VNS is applied over the population's individuals. By doing so, the algorithm is guided towards new regions within our solution space. The algorithm's chance to find better solutions will therefore increase.

Furthermore, applying VNS will ensure that the genetic algorithm maintains a sufficient diversity level that prevents premature convergence. The mutation operation has been applied at each time we perform the crossover. The conventional mutation operation used in GAs is usually applied with low probability. However, in the proposed hybrid approach, the mutation is always applied after crossover. The purpose of doing that is to allow the algorithm to avoid local minima by preventing the population of chromosomes from becoming too similar to each other, thus slowing or even stopping evolution.

Other mutation techniques have been applied such as order changing, but it has been realized that the mutation becomes less performant and it may provide invalid paths. An additional process should therefore be applied to reform infeasible paths. As a result, the mutation computational time will increase. It is decided thereby against using such traditional mutation techniques.

In the next figure, the scheme of the mutation operation is presented. As input, the VNS approach that is used to perform the mutation, takes a population of individuals. After performing the VNS after each individual in the population, the output of the mutation would be a set of better individuals in terms of quality and diversification level.

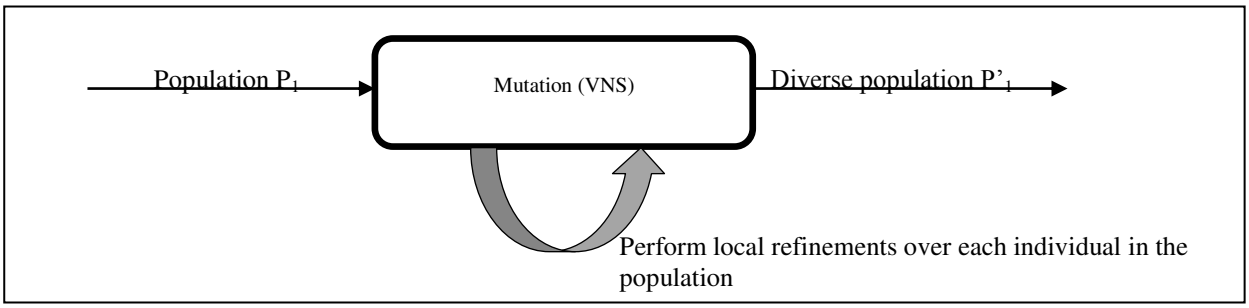

Figure 9. Mutation Scheme

4.8. Terminating Condition

Heuristic approaches do not guarantee finding the optimal Pareto front set. They do not therefore have the ability to automatically stop performing when a set of solutions is detected. Additional terminating conditions should then be introduced in order to allow the convergence of the algorithm.

Maximum number of generations, fixed execution time, and no modifications in population elements can be considered as algorithm stopping criteria. Two stopping criteria have been used in this study. Firstly, the algorithm stops when it fails to find interesting solutions during several continuous steps. An interesting solution is a new generated individual that is not dominated by any of the individuals in the current population *P* or is a solution that at least dominates one individual in *P*. 100 generations is used as a number to ensure a fixed state in the population. Another stopping criterion is until the algorithms reaches the maximum number (500) of generations. It has been noticed after some experimentations that the algorithm visits wide range of the search space rapidly. Thus, there is a big chance that the algorithm converges after few generations. That explains the small number of generations used as stopping criteria.

5 Experimental Results

To assess the performance of the proposed work, we have developed an advanced web-based routing application based on the real data of the French region Île-de-France that includes the city of Paris and its suburbs. Data are provided by the transport organization authority that controls the Paris public transport

network and coordinates the different transport companies operating in Île-de-France, mainly the RATP, the SNCF and Optile.

Data comprise geographical information, as well as, timetable information for four transport modes (Bus, Metro, Railway, and Tram). More precisely, data encompass 17950 stations; 41047 platforms; 195000 transfers; 303000 trips and 6800000 events for one day. For more information about the geographical perimeter of this study, we show in Figure 10 a screenshot taken from Google Maps API for the French region of Ile-de-France that include the city of Paris and its suburbs.

Before analysing results, it is worth to clarify that several models from the literature suffer from high computational effort when reading and compiling data. However, in the proposed modelling approach, the generation and integration of the different sub-networks take less than 12 seconds. This validates the efficiency of the proposed model, as well as, the appropriateness of the data structures used during the implementation phase. For instance, an event node with its associated edges corresponding to a) waiting and boarding and b) alighting are not represented as three separated entities during the implementation. In contrast, they do belong to the same object. Information are then added inside one object to refer to that information. Thanks to such implementation tricks, we succeeded at reducing the computational space to store all the model's components. Therefore, the size of the network resulting from the modelling phase does not constitute a bottleneck for the proposed approach to be integrated in real world routing system.

