Received: 24 April 2018 | Accepted: 24 April 2018

DOI: 10.1002/net.21827

SPECIAL ISSUE ARTICLE

WILEY

A two-phase Pareto local search heuristic for the bi-objective pollution-routing problem

Luciano Costa[1] | **Thibaut Lust**[2] | **Raphael Kramer**[3] | **Anand Subramanian**[4]

[1]Département de Mathématiques et de Génie Industriel, École Polytechnique de Montréal and GERAD, Montréal, Québec H3C 3A7, Canada

[2]Sorbonne Universités, UPMC, Université Paris 06, CNRS, LIP6, UMR 7606, Paris F-75005, France

[3]Università degli Studi di Modena e Reggio Emilia, Dipartimento di Scienze e Metodi dell'Ingegneria, Reggio Emilia 42122, Italy

[4]Departamento de Sistemas de Computação, Centro de Informática, Rua dos Escoteiros, Mangabeira, Universidade Federal da Paraíba, João Pessoa, PB 58058-600, Brazil

Correspondence
Thibaut Lust, Sorbonne Universités, UPMC, Université Paris 06, CNRS, LIP6, UMR 7606, Paris F-75005, France.
Email: thibaut.lust@lip6.fr

Funding information
Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq/Brazil), Grant/Award Number: 132610/2014-0, 132789/2015-9, 305223/2015-1, 428549/2016-0, GDE 201222/2014-0

Abstract

This article deals with the bi-objective pollution-routing problem (bPRP), a vehicle routing variant that arises in the context of green logistics. The two conflicting objectives considered are the minimization of the CO_2 emissions and the costs related to driver's wages. A multi-objective approach based on the two-phase Pareto local search heuristic is employed to generate a good approximation of the Pareto front. During the first phase of the method, a first set of potentially efficient solutions is obtained by solving a series of weighted sum problems with an efficient heuristic originally developed to solve the single-objective PRP. A dichotomous scheme is used to generate the different weight sets in an automatic way. In the second phase, the set is improved with an efficient Pareto local search (PLS) procedure. The use of PLS allows to limit the number of computational demanding weighted sum problems solved in the first phase, while keeping high-quality results. Extensive computational experiments over existing benchmark instances show that the proposed approach leads to better results in less CPU time when compared to those obtained by state-of-the-art methods.

KEYWORDS

combinatorial optimization, heuristics, multi-objective optimization, Pareto local search, pollution-routing problem

1 | INTRODUCTION

This article deals with a green variant of the vehicle routing problem (VRP), a well-known combinatorial optimization problem that arises in the fields of operations research and logistics. In the VRP, a set of routes that start and end in the same (and unique) depot must be designed in such a way that each customer has its demand satisfied in a single visit and the vehicle capacity is not exceeded. In practical situations, one usually aims at minimizing the operational costs such as the total distance traveled, number of vehicles, labor costs, etc.

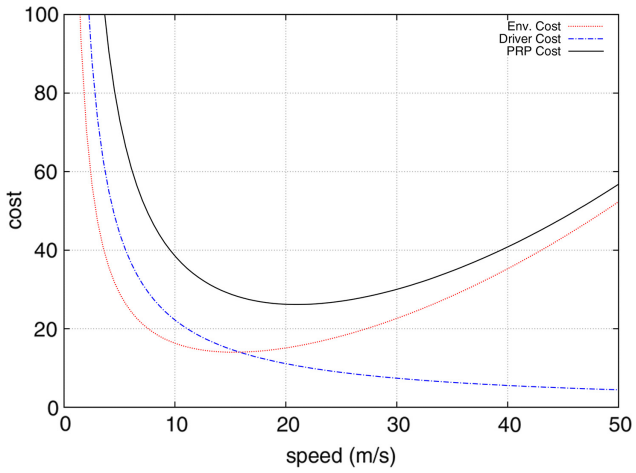

FIGURE 1 PRP objective functions [Color figure can be viewed at wileyonlinelibrary.com]

The continuous research efforts on VRPs are often motivated by economic aspects [56]. However, given that transportation operations are responsible for a large amount of greenhouse gases (GHG) emissions, environmental issues may be also taken into account when solving such problems.

Due to the growing worldwide concern about the impact of human activities on the environment, research works related to this issue started to increase. In fact, in the last two decades, the number of operations research works including environmental aspects in their scope, specially those related to logistics and freight transportation, also increased [40]. Sbihi and Eglese [53] presented several combinatorial optimization problems that could be analyzed in the environment context as well as some concepts and definitions about green logistics.

According to Lin et al. [40], VRP variants considering sustainable transportation issues are called *green vehicle routing problems* (GVRPs). As an attempt to characterize the different types of GVRPs, the authors proposed a classification that divides the GVRPs into three categories: (i) green-VRP, which addresses the optimization of energy consumption of transportation; (ii) pollution routing problem (PRP), which aims at reducing the emissions of GHG; and (iii) VRP in reverse logistics (e.g., waste collection VRP).

Data from International Energy Agency (IEA) show that the majority of CO_2 emissions arising from fuel combustion in the year 2015 came from the energetic and transportation sectors, each one responsible for 42% and 24% of total emissions, respectively [26]. The emissions related to the latter sector are directly proportional to the amount of fuel consumed by a vehicle. The way such emissions are estimated might vary depending on which aspects are considered by the emission model. Typically, the speed, the acceleration, the load carried and the distance traveled by the vehicle are some of the factors incorporated by most models [10]. A comparison highlighting the weakness and strengths of several emission models can be found in Demir et al. [10]. VRPs where the emissions are estimated according to the *vehicle load* $\times$ *distance* indicator have been proposed by Figliozzi [16], Kopfer et al. [35] and Xiao et al. [58]. On the other hand, Franceschetti et al. [18], Franceschetti et al. [19], Jabali et al. [27], Kopfer and Kopfer [34], Bektaş and Laporte [5], Demir et al. [11], Kramer et al. [36], Kramer et al. [37], Kuo [39] and Soysal et al. [55] considered VRPs where the level of emissions is proportional to the traveling speeds between pairs of customers. Most of the methods adopted to solve these problems are based on heuristics. Concerning the exact approaches, one can cite the column generation-based algorithms proposed by Fukasawa et al. [21], Fukasawa et al. [20], and Dabia et al. [7], and the disjunctive convex programming models by Fukasawa et al. [22]. The interested reader is referred to the surveys of Demir et al. [13] and Lin et al. [40] for a more complete analysis of the state-of-the-art on VRPs with environmental considerations.

Despite the growing interest on the environmental preservation, economic aspects still remain highly important. Therefore, examining their relationship allows for a more complete analysis on the behavior of such aspects in this integrated context. In light of this, Bektaş and Laporte [5] proposed the pollution-routing problem (PRP), an extension of the VRP with time windows (VRPTW), whose objective is to minimize the GHG emission costs plus the labor costs.

The objectives considered in all works mentioned above were embedded in the same cost function. However, it should be pointed out that the two PRP objective functions (i.e., GHG emission and labor costs) are conflicting. As depicted in Figure 1, this happens when the vehicles travel with speeds greater than $\approx$ 15 m/s, where optimizing one objective implies degrading the other. Since labor costs increase with the routes duration, one may intend to generate shorter routes, in terms of travel time.

Nevertheless, at the same time, this may imply in larger emission costs, since higher vehicle speeds are necessary to minimize the total travel time of a route. On the other hand, in order to minimize the emission costs, one needs to adopt lower vehicle speeds. In this case, routes will have a larger duration, thus increasing the labor costs.

Recently, Lin et al. [40] remarked that it could be interesting to explore the existing trade-off between travel distance and environmental impacts. According to Ehrgott [15], the best way to analyze trade-off in problems with multiples objectives is through multi-objective optimization (MO). The main goal of MO is to find all Pareto optimal solutions (or a good approximation of these solutions [8, 23]), that is, solutions that cannot be improved in any of the objectives without degrading at least one of the other objectives.

Despite the conflicting nature between the environmental and economic costs, there are few works in which a MO approach is explicitly applied to tackle problems of this nature. Jemai et al. [30] defined the bi-objective green VRP, where the two objectives to be minimized are the total traveled distance and CO_2 emissions. The non-dominated sorting genetic algorithm II (NSGA-II) [9] was applied to solve the problem. Siu et al. [54] devised a genetic algorithm for dealing with the multi-objective green cargo routing, in which different plans must be defined for distinct types of transportation modes. Nevertheless, the objectives still remain the same, that is, minimizing transportation costs and CO_2 emissions. The authors compared their results with those obtained by an adapted version of Martins' algorithm [46]. Molina et al. [47] dealt with a real-life application for the multi-objective heterogeneous fleet VRP with environmental considerations. Three objectives were considered: (i) minimization of internal costs and minimization of (ii) CO_2 and (iii) NO_x emissions. An augmented weighted Tchebycheff method was used to solve it.

Demir et al. [12] proposed the bi-objective PRP (bPRP) which extends the PRP defined in [5], but they considered the two objectives separately through a MO approach. The authors solved such problem by directly applying straightforward and well-known MO methods from the literature. Recently, Kumar et al. [38] implemented a so-called self-learning particle swarm optimization approach for solving a multi-objective PRP with time windows and multiple time periods, where the objectives considered were related to environmental, routing, and production costs.

