Solving a Dial-a-Ride Problem with a Hybrid Multi-objective Evolutionary Approach: Application to Demand Responsive Transport

R. Chevrier, Arnaud Liefooghe, Laetitia Jourdan, Clarisse Dhaenens

Solving a Dial-a-Ride Problem with a Hybrid Evolutionary Multi-objective Approach: Application to Demand Responsive Transport

Rémy Chevrier[*,a], Arnaud Liefooghe[b,c], Laetitia Jourdan[b,c], Clarisse Dhaenens[b,c]

[a] *Université Lille Nord de France, IFSTTAR – ESTAS*
20 rue Élisée Reclus, 59650 Villeneuve d'Ascq, France
[b] *Université Lille 1, Laboratoire d'Informatique Fondamentale de Lille, UMR CNRS 8022*
Cité Scientifique, Bâtiment M3, 59655 Villeneuve d'Ascq cedex, France
[c] *INRIA Lille-Nord Europe, 40 avenue Halley, 59650 Villeneuve d'Ascq, France*

Abstract

Demand responsive transport allows customers to be carried to their destination as with a taxi service, provided that the customers are grouped in the same vehicles in order to reduce operational costs. This kind of service is related to the dial-a-ride problem. However, in order to improve the quality of service, demand responsive transport needs more flexibility. This paper tries to address this issue by proposing an original evolutionary approach. In order to propose a set of compromise solutions to the decision-maker, this approach optimizes three objectives concurrently. Moreover, in order to intensify the search process, this multi-objective evolutionary approach is hybridized with a local search. Results obtained on random and realistic problems are detailed to compare three state-of-the-art algorithms and discussed from an operational point of view.

Key words:
Demand responsive transport, Dial-a-ride problem, Evolutionary algorithm, Multi-objective combinatorial optimization, Local search

1. Introduction

Sparsely inhabited areas usually suffer from a lack of transport service, given that the authorities do not want to accept the cost of a transport service insufficiently used [1]. Demand responsive transport (DRT) tries to address this issue. Indeed, this service of people transportation is activated on demand only and involves the satisfaction of customers' demands. It is necessary for the customers to have booked the service by defining a pick-up point and a destination (delivery) at an arranged time. A DRT service manages a fleet of vehicles and aims at grouping as many customers as possible in the same vehicles in order to reduce the operational costs. Given that each customer has his own destination, the grouping and the routing are optimized according to several criteria and a set of constraints imposed by the capacity of the vehicles (number of seats) and the timetable which is defined by the pre-arranged pick-up and delivery times. One vehicle starts from a depot, then follows an itinerary along which it picks up customers and carries them to the destination while respecting the predefined timetable.

In its usual form, DRT is related to the dial-a-ride problem (DARP) [2, 3] or to the vehicle routing problem (VRP) [4]. Indeed, both problems consist in optimizing the vehicle routes by reducing the number of vehicles and the journey times. The VRP is the formulation of routing problems with loads to pick up and deliver [5], whereas the DARP formulates routing problems with passengers (one load equals 1). Another main difference between the DARP and the VRP lies in the precedence constraints imposed by the customer's journeys [6], and in the acceptance

[*]Corresponding author
Email addresses: remy.chevrier@ifsttar.fr (Rémy Chevrier), arnaud.liefooghe@univ-lille1.fr (Arnaud Liefooghe), laetitia.jourdan@lifl.fr (Laetitia Jourdan), clarisse.dhaenens@lifl.fr (Clarisse Dhaenens)

of delays (quality of service). Thus, a DRT service is a specific case of the DARP which is the academic formulation of a routing service with passengers.

Globally, the DARP may involve a set of objectives, usually conflicting, and which have to be optimized simultaneously. However, optimizing one objective often happens at the expense of the others. This is the reason why a multi-objective approach may be more than relevant in this context. In this paper, the DARP addressed is a multi-objective combinatorial optimization problem (MCOP) with conflicting criteria. Therefore, the search process aims at concurrently optimizing three objectives. The first one is *economic* and consists in minimizing the number of vehicles used in order to reduce the operational costs. The second one looks to reduce the duration of the vehicles' journeys. This objective could be seen as an *environmental* objective in so far as we try to limit the emission of pollutants (and also to reduce the carbon tax). The last objective minimizes the likely delays which may occur (*quality of service*). Besides, the approach is focused on finding a set of sub-optimal solutions, known as an approximation of the Pareto front when mapped into the objective space. A large number of MCOPs are known to be NP-hard and intractable [7], so that large-size problem instances cannot, in general, be solved exactly. Some exceptions can be noticed for small bi-objective [8, 9] and multi-objective [10] problems. Since the DARP is known to be NP-hard in its single-objective formulation [2], so is its multi-objective variant.

Although there are a lot of approaches solving the DARP [2], very few use evolutionary algorithms with a multi-objective approach. Indeed, they usually aggregate multiple objective functions for optimizing a single-objective problem [11, 12, 13, 14]. Although methods based on multi-objective evolutionary algorithms exist to solve the VRP [5, 15], these are not necessarily adapted to solve a DARP. However, a recent study deals with a multi-objective DARP, but not necessarily with evolutionary algorithms. In [16], a two-phase heuristic is proposed, but for two-objective cases only.

In this paper, the approach proposed to solve the DRT problem (DRTP) is the result of a preliminary work based on evolutionary algorithms [17]. An encoding mechanism based on a two-dimensional representation is provided, as well as a specific initialization strategy and adapted variation and improvement operators. Furthermore, a body of additional values are proposed to significantly improve the DRTP solving. To this end, a set of performance analysis is proposed, detailed and discussed, in particular: a calibration of crossover and mutation rates is carried out through two performance indicators; an iterative local search (ILS) is added in the mutation operator; an experimental analysis of performance along time is performed on several sets of instances (random, realistic and large-size). Finally, a comparison with and without the ILS is provided and analyzed.

The topic of the paper is to study the relevance of this kind of approach in an operational context. That is why several aims have to be achieved. One of them is to be capable of producing solutions in a short period of time. Indeed, a lot of DRT services usually impose booking several hours in advance and need methods allowing the reduction of this time. Besides, the use of a multi-objective approach would help decision-makers by providing them with a set of compromise solutions. This method will benefit from the features of evolutionary multi-objective optimization algorithms, which have received a particular interest over the last decades. Such methodologies have shown their efficiency to solve real-life problems [18]. In this perspective, the ILS will be used in the mutation operator in the same way as in a memetic algorithm [19]. The relevance of using an iterated local search in the mutation operator will be analyzed and discussed. As the instances used in the benchmarks are very often not relevant to real life, realistic instances will be used to assess the performance of the proposed approach. That is why the second aim of the approach is to be able to cope with both realistic and random instances. The third aim of this work consists of the comparison of three state-of-the-art evolutionary algorithms: the Non-dominated Sorting Genetic Algorithm II (NSGA-II) [20], the Strength Pareto Evolutionary Algorithm 2 (SPEA-2) [21] and the Indicator Based Evolutionary Algorithm (IBEA) [22]. The candidate algorithms are compared according to a set of different parameters: mutation and crossover rates, instances sizes and structures, computation time. These experiments tend to highlight the best candidate algorithm in an operational context.

The paper is organized as follows. The formulation of the problem under consideration is provided in section 2. The main principles of multi-objective optimization and the three candidate algorithms used in this work are presented in section 3. The encoding of the problem, as well as the initialization and the variation operators, are detailed in section 4. Results are provided and discussed in section 5. Finally section 6 concludes the paper.

2. Problem definition

In order to help the reader, the symbols used in the paper are summarized in Table 1.

2.1. Objectives formulation

The multi-objective problem under study can be formulated as a set of three objective functions to optimize ($f = (f_1, f_2, f_3)$) and a set of constraints to be taken into account. Problem solving is based on specific parameters, such as a relaxation and time windows which introduce more tolerance and flexibility to the slight delays which may occur during the journey. Since the DRTP is analogous to a DARP, the reader can refer to [2] to have mathematical models. Here we only detail the specificities of the DRTP and its multi-objective formulation. Therefore, in the DRTP under study, we aim to optimize three objectives: 1. minimize the number of vehicles used: function f_1 (eq. (2)); 2. minimize the journey durations: function f_2 (eq. (3)); 3. minimize the delays: function f_3 eq. (4).

$$f = (f_1, f_2, f_3) \tag{1}$$

$$f_1 = \min |\Lambda| \tag{2}$$

$$f_2 = \min \sum_{v \in \Lambda} t_v \tag{3}$$

$$f_3 = \min \sum_{v \in \Lambda} d_v \tag{4}$$

2.2. Introduction of delay tolerance and time windows

A usual DRT service uses the tolerance of the customers to accept delays. Making detours allows a vehicle to group the customers even if that produces delays. To deal with these delays, a coefficient of relaxation k_r is introduced and applied to the journey duration for defining a maximal delivery time. Let $t'_{r^+ \to r^-}$ be the slackened journey duration when $k_r > 1$: $t'_{r^+ \to r^-} = k_r . t_{r^+ \to r^-}$. Consequently, the maximal delivery time h'_{r^-} is defined as follows: $h'_{r^-} = h_{r^+} + t'_{r^+ \to r^-}$

In routing problems, the time windows define the time slots during which the picking up and the deliveries can be done. A time window at point r^+ is denoted w_{r^+} and its size is proportional to the theoretical journey duration to the point r^-: $w_{r^+} = k_w . t_{r^+ \to r^-}$, where k_w is a coefficient indicating the percentage of the duration allocated to the time window. In practice, the time window should not exceed a few minutes: for a ten-minute journey, the time window would last one minute if $w = 0.1$. Of course, for the very short journeys, a minimum time window should be defined (30 seconds for example).