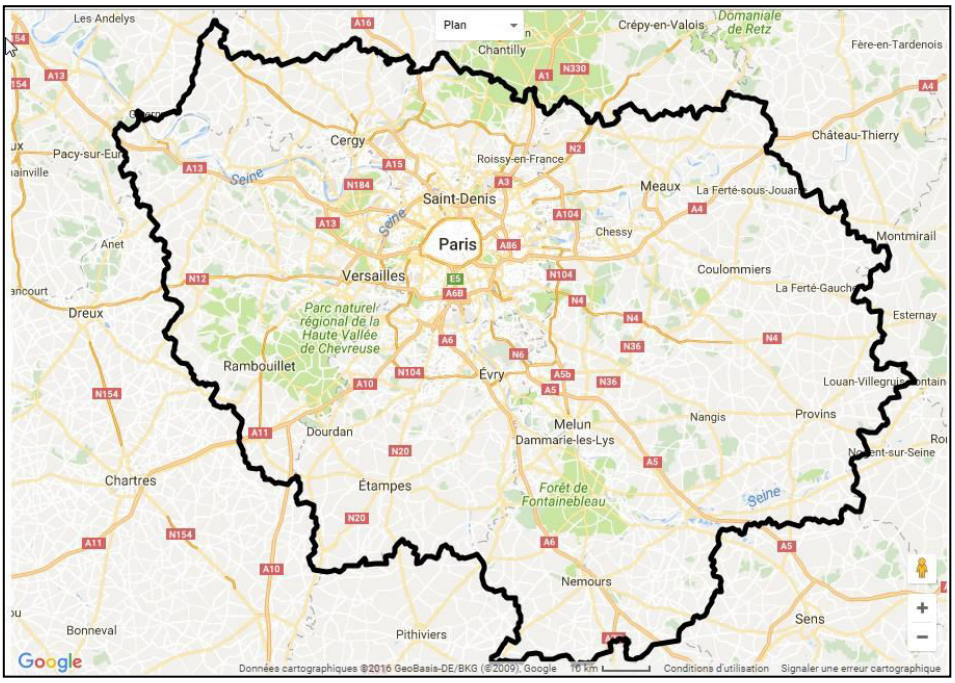

Figure 10. Case study: Ile-de-France region

The following parameters have been used in the proposed GA-VNS and the pure GA: the initial population size is 5; the probability of crossover is 0.9; the mutation rate is 0.9; the population size is 5; the maximum number of generation is 500; The number of generations used to ensure a fixed state in the population is 100. Algorithms have been tested on an Intel core I5 machine of 8 GB RAM and Java is used as a programming language. Efficient and appropriate data structures have been used to guarantee fast access to information when needed and thereby improve the performance of the proposed approach.

Since the increased time in solve route planning issues decreases the utility of the journey planning services, the computational effort of any routing algorithm constitutes a critical success factor for its integration within an online journey planning decision support system. Therefore, in this work, the evaluation of the proposed hybrid approach GA-VNS is done by comparing its running time with other approaches such as the algorithm of Dijkstra, a pure genetic algorithm and a pure VNS approach. The algorithm of Dijkstra has been used as a reference for comparing other approaches (GA,VNS, GA-VNS) since it provides the optimal set of nondominated solutions to go from one station at a certain departure time

to another station. Although the bidirectional Dijkstra algorithm may be seen as an advanced way to speed up the standard Dijkstra algorithm, however, it cannot be used in this work since the exact arrival time at the destination node is not known a priori. Therefore, the standard algorithm of Dijkstra has been selected for evaluations. The pure GA applied has the same scheme of the proposed GA-VNS except that VNS is not applied neither to enhance initial solutions, nor to perform the mutation operation. Rather, a standard mutation that consists of taking a subpath from an individual and replacing it with a newly generated path using the algorithm of generating initial solutions. The mutation in this case is applied with low probability of 0.1. Regarding the VNS approach, the moving strategy used is the best enhancing neighbour since this strategy is the most efficient one for the studied problem. Moreover, the starting solution for the VNS is chosen arbitrary. It is worth to mention that the starting solution may affect the performance of the VNS approach. After empirically testing many starting solution, it has been found that the performance of the VNS for the studied problem is not largely impacted by the starting point. Therefore, in this work, a random starting solution has been given as an input solution for the VNS.

Furthermore, since the proposed GA-VNS approach is based on two heuristic approaches, there is no guarantee to find the exact Pareto front set. Therefore, another key factor to assess the performance of one heuristic approach is to measure the quality of solutions found w.r.t optimal solutions. To do so, we compute the GAP by dividing the Global Rank (GR) of each solution found by the highest GR among all non-dominated solutions found using the exact approach. This latter is based on the classical shortest path algorithm of Dijkstra, which has been generalized to compute the exact set of nondominated solutions to go from one station to another w.r.t some predefined criteria. Details about the algorithm can be found here (Bast et al. 2015).

Finally, comparing the running time and the solutions quality of the proposed GA-VNS has been done by solving 10000 routing queries and w.r.t the algorithm of Dijkstra a pure GA and a pure VNS. Each routing request consists of a source node, a target node, and a departure time. Those parameters are uniformly generated at random. It is assumed in this work that requests allow all transport modes. We present in Figure 11 the details of the experimental process.