In this article, we focus our attention on solving the bPRP by adapting the two-phase Pareto local search (2PPLS) approach proposed by Lust and Teghem [43] and originally developed for solving the bi-objective TSP (bTSP). State-of-the-art results have been obtained by such method when solving other classic MO combinatorial optimization problems like the MO knapsack problem [44] and MO set covering problem [45]. In the first phase of the method, an approximation of the set of supported efficient solutions is generated by solving single-objective problems (SOPs) obtained from the linear weighted aggregation of the objectives (weighted sum). In our case, we used a simplified version of the state-of-the-art PRP algorithm by Kramer et al. [37] to solve such problems. In the second phase, new solutions are generated by applying Pareto local search (PLS) [2, 48], a method based on neighborhood search. In contrast to the bTSP, finding good approximations for the Pareto Front for the bPRP in an acceptable CPU time is not an easy task. Firstly, solving single-objective PRP instances obtained with weighted sums is much harder and more computational demanding than solving single-objective TSP instances. Therefore, it is necessary to limit as much as possible the number of SOPs solved during the first phase of the algorithm. Secondly, during the application of the neighborhood search, one needs to build not only the routes themselves, but also determine the vehicle speeds associated with each arc of the route, which makes the problem much harder to be solved in practice.

The main contributions of this article are threefold: firstly we show how to limit the number of weighted sum problems solved during the first phase by using a dichotomous scheme to generate automatically the different weight sets, allowing to obtain good quality initial sets in small CPU times. Secondly, we study different combinations of neighborhood functions and propose an efficient mechanism that avoids a prohibitive number of calls to the speed optimization algorithm (SOA) throughout the PLS. Thirdly, we show that the proposed algorithm obtains high-quality results for several MO indicators, in less CPU time than state-of-the-art MO methods used in [12] to solve the same problem. To our knowledge, this is the first time that an adaptation of 2PPLS to solve a multi-objective mixed integer linear program problem (MOMILP) is proposed. Although this MO method is more enhanced than those considered in [12], it is still simple and straightforward to be implemented, since the method is mainly based on the single-objective solver previously developed for solving the PRP. Please note that the goal of this article is not only to produce high-quality results for the bPRP, but also to show that it is possible to easily use an efficient single-objective solver of a specific problem to obtain high-quality results for the bi-objective version of the same problem. We only consider two objectives in this article, as for problems with more than two objectives, particular adaptations are often needed to manage the growing number of solutions [6].

The remainder of this article is organized as follows. Section 2 describes the bPRP. Section 3 explains the proposed 2PPLS based approach. Quality indicators are listed in Section 4. Computational experiments are reported and discussed in Section 5. Finally, Section 6 presents some concluding remarks.

2 | PROBLEM DESCRIPTION

Let $\mathcal{G} = (\mathcal{V}, \mathcal{A})$ be a complete and directed graph with a set $\mathcal{V} = \{0, \ldots, n\}$ of vertices and a set $\mathcal{A} = \{(i,j) : i,j \in \mathcal{V}; i \neq j\}$ of arcs. The depot is represented by vertex 0 whereas the set of customers is denoted by $\mathcal{V}' = \mathcal{V} \setminus \{0\}$. Each customer $i \in \mathcal{V}'$ has a non-negative demand q_i, a time interval $[a_i, b_i]$ where it can be served, and a service time t_i. The travel distance between a pair of vertices i and j is given by d_{ij}, $(i,j) \in \mathcal{A}$. A set $K = \{1, \ldots, r\}$ of homogeneous vehicles with capacity Q is available at the depot.

The bPRP considers the same constraints found in the VRPTW, where in addition to the constraints defined for the classical VRP, it also imposes that each customer $i \in \mathcal{V}'$ can only start to be served within its time window $[a_i, b_i]$ [14]. The bPRP has two objective functions, which are treated separately: the first aims at minimizing the costs from CO_2 emissions while the second aims at minimizing the total costs associated with drive wages.

From [5, 10], drivers are assumed to be paid per unit of time, and CO_2 emissions are assumed to be proportional to vehicle fuel consumption, which in turn is dependent on environment and traffic-related parameters such as vehicle type, speed, load, acceleration, and congestion. Let v_{ij} and f_{ij} be the vehicle speed and the vehicle load on arc (i,j), respectively. The amount of CO_2 emissions associated with a travel from i to j can be computed as

$$F_{ij}(v_{ij}, f_{ij}, d_{ij}) = \xi(\mu NV + w\gamma\alpha_{ij}v_{ij} + \gamma\alpha_{ij}f_{ij}v_{ij} + \beta\gamma v_{ij}^3)d_{ij}/v_{ij}, \tag{1}$$

where ξ and γ are constants related to fuel properties, β and w are associated with vehicle characteristics and α_{ij} is a constant that depends on road characteristics and vehicle acceleration. Moreover, μ is the engine friction factor, N is the engine speed, V is the engine displacement. Equation (1) is based on the comprehensive emissions model described by Barth et al. [4] and Barth and Boriboonsomsin [3]. The parameter values adopted in the bPRP can be found in [11, 37].

The bPRP aims at designing a set of routes by deciding the arcs to be included in the solution as well as their associated vehicle speeds, in order to minimize the two aforementioned objectives, while respecting VRPTW constraints. Hence, if we consider f_c and f_d as the cost associated with fuel consumption and labor activities, respectively, the bPRP objective functions can be expressed as follows:

$$\min f_1(x) = f_c \sum_{(i,j)\in S} F_{ij}(v_{ij}, f_{ij}, d_{ij}) \tag{2}$$

$$\min f_2(x) = f_d \sum_{i\in\mathcal{V}'} s_i, \tag{3}$$

where x is a feasible solution (set of routes), S is the set of arcs in x, $F_{ij}(.)$ corresponds to the amount of CO_2 emissions as given in Equation (1), and$_i s$ represents the total time spent on a route that has the vertex $i \in \mathcal{V}$ as the last visit before returning to the depot.

Note that we can consider the bPRP as a MOMILP since one has to perform binary decisions, that is, by deciding whether an arc must be included in the solution or not, as well as continuous decisions, that is, by deciding the vehicle speed over each selected arc. Motivated by real-life aspects, the selected speeds must respect lower (V_{MIN}) and upper (V_{MAX}) bounds, which come from the problem definition. According to Bektaş and Laporte [5], the speed at which a vehicle travels on arc (i,j) is imposed by traffic regulations. Nevertheless, it is important to emphasize that in the PRP definition, traffic conditions are not taken into account. Hence, the vehicles are allowed to travel at any speed within a given interval.

In practice, we are interested in finding the set of *Pareto optimal* (or efficient) solutions for the bPRP. We recall the definitions related to MO optimization in the following.

We consider a general MO problem (MOP), with a feasible set X and k objective functions $f_i(x)$ ($i \in \{1, \ldots, k\}$), where $x \in X$ is a solution vector. In MO optimization, solutions are usually compared according to the *Pareto dominance relation*:

Definition 1. Pareto dominance relation: we say that a vector $z^* = (z_1^*, \ldots, z_k^*)$ dominates a vector $z = (z_1, \ldots, z_k)$ if, and only if, $z_i^* \leq z_i \, \forall \, i \in \{1, \ldots, k\} \wedge \exists \, i \in \{1, \ldots, k\} : z_i^* < z_i$. We denote this relation by $z^* \succ z$.

We define an efficient solution as follows:

Definition 2. Efficient solution: a feasible solution $x^* \in X$ is called efficient if there does not exist any other feasible solution $x \in X$ such that $f(x) \succ f(x^*)$.

For these solutions, it is not possible to improve the value of one of the criteria without deteriorating at least one other criterion.

The image $z = f(x)$ of an efficient solution is called a nondominated point ($z \in \mathbb{R}^k$). The efficient set denoted by X_E contains all efficient solutions, whereas the Pareto front, denoted by Y_N, contains all nondominated points (it corresponds to the image of the efficient set in objective space). In this work we will mainly work with approximations of the efficient set, also called set of potentially efficient solutions, and denoted by $\hat{X}_E$.

At some point, we will also need to use the *weakly Pareto dominance relation*, defined as follows.

Definition 3. Weakly Pareto dominance relation: we say that a vector $z^* = (z_1^*, \ldots, z_k^*)$ weakly dominates a vector $z = (z_1, \ldots, z_k)$ if, and only if, $z^* \succ z$ or $z = z^*$. We denote this relation by $z^* \succeq z$.

Finally, we make the important distinction between two types of efficient solutions: supported and nonsupported [15]. The set of supported efficient solutions (X_{SE}) can be obtained by solving weighted sum SOPs of the form $\min_{x \in X} \sum_{i=1}^{k} \lambda_i f_i(x)$, with non-negative weights λ_i. The set of non-supported efficient solutions (X_{NSE}) are the remaining efficient solutions, not located in the convex hull of the Pareto front, and that cannot be obtained by solving weighted sum problems. In many cases, nonsupported solutions may provide a better compromise between the objectives of the problem [42].

3 | PROPOSED METHOD

We developed a 2PPLS-based method [43] for solving the bPRP, which makes use of approximation schemes in both phases, as described in the following.