2.3. Constraints

Flexibility is introduced by using relaxation and time windows while authorizing delays on the journeys. Nevertheless, in order to limit delays and keep a good quality of service, the likely delays must be limited by adding a set of constraints. Firstly, it is necessary to define what a feasible journey is. Let x, y be two points (pick-up or delivery) to be potentially connected, and H_x the effective starting time at point x. Indeed, as there may be some accumulated delays, H_x must take them into account. So, the journey from x to y is feasible *iff*:

$$\begin{array}{ll} H_x + t_{x \to y} \leq h_y + tw_y & \text{if } y \in V^+ \\ H_x + t_{x \to y} \leq h'_y & \text{if } y \in V^- \end{array}$$

Note that H is evaluated after each journey between two points and depends on the journey durations and whether the vehicle arrives before or after a theoretical time. Therefore, two cases may arise for determining H_y which corresponds to the effective time at point y coming from point x:

$$H_y = \begin{cases} h_y & \text{if } H_x + t_{x \to y} < h_y \\ H_x + t_{x \to y} & \text{otherwise.} \end{cases}$$

After calculating the effective visiting time, it is possible to evaluate the delay at a delivery point. A delay d corresponds to the difference between H and the theoretical arrival time, thus the delay d_y in a delivery point y is defined as follows:

$$d_y = \begin{cases} H_y - h_y & \text{if } H_y > h_y \\ 0 & \text{otherwise.} \end{cases}$$

Table 1: Definition of symbols used for the DRTP

Input data	
f_i	Objective function i
Λ	Set of the used vehicles
D	Depot of the vehicles
V	Set of the pick-up (V^+) and delivery (V^-) points such that $V = V^+ \cup V^-$
x, y	Arbitrary points such that $\{x, y\} \in V$
$t_{x \to y}$	Journey duration from x to y
d_x	Delay at a point x
R	Set of the requests
r	A request such that $r \in R$
r^+ (resp. r^-)	Pick-up (resp. delivery) point of the request r such that $r^+ \in V^+, r^- \in V^-$
q_r	Number of people of request r to be carried
v	A vehicle such that $v \in \Lambda$ When crossing point x, it is denoted: v_x
Q_v	Capacity of vehicle v, $v \in \Lambda$
h_{r^+}	Desired pick-up time
h_{r^-}	Theoretical arrival time
k_r	Relaxation coefficient
k_w	Coefficient for the time windows

Variables	
R_{min}	Set of minimal requests such that $R_{min} \subset R$
t_v	Amount of all journey durations between each point visited by vehicle v
d_v	Sum of each delay of vehicle v at delivery points
p_v	Number of passengers in vehicle v
H_x	Effective arrival time at point x
w_x	Time window at point x

Let p_v be the number of passengers currently in a vehicle v (initially, $p_v = 0$). p_v is updated whenever customers are delivered or picked up. In this latter case, the capacity constraint must be checked if customers (w_r) are picked up at a point r^+: $p_v + q_r \leq Q_v$

2.4. Definition of the minimal requests and the starting points

A request $r \in R$ is minimal *iff* no request $s \in R$ exists such that point s^+ can be crossed before r^+:

$$\nexists s \in R \setminus \{r\}, h_{s^+} + t_{s^+ \to r^+} \leq h_{r^+} \Leftrightarrow r \in R_{min}$$

In other words, all the requests whose pick-up points cannot be reached after another point have to be served first (if we except the depot D of course). Therefore, the pick-up point of a minimal request will be necessarily a starting point of one vehicle route. However, there is no reciprocity. A starting point can be the pick-up point of a non-minimal request.

As a consequence, a vehicle v, serving request r first, starts from depot D at time $h_D(v) \leq h_{r^+} - t_{D \to r^+}$ in such a way that it does not start its journey late.

3. Evolutionary Multiobjective Optimization

The DRTP under study is a multi-objective and NP-hard combinatorial optimization problem, so that large-size problem instances cannot generally be solved by exact methods. Therefore, we propose to adopt an evolutionary approach. First, we present multi-objective optimization principles and concepts. Then, we introduce three evolutionary multi-objective optimization algorithms which will be compared with each other in the experimental analysis.

3.1. Multi-objective Combinatorial Optimization

A general Multi-objective Combinatorial Optimization Problem (MCOP) can be defined by a set of n objective functions $f = (f_1, f_2, \ldots, f_n)$, a discrete set U of feasible solutions in the decision space. Let Z be the set of feasible points in the objective space $Z = f(U)$. Without loss of generality, we here assume that each objective function is to be minimized. To each solution $u \in U$ is assigned an objective vector $z \in Z$ on the basis of the vector function $f : U \to Z$ with $z = f(u) = (f_1(u), f_2(u), \ldots, f_n(u))$. An objective vector $z \in Z$ is said to *dominate* another objective vector $z' \in Z$ *iff* $\forall i \in \{1, 2, \ldots, n\}$, $z_i \leq z'_i$ and $\exists j \in \{1, 2, \ldots, n\}$ such that $z_j < z'_j$. We will also say that a decision vector $u \in U$ *dominates* a decision vector $u' \in U$ if $f(u)$ dominates $f(u')$. An objective vector $z \in Z$ is said to be *non-dominated iff* no other objective vector $z' \in Z$ exists such that z' dominates z. A solution $u \in U$ is said to be *efficient*, *Pareto optimal* if its mapping in the objective space results in a non-dominated point. The set of all efficient solutions is the *efficient set*, denoted by U_E. The set of all non-dominated vectors is the *Pareto front*, denoted by Z_N. A possible approach in MCOP solving is to find a minimal complete set of efficient solutions, *i.e.* one solution $u \in U_E$ for each non-dominated vector $z \in Z_N$ such that $f(u) = z$. However, generating the entire efficient set is usually unfeasible due to the complexity of the underlying problem. Therefore, the overall goal is often to identify a good approximation of it. Population-based metaheuristics in general, and evolutionary algorithms in particular, are commonly used to this end, as they are capable of finding multiple and well-spread non-dominated solutions in a single run [23].

3.2. Evolutionary Multi-objective Optimization Algorithms

Over the last decades, a very large number of evolutionary algorithms for MCOP solving have been proposed in the literature [23]. These approaches can be seen as white-boxes in which problem-related components have to be defined. The reader who wants more details on the outlines of a canonical evolutionary algorithm can refer to [23, 18]. In this paper, we will discuss three state-of-the-art evolutionary multi-objective optimization algorithms, namely NSGA-II [20], SPEA-2 [21] and IBEA [22]. Those algorithms all follow the same main steps [24], as illustrated in the flowchart of Fig. 1.

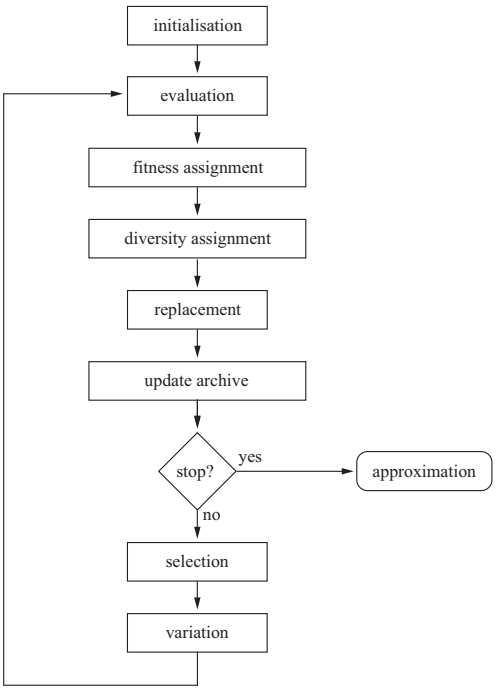


Figure 1: Flowchart of an evolutionary multi-objective optimization algorithm. The differences between NSGA-II, SPEA-2 and IBEA appear at the fitness and diversity assignment steps.