Although the proposed routing method is not limited to specific criteria, experimentations are done w.r.t the following criteria: i) travel time ii) cost iii) number of transfers and the total walking time. One can add more criteria such as distance or co2 emission by charging their values on corresponding edges.

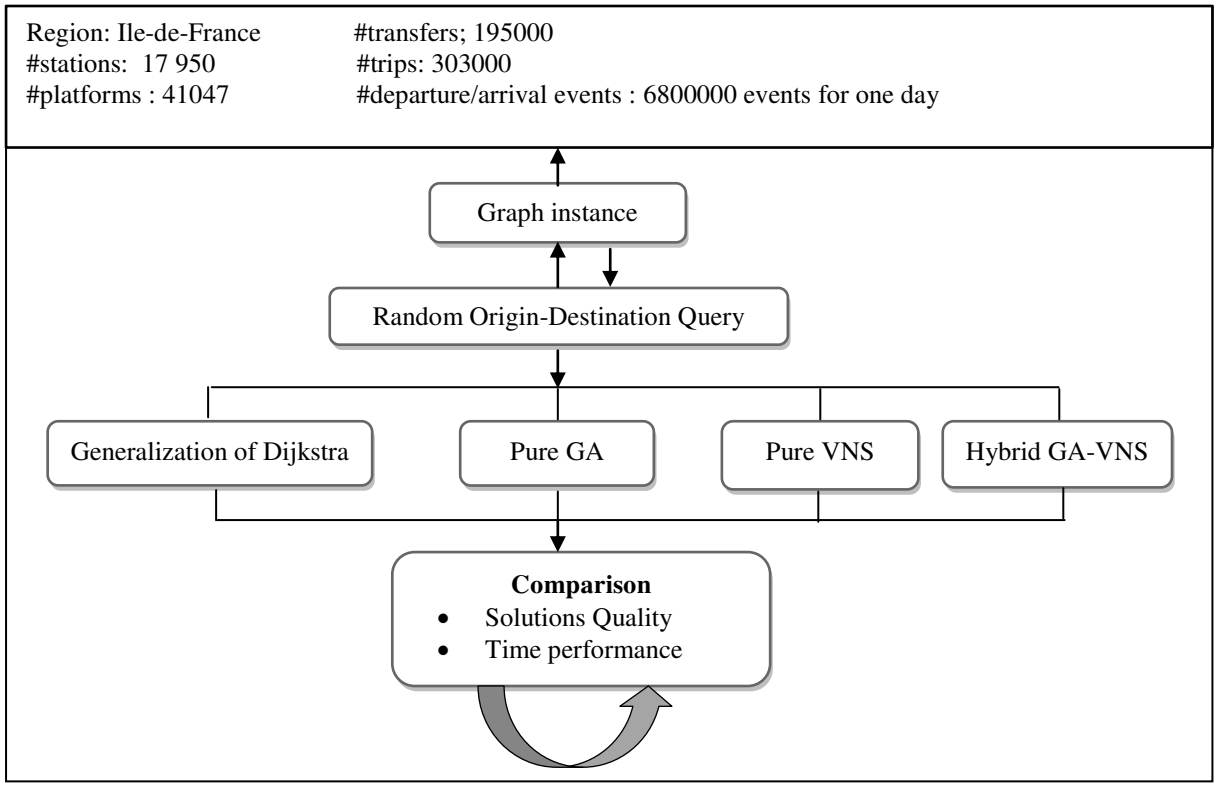

Figure 11. Experimental Process

Before comparing the different approaches, it is worth to mention that the performance of the VNS approach may be affected by the choice of the strategy adopted to move from one solution to another. Several strategies can be used such as the first, best or least enhancing solution among all solutions belonging to the set of neighborhood list. Theoretically speaking, deciding which strategy is the most efficient one is very difficult task since it largely depends on the problem itself. For this seek, we have studied the impact of several moving strategies.

Table 5. Impact of different moving strategies on the performance of the VNS approach.

Criterion	*Running time of the exact approach of Dijkstra (s)*	*First enhancing*		*Least enhancing*		*Best enhancing*	
		Time (ms)	Gap (%)	Time (ms)	Gap (%)	Time (ms)	Gap (%)
T	1.6	51	5.88	58	4.44	55	4.51
C	1.5	52	6.64	61	4.14	53	4.16
E	1.52	51	7.14	64	5.36	51	5.31
W	1.54	55	6.83	57	5.68	56	5.66
T & C	2.18	75	12.23	93	8.25	81	8.15
T & C & E	20.60	84	13.54	101	10.29	92	9.16
T & C & E & W	190.17	99	15.45	124	13.36	110	12.90

T: travel time C: cost E: number of exchanges W: total walking time

Results in Table5 show that the moving strategy has an impact on the performance of the VNS approach regardless the number of criteria taken into account. While the first enhancing strategy may help to reduce the computational time of the VNS approach, it may decrease the quality of solutions. Furthermore, when using the worst enhancing strategy, it has been realized that the quality of solutions may be enhanced in comparison with the first enhancing technique. However, that improvement will be at the expense of additional computational time. Finally, using the best enhancing outperforms the other two strategies in terms of the quality of solutions found. However, it takes more time than the first improving strategy. Therefore, we decided to use the strategy that gives the best solutions since the additional time required to get the best enhancing neighbor is not very limiting.