- *Phase 1*: aims at finding a good approximation for the set of supported efficient solutions. This can be done, for example, by solving weighted linear aggregated problems related to the two bPRP objectives.
- *Phase 2*: aims at obtaining a set of non-supported efficient solutions by exploring the search space between supported efficient solutions. These new solutions are obtained using a PLS procedure. PLS [2, 48] is an adaptation of simple local search based on improving moves to multi-objective optimization. PLS works directly with the current approximation of the efficient set. For each solution in the approximation its neighborhood is searched to find new potentially efficient solutions and to update the approximation. PLS is especially good for finding solutions along the Pareto front but can be very slow to find solutions toward the Pareto front. That is why it is important to start PLS from a good initial population. In this configuration, PLS can generate a large number of new potentially efficient solutions in a very short CPU time.

We present below how we have adapted these two phases to the bPRP.

3.1 | First phase

In this phase, aggregated problems of the form (4) are solved for the feasible space X, where $\lambda = [\lambda_1, \lambda_2, \ldots, \lambda_k]$ and $z = [f_1(x), f_2(x), \ldots, f_k(x)]$ represent the weight and objective function vectors, respectively.

$$\min \left\{ \sum_{i=1}^{k} \lambda_i f_i(x) : x \in X \right\}. \tag{4}$$

One way of generating the weight sets is by applying the dichotomous scheme proposed by Aneja and Nair [1] and later improved by Przybylski et al. [51]. These methods employ exact approaches capable of generating all the weight sets, thus enabling one to obtain all the supported efficient solutions for bi-objective problems. Lust and Teghem [43] adapted the former method by applying heuristic approaches for solving the aggregated problems, in order to decrease the runtime of the method. As a result, the solutions found are not necessarily supported, or even efficient, but constitute a set that is close to the complete

set of supported efficient solutions. Two main advantages of using this method for solving the bPRP are that the number of weight sets to be used is automatically determined and the objective functions need not be normalized as the weight sets are computed from the representation of the solutions in the objective space.

This method is adopted in the present work and its adaptation to the bPRP is detailed in the following.

3.1.1 | Heuristic procedure for obtaining supported efficient solutions

The dichotomous scheme used for obtaining a good approximation of the set of supported efficient solutions is presented in Algorithm 1. For all pseudo-codes presented hereafter, the symbols ↑, ↓, and ↕ denote the passing of parameters in mode IN, OUT, and IN/OUT, respectively. The comments are identified by the symbol ▷.

Algorithm 1 receives as input a bPRP instance and returns an approximation $\hat{X}_{SE}$ of the set of supported efficient solutions. Herein, when the procedure `SolvePRP` is called, a SOP of the form $\lambda_1 f_1 + \lambda_2 f_2$ is solved.

The approximation $\hat{X}_{\text{SE}}$ is initialized with two solutions of SOPs. Firstly, the objective that aims at minimizing the costs from CO_2 emissions is considered (line 4). Secondly, the total cost associated with drive wages is minimized (line 6). Once both solutions are obtained, the method `SolveRecursionHeuristic` (line 8) is called in order to generate new weight sets and solutions.

Algorithm 1 `bPRP_Phase1_Heuristic`

1: Parameters ↑: bPRP instance c
2: Parameters ↓: An approximation $\widehat{X}_{SE}$ of X_{SE}
3: $\lambda = (1, 0)$
4: `SolvePRP`($\lambda \downarrow, x_1 \uparrow$) ▷ Solve the PRP considering only the objective that minimizes the emission costs
5: $\lambda = (0, 1)$
6: `SolvePRP`($\lambda \downarrow, x_2 \uparrow$) ▷ Solve the PRP considering only the objective that minimizes the costs associated with the driver wages
7: $\widehat{X}_{SE} \leftarrow \{x_1, x_2\}$
8: `SolveRecursionHeuristic`($x_1 \downarrow, x_2 \downarrow, \widehat{X}_{SE} \updownarrow$) ▷ Compute an approximation of X_{SE}

The method `SolveRecursionHeuristic` is described in Algorithm 2. In this method, the property that the images of the supported efficient solutions are always located in the convex hull of the Pareto front is used: by considering weight sets $\lambda = (\lambda_1, \lambda_2)$ defined by the normal line across the images in the objective space of the two solutions x_1 and x_2 considered, new supported solutions can be recursively generated [51]. As the new weight set is a vector normal to the line joining the two points $f(x_r)$ and $f(x_s)$, we have $\lambda_1 = \frac{f_2(x_r) - f_2(x_s)}{f_1(x_s) - f_1(x_r)}$ and $\lambda_2 = 1$. Then, we normalize the weight sets ($\lambda_1 + \lambda_2 = 1$) and we obtain $\lambda_1 = \frac{f_2(x_r) - f_2(x_s)}{f_2(x_r) - f_2(x_s) + f_1(x_s) - f_1(x_r)}$ and $\lambda_2 = 1 - \lambda_1$.

The solution obtained considering the new weight set is denoted as x_t. If x_t is not dominated by any of the solutions of the set of potentially efficient solutions, then x_t is added to this set. This verification is performed using the method `UpdateXE` (see Algorithm 3). The procedure simply consists in updating a set of potentially efficient solutions $\hat{X}_E$ with a new solution p of cost $f(p)$. The solution p is compared with all solutions from $\hat{X}_E$ (lines 4–11) and in case p is weakly dominated by at least one solution from the set, p is not added to $\hat{X}_E$ and the algorithm stops (lines 5–7). Otherwise, p is added to $\hat{X}_E$ (line 12). Moreover, if p dominates some solutions of $\hat{X}_E$, then these solutions are removed from $\hat{X}_E$ (lines 8–10).

Let $\triangle z(x_r)z(x_s)$ be the right triangle defined by the points $z(x_r)$, $z(x_s)$ (x_r and x_s images) and their associated local ideal point (minimum value of the coordinate of each point). Every time a solution x_t whose image $z(x_t)$ is inside the interior of the triangle $\triangle z(x_r)z(x_s)$, Alg. 2 is recursively called twice: one for x_r and x_t, and another one for x_t and x_s. Otherwise, the `SolveRecursionHeuristic` function is not called. Note that the area of $\Delta z(x_r)z(x_s)$ decreases between two iterations of `SolveRecursionHeuristic` and that is why the method is often called "dichotomic" search.

In practice, we do not consider the entire region defined by $\Delta z(x_r)z(x_s)$. When applying the dichotomous scheme, since in the bPRP there are decisions associated with continuous variables, there might be too many solutions that are close to each other. As a result, the dichotomous scheme will possibly generate a considerable number of nondominated points, and thus the first phase could be time consuming. Given that PLS employed in the second phase is likely to generate a large number of points

Algorithm 2 `SolveRecursionHeuristic`

1: Parameters $\downarrow$: x_r and x_s
2: Parameters $\updownarrow$: $\widehat{X}_{SE}$
3: $\lambda = (\lambda_1, \lambda_2)$ $\quad \lambda_1 = \frac{f_2(x_r) - f_2(x_s)}{f_2(x_r) - f_2(x_s) + f_1(x_s) - f_1(x_r)}$, $\lambda_2 = 1 - \lambda_1$ ▷ New weight set λ
4: `SolvePRP`($\lambda \downarrow, x_t \uparrow$) ▷ Solve PRP $\lambda = (\lambda_1, \lambda_2)$
5: `UpdateXE`($\widehat{X}_{SE} \updownarrow, x_t \downarrow, z(x_t) \downarrow$) ▷ Add x_t to set $\widehat{X}_{SE}$
6: **if** $z(x_t) \cap \text{int} \triangle z(x_r)z(x_s) \neq 0$ **then**
7: `SolveRecursionHeuristic`($x_r \downarrow, x_t \downarrow, \widehat{X}_{SE} \updownarrow$)
8: `SolveRecursionHeuristic`($x_t \downarrow, x_s \downarrow, \widehat{X}_{SE} \updownarrow$)

Algorithm 3 `UpdateXE`

1: Parameters $\updownarrow$: A set $\widehat{X}_E$ of potentially efficient solutions
2: Parameters $\downarrow$: A solution p and its evaluation $f(p)$
3: Parameters $\uparrow$: A Boolean variable which is true if p has been added and false otherwise
4: **for all** $x \in \widehat{X}_E$ **do**
5: **if** $f(x) \succeq f(p)$ **then**
6: **return** False ▷ p is weakly dominated
7: **if** $f(p) \succ f(x)$ **then**
8: $\widehat{X}_E \leftarrow \widehat{X}_E \setminus \{x\}$ ▷ We remove the dominated solutions from $\widehat{X}_E$
9: $\widehat{X}_E \leftarrow \widehat{X}_E \cup \{p\}$ ▷ p is added to $\widehat{X}_E$
10: **return** True

in a small amount of time, one potential strategy is to accept less solutions during the first phase so as to improve the overall runtime of the algorithm.