3.2.1. NSGA-II

NSGA-II [20] is probably the most widely used multi-objective resolution method. With each generation, solutions from the current population are ranked in several classes. Individuals mapping to vectors from the first non-dominated set all belong to the best efficient set; individuals mapping to vectors from the second non-dominated set all belong to the second best efficient set; and so on. Two values are then assigned to each population member. The first one corresponds to the *rank* to which the corresponding solution belongs, and represents the quality of the solution in terms of convergence. The second one, the *crowding distance*, consists in estimating the density of solutions surrounding a particular point of the objective space, and represents the quality of the solution in terms of diversity. One solution is said to be better than another if it has a best rank value, or in the case of equality, if it has the best crowding distance. The selection strategy is a deterministic tournament between two random solutions. At the replacement step, only the best individuals are kept with respect to a predefined population size. Notice that, in addition to the original NSGA-II, an external population is added, the so-called *archive*, in order to store the whole set of potentially efficient solutions found during the search.

3.2.2. SPEA2

In [21], the authors propose an extension of SPEA, in which an improved fitness assignment strategy is proposed. It intrinsically handles an internal archive of fixed size that is used during the selection step to create offspring solutions. At a given iteration of SPEA2, to each population and archive member u is assigned a strength value $S(u)$ representing the number of solutions it dominates. Then, the fitness value $F(u)$ of solution u is calculated by summing up the strength values of all individuals that solution x currently dominates. In addition, a diversity preservation strategy, based on a technique of the nearest neighbor, is incorporated. The selection step consists of a binary tournament with replacement applied on the internal archive only. Finally, given that the SPEA2 archive has a fixed-size storage capacity, a bounding mechanism based on fitness and diversity information is used when the size of the non-dominated set is too high. On the contrary, when the size of the non-dominated set is too small, some dominated solutions are allowed to be incorporated. Furthermore, an external archive is also added to store the whole set of non-dominated

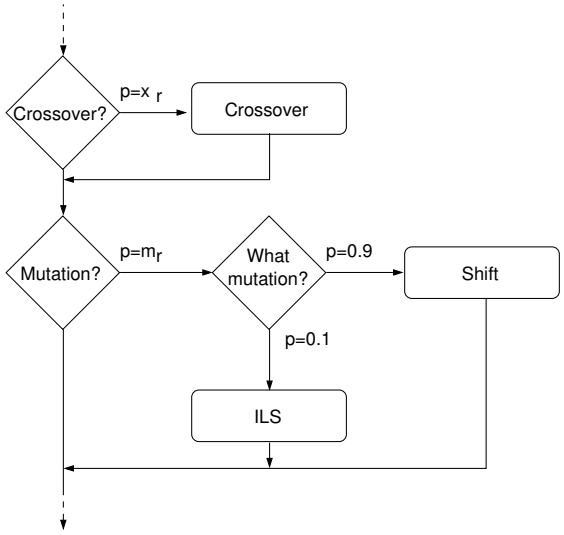


Figure 2: Flowchart of the variation step: description of the probabilistic choices of crossover and mutation (shift or ILS).

solutions found during the search process.

3.2.3. IBEA

Zitzler and Künzli [22] developed a more modern method which characterizes the new trend in evolutionary computation dealing with indicator-based search and which has become popular over recent years. The main idea is to introduce a total order between solutions by means of a binary quality indicator. The fitness assignment scheme is based on a pairwise comparison of solutions from the current population with regard to an indicator I. To each individual u is assigned a fitness value $F(u)$ measuring the 'loss in quality' if u was removed from the current population P, *i.e.* $F(u) = \sum_{u' \in P \setminus \{u\}} (-e^{-I(u',u)/\kappa})$, where $\kappa > 0$ is a user-defined scaling factor. Different indicators can be used for such a purpose and we here choose to use the binary additive ϵ-indicator ($I_{\epsilon+}$) as defined in [22]. $\mathrm{I}_{\epsilon+}(u, u')$ gives the minimum value by which a solution $u \in U$ has to or can be translated in objective space to weakly dominate another solution $u' \in U$. The selection for reproduction consists of a binary tournament between randomly chosen individuals and the selection for replacement consists in iteratively removing the worst solution from the current population until the required population size is reached. The fitness information of the remaining individuals is updated whenever one is deleted. Similarly, an archive is added to store non-dominated solutions in order to prevent their loss during the stochastic search process.

4. Evolutionary Multi-objective Optimization for the DRTP

In order to solve the DRTP, it is necessary to define a relevant representation for the problem under consideration. Firstly, an appropriate encoding has to be chosen in order to depict the vehicle paths accurately. The other evolutionary mechanisms, including initialization, crossover and mutation, are next detailed. Flowchart of Fig. 2 details the variation step, that is, the probabilistic choices of crossover and mutation, where the ILS can be invoked.

4.1. Solution representation

The main idea of the proposed representation is to facilitate the reading of the vehicle routes. So, we use a two-dimensional representation: a vector of routes. Each of them corresponds to a single vehicle route. The sequence of data literally indicates the sequence of points by which the vehicle passes. Even if it is not mentioned in the solution, D is the start and final point of a route because it is the depot of the vehicle. To avoid the problem of points precedence, a cell indicates a request identifier and not the point itself in such a way that we can retrieve the associated point by

counting the number of times the corresponding request identifier is encountered. If a request r appears for the first time, it is necessarily the pick-up point r^+, otherwise it is the delivery point r^-. Note that each route has an even number of points.

The examples of Fig. 3 (b,c,d) are solutions to a DRTP instance. For each solution, the associated encoding is depicted. Each vector i translates the route of vehicle v_i. The order of the values in the cells indicates the sequence of the points by which a vehicle passes. As example, the encoding of solution in Fig. 3(d) is detailed and explained hereafter.

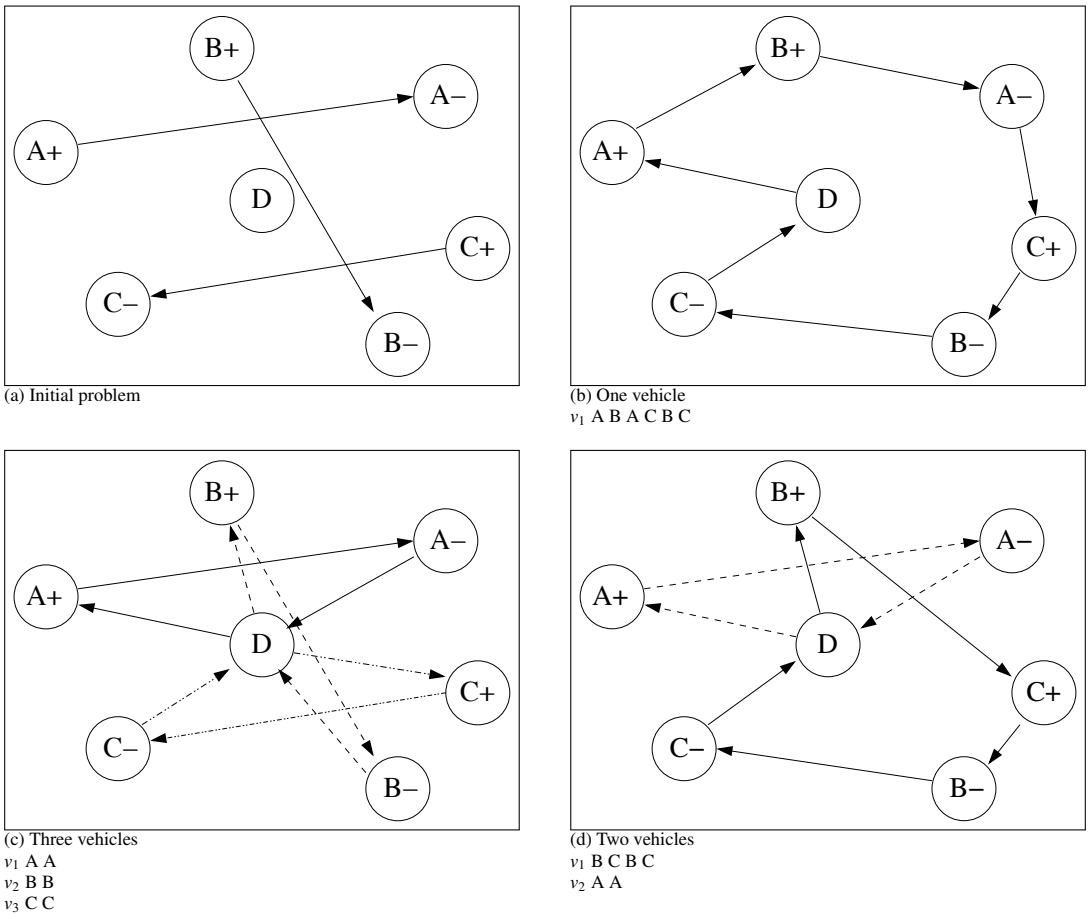


Figure 3: Examples of solution encoding: (a) Initial problem with three customers A, B, C, (b) a solution with one vehicle (one route), (c) a solution with three vehicles (three routes), (d) a solution with two vehicles (two routes).

Reading of a solution. As example, the reading mechanism of solution in Fig. 3(d) is detailed in Fig. 4. Given that it is composed of two vector-routes (v_1, v_2), two vehicles are used to transport the customers. Each value is counted in order to identify whether it is a pick-up point or a delivery. Therefore, when a value occurs for the first time, the associated point is inevitably the pick-up point of the request, whereas, when the occurrence reaches two, the point is necessarily the delivery of the request. As soon as the associated points are defined, the vehicle route is retrieved by simply adding the depot D as first and final point of the route.