After empirically identifying the most efficient moving strategy for the VNS approach, we start evaluating the proposed approach by limiting the routing query to only one criteria (i.e. we start by travel time). That is, we want to ensure that the proposed model allows computing multimodal routes in a single criteria context and the proposed algorithm provides correct results. Analyzing and comparing results with real-world routing applications have verified the ability of the proposed model in providing correct itineraries when applying the routing algorithm.

When it comes to assessing the computational performance, results in Table 6 indicate that the hybrid GA-VNS does not exceed the ratio of 70 milliseconds to handle a routing request. Conversely, the running time of the exact approach of Dijkstra increases to 160 milliseconds. Thus, the speed column of the proposed GA-VNS, which is computed by dividing the running time of the algorithm of Dijkstra over the running time of the proposed GA-VNS, indicates a maximal gain of 2.42.

While the exact approach guarantees finding the exact Pareto front set, the average Gap to the optimality of the proposed GA-VNS in a single criterion does not exceed the ratio of 3%. Results in Table 6 show also that the running time of the other heuristic approaches GA and VNS is better than the algorithm of Dijkstra and the hybrid GA-VNS. This can be explained by the fact that such approaches rapidly converges in comparison with the proposed GA-VNS. The rapid converge of the pure GA and VNS can be also noticed from their average GAP w.r.t the hybrid GA-VNS. While the ~GAP of the pure GA increases to 3.45, the ~GAP of the pure VNS may reach 5.66.

By analyzing the result below, it can be said that the proposed hybrid GA-VNS provide better solutions than the pure GA and VNS. However, that will be at the expense of additional computational time. It can also noticed that using a heuristic approach to solve the single criterion shortest path is not very attractive since the exact algorithm of Dijkstra is efficient enough to provide optimal solutions within reasonable computational time. Finally, Results in Table 6 also indicate that when changing the criterion, the algorithm's search space does not relatively change and so does the ultimate running time.

Table 6. Single criterion experimentations.

Criterion	*Running time of the exact approach of Dijkstra (ms)*	*(GA-VNS)*			*GA*		*VNS*	
		TIME (*ms*)	GAP (%)	SPEED	GAP (%)	TIME (*ms*)	GAP (%)	TIME (*ms*)
T	160	70	2.88	2.28	3.19	62	4.51	55
C	155	64	2.58	2.42	3.25	58	4.16	53
E	152	63	2.69	2.41	3.45	57	5.31	51
W	154	66	2.43	2.33	3.16	61	5.66	56

T: travel time C: cost E: number of exchanges W: total walking time

After studying the performance of the proposed approaches in a single criterion environment, we now analyze their behaviors in a multicriteria context.

While using the exact approach to compute routes in a single criteria context was done without any limitations, computing Pareto routes turns out to be more challenging.

Results in Table 7 have indicated that the average running time of the exact approach in two criteria context may reach 2.18 seconds. It, however, increases to 20.60 seconds when considering three criteria and to 190.17 seconds when dealing with four criteria. As can be easily noticed, using the exact approach to support online users' queries is not satisfying in a real context due to its high computational time.

It can be also noticed that the average running time of the exact approach exponentially varies with the number of criteria. More criteria leads to an exponential increase in the search space. Thereby an important increase in the running time.

When it comes to assessing the performance of the proposed heuristic approaches when handling more than one criterion, results indicate that their average running is highly better than the exact approach. While answering a routing request may take more than 3 minutes when using the exact approach and considering 4 criteria, heuristic approaches provide answers within no more than 170 milliseconds.

It can be also seen from Table 7 that the running time of the pure VNS is the best one. However, that decreases the quality of its provided solutions. While the ~GAP to the optimality of the proposed GA-VNS reaches 3.24%, the ~GAP of the pure GA increases to 7.46% and 12.90 for the pure VNS.

By analysing those results, we can notice that the proposed hybrid GA-VNS represents the best compromise in terms of running time and the quality of solutions provided. Consequently, our algorithm is efficient enough to replace other approaches and to be used in real world routing system.

Table 7. Multiple criteria experimentations.

Criterion	*Running time of the exact approach of DIJKSTRA (s)*	*Our Algorithm (GA-VNS)*			*GA*		*VNS*	
		TIME (*s*)	GAP (%)	SPEED	GAP (%)	TIME (*ms*)	GAP (%)	TIME (*ms*)
T & C	2.18	0.11	3.14	20	4.25	0.087	8.15	0.081
T & C & E	20.60	0.14	3.19	147	5.90	0.095	9.16	0.092
T & C & E & W	190.17	0.17	3.24	1118	7.46	0.120	12.90	0.11

t: travel time c: cost e: number of exchanges w: total walking time Seconds: s

As in most of metaheuristics, tuning the parameters is one of the hardest part while accomplishing the implementation phase. Unfortunately, there is no standard parametrization rule since parameters highly depend on the structure of the optimization problem itself.