More precisely, we propose a mechanism that reduces the area of the triangle in order to avoid the acceptance of several nondominated points within a small region of the solution space (see Figure 2). Such reduction is controlled by a parameter τ: the larger the value of τ, the smaller the number of nondominated points accepted for restarting the recursion. In other words, parameter τ is used to limit the generation of supported efficient solutions located between two close efficient supported solutions. In our case, we assume that τ is proportional to the Euclidean distance between the points $z(x_r)$ and $z(x_s)$, that is, $\tau = \rho \times \|z(x_r), z(x_s)\|$, where ρ is an input parameter. The value of this parameter should be chosen in such way that enough potentially nondominated points are generated in the first phase so as to provide to PLS a good initial set and to attain high-quality approximations front in reasonable CPU times (see Section 5.2).

3.1.2 | Solving the single-objective PRP

For solving the single-objective PRPs obtained from the linear weighted aggregation of the objectives, we have decided to use the algorithm proposed by Kramer et al. [37], which is to the best of our knowledge the best heuristic method available for solving the single-objective PRP. Their multi-start matheuristic, called ILS-SP-SOA, combines iterated local search (ILS) [41] with a set partitioning (SP) approach and SOA. Nevertheless, since many SOPs are solved during the execution of the algorithm, we chose to drop the SP phase of the method so as to improve the runtime performance. Of course, removing this component of the method may possibly affect the quality of the solutions generated. However, this simplified version (ILS-SOA) of the algorithm developed in [37] still yields good solutions. This can be verified in the results reported in Appendix.

Algorithm 4 presents the pseudo-code of ILS-SOA. The algorithm restarts n_R times (lines 5–21) where at each iteration the speed matrix $\mathbf{v}$ is initialized with the maximum speed allowed in the instance (line 9) and a solution is generated using an insertion based heuristic [50] that allows infeasible solutions with respect to the time windows constraints (line 10). Local search and perturbation moves as well as speed optimization are then alternatively performed (line 13) for n_{ILS} consecutive iterations without improvements (lines 11–17). The algorithm returns the best solution found among all restarts.

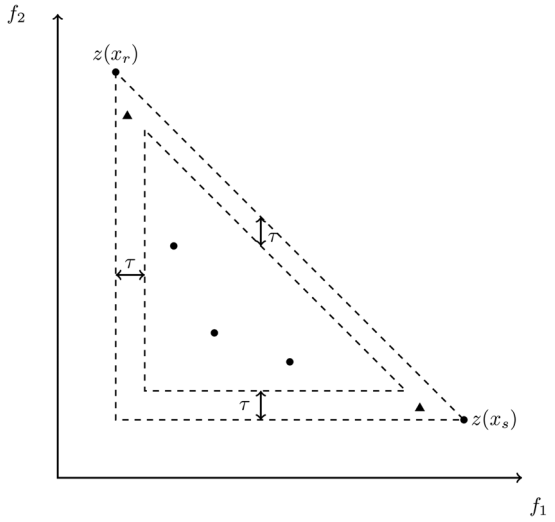

FIGURE 2 Adapted Stopping Criterion

ILS-SOA makes use of the auxiliary data structures (ADSs) presented in Vidal et al. [57] in order to enhance the runtime performance of the method. These ADSs store a series of information (total duration, earliest arrival time, latest arrival time, cumulated load, etc.) for all the subsequences of a given solution, and they allow to perform move evaluation and feasibility check in amortized $\mathcal{O}(1)$ time during the local search. The reader is referred to [37] for a complete and detailed description of all individual components of the algorithm, including local search, perturbation, and speed optimization procedures.

Algorithm 4 ILS-SOA

```
Parameters ↓: n_R, n_ILS, seed
Parameters ↑: S*
S* ← ∅;
f(S*) ← ∞;
while i_R < n_R do
    i_R ← i_R + 1
    S' ← ∅;
    f(S') ← ∞;
    v ← InitializeSpeedMatrix(v_MAX)
    S ← SpeedOptimization(LocalSearch(GenInitSol(seed)))
    while i_ILS < n_ILS do
        i_ILS ← i_ILS + 1
        S ← SpeedOptimization(LocalSearch(Perturbation(S', seed)))
        if f(S) < f(S') then
            S' ← S; i_ILS ← 0
    if f(S' < f(S*)) then
        S* ← S'
```

3.2 | Second phase

In this phase one aims at finding potentially nonsupported efficient solutions with a view of improving the quality of the Pareto front by applying PLS [43, 48, 49] over the set of potentially efficient solutions $\hat{X}_{\text{SE}}$ obtained in first phase, as shown in Algorithm 5.

Algorithm 5 Pareto Local Search (PLS)

1: Parameters ↓: An initial set $\widehat{X}_{SE}$ and a bPRP instance
2: Parameters ↑: An approximation $\widehat{X}_E$ of the set of efficient solutions
3: $\widehat{X}_E \leftarrow \widehat{X}_{SE}$
4: $P \leftarrow \widehat{X}_{SE}$ ▷ P is the set of solutions to be explored
5: $P_a \leftarrow \emptyset$ ▷ P_a is an auxiliary set
6: **while** $P \neq \emptyset$ **do**
7: **for all** $p \in P$ **do**
8: **for all** $p' \in \mathcal{N}(p)$ **do** ▷ The neighborhood of p is explored
9: **if** $f(p) \not\geq f(p')$ **then**
10: **if** `UpdateXE`($\widehat{X}_E \updownarrow, p' \downarrow, f(p') \downarrow$) **then** ▷ Update $\widehat{X}_E$ with p'
11: `UpdateXE`($P_a \updownarrow, p' \downarrow, f(p') \downarrow$) ▷ Update P_a with p'
12: $P \leftarrow P_a$ ▷ P_a is the new set to explore and we copy it in P
13: $P_a \leftarrow \emptyset$

A set P and the set of potentially efficient solutions $\hat{X}_E$ are both initialized with the same set of solutions $\hat{X}_{\text{SE}}$ (lines 4–5), generated during phase 1. Next, while P is not empty (lines 6–18), each neighbor solution p' of every $p \in P$ is generated (lines 7–15) using the following VRP based neighborhoods, that are all well-known for providing good computational results in the VRP context (and also used in ILS-SOA):

- `Shift(1,0)`—a customer is moved from one route to another one.
- `Swap(1,1)`—two customers in two different routes are switched.
- `Cross` (a.k.a. 2-opt*)—two arcs are removed: one from route r_1 and another one from route r_2; and another two are inserted: one connecting r_1 with r_2 and another one connecting r_2 with r_1.
- `Reinsertion`—a customer is removed and inserted in another position of the same route.
- `Exchange`—two customers in the same route are switched.

In Section 5.2, we will empirically justify the choice of the neighborhood structures described above.

Next, each nondominated solution p' is added to $\hat{X}_E$ and also to an auxiliary set P_a (lines 9–13), which in turn becomes the new set P in line 16.

Note that the moves performed in the neighborhood only modify the routes. However, the optimal speeds associated with the arcs of a modified route may not be the same as the original route. One thus needs to apply the SOA in order to compute the new optimal speeds for the arcs. Nevertheless, since the size of each neighborhood structure is of the order of $\mathcal{O}(n^2)$, and the SOA complexity is $\mathcal{O}(n^2)$, the worst-case complexity of the entire procedure is $\mathcal{O}(n^4)$. Hence, the overhead of calling the SOA every time a move is evaluated can be prohibitively expensive in terms of CPU time.

To overcome the aforementioned issue, we propose a simple yet effective approach to improve the runtime performance without severely compromising the quality of the solutions found during the local search. In particular, whenever a move is evaluated, the method computes the objective functions considering a maximum speed allowed in the instance. If this solution turns out to be potentially efficient, then SOA is called in order to determine the actual optimal speeds for each arc of the modified route. This dramatically reduces the number of calls to SOA, thus preventing the algorithm's runtime to increase excessively.

PLS does not accept infeasible solutions that violate the vehicle capacity, and the feasibility check can be easily performed using the ADSs mentioned in Section 3.1.2. However, infeasible solutions with respect to time windows are accepted, but the algorithm penalizes such solutions, which will be then naturally dominated and thus disregarded during the search. In this case, the move evaluation (considering the penalties due to time windows violation) can still be performed in amortized $\mathcal{O}(1)$ time as shown in [37, 57]. Note that this is also the same policy adopted by ILS-SOA.

4 | QUALITY INDICATORS

In contrast to SOPs, comparing the quality of solutions in MOPs is not an easy task. While in the first the solutions can be directly compared by simply observing the value of the objective function, in the latter one has to compare a set of points, which is not straightforward. Nevertheless, according to Zitzler et al. [60], when assessing the quality of solutions in MOPs, it is desirable that: the distance between the potentially nondominated points found by the method and the optimal Pareto front (reference set) is the minimum possible; there should be a maximal number of points, well-distributed along the Pareto front. When is not practical to determine the optimal Pareto front, the reference set can be derived by merging the fronts generated by state-of-the-art algorithms and removing dominated points [31].

The MO assessment methods available in the literature can be classified into two main categories: unary and binary. While in the first category a score is attributed to a set of solutions, in the second one a score is attributed to a pair of sets of solutions. Such score is generally computed by comparing both sets to a reference set. Zitzler et al. [61] performed a detailed analysis on the main features of the most popular MO assessment methods.