4.2. Population initialization

Before initializing the population, set R_{min} has to be determined by searching in R all the minimal requests as defined before. The initialization strategy consists in randomly adding a request to a vehicle in such a way that the

Vehicle v_1				
Encoding	B	C	B	C
Occurrence	1	1	2	2
Associated point	B^+	C^+	B^-	C^-
Final route: $D \rightarrow$	$B^+ \rightarrow$	$C^+ \rightarrow$	$B^- \rightarrow$	$C^- \rightarrow D$

Vehicle v_2			
Encoding	A	A	
Occurrence	1	2	
Associated point	A^+	A^-	
Final route: $D \rightarrow$	$A^+ \rightarrow$	$A^- \rightarrow$	D

Figure 4: Reading mechanism of solution of Fig. 3(d)

search process starts with a population of feasible solutions. Initially, we begin by randomly assigning each minimal request to a distinct random vehicle. Then, each non minimal request is, if possible, randomly added to an existing vehicle route. Otherwise, if the request cannot be added, a new vehicle is assigned for serving the request: one route is added to the solution.

Let us note that the solution size l can be bounded. Indeed this size corresponds to the number of vehicles used by the solution: $l = |\Lambda|$. Hence, l belongs to a range between the number of minimal requests $|R_{min}|$ which is the lower bound of the number of starting points, and the total number of requests R which is the upper bound (one vehicle *per* request):

$$|R_{min}| \leq l \leq |R|$$

Fig. 5 illustrates how the initialization process works. In the example, there are four customers: $R = \{r_A, r_B, r_C, r_X\}$ and $R_{min} = \{r_A, r_B, r_C\}$ ($R_{min} \subset R$). Each minimal request is randomly assigned to a single vehicle (Fig. 5(a)). Then, the second step consists in adding all non-minimal requests to the solution. So, each of them is randomly chosen and then we check whether it may be added to a random route, or not. For example, we try to add request r_X to randomly chosen route v_2 (Fig. 5(b)). If this does not work, we try to add it to another route (Fig. 5(c,d)) and so on until a feasible route expansion is possible or until no other vehicle, that has already been used, is available. In the latter case (Fig. 5(e)), no route can receive the request so that we add a new route v_4 with request r_X. In other words, an additional vehicle is used to carry the customer.

v_1 C C
v_2 A A
v_3 B B

(a) Initial encoding: three minimal requests r_A, r_B, r_C denoted A, B and C

v_1 C C
v_2 A A + X X ?
v_3 B B

(b) Try 1: add r_X to v_2

v_1 C C + X X ?
v_2 A A
v_3 B B

(c) Try 2: add r_X to v_1

v_1 C C
v_2 A A
v_3 B B + X X ?

(d) Try 3: add r_X to v_3

v_1 C C
v_2 A A
v_3 B B
v_4 **X X**

(e) No feasible assignment to existing genes: addition of new gene v_4

Figure 5: Example of initialization with four requests: r_A, r_B, r_C, r_X. $\{r_A, r_B, r_C\} \in R_{min}$

4.3. Evaluation

The evaluation function $F = (f_1, f_2, f_3)$ applied to a solution S computes a 3-vector of values corresponding to the respective objective functions. At this level, the feasibility of the solution is checked by ensuring that the journeys from one point to another respect the constraints mentioned before: the time conditions and the vehicle capacities. In the case of unfeasibility of the solution, it is penalized by receiving the worst objective values: given that the objective functions have to be minimized, huge values are assigned.

Now, solution S is defined by a 3-vector $(f_1(S), f_2(S), f_3(S))$ representing it in objective space. This vector is used to compute the dominance of the solution according to the algorithm under consideration. NSGA-II searches for its

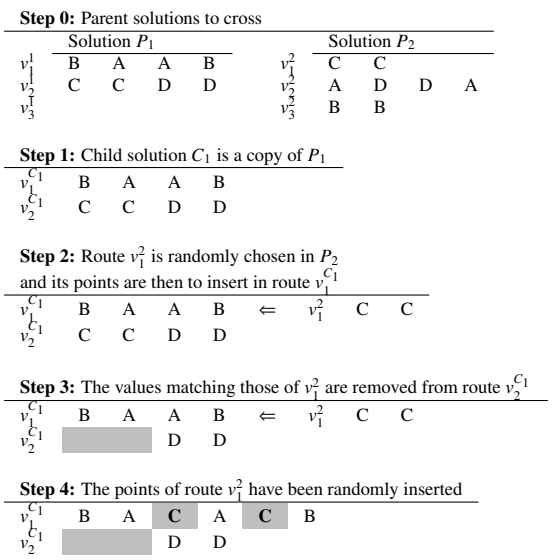


Figure 6: Example of crossover of two solutions P_1, P_2 for producing a new individual C_1 (Same construction for solution C_2 by copying P_2 and inserting a random route v_λ^1)

dominance rank, SPEA2 determines its Pareto strength and IBEA computes a fitness value indicating the quality of the solution.

4.4. Crossover operator

The crossover operator aims at producing new individuals (offspring solutions) from two individuals in the current population (parents). Let P_1, P_2 be two solutions selected from the population and C_1, C_2 two solutions built from P_1, P_2. Note that C_1 (resp. C_2) is the modified copy of P_1 (resp. P_2).

The construction of solution C_1 is done as follows: (step 1) copy P_1 in C_1 then (step 2) randomly choose a route of P_2 at position λ (route v_λ^2). v_λ^2 contains the requests that will be reassigned to C_1. However, in order to avoid duplicated data, (step 3) the requests of C_1 matching those of v_λ^2 are first removed. Thereafter, (step 4) the v_λ^2 data can be randomly inserted in the counterpart route of C_1 at position λ.

Fig. 6 illustrates how the crossover works. In this example, route v_1^2 of P_2 is randomly chosen and its values (C C) have to be assigned to the counterpart route of P_1. These values are randomly inserted into route v_1 of C_1. If the solutions to be crossed are not the same size (*i.e.* different number of routes) and if the route to be filled does not exist, a new route (one vehicle) is added to the solution. Producing C_2 follows the same process by inverting P_1 and P_2.

Let us note that an unfeasible solution can be produced. At this level, no mechanism is used to fix the solution. Nevertheless, it will be penalized during the evaluation step as explained before, in such a way that it does not appear in the next population.

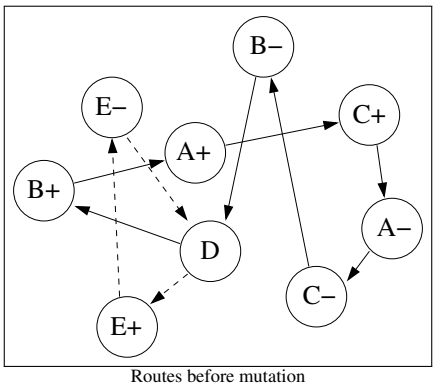
4.5. Mutation operator and local search

A mutation operator generally aims at bringing diversity into a population. However, the mutation step can be used to intensify the search with a local search, for example. That is why we propose to introduce two candidate mechanisms: shifting one customer to diversify the solutions and using a local search (LS) to improve a route. The mechanism used is randomly chosen according to predefined rates in such a way there is an alternation between a shift and an LS.

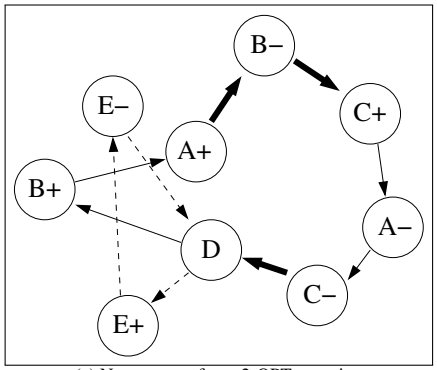
Thus, we propose to use a Hill Climbing (HC) as local search at this level. Indeed, HC is a very easy method to design and implement and gives fairly good results quickly [18]. As the main drawback of HC is the convergence toward local optima, some variants to HC have been proposed to avoid converging. That is why we use an ILS, which iterates from different initial solutions. The operator used by the ILS is the well-known 2-OPT operator which is applied on a sequence of points of one route. This operator has been chosen because it is known to be very efficient

on a single vehicle routing problem. Its working is illustrated on Fig. 7(a) and consists in reversing a sequence of points of one route. In the example, the sequence (C A C B) becomes (B C A C). The sequence length to reverse as well as the choice of the route are random. Thus, the use of the 2-OPT operator will quickly improve the second objective function. Other approaches where a vehicle routing problem-related heuristic is successfully integrated into an evolutionary multi-objective optimization algorithm can be found in [25, 26].