Several parameters have to be tuned for the proposed algorithms in order to achieve the best performance. We start with the initial population size. As previously mentioned, we limited the initial population size to 5. One reason for that is that, the algorithm used for generating initial feasible solutions is not always capable of providing as much solutions as we want. Another reason originates from the experimentations. Experimental results show that having less than 5 initial solutions will lead to premature

convergence. In another word, the initial population becomes less diverse. Consequently, the algorithm usually converges to poor local minima very rapidly.

On the other hand, having more than 5 initial individuals may increase the diversity in the first population. However, that will be at the expense of additional computational time and in most cases to the same quality of final computed solutions. Thus, we have fixed the initial population size to 5.

We have also analysed the crossover and mutation rates. As in most traditional GA schemes, the crossover rate is always high, while the mutation probability is very low. However, in our case the mutation is always applied after the crossover. Experimental results have shown that the best crossover and mutation rates for our problem is 0.9. Since the crossover is a convergence operation, which is intended to pull the population towards a local minimum/maximum, performing the crossover over all individual will lead to premature convergence. On the other side, results indicate that using less than 0.9 as a crossover probability will affect the final quality of solutions. That is, the average Gap to the optimality will increase as the crossover rate decreases.

Results have also indicated that decreasing the mutation rate will prevent the algorithm from converging towards interesting regions within the search space. This can be explained by the fact the mutation is a divergence operation and is usually intended to occasionally break one or more members of a population out of a local minimum/maximum space and potentially discover a better minimum/maximum space. On the other hand, decreasing the mutation rate to less than 0.9 will lead to premature convergence. Thus, the quality of final solutions found will be degraded.

Therefore, parameters used to tune the proposed GA have been experimentally found. Using such parameters will lead to achieving the best performance and thus increasing the efficiency of the introduced GA-VNS. It is worth mentioning that other values can be used to increase the convergence rate of the algorithm, however, that will surely decrease the quality of solutions.

Real Case Scenario

To have better evaluations of the proposed work, an advanced web routing application has been developed. The user interface has been developed using HTML, JavaScript, JQuery. Algorithm has been implemented in Java. To visualize itineraries we have used the Google Maps API.

We show in Figure 13, a real case scenario: it consists of a request to go from the station “Tour Eiffel” to the station “La defense”, which are both located in the region Ile-de-France. The departure time is “10:45”. Criteria are travel time, monetary cost, number of transfers and walking time. Modes used are Bus, Railway, Tram, Metro. After applying the proposed GA, three non-dominated paths have been found. In Figure 12, we present the stations constructing such paths, as well as, the values of the non-dominated paths.

We present in Figure 13, the user interface of the developed routing application and the results of the previously mentioned query. Three paths among many others are presented and visualized using the Google Maps API.