In this work, we make use of two indicators: hypervolume ($\mathcal{H}$) and R measure. We chose them because they are unary indicators and thus do not depend on a reference set, which is quite complicated to determine, not only because solving the bPRP exactly is far from being a simple task, but also because the approximations found in [12] are no longer available. We also use the concept of attainment surfaces to take into account the stochasticity of the algorithms and to compare the approximations in objective space. We briefly present these concepts in the following subsections.

4.1 | Hypervolume

The hypervolume ($\mathcal{H}$) indicator was proposed by Zitzler [59] and it computes the approximated area over the curve formed by the set of potentially nondominated points bounded by a *low-quality* point. The low-quality point should be defined such that every point of the Pareto front approximation weakly dominates the low-quality point. A point close to the Nadir point (that is the virtual point composed of the worst values for each objective among the nondominated points) can be used. The value of $\mathcal{H}$ must be maximized to obtain potentially nondominated points located far from the low-quality point and close to the ideal point (the virtual point composed of the best values for each objective).

4.2 | R measure

The R measure ($\mathcal{R}$) [28] depends on the average of the minimum value of the weighted Tchebycheff function (Γ) over a set of systematically generated normalized weight vectors. For each weight λ, the Tchebycheff weighted distance is determined for all the points $y = f(x)$ of an approximation $\hat{Y}_N$, by means of the expression $\Gamma_y^\lambda = \|y - y^0\|_\lambda = \max_{i=1,\dots,k} \lambda_i (y_i - y_i^0)$. Next, the minimum (best) value associated with each λ is computed, that is, $\Gamma_\lambda^* = \min_{y \in \hat{Y}_N} \{\Gamma_y^\lambda\}$. We finally normalize the value of $\mathcal{R}$ with this expression: $\mathcal{R} = 1 - \frac{\sum_{\lambda \in \Psi} \Gamma_\lambda^*}{|\Psi|}$. In this way, values between 0 and 1 are obtained, and as in the previous case, the value of $\mathcal{R}$ must be maximized (that is close to 1).

4.3 | Attainment surfaces

Proposed by Fonseca and Fleming [17], attainment surfaces are a nice way to represent the different outputs of a stochastic multi-objective optimizer in objective space. Indeed, when running one algorithm several times, plots of approximation of Pareto fronts quickly become confusing and unusable. In this work, we will use the concept of *summary* attainment surface proposed by Knowles [32], which can be defined as the union of all tightest goals that have been attained (independently) in precisely s of the runs of a sample of m runs, for any $s \in 1, \dots, m$. For example, the sample median quality is the best estimator of what one would expect to achieve in 50% of the runs. In this work, we will use the best (1st) summary attainment surface (best-case performance of the algorithm), the median summary attainment surface and the worst summary attainment surface (worst-case performance of the algorithm). We have used the code of Knowles [32] to generate the summary attainment surfaces.

5 | COMPUTATIONAL EXPERIMENTS

The proposed algorithm was coded in C++ and the computational experiments were conducted on an Intel Xeon E3-1226 3.30 GHz with 16 GB of RAM running Oracle Linux Server. The method was executed 10 times for each instance using a single thread. The statistical tests were performed using the R package [52].

TABLE 1 Different settings adopted for the 2PPLS

Setting	ρ (%)	Neighborhoods PLS
1	1.25	`Shift(1,0)`, `Swap(1,1)`, `Reinsertion`
2	1.25	`Shift(1,0)`, `Swap(1,1)`, `Reinsertion`, `Exchange`
3	1.25	`Shift(1,0)`, `Swap(1,1)`, `Reinsertion`, `Exchange`, `Cross`
4	2.50	`Shift(1,0)`, `Swap(1,1)`, `Reinsertion`
5	2.50	`Shift(1,0)`, `Swap(1,1)`, `Reinsertion`, `Exchange`
6	2.50	`Shift(1,0)`, `Swap(1,1)`, `Reinsertion`, `Exchange`, `Cross`
7	5.00	`Shift(1,0)`, `Swap(1,1)`, `Reinsertion`
8	5.00	`Shift(1,0)`, `Swap(1,1)`, `Reinsertion`, `Exchange`
9	5.00	`Shift(1,0)`, `Swap(1,1)`, `Reinsertion`, `Exchange`, `Cross`

Since the bPRP instances used in [12] are no longer available, we decided to use those originally proposed for the PRP, which are divided into three groups involving different sizes, each of them containing 20 instances. Group A, available at http://www.apollo.management.soton.ac.uk/prplib.htm, was suggested in [11]. Groups B and C, available at http://w1.cirrelt.ca/ vidalt/en/VRP-resources.html, were introduced in [37] and they were derived from the instances of group A by tightening the customers' time windows. More precisely, Groups A, B, and C contain instances with large, tight and moderate time windows, respectively. For comparison purposes, we only performed experiments for 100-customer instances, as in [12].

5.1 | Assessment of results

As already mentioned, the indicators $\mathcal{H}$ and $\mathcal{R}$ were the ones adopted to assess the quality of the solutions found by the algorithm. For obtaining the value of $\mathcal{R}$, the number of weights $|\Psi|$ generated for determining the weighted Tchebycheff function was given by $\binom{n+k-1}{k-1}$ [29], where n is the number of customers and k is the number of objectives. In our case, since $k = 2$ and $n = 100$, it follows that $|\Psi| = 101$ weights.

The nondeterministic values of $\mathcal{H}$ and $\mathcal{R}$, obtained through several stochastic runs of 2PPLS and also of other MO methods from the literature (see Section 5.3.1), were evaluated by means of the Mann-Whitney (MW) nonparametric statistical test. The reason for performing a nonparametric test is that the statistical distribution of the data is unknown. Since two hypothesis were simultaneously tested (differences between the values of $\mathcal{R}$ and of $\mathcal{H}$), the risk level α, which determines the probability of error of the test, was adjusted according to the Holm sequential rejective method [25], with $\alpha = 0.05$. The smallest P value obtained during the statistical tests were compared with $\alpha/2$, while the second smallest value was compared with α. The statistical results are reported in the tables of Section 5.3.1. The analysis was always performed with respect to 2PPLS using the following scheme. On the one hand, signs "$>$" and "$<$" indicate that the values of the quality indicator associated with 2PPLS are, respectively, statistically greater and smaller than the values found using other MO methods from the literature. On the other hand, sign "$=$" indicates that the values are not statistically different. Moreover, the number of solutions found (**#Sol.**), as well as the CPU time in seconds (**T(s)**) are also reported.

5.2 | Parameter tuning

In this section, we show how the parameters of the proposed algorithm were calibrated. The overall performance of the method mainly depends on: (i) the reduction of the ideal local triangle $\Delta z(x_r)z(x_s)$ that is measured by the parameter ρ described in Section 3.1.1; and (ii) the neighborhoods used in the second phase of the method.

We have tested three values of ρ, namely: 1.25%, 2.50%, and 5.00%, and three neighborhood schemes, thus leading to a total of nine different settings, as shown in Table 1.

We have performed a series of experiments on the set of 100-customers instances of Group B (20 instances, in total), which can be considered the most challenging set. The proposed algorithm was executed 10 times for each instance and the results obtained for the different settings can be found in Table 2. We report the number of times a particular setting is superior to the best-known methods of [12] in terms of number of nondominated solutions (#Sol.), hypervolume ($\mathcal{H}$) and R measure ($\mathcal{R}$), as well as the CPU time (T(s)) in seconds. The results show that setting 6 seems to be the most interesting configuration, with

TABLE 2 Number of times each setting of the 2PPLS outperformed the best known methods of Demir et al. [12]

Setting	#Sol.	$\mathcal{H}$	$\mathcal{R}$	CPU time
1	20	20	19	2
2	20	20	19	5
3	20	20	20	5
4	20	18	16	15
5	20	19	16	17
6	**20**	**19**	**17**	**15**
7	20	15	13	17
8	20	15	9	16
9	20	17	14	14

a good compromise between solution quality and CPU time. Therefore, we have set $\rho = 2.50\%$ and we have adopted the neighborhoods `Shift(1,0)`, `Swap(1,1)`, `Cross`, `Reinsertion` and `Exchange`.

5.3 | Results

5.3.1 | Comparing 2PPLS with other MO methods from the literature

With a view of evaluating the performance of the proposed 2PPLS heuristic in solving the bPRP, we have implemented two other MO methods considered in [12] for solving the bPRP, namely: weighted sum of the objective functions (WM) and weighed sum of the normalized objective functions (WMN). The other methods presented in [12] could not be reproduced because they are based on the ε-Constraint approach, which should be implemented in an exact fashion and the authors did not mention how this issue was dealt in their work.

As proposed by the authors, we considered the set $W = \{(1,0)\ (0.9,0.1)\ldots(0.1,0.9)\ (0,1)\}$, containing 11 weights, for WM and WMN. For the sake of comparison with 2PPLS, ILS-SOA was also used to solve the SOPs that arise in such methods. Hence, all the conclusions drawn can be attributed to the multi-objective methods.

We chose to compare 2PPLS with WM and WMN for two important reasons: (i) WM and WMN have been successfully applied to the bPRP and it is easy to reproduce the results obtained to the new instances used in the article (ii) WM, WMN, and 2PPLS are quite similar since weighted sums problems are solved in the three methods. Therefore, the comparison between these methods will also tell us, for a given CPU time, if it is worth to apply PLS rather than solving more weighted sum problems. We did not compare 2PPLS with the more popular MO algorithm NSGA-II [9] since Kumar et al. [38] already showed that the performances attained by NSGA-II were not very good for the bPRP.