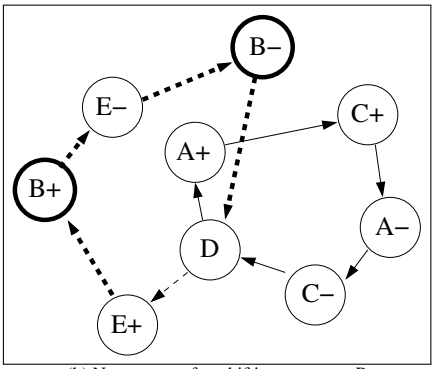
In most cases (90%), a random customer is simply shifted from one route to another (Fig. 7(b)) and the pair values are randomly inserted in the host route. In the remaining cases (10%), an ILS is used to optimize a single objective function: f_2. Even if the likelihoods of mechanism choice have been empirically decided, it seems obvious that shifting customers more frequently than using ILS will allow us to modify routes and hence to create new combinations. Indeed, using ILS too frequently may risk in not improving existing routes and in saturating. This empirical balance has been chosen due to the good results observed.



Routes before mutation



(a) New routes after a 2-OPT mutation



(b) New routes after shifting customer B

2-OPT operator in ILS

C_1		C'_1
B A **C A C B**	⇒	B A **B C A C**
E E		E E

Shifting one customer

C_1		C'_1
B A C A C **B**	⇒	A C A C
E E		E **B** E **B**

Figure 7: Example of mutation of chromosome C_1 into chromosome C'_1

4.6. Experimental Design

4.6.1. Benchmark Test Instances

Three sets of instances are used for the experiments[1]. These instances are built on realistic geographical data [1]. All pick-up or delivery points are geo-localized stops and the point-to-point matrix of the shortest durations was built

[1]These benchmark test instances can be retrieved on the web from the following URL: `http://sites.google.com/site/remychevrier/chevrier_instances.tgz`.

on a geographical information system.

The first set, denoted *'Rnd100'*, contains 10 instances of 100 randomly generated requests, whereas the second set named *'Gravit100'* contains 10 instances of 100 requests generated according to a geographical model of people or freight flows. This model called *'gravity model'*, as well as other interaction models, allows us to quantify people flows in a geographical space according to a calibration of the model with data, such as inhabitants densities, distances, time dimension [27]. The third set, denoted *'Gravit1000'*, has 10 instances of 1,000 requests generated according to the same model as for set *Gravit100*.

Globally, the *'Rnd100'* set has instances with quasi-homogeneous distribution of customers, while the sets *'Gravit100'* and *'Gravit1000'* have instances with non-homogeneously distributed flows representing a time slot between 8:00am and 9:00am. This distribution is the result of the existence of attractive areas to which most flows converge, instead of less attractive areas, such as residential downtowns which are rather areas of emission.

4.6.2. Performance Assessment

The experiments are performed on PC (3.0 GHz with 6 GiB) running Linux release of framework ParadisEO [24]. Its implementation of ILS has been used in our program. Note that parallel computing options are not used.

In order to evaluate the quality of the approximations obtained for every instance, the protocol proposed in [28] has been followed. For a given instance, let Z^{all} denote the union of the outputs we obtained during all our experiments. We first compute a reference set $Z_N^\star$ containing all the non-dominated points of Z^{all}. Second, we define $z^{min} = (z_1^{min}, \ldots, z_n^{min})$ and $z^{max} = (z_1^{max}, \ldots, z_n^{max})$, where z_k^{min} (resp. z_k^{max}) denotes the lower (resp. upper) bound of the k^{th} objective for all the points contained in Z^{all}. In order to give a roughly equal range to all objective functions, values are normalized with respect to z^{min} and z^{max}. Then, to measure the quality of a given output set A in comparison to $Z_N^\star$, we compute the difference between these two sets by using the unary hypervolume metric [29], z^{max} being the reference point. The hypervolume difference indicator (I_H^-, Fig. 8(a)) computes the portion of the objective space that is dominated by $Z_N^\star$ and not by A. Furthermore, we also consider the additive ϵ-indicator proposed in [29]. The unary additive ϵ-indicator ($I_{\epsilon+}^1$, Fig. 8(b)) gives the minimum factor by which an approximation A has to be translated in objective space to dominate the reference set $Z_N^\star$. Note that both I_H^-- and $I_{\epsilon+}^1$-values are to be minimized.

Since we want to assess the evolution of the algorithms efficiency over time, a run is divided into a certain number of 'tops', which happen every five seconds over a run of one minute (12 tops) or every minute over a run of 10 minutes (10 tops). Hence, the computation time is input data and is used as the stopping criterion. So, the best population is archived every top along the run. Every instance is optimized 20 times (20 different seeds used for each instance). Then, we obtain 20 I_H^- measures and 20 $I_{\epsilon+}$ measures, corresponding to the 20 runs, *per* algorithm and *per* top. When all these values are computed for a given time, an average value for each metric *per* algorithm can be computed.

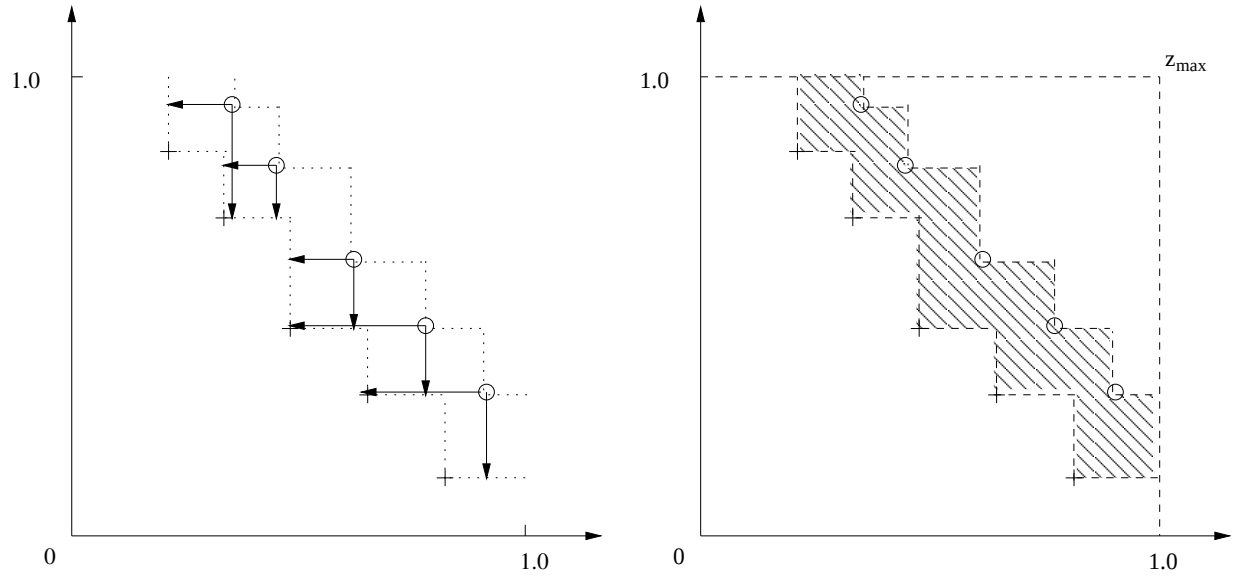
4.6.3. Parameter Setting

In every run, the population is composed of 100 individuals. Fixing the computation time at one minute allows to outline the feasibility of a real-time service whereas a computation time of 10 minutes allows us to show the gain in terms of quality of solution. In practice, some services just need a booking time fixed at 30 minutes, such as Modulobus or Tad'Mod services experimented and utilized in Montbéliard (Eastern France) [1].

Crossover and mutation rates are calibrated as explained in the following paragraph. In addition, specific parameters exist for some algorithms. Hence, SPEA2 requires an internal, fixed-size archive whose size is set to 100 individuals. Moreover, following [22], the scaling factor κ of IBEA is set at 0.05.

Calibration of mutation and crossover rates. In order to calibrate mutation and crossover rates, three rates are compared for each recombination operator. hybrid NSGA-II (NSGA-II$_H$) is used to solve the instances of set *Gravit100* over 1 minute of computation. We first deal with the mutation and then with the crossover.

With a crossover rate fixed at 0.9, three mutation rates are compared: 0.1, 0.5, 1.0. The evolution of the quality of solutions is computed according to the protocol described before and three curves are depicted on Fig. 9(a,b) (one curve per mutation). Whatever the indicator under consideration ($I_{\epsilon+}^1$ on the left and I_H^- on the right), the clear result of this comparison is that a mutation rate of 0.5 allows the algorithm to have the best performance all along the run. In fact, we can assume that the mutation rate of 1.0 may limit the total number of iterations over the run due to the ILS occurring, but also to the number of crossovers. That is why we keep a mutation rate of 0.5 for the continuation.



(a) Indicator $I^1_{\epsilon+}$ (b) Indicator I^-_H

Figure 8: Comparison of the quality of Pareto solution sets according two indicators: $I^1_{\epsilon+}$ and Indicator I^-_H in the objective space (approximation set 'o' is compared with reference set '+')

Furthermore, this study of the mutation rates also shows the significant contribution of the mutation in the improvement of the quality of solutions and hence, the contribution of the ILS in the mutation.