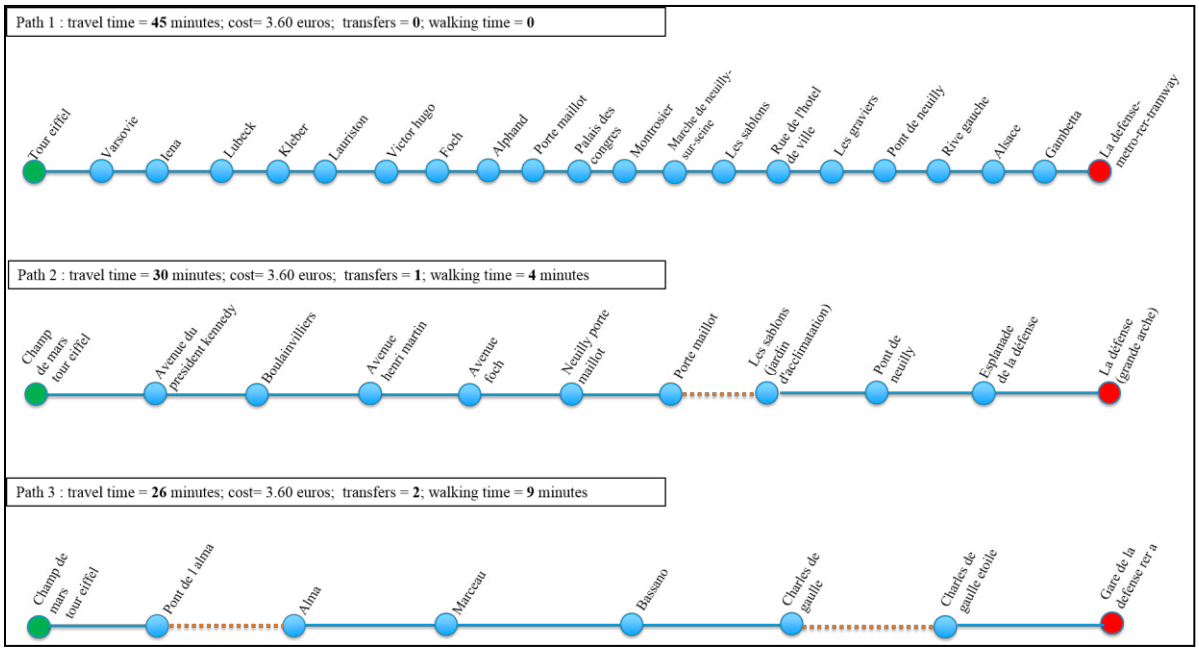

Figure 12: Example of three non-dominated paths found while going from the station of "tour Eiffel" to "La defense"

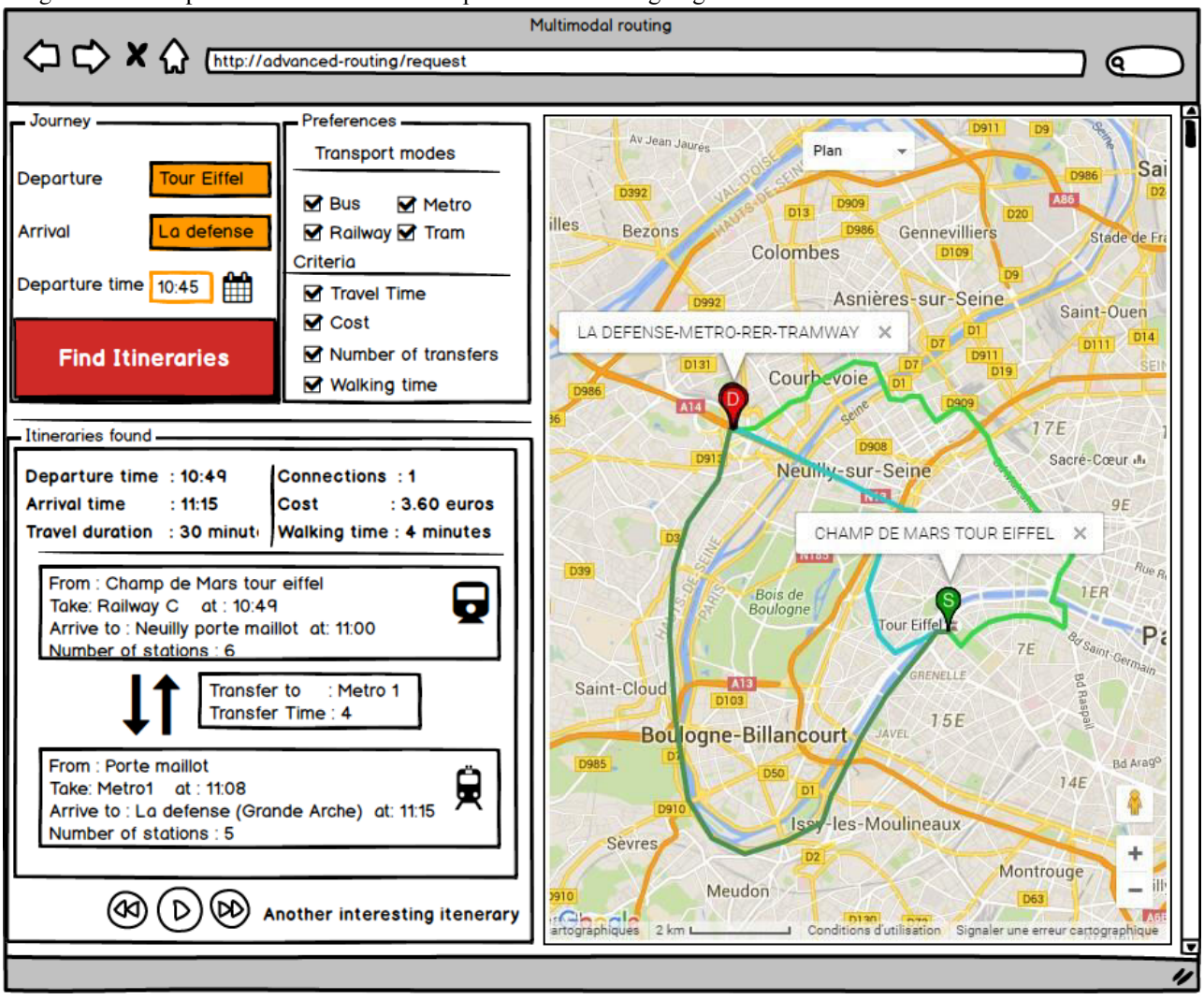

Figure 13: An advanced web-based routing application developed to assess the performance of our works

6 Conclusion

We have addressed in this paper the multicriteria routing issue that may arise while passengers accomplishing their journeys. After modelling the transportation network, a hybrid GA-VNS has been proposed to solve the emerging problem. We focused in this paper on four criteria: the travel time, the total monetary cost of a travel, the number of transfers and the total walking time. As transport modes, we used Railway, Bus Tram and Metro. The proposed work has been assessed by solving a wide range of real life itinerary planning problems defined on the Urban Public Transportation System of the French region Ile-De-France that includes the city of Paris and its suburbs.