Tables 3–5 compare 2PPLS either with WM or with WMN. Each table reports the average values (of 10 runs) for the indicators **#Sol**, $\mathcal{H}$, and $\mathcal{R}$ obtained by each algorithm (Bold values correspond to the best values). In addition, gaps with respect to WM and WMN are presented (columns below the label **Gap (%)**) and they are computed as $(v_{2PPLS} - v_{WM})/v_{WM}$ or $(v_{2PPLS} - v_{WMN})/v_{WMN}$, where v_{2PPLS}, v_{WM}, and v_{WMN} refer to the values of the indicators achieved by using 2PPLS, WM, and WMN, respectively.

From Table 3, it is possible to observe that a large number of potentially efficient solutions were generated by 2PPLS. This can be explained by the wide time windows of the instances of set A, allowing many feasible solutions to be generated during the application of PLS. Moreover, the gap between the quality indicators show that 2PPLS is clearly superior than the other methods for this set of instances, which is also confirmed by the result of the MW statistical test. In addition, the proposed algorithm was capable of finding six to eight times more solutions than the other ones. Finally, the average CPU time of 2PPLS was slightly slower than WM, but faster than WMN.

Note that the CPU time of 2PPLS depends on the stopping criterion used in the first phase. As a matter of fact, it is not possible to specify a priori the number of calls to ILS-SOA, as opposed to WM and WMN, since the first phase of 2PPLS stops when a good estimation of the supported efficient solutions is obtained. Nevertheless, depending on the reduction of the ideal triangle considered during the dichotomous scheme (parameter ρ described in Section 3.1.1) and also on the feasible region provided by the instance, a large number of solutions may be generated, which can be time consuming. This somewhat explains the differences in CPU time between 2PPLS and the other two methods. Further conclusions in this matter can be drawn from Table 6, where we summarize the computational results presented in Tables 3–5.

TABLE 3 Comparison between 2PPLS, WM, and WMN – Set A

	2PPLS				WM				Gap (%)			MW		WMN				Gap (%)			MW	
Instance	**#Sol.**	$\mathcal{H}$	$\mathcal{R}$	**T (s)**	**#Sol.**	$\mathcal{H}$	$\mathcal{R}$	**T (s)**	$\mathcal{H}$	$\mathcal{R}$	**T (s)**	$\mathcal{H}$	$\mathcal{R}$	**#Sol.**	$\mathcal{H}$	$\mathcal{R}$	**T (s)**	$\mathcal{H}$	$\mathcal{R}$	**T (s)**	$\mathcal{H}$	$\mathcal{R}$
UK100_01-A	**66.8**	**52813.9**	**0.8959**	**319.5**	9.1	50775.2	0.8815	412.7	4.02	1.63	−22.58	>	>	9.3	50674.7	0.8807	425.1	4.22	1.73	−24.84	>	>
UK100_02-A	**80.0**	**49074.3**	**0.9002**	412.1	10.7	47726.7	0.8852	**364.6**	2.82	1.69	13.03	>	>	10.5	47612.5	0.8850	420.7	3.07	1.72	−2.04	>	>
UK100_03-A	**69.0**	**41192.5**	**0.9020**	436.8	9.7	40014.8	0.8867	**418.7**	2.94	1.73	4.32	>	>	10.3	40091.3	0.8869	429.9	2.75	1.70	1.61	>	>
UK100_04-A	**72.7**	**50616.0**	**0.9108**	518.2	9.8	48691.4	0.8898	**338.2**	3.95	2.36	53.22	>	>	9.7	48450.8	0.8883	411.4	4.47	2.53	25.96	>	>
UK100_05-A	**69.8**	**38687.3**	**0.8991**	476.5	9.0	37626.3	0.8844	**348.4**	2.82	1.66	36.77	>	>	9.0	37164.3	0.8831	421.8	4.10	1.81	12.97	>	>
UK100_06-A	**65.6**	**50171.2**	**0.8950**	310.3	9.2	48726.3	0.8812	**307.6**	2.97	1.57	0.88	>	>	9.8	48340.7	0.8811	376.8	3.79	1.58	−17.65	>	>
UK100_07-A	**63.8**	**40732.5**	**0.8971**	402.1	8.5	38554.4	0.8827	**294.2**	5.65	1.63	36.68	>	>	9.1	39235.4	0.8806	361.7	3.82	1.87	11.17	>	>
UK100_08-A	**62.9**	**42739.5**	**0.8958**	370.4	9.0	41705.2	0.8831	**287.8**	2.48	1.44	28.70	>	>	9.2	41650.3	0.8828	357.8	2.62	1.47	3.52	>	>
UK100_09-A	**74.3**	**44644.4**	**0.9048**	347.8	9.5	43207.8	0.8871	**336.7**	3.32	2.00	3.30	>	>	9.7	43105.9	0.8873	411.5	3.57	1.97	−15.48	>	>
UK100_10-A	**67.1**	**43744.1**	**0.8961**	352.4	9.5	42134.8	0.8814	**287.8**	3.82	1.67	22.45	>	>	9.8	42259.3	0.8814	361.7	3.51	1.67	−2.57	>	>
UK100_11-A	**74.0**	**57958.2**	**0.9002**	**355.8**	8.7	55139.1	0.8780	365.2	5.11	2.53	−2.57	>	>	8.3	54612.8	0.8766	453.7	6.13	2.69	−21.58	>	>
UK100_12-A	**65.2**	**40390.8**	**0.9003**	505.6	8.6	38750.8	0.8823	**304.7**	4.23	2.04	65.93	>	>	9.3	38642.1	0.8808	383.1	4.53	2.21	31.98	>	>
UK100_13-A	**72.7**	**46537.7**	**0.8972**	380.8	10.1	45480.0	0.8849	**306.9**	2.33	1.39	24.08	>	>	10.0	45268.8	0.8833	369.0	2.80	1.57	3.20	>	>
UK100_14-A	**63.9**	**49461.8**	**0.9019**	**321.5**	9.0	47053.2	0.8806	338.0	5.12	2.42	−4.88	>	>	9.8	47587.9	0.8808	412.1	3.94	2.40	−21.98	>	>
UK100_15-A	**75.1**	**64470.5**	**0.9012**	460.1	9.9	62555.1	0.8864	**371.9**	3.06	1.67	23.72	>	>	10.2	62742.1	0.8865	453.1	2.75	1.66	1.54	>	>
UK100_16-A	**70.5**	**41059.4**	**0.9003**	334.4	8.6	39694.0	0.8865	**301.4**	3.44	1.56	10.95	>	>	9.4	39767.8	0.8864	367.5	3.25	1.57	−9.01	>	>
UK100_17-A	**68.1**	**58994.6**	**0.8950**	460.6	9.8	56708.5	0.8764	**389.1**	4.03	2.12	18.38	>	>	9.4	56344.2	0.8762	466.4	4.70	2.15	−1.24	>	>
UK100_18-A	**69.3**	**45526.0**	**0.8988**	476.0	9.4	43952.5	0.8845	**350.1**	3.58	1.62	35.96	>	>	9.8	44181.1	0.8852	430.4	3.04	1.54	10.59	>	>
UK100_19-A	**73.6**	**44306.6**	**0.9055**	**292.5**	9.1	43041.1	0.8932	326.3	2.94	1.38	−10.36	>	>	9.5	42857.9	0.8926	398.9	3.38	1.45	−26.67	>	>
UK100_20-A	**74.3**	**56893.4**	**0.8969**	354.1	10.4	55564.5	0.8844	**340.8**	2.39	1.41	3.90	>	>	10.1	55362.5	0.8841	416.3	2.77	1.45	−14.94	>	>
Average	**69.9**	**48000.7**	**0.8997**	394.4	9.4	46355.1	0.8840	**339.6**	3.55	1.78	17.09			9.6	46297.6	0.8835	425.1	3.66	1.84	−2.77		