In the same way, three crossover rates are compared: 0.1, 0.5, 1.0. The three curves depicted in Fig. 9(c,d) tend to indicate that the greater the crossover rate, the better the quality of solutions. Nevertheless, the curves representing the rates 0.5 and 1.0 also seem to show that there are not so many differences between both rates, and, hence, that the contribution of a more frequent crossover is not really significant beyond 0.5. That is why we choose a crossover rate fixed at 0.9 for the continuation of the paper.

5. Results and Discussion

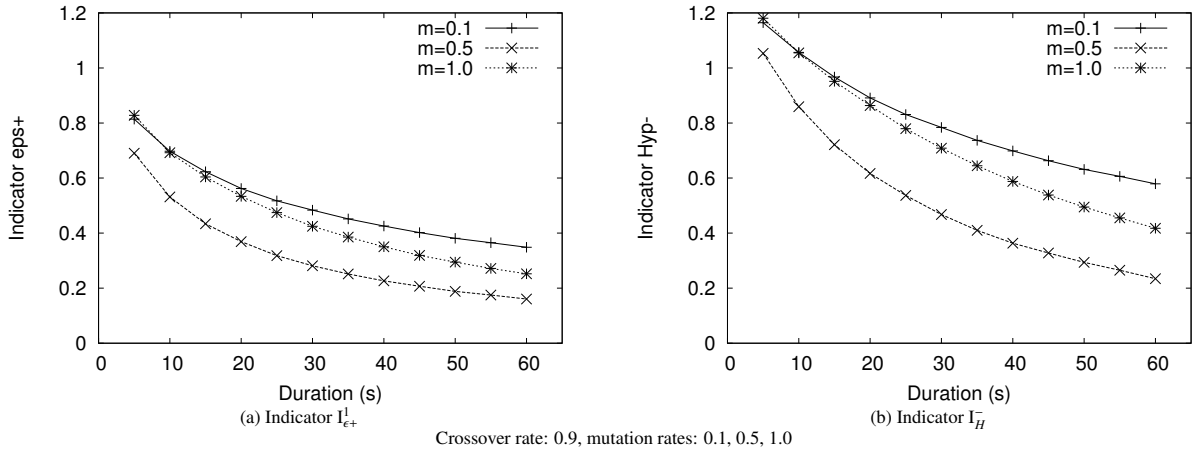
After the preliminary study based on NSGA-II$_H$ for calibrating the parameters, the comparison of the three candidate algorithms is performed according to the same protocol mentioned and used before. First, two studies of performance along time are considered: sets *Gravit100* and *Rnd100* over **10 minutes** of computation; set *Gravit1000* over **10 minutes** of computation. Then, a comparison between hybrid and standard algorithms after 10 minutes of computation is provided. The hybrid algorithms are denoted: NSGA-II$_H$, SPEA2$_H$, IBEA$_H$.

5.1. Case study 1: 100-request instances

In this first case study, we are interested in the performance of the three candidate algorithms over ten minutes of computation applied to two different sets of instances. The protocol used is exactly the same as before and allows us to show the evolution of the quality of solutions all along the run. The results of these evolutions are represented by the curves of Fig. 10. In detail, the curves of Fig. 10(a,b) and 10(c,d) are respectively obtained for set *Rnd100* and set *Gravit100*.

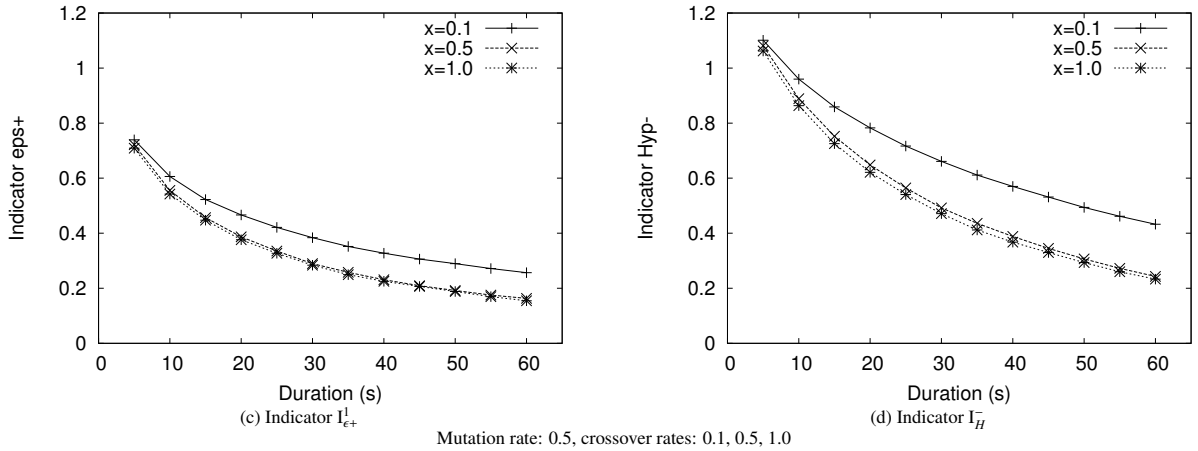
The runs are performed with a top every minute and two additional tops at 5 seconds and 30 seconds are taken into account in order to see the average evolution during the first minute of computation. The results are illustrated by the curves on Fig. 10.

First, we focus on the random instances (Fig. 10(a, b)). At first sight, whatever the indicator under consideration, the outcomes obtained show that IBEA$_H$ converges far more quickly towards the best-found solutions than the two



(a) Indicator $I^1_{\epsilon+}$ (b) Indicator I^-_H

Crossover rate: 0.9, mutation rates: 0.1, 0.5, 1.0



(c) Indicator $I^1_{\epsilon+}$ (d) Indicator I^-_H

Mutation rate: 0.5, crossover rates: 0.1, 0.5, 1.0

Figure 9: Average evolution of the quality of solutions of 10 realistic instances over 20 runs (1 minute *per* instance, NSGA-II$_H$ used) according to three crossover and three mutation rates.

other algorithms. In detail, we can see that a great part of the optimization is achieved around the first minute and that the improvement of the quality of solutions is quite slow after two minutes.

Even if the evolution of the quality of solutions obtained by NSGA-II$_H$ is approximately the same as IBEA$_H$, NSGA-II$_H$ appears all the same less efficient than IBEA$_H$. Besides, its results are equivalent to those of IBEA$_H$ only from ten minutes. Concerning SPEA2$_H$, it is always outperformed by its two challengers even if it reaches practically the same quality after ten minutes of computation according to indicator I^-_H. Finally, the algorithms can be ranked as follows: 1. IBEA$_H$, 2. NSGA-II$_H$, 3. SPEA2$_H$. These first results tend to state that IBEA$_H$ converges faster than the other two algorithms.

If we are now interested in the realistic instances (Fig. 10(c, d)), the quality of solutions follows practically the same evolution as before, whatever the indicator. Furthermore, even if the evolution seems to be a bit slower than for random instances, a great part of the optimization is achieved after around two minutes according to indicator $I^1_{\epsilon+}$ or after around three minutes according to indicator I^-_H. Moreover, the ranking of the algorithms is exactly the same. These results come to strengthen the previous assumptions which assert that IBEA$_H$ is the fastest algorithm to solve a DRTP instance if we assume that optimizing better over the same period of time means that it is quicker.

It is interesting to note that after ten minutes of computation the quality of solutions obtained by NSGA-II$_H$ or SPEA2$_H$ are a bit weaker than those obtained by IBEA$_H$ (contrary to the random instances for which NSGA-II$_H$ reaches the same results after the same time). In addition, the different kinds of instance do not seem to influence the