Experimental results have shown that the computational time when using heuristic approaches is not prohibitive to integrate them within an online journey planning system. However, special attention should be paid to the quality of solutions that metaheuristics provide. In this work, we have proved that combining two metaheuristics (GA&VNS) is better than using each one solely. Therefore, we have developed an advanced routing system and we integrated the proposed GA-VNS into it as an advanced heuristic routing method.

As future works, we have planned to integrate train delays and other stochastic parameters. We believe that metaheuristic and especially GAs and VNS are good candidates to handle such additional constraints. Moreover, transport systems do not only encompass public transportation modes. Other modes such as Bike and Car Sharing also represent efficient alternatives for many passengers. Integrating such modes into our model is therefore one of our future goals. We also believe that adding such modes will increase the search space of the proposed GA-VNS. Therefore, it may not be efficient enough in its current version. To overcome that, we have planned to parallelize the genetic operators so that we enhance the global computational time, as well as, the quality of solutions.

Acknowledgments

This research work has been carried out in the framework of the Technological Research Institute SystemX, and therefore granted with public funds within the scope of the French Program “Investissements d’Avenir”.

References

Aarts, Emile, Jan Korst, and Wil Michiels. "Simulated annealing." Search methodologies. Springer US, 2014. 265-285.

Abbaspour, Rahim A., and Farhad Samadzadegan. "An evolutionary solution for multimodal shortest path problem in metropolises." Comput. Sci. Inf. Syst. 7.4 (2010): 789-811.

Ayed, H., & Khadraoui, D. (2008). Transfer graph approach for multimodal transport problems. Second International Conference MCO, Metz, France – Luxembourg.

Bast, H., Carlsson, E., Eigenwillig, A., Geisberger, R., Harrelson, C., Raychev, V., & Viger, F. (2010). Fast routing in very large public transportation networks using transfer patterns. In Algorithms–ESA 2010 (pp. 290-301). Springer Berlin Heidelberg.

Bast, Hannah, et al. "Route planning in transportation networks." arXiv preprint arXiv:1504.05140 (2015).

Behzadi, Saeed, and ALIA ALESHEIKH. "Developing a Genetic Algorithm for Solving Shortest Path Problem." NEW ASPECTS OF URBAN PLANNING AND TRANSPORTATION (2008).

Chakraborty, B., 2005. Simultaneous multiobjective multiple route selection using GA for car navigation. Pattern Recognition and Machine Intelligence. Springer-Verlag, Berlin, pp. 696-701.

Chakraborty, B., 2004. GA-Based Multiple Route Selection for Car Navigation. Applied Computing. Springer-Verlag, Berlin,pp. 76-83.

Civicioglu, Pinar, and Erkan Besdok. "A conceptual comparison of the Cuckoo-search, particle swarm optimization, differential evolution and artificial bee colony algorithms." Artificial Intelligence Review 39.4 (2013): 315-346.

Cotta, Carlos. Handbook of memetic algorithms. Eds. Ferrante Neri, and Pablo Moscato. Vol. 379. Heidelberg: Springer, 2012.

Corley H.W. and Moon I.D. (1985): Shortest paths in networks with vector weights. — J. Optim. Th. Applic., Vol. 46,No. 1, pp. 79–86.

Davies, C. and Lingras, P., 2003. Genetic algorithms for rerouting shortest paths in dynamic and stochastic networks. European Journal of Operational Research, 144(1), pp. 27-38.

Delling, D., Pajor T. & Wagner. D. (2009) Accelerating Multi-Modal Route Planning by Access-Nodes. 17th Annual European Symposium on Algorithms (ESA'09), Copenhagen, Denmark.

Deb, Kalyanmoy, et al. "A fast and elitist multiobjective genetic algorithm: NSGA-II." IEEE transactions on evolutionary computation 6.2 (2002): 182-197.

Delavar, M.R.; Samadzadegan, F. and Pahlavani, P., 2001. A GIS-assisted optimal urban route finding approach based on genetic algorithms. Dept. of Surveying and Geometrics, Faculty of Engineering, University of Tehran, Tehran, Iran, Commission II, Working Group II/5.

Dib, Omar, Marie-Ange Manier, and Alexandre Caminada. "Memetic algorithm for computing shortest paths in multimodal transportation networks." Transportation Research Procedia 10 (2015): 745-755.

Dib, Omar, Marie-Ange Manier, and Alexandre Caminada. "A Hybrid Metaheuristic for Routing in Road Networks." 2015 IEEE 18th International Conference on Intelligent Transportation Systems. IEEE, 2015.