TABLE 4 Comparison between 2PPLS, WM, and WMN – Set B

	2PPLS				WM				Gap (%)			MW		WMN				Gap (%)			MW	
Instance	**#Sol.**	$\mathcal{H}$	$\mathcal{R}$	**T (s)**	**#Sol.**	$\mathcal{H}$	$\mathcal{R}$	**T (s)**	$\mathcal{H}$	$\mathcal{R}$	**T (s)**	$\mathcal{H}$	$\mathcal{R}$	**#Sol.**	$\mathcal{H}$	$\mathcal{R}$	**T (s)**	$\mathcal{H}$	$\mathcal{R}$	**T (s)**	$\mathcal{H}$	$\mathcal{R}$
UK100_01-B	**62.3**	**137170.1**	**0.9126**	**482.1**	10.1	133931.9	0.9010	506.8	2.42	1.29	−4.87	>	>	9.6	133092.5	0.9023	614.8	3.06	1.14	−21.58	>	>
UK100_02-B	**67.1**	**217810.1**	**0.9261**	**461.8**	10.4	212317.4	0.9210	477.0	2.59	0.55	−3.19	>	>	9.6	212211.1	0.9219	582.2	2.57	0.45	−20.68	>	>
UK100_03-B	**98.0**	**212508.6**	**0.9211**	659.1	10.7	209034.4	0.9152	**538.8**	1.66	0.64	22.33	>	>	10.2	210049.1	0.9165	635.0	1.16	0.50	3.80	>	>
UK100_04-B	**52.9**	**202401.9**	**0.9171**	**447.2**	10.1	199315.6	0.9119	543.3	1.55	0.57	−17.69	>	>	9.9	199753.8	0.9131	596.2	1.31	0.44	−24.99	>	>
UK100_05-B	**69.9**	**230489.6**	**0.9057**	**562.9**	10.4	227078.8	0.9012	573.8	1.50	0.50	−1.90	>	>	9.2	225797.1	0.9011	586.6	2.04	0.51	−4.04	>	>
UK100_06-B	**60.1**	226259.5	0.9296	**413.2**	9.4	225387.2	0.9288	470.9	0.39	0.09	−12.25	=	=	8.9	**227001.3**	**0.9302**	596.9	−0.33	−0.06	−30.78	=	=
UK100_07-B	**49.0**	**234897.4**	**0.9282**	**339.7**	10.4	232030.5	0.9263	461.3	1.24	0.21	−26.36	>	=	9.5	233307.4	0.9278	565.6	0.68	0.04	−39.94	>	=
UK100_08-B	**73.3**	**234991.1**	**0.9392**	**463.8**	10.3	233069.7	0.9354	474.2	0.82	0.41	−2.19	>	>	10.0	232170.2	0.9347	598.0	1.20	0.48	−22.44	>	>
UK100_09-B	**69.1**	**226787.6**	**0.9409**	500.6	9.9	221786.0	0.9352	**444.5**	2.26	0.61	12.62	>	>	8.8	222142.5	0.9350	545.7	2.05	0.63	−8.26	>	>
UK100_10-B	**77.4**	**210447.0**	**0.9345**	512.1	10.4	206756.3	0.9293	**448.1**	1.79	0.56	14.28	>	>	9.6	206596.7	0.9291	563.8	1.83	0.58	−9.17	>	>
UK100_11-B	**71.1**	**226803.7**	**0.9299**	615.3	10.8	223769.2	0.9264	**539.7**	1.36	0.38	14.01	>	>	10.3	223752.4	0.9254	670.4	1.35	0.48	−8.22	>	>
UK100_12-B	**62.5**	**155418.0**	**0.9255**	**412.4**	8.8	152473.6	0.9229	462.4	1.93	0.28	−10.81	=	=	8.5	154454.2	0.9229	611.4	0.62	0.28	−32.55	=	=
UK100_13-B	**55.2**	**250030.6**	**0.9258**	**491.5**	10.2	246466.2	0.9248	512.5	1.45	0.11	−4.10	>	=	10.0	246725.5	0.9252	634.4	1.32	0.06	−22.53	>	=
UK100_14-B	**55.3**	**216549.3**	**0.9352**	**416.9**	9.7	213399.6	0.9310	504.3	1.48	0.45	−17.33	>	>	9.1	214458.2	0.9324	578.8	0.97	0.30	−27.97	=	>
UK100_15-B	**46.9**	**190124.9**	**0.9248**	**453.1**	10.2	185266.3	0.9181	540.9	2.62	0.73	−16.23	>	>	9.5	185755.9	0.9195	606.2	2.30	0.57	−25.26	>	>
UK100_16-B	**54.6**	**193615.0**	**0.9299**	**435.0**	10.1	190496.3	0.9292	482.7	1.64	0.08	−9.88	>	=	9.7	191060.6	0.9293	605.8	1.32	0.06	−28.19	>	=
UK100_17-B	**71.3**	**193250.2**	**0.9168**	562.4	10.3	189269.4	0.9096	**523.0**	2.10	0.79	7.53	>	>	9.6	189383.5	0.9091	629.1	2.00	0.84	−10.60	>	>
UK100_18-B	**59.7**	**200926.6**	0.9215	**392.9**	10.1	197711.8	0.9207	462.4	1.63	0.09	−15.03	=	=	9.2	200371.2	**0.9227**	580.2	0.28	−0.13	−32.28	=	=
UK100_19-B	**65.0**	**266002.1**	**0.9366**	**447.9**	10.6	263407.2	0.9334	467.7	0.99	0.34	−4.23	>	>	10.0	263429.8	0.9331	566.4	0.97	0.37	−20.92	>	>
UK100_20-B	**52.0**	**209712.5**	0.9285	**376.1**	10.3	207994.6	0.9279	438.6	0.83	0.06	−14.25	=	=	9.8	207317.0	**0.9298**	547.1	1.14	−0.14	−31.26	>	=
Average	**63.6**	**211809.8**	**0.9265**	**472.3**	10.2	208548.1	0.9225	493.6	1.61	0.44	−4.48			9.6	208941.5	0.9231	595.7	1.39	0.37	−20.89		

TABLE 5 Comparison between 2PPLS, WM, and WMN – Set C

	2PPLS				WM				Gap (%)			MW		WMN				Gap (%)			MW	
Instance	**#Sol.**	$\mathcal{H}$	$\mathcal{R}$	**T (s)**	**#Sol.**	$\mathcal{H}$	$\mathcal{R}$	**T (s)**	$\mathcal{H}$	$\mathcal{R}$	**T (s)**	$\mathcal{H}$	$\mathcal{R}$	**#Sol.**	$\mathcal{H}$	$\mathcal{R}$	**T (s)**	$\mathcal{H}$	$\mathcal{R}$	**T (s)**	$\mathcal{H}$	$\mathcal{R}$
UK100_01-C	**91.3**	**162461.7**	**0.9199**	448.5	10.3	157607.9	0.9104	**444.0**	3.08	1.04	1.01	>	>	10.3	158069.1	0.9095	542.9	2.78	1.14	−17.39	>	>
UK100_02-C	**88.7**	**183414.7**	**0.9242**	442.4	10.1	179111.3	0.9168	**438.6**	2.40	0.81	0.87	>	>	9.4	178585.4	0.9169	545.8	2.70	0.80	−18.94	>	>
UK100_03-C	**89.8**	**139473.5**	**0.9242**	533.8	10.2	135063.3	0.9121	**453.8**	3.27	1.33	17.63	>	>	9.3	135331.0	0.9133	562.5	3.06	1.19	−5.10	>	>
UK100_04-C	**83.5**	**203208.2**	**0.9391**	530.2	10.5	199436.1	0.9293	**425.2**	1.89	1.05	24.69	>	>	10.4	198719.5	0.9279	531.1	2.26	1.21	−0.17	>	>
UK100_05-C	**90.4**	**157937.2**	**0.9325**	490.5	10.2	154928.9	0.9250	**416.4**	1.94	0.81	17.80	>	>	10.3	154577.5	0.9234	509.3	2.17	0.99	−3.69	>	>
UK100_06-C	**99.8**	**170646.0**	**0.9264**	464.0	10.4	168222.6	0.9209	**429.9**	1.44	0.60	7.93	>	>	10.4	167543.9	0.9204	536.1	1.85	0.65	−13.45	>	>
UK100_07-C	**82.1**	**157590.5**	**0.9281**	**401.1**	9.9	155526.2	0.9235	409.7	1.33	0.50	−2.10	>	>	9.4	154879.1	0.9234	511.0	1.75	0.51	−21.51	>	>
UK100_08-C	**74.8**	**207032.6**	**0.9381**	398.9	10.5	203222.4	0.9324	**391.7**	1.87	0.61	1.84	>	>	9.7	203276.2	0.9326	480.2	1.85	0.59	−16.93	>	>
UK100_09-C	**84.3**	**190909.1**	**0.9270**	415.3	9.6	184317.5	0.9158	**371.4**	3.58	1.22	11.82	>	>	9.5	184470.2	0.9148	448.4	3.49	1.33	−7.38	>	>
UK100_10-C	**76.8**	**141106.8**	**0.9224**	428.1	10.0	137885.5	0.9152	**400.8**	2.34	0.79	6.81	>	>	10.2	138333.9	0.9163	506.6	2.00	0.67	−15.50	>	>
UK100_11-C	**75.5**	**229173.0**	**0.9376**	519.0	10.1	223793.0	0.9256	**441.6**	2.40	1.30	17.53	>	>	9.9	224391.4	0.9255	547.6	2.13	1.31	−5.22	>	>
UK100_12-C	**75.0**	**101181.4**	**0.9255**	**388.3**	9.5	99289.6	0.9194	389.9	1.91	0.66	−0.41	>	>	9.6	99795.4	0.9200	481.8	1.39	0.60	−19.41	>	>
UK100_13-C	**93.4**	**180856.1**	**0.9268**	474.0	10.4	177292.8	0.9197	**434.0**	2.01	0.77	9.22	>	>	10.1	177071.0	0.9202	541.2	2.14	0.72	−12.42	>	>
UK100_14-C	**85.1**	**185454.4**	**0.9258**	444.1	9.8	178732.2	0.9115	**431.6**	3.76	1.57	2.90	>	>	10.4	179568.3	0.9129	540.2	3.28	1.41	−17.79	>	>
UK100_15-C	**93.3**	**180755.8**	**0.9278**	461.2	10.3	173503.3	0.9100	**457.4**	4.18	1.96	0.83	>	>	10.0	174372.9	0.9103	577.9	3.66	1.92	−20.19	>	>
UK100_16-C	**76.4**	**129825.3**	**0.9352**	422.9	9.5	127382.9	0.9265	**387.4**	1.92	0.94	9.16	>	>	9.4	127690.7	0.9270	480.8	1.67	0.88	−12.04	>	>
UK100_17-C	**85.9**	**225831.9**	**0.9372**	**440.6**	10.0	222202.4	0.9278	466.4	1.63	1.01	−5.53	>	>	10.4	221603.8	0.9273	561.4	1.91	1.07	−21.52	>	>
UK100_18-C	**67.3**	**150931.3**	**0.9401**	**406.9**	9.7	148148.9	0.9274	425.8	1.88	1.37	−4.44	>	>	9.0	147608.4	0.9256	519.6	2.25	1.57	−21.69	>	>
UK100_19-C	**93.8**	**163494.9**	**0.9287**	520.8	9.7	158382.6	0.9187	**425.4**	3.23	1.09	22.43	>	>	9.8	159117.1	0.9188	540.1	2.75	1.08	−3.57	>	>
UK100_20-C	**102.4**	**148553.0**	**0.9198**	**425.1**	10.3	143872.7	0.9093	428.6	3.25	1.15	−0.82	>	>	10.4	145149.7	0.9097	540.5	2.34	1.11	−21.35	>	>
Average	**85.5**	**170491.9**	**0.9293**	452.8	10.1	166396.1	0.9199	**423.5**	2.47	1.03	6.96			9.9	166507.7	0.9198	525.3	2.37	1.04	−13.76		