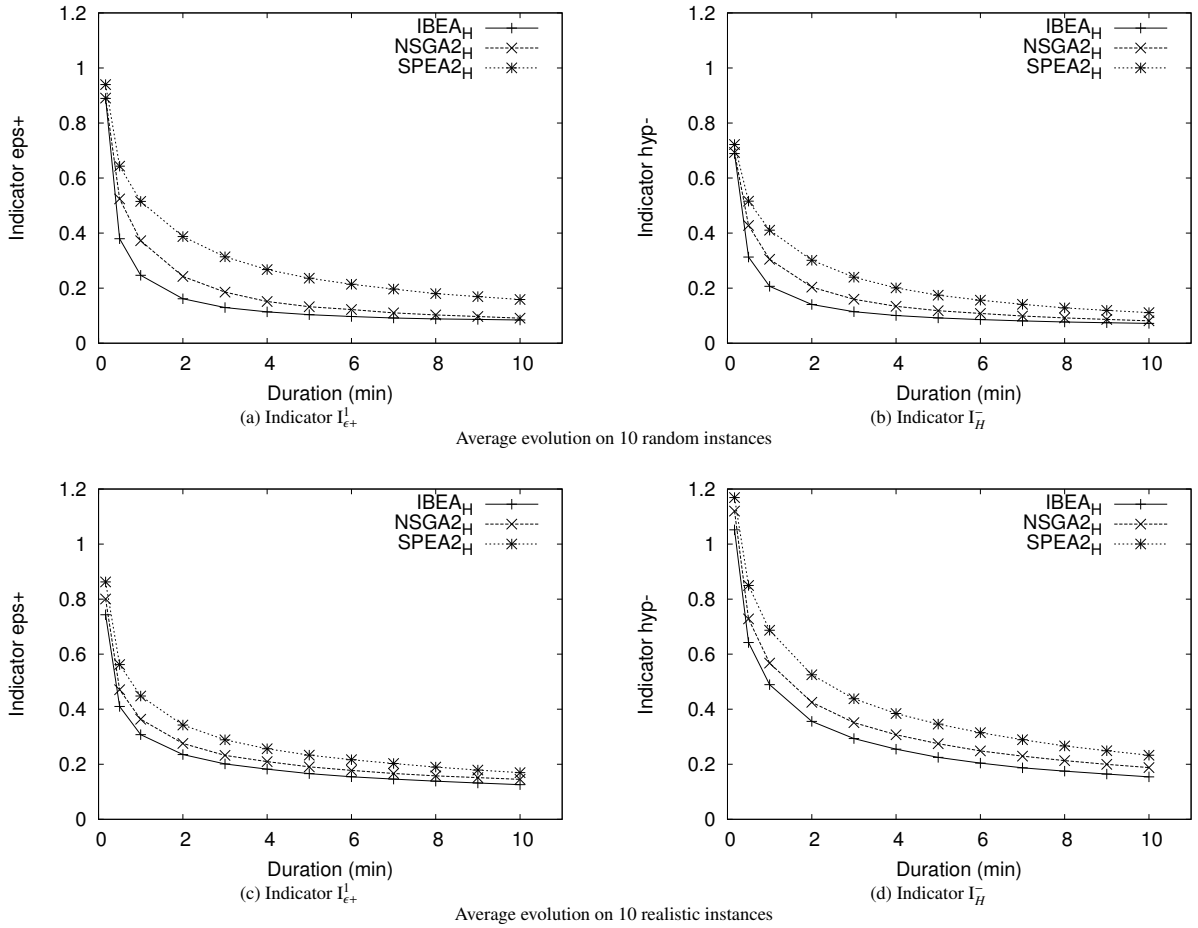


Figure 10: Average evolution of quality of solutions on 10 realistic and 10 random instances over 20 runs (100 requests *per* instance, **10 minutes *per* run**, mutation: 0.5, crossover: 0.9).

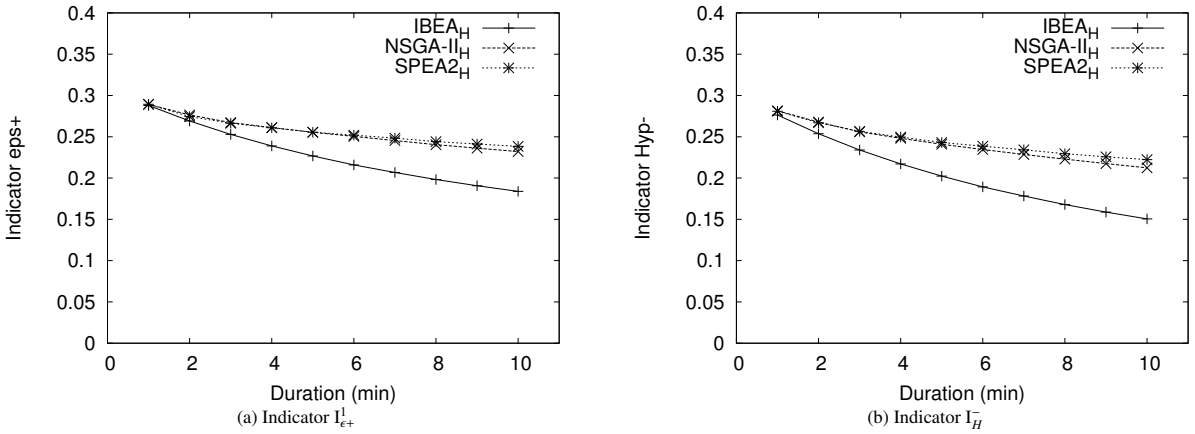
efficiency of the algorithm. Indeed the average evolution of the quality of solutions seems to have the same behavior whatever the indicator or the computation time (the tendency is the same at the beginning as at the end).

From an operational point of view, the capability of producing solutions in a short period of time, such as two minutes, would allow us to reduce the booking time and, hence, to consider very reactive DRT services.

5.2. Case study 2: 1,000-request instances

The second study is performed on realistic instances only, because the same behavior of the algorithms has been shown, whatever the kind of instances. Here we are interested in the capacity of the approach to face a very great workload even if it is highly unlikely to have so many customers at the same time. The instances of set *Gravit1000* have 1,000 requests to be served and each run is performed over 10 minutes with a 'top' every minute. The results and the underlying curves depicted in Fig. 11 are produced in the same way as before. Contrary to the previous experiments, it is obvious that optimizing in this context is far more difficult. Therefore, the average evolution of the quality of solutions should highlight this difficulty.

As could be assumed, the improvement of the quality of solutions is quite different according to the algorithm under study. Whatever the indicator under consideration, for algorithms NSGA-II$_H$ and SPEA2$_H$, the evolution is almost stagnant and the improvement is weak during the run. Even if they seem to have the same efficiency, NSGA-II$_H$ is very slightly better than SPEA2$_H$ at the end of the run.



(a) Indicator $I^1_{\epsilon+}$ (b) Indicator I^-_H

Figure 11: Average evolution of quality of solutions on 10 realistic instances over 20 runs (1,000 requests *per* instance, **10 minutes *per* run**, mutation: 0.5, crossover: 0.9)

On the other hand, it clearly appears that IBEA$_H$ is the lone algorithm capable of improving the quality of solutions all along the run. Even if the improvement is weak, it is significantly better than for NSGA-II$_H$ and SPEA2$_H$. The difference of evolution between the algorithms underlines that IBEA$_H$ suffers less because of the workload than the other two candidate algorithms. This shows IBEA$_H$ is more robust than SPEA2$_H$ and NSGA-II$_H$ to tackle the problem scalability.

5.3. Standard vs hybrid algorithms

The candidate algorithms, with and without hybridization, are compared on the mentioned sets of instances after 10 minutes of computation. The protocol used is the same as before: two performance indicators are used. That allows the ranking of the algorithms according to their performance at the end of the computation. Table 2 provides detailed results and it appears that algorithms, hybridized with an ILS, are better than their standard counterparts. Furthermore, it has to be noted that IBEA$_H$ is always the best algorithm, whatever the indicator or the kind of instance. It is also interesting to see that IBEA's performance is globally very close to NSGA-II$_H$'s one: NSGA-II$_H$ is second for all ranking except that considering indicator I^-_H for random instances, where IBEA is the second best algorithm. For the remainder, SPEA2 is outperformed in all cases and its hybrid variant is better than NSGA-II in just two cases: for the realistic instances whatever the indicator under consideration.

Table 2: Comparison and ranking of the algorithms (standard and hybrid) applied to 10 realistic and 10 random instances over 20 runs through two indicators (100 requests *per* instance, **10 minutes *per* run**, mutation: 0.5, crossover: 0.9).

Ranking of the algorithms applied to random instances

Rank	Algorithm	Avg. value $I^1_{\epsilon+}$	Rank	Algorithm	Avg. value I^-_H
1	IBEA$_H$	0.0888769	1	IBEA$_H$	0.0761573
2	NSGA-II$_H$	0.0951126	2	IBEA	0.0854913
3	IBEA	0.0984397	3	NSGA-II$_H$	0.0858838
4	NSGA-II	0.1059630	4	NSGA-II	0.0964697
5	SPEA2$_H$	0.1655520	5	SPEA2$_H$	0.1162990
6	SPEA2	0.1757240	6	SPEA2	0.1234380

Ranking of the algorithms applied to realistic instances

Rank	Algorithm	Avg. value $I^1_{\epsilon+}$	Rank	Algorithm	Avg. value I^-_H
1	IBEA$_H$	0.144561	1	IBEA$_H$	0.181033
2	NSGA-II$_H$	0.165714	2	NSGA-II$_H$	0.217364
3	IBEA	0.180698	3	IBEA	0.243825
4	SPEA2$_H$	0.195997	4	SPEA2$_H$	0.267167
5	NSGA-II	0.198705	5	NSGA-II	0.277367
6	SPEA2	0.246288	6	SPEA2	0.355923

6. Conclusions and Perspectives

In this paper, the multi-objective DRTP has been tackled by a new approach in order to be used in a DRT service. This approach based on a three-objective optimization proposes a new encoding of the problem under study, as well as new variation and improvement operators. The originality of one of these operators lies in the hybridization at the mutation step. Indeed, the mutation usually consists of the diversification of the population while randomly mutating individuals, but it is possible to introduce a local mechanism of optimization, as in memetic algorithms. In our approach, it consists in using an ILS which optimizes one objective only. This ILS applies the well-known 2-OPT operator on a random sequence of locations of one route in order to reduce the journey duration.