Dib, Omar, et al. "Combining VNS with Genetic Algorithm to solve the one-to-one routing issue in road networks." Computers & Operations Research (2015).

E. Pyrga, F. Schulz, D. Wagner, C. Zaroliagis Efficient Models for Timetable Information in Public Transportation Systems ACM Journal on Experimental Algorithms, 12 (2.4) (2007), p. 39.

Evangelia Pyrga, Frank Schulz, Dorothea Wagner, and Christos Zaroliagis. Efficient models for timetable information in public transportation systems. ACM Journal of Experimental Algorithmics, 12(2.4):1–39, 2008.

Gen, M., Cheng, R. and Wang, D., 1997. Genetic algorithms for solving shortest path problems, In: Proceedings of 1997 IEEE International Conference on Evolutionary Computing, pp. 401- 406.

Glover, Fred, and Manuel Laguna. Tabu Search∗. Springer New York, 2013.

Hamacher, Horst W., Stefan Ruzika, and Stevanus A. Tjandra. "Algorithms for time-dependent bicriteria shortest path problems." Discrete optimization 3.3 (2006): 238-254.

Hansen, Pierre, and Nenad Mladenović. "Variable neighborhood search." Search methodologies. Springer US, 2014. 313-337.

Hochmair, H. H., 2008. Grouping of optimized pedestrian routes for multi-modal route planning: a comparison of two cities. The European Information Society. Springer, Berlin, pp.339-358.

Huang, B. and Cheu, R. L. et al. 2004. GIS and genetic algorithms for HAZMAT route planning with security considerations. International Journal of Geographical Information Science, 18(8), pp. 769-787.

Ikeda, Takahiro, et al. "A fast algorithm for finding better routes by AI search techniques." Vehicle Navigation and Information Systems Conference, 1994. Proceedings., 1994. IEEE, 1994.

Kumar, Rakesh, and Mahesh Kumar. "Reliable and Efficient Routing Using Adaptive Genetic Algorithm in Packet Switched Networks." IJCSI International Journal of Computer Science Issues 9.1 (2012): 1694-0814.

Kumar, Dr Rakesh, and Mahesh Kumar. "Exploring Genetic algorithm for shortest path optimization in data networks." Global Journal of Computer Science and Technology 10.11 (2010).

Liu, Lu, and Liqiu Meng. "Algorithms of multi-modal route planning based on the concept of switch point." Photogrammetrie Fernerkundung Geoinformation 2009.5 (2009): 431-444.

Martins E.Q.V. (1984): On a multicriteria shortest path problem. Europ. J. Oper. Res., Vol. 16, No. 1, pp. 236–245

Modesti, Paola, and Anna Sciomachen. "A utility measure for finding multiobjective shortest paths in urban multimodal transportation networks." European Journal of Operational Research 111.3 (1998): 495-508.

Pajor, T. (2009). Multi-modal route planning. Master thesis of Karlsruhe Institute of Technology.

Pyrga, E., Schulz, F., Wagner, D., & Zaroliagis, C. (2004). Towards realistic modeling of timetable information through the time-dependent approach. Electronic Notes in Theoretical Computer Science, 92, 85-103.

Skriver A.J.V. and Andersen K.A. (2000): A label correcting approach for solving bicriterion shortest path problems. — Comput. Oper. Res., Vol. 27, No. 6, pp. 507–524.

SteadieSeifi, M., et al. "Multimodal freight transportation planning: A literature review." European journal of operational research 233.1 (2014): 1-15.Liu, L. (2011). Data model and algorithms for multimodal route planning with transportation networks. Technische Universität München, München.

Surender Baswana, Somenath Biswas, Benjamin Doerr, Tobias Friedrich, Piyush P. Kurur, Frank Neumann: Computing single source shortest paths using single-objective fitness. FOGA 2009: 59-66

Talbi, El-Ghazali. "A unified taxonomy of hybrid metaheuristics with mathematical programming, constraint programming and machine learning."Hybrid Metaheuristics. Springer Berlin Heidelberg, 2013. 3-76.

Van Nes, R. (2002) Design of multimodal transport networks: a hierarchical approach. Ph.D thesis of Delft University of Technology.

Wang, M..Optimal paths based on Time and fare in transit network. Master thesis of National Cheng Kung University. 2008.

Yaseen, Saad Ghaleb, and Nada MA Al-Slamy. "Ant colony optimization."IJCSNS 8.6 (2008): 351.

Zhang, L., Li, J.Q., & Zhou, K. (2011). Design and implementation of a traveler in-formation tool with integrated real-time transit information and multi-modal trip planning. Transportation Research Board 2011 Annual Meeting, Washington, D.C.

Zografos, Konstantinos G.; Androutsopoulos, Konstantinos N.; Spitadakis, Vassilis "Design and Assessment of an Online Passenger In-formation System for Integrated Multimodal Trip Planning", IEEE Transactions on Intelligent Transportation Systems, Volume.10, Issue.2, pp.311, 2009, ISSN: 15249050.