Several points have been dealt with. The first was, of course, to show the relevance of the ILS in the mutation step of the evolutionary algorithm. By using several mutation rates, the efficiency of the approach has been observed and that has allowed us to calibrate the parameters for the next experiments.

One part of the experiments was performed in order to analyze the performance of three hybrid algorithms along time: NSGA-II$_H$, SPEA2$_H$, IBEA$_H$. The outcome of the experiments clearly shows that IBEA$_H$ is more efficient than its two challengers. Indeed, IBEA$_H$ has the best performance all along the run of ten minutes, whatever the indicator under consideration. Moreover, this analysis has been enhanced by a study of the efficiency of the approach on huge instances (1,000 customers). Although it is unlikely to have so many customers at the same time, it is quite interesting to observe the behavior of the algorithms in the case of a great workload. It has to be noted that IBEA$_H$ is also more efficient than NSGA-II$_H$ and SPEA2$_H$ in that case. The next experiment was performed in order to compare the algorithms with and without the ILS. It has been observed that all the algorithms hybridized with the ILS are better than their respective standard counterparts (IBEA$_H$ better than IBEA, NSGA-II$_H$ better than NSGA-II, SPEA2$_H$ better than SPEA2).

Even if the rationale behind the better performance of IBEA compared with its challengers is not well comprehended, some assumptions can be formulated. The first is that, contrary to NSGA-II and SPEA2, the IBEA search process explicitly aims at improving the quality indicator under consideration. As a consequence, the search process is directly related to the quality indicators generally used to assess the performance of evolutionary multi-objective optimization algorithms. The second concerns the large complexity of SPEA2, which is very time-consuming with respect to the other algorithms, so that it is penalized when algorithms are compared according to the computation time. It may be the same phenomenon which happens to NSGA-II compared to IBEA.

Finally, we can state that a DRT service with little time dedicated to the optimization process (one or two minutes) should be possible or, at least, a short time to compute should help to reduce the booking time (generally less than half an hour). Consequently, it is possible to consider more flexible DRT services, which respond more to the customers' wishes. Indeed, the availability and the immediacy of the service are the crucial points to be continuously improved for providing more attractive services. Besides, from the decision-maker's point-of-view, a set of compromise solutions can be proposed in a short time, which is relevant in practice if the decision-maker wants to integrate parameters which cannot be formalized (different policy...). For all these reasons, DRT can be utilized as a complement to the classical public transportation in the badly-served areas.

In the future, the research for multi-objective transportation problems will intensify and will tend more and more towards real-time and dynamic services. In this context, the combination or hybridization of methods will help to face this challenge, as we have seen with an ILS in an evolutionary algorithm. New results based on a combination with variants of the local search may help to improve the performance of the approach.

Acknowledgment. The authors would like to acknowledge the anonymous reviewers for their valuable comments and suggestions that largely contributed to improve the quality of the paper.

References

[1] R. Chevrier, Optimization of demand responsive transport in polarized territories, Ph.D. thesis, UMR ESPACE (CNRS 6012), University of Avignon (France), 244 p. (november 2008).

[2] J.-F. Cordeau, G. Laporte, The dial-a-ride problem: Models and algorithms, Annals of Operations Research 153 (1) (2007) 29–46.

[3] J. Paquette, J.-F. Cordeau, G. Laporte, Quality of service in dial-a-ride operations, Computers & Industrial Engineering 56 (4) (2009) 1721 – 1734.

[4] B. Golden, S. Raghavan, E. Wasil (Eds.), The Vehicle Routing Problem: Latest Advances and New Challenges, Springer, New York, 2008, 589 p.

[5] N. Jozefowiez, F. Semet, E.-G. Talbi, Multi-objective vehicle routing problems, European Journal of Operational Research 189 (2) (2008) 293 – 309.
[6] M. W. Savelsberg, M. Sol, The general pickup and delivery problem, Transportation Science 1 (29) (1995) 17–29.
[7] M. Ehrgott, Multicriteria optimization (2nd edition), Springer, 2005.
[8] E. L. Ulungu, J. Teghem, The two phases method: An efficient procedure to solve bi-objective combinatorial optimization problems, Foundations of Computing and Decision Sciences 20 (2) (1995) 149–165.
[9] A. Przybylski, X. Gandibleux, M. Ehrgott, Two phase algorithms for the bi-objective assignment problem, European Journal of Operational Research 185 (2) (2008) 509–533.
[10] C. Dhaenens, J. Lemesre, E. Talbi, K-PPM: A new exact method to solve multi-objective combinatorial optimization problems, European Journal of Operational Research 200 (1) (2010) 45–53.
[11] G. Pankratz, A grouping genetic algorithm for the pickup and delivery problem with time windows, OR Spectrum 27 (1) (2005) 21–41.
[12] R. Jørgensen, J. Larsen, K. Bergsvindottir, Solving the dial-a-ride problem using genetic algorithms, Journal of the Operational Research Society 58 (2007) 1321–1331.
[13] C. Cubillos, N. Rodriguez, B. Crawford, Bio-inspired Modeling of Cognitive Tasks, Lecture Notes in Computer Science, Springer, Berlin / Heidelberg, 2007, Ch. A Study on Genetic Algorithms for the DARP Problem, pp. 498–507.
[14] S. Jung, A. Haghani, Genetic algorithm for a pickup and delivery problem with time windows, Transportation Research Record: Journal of the Transportation Research Board 1733 / 2000 (2007) 1 – 7.
[15] N. Jozefowiez, F. Semet, E.-G. Talbi, An evolutionary algorithm for the vehicle routing problem with route balancing, European Journal of Operational Research 195 (2009) 761–769.
[16] S. N. Parragh, K. F. Doerner, R. F. Hartl, X. Gandibleux, A heuristic two-phase solution approach for the multi-objective dial-a-ride problem, Networks 54 (2009) 227–242.
[17] R. Chevrier, A. Liefooghe, L. Jourdan, C. Dhaenens, On optimizing a demand responsive transport with an evolutionary multiobjective approach, in: ITSC'2010, 13th IEEE Intelligent Transport Systems Conference, Madeira, Portugal, 2010, pp. 575–580.
[18] E.-G. Talbi, Metaheuristics – From Design to Implementation, Wiley, Hoboken, New Jersey, 2009.
[19] P. Moscato, C. Cotta, A gentle introduction to memetic algorithms, in: F. Glover, G. Kochenberger (Eds.), Handbook of Metaheuristics, Vol. 57 of International Series in Operations Research & Management Science, Springer New York, 2003, pp. 105–144.
[20] K. Deb, S. Agrawal, A. Pratap, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II, IEEE Transactions on Evolutionary Computation 6 (2) (2002) 182–197.
[21] E. Zitzler, M. Laumanns, L. Thiele, SPEA2: Improving the strength pareto evolutionary algorithm, Tech. Rep. 103, Computer Engineering and Networks Lab (TIK), Swiss Federal Institute of Technology (ETH), Zurich, Switzerland (2001).
[22] E. Zitzler, S. Künzli, Indicator-based selection in multiobjective search, in: Conference on Parallel Problem Solving from Nature (PPSN VIII), Vol. 3242 of Lecture Notes in Computer Science, Springer-Verlag, Birmingham, UK, 2004, pp. 832–842.
[23] K. Deb, Multi-objective Optimization using Evolutionary Algorithms, Wiley, 2001, 517 p.
[24] A. Liefooghe, L. Jourdan, E.-G. Talbi, A software framework based on a conceptual unified model for evolutionary multiobjective optimization: ParadisEO-MOEO, European Journal of Operational Research 209 (2) (2011) 104–112.
[25] N. Jozefowiez, F. Semet, E.-G. Talbi, The bi-objective covering tour problem, Computers & Operations Research 34 (2007) 1929–1942.
[26] A. Liefooghe, L. Jourdan, N. Jozefowiez, E.-G. Talbi, On the integration of a TSP heuristic into an EA for the bi-objective ring star problem, in: International Workshop on Hybrid Metaheuristics (HM 2008), Vol. 5296 of Lecture Notes in Computer Science, Springer-Verlag, Malaga, Spain, 2008, pp. 117–130.
[27] P. Haggett, Locational Analysis in Human Geography, Edward Arnold, London, 1977, 2nd edition, Volume I: "Locational Models" (298 pages) and Volume II: "Locational Methods" (346 pages).
[28] J. Knowles, L. Thiele, E. Zitzler, A tutorial on the performance assessment of stochastic multiobjective optimizers, Tech. rep., Computer Engineering and Networks Laboratory (TIK), ETH Zurich, Switzerland, (revised version) (2006).
[29] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, V. Grunert da Fonseca, Performance assessment of multiobjective optimizers: An analysis and review, IEEE Transactions on Evolutionary Computation 7 (2) (2003) 117–132.