TABLE 6 Summary results

		Set A				Set B				Set C			
Method	**Statistic**	**#Sol.**	$\mathcal{H}$	$\mathcal{R}$	T(s)	**#Sol.**	$\mathcal{H}$	$\mathcal{R}$	T (s)	**#Sol.**	$\mathcal{H}$	$\mathcal{R}$	T(s)
2PPLS	Min.	58.2	47400.5	0.8950	236.4	51.1	208392.3	0.9201	372.3	70.2	168126.6	0.9232	350.7
	Max.	82.5	48732.4	0.9050	976.6	76.8	215062.7	0.9330	589.4	101.4	173363.9	0.9361	621.4
	Avg.	69.9	48000.7	0.8997	394.4	63.6	211809.8	0.9265	472.3	85.5	170491.9	0.9293	452.8
WM	Min.	8.1	45402.8	0.8822	325.9	9.0	204296.3	0.9197	469.5	8.9	163405.2	0.9172	406.9
	Max.	10.5	46787.7	0.8857	356.5	11.0	211301.5	0.9251	519.5	11.0	168231.4	0.9228	439.4
	Avg.	9.4	46355.1	0.8840	339.6	10.2	208548.1	0.9225	493.6	10.1	166396.1	0.9199	423.5
WMN	Min.	8.3	45585.5	0.8814	394.3	8.2	205874.0	0.9207	574.8	8.8	163894.6	0.9171	505.3
	Max.	10.7	46725.0	0.8854	420.0	10.6	211081.8	0.9253	621.0	10.8	168223.7	0.9225	545.7
	Avg.	9.6	46297.6	0.8835	406.4	9.6	208941.5	0.9231	595.7	9.9	166507.7	0.9198	525.2

With respect to the instances of sets B and C, as can be observed in Tables 4 and 5, 2 PPLS was capable of finding a high number of potentially efficient solutions. In addition, it also visibly outperforms, on average, WM and WMN when considering $\mathcal{H}$ and $\mathcal{R}$ indicators. In terms of CPU time, the average performance of 2PPLS and WM are equivalent and both are faster than WMN.

The average results for each instance group is summarized in Table 6. For each method, it reports the minimum, maximum, and average values of #Sol., $\mathcal{H}$, $\mathcal{R}$, and T(s). The three methods usually require more CPU time for solving instances with tighter time-windows (i.e., those from sets B and C), which coincides with the results reported by Kramer et al. [37] for the mono-objective PRP. 2PPLS obtained more potentially efficient solutions for instances from the set C. This may be justified by the larger heterogeneity of the time-window widths of these instances. In contrast to 2PPLS, the number of potentially efficient solutions obtained by WM and WMN does not vary significantly between the instances sets. This is a consequence of the chosen weight set W (few elements with uniformly spread values).

5.3.2 | Evaluating the impact of the PLS

In this section, we discuss the impact of PLS in improving the quality of the Pareto fronts obtained during the first phase (Tables 7–9). In each table, we provide the number of potentially supported (**#St.**) and nonsupported efficient solutions (**#NSt.**) obtained

TABLE 7 Characterization of the solutions obtained by 2PPLS in each phase of the algorithm - Instances A

	First phase							Second phase					
Instance	**#Sol**	**#St.**	**#NSt.**	**#C.**	$\mathcal{H}$	$\mathcal{R}$	T(s)	**#Sol**	**#St.**	**#NSt.**	$\mathcal{H}$	$\mathcal{R}$	T(s)
UK100_01-A	8.9	8.3	0.6	9.4	50467.6	0.8819	302.8	66.8	11.9	54.9	52813.9	0.8959	16.7
UK100_02-A	11.6	9.6	2.0	12.4	47621.3	0.8862	390.5	80.0	12.9	67.1	49074.3	0.9002	21.0
UK100_03-A	11.8	9.8	2.0	13.0	40289.7	0.8903	413.8	69.0	12.8	56.2	41192.5	0.9020	23.0
UK100_04-A	11.1	7.8	3.3	14.4	48199.6	0.8879	449.8	72.7	14.7	58.0	50616.0	0.9108	68.4
UK100_05-A	10.1	6.7	3.4	14.0	36667.2	0.8843	450.5	69.8	11.9	57.9	38687.3	0.8991	26.0
UK100_06-A	10.1	8.5	1.6	10.2	48588.1	0.8822	290.4	65.6	14.8	50.8	50171.2	0.8950	19.9
UK100_07-A	11.2	7.4	3.8	14.4	38708.8	0.8844	385.8	63.8	10.8	53.0	40732.5	0.8971	16.3
UK100_08-A	10.9	8.2	2.7	13.4	41295.6	0.8835	358.2	62.9	12.8	50.1	42739.5	0.8958	12.2
UK100_09-A	8.1	6.8	1.3	9.6	42371.6	0.8833	296.3	74.3	11.1	63.2	44644.4	0.9048	51.5
UK100_10-A	10.7	8.4	2.3	13.0	42217.3	0.8822	335.8	67.1	13.1	54.0	43744.1	0.8961	16.6
UK100_11-A	7.5	6.3	1.2	9.2	52801.9	0.8731	297.5	74.0	12.3	61.7	57958.2	0.9002	58.4
UK100_12-A	10.6	7.1	3.5	16.6	38677.7	0.8825	478.5	65.2	12.6	52.6	40390.8	0.9003	27.1
UK100_13-A	11.2	8.4	2.8	13.2	45240.8	0.8850	367.0	72.7	13.3	59.4	46537.7	0.8972	13.8
UK100_14-A	9.1	8.5	0.6	9.8	47056.4	0.8822	300.9	63.9	13.7	50.2	49461.8	0.9019	20.6
UK100_15-A	11.9	9.8	2.1	12.8	62612.2	0.8871	437.3	75.1	13.7	61.4	64470.5	0.9012	22.7
UK100_16-A	10.1	7.5	2.6	11.6	39195.8	0.8872	312.1	70.5	13.5	57.0	41059.4	0.9003	22.3
UK100_17-A	10.8	8.9	1.9	11.6	56252.3	0.8779	412.6	68.1	12.0	56.1	58994.6	0.8950	47.9
UK100_18-A	11.5	8.7	2.8	14.2	44208.1	0.8854	458.4	69.3	13.8	55.5	45526.0	0.8988	17.6
UK100_19-A	8.5	7.5	1.0	9.0	42779.4	0.8902	267.5	73.6	14.3	59.3	44306.6	0.9055	24.9
UK100_20-A	10.7	8.7	2.0	10.8	55180.7	0.8847	338.7	74.3	13.8	60.5	56893.4	0.8969	15.4
Average	10.3	8.1	2.2	12.1	46021.6	0.8841	367.2	69.9	13.0	56.9	48000.7	0.8997	27.1

TABLE 8 Characterization of the solutions obtained by 2PPLS in each phase of the algorithm - Instances B

	First phase							Second phase					
Instance	**#Sol**	**#St.**	**#NSt.**	**#C.**	$\mathcal{H}$	$\mathcal{R}$	***T*(s)**	**#Sol**	**#St.**	**#NSt.**	$\mathcal{H}$	$\mathcal{R}$	***T*(s)**
UK100_01-B	7.3	7.1	0.2	7.4	131220.4	0.8968	353.6	62.3	12.2	50.1	137170.1	0.9126	128.5
UK100_02-B	6.4	6.0	0.4	7.0	206733.5	0.9141	313.1	67.1	11.3	55.8	217810.1	0.9261	148.7
UK100_03-B	8.2	7.5	0.7	8.8	203342.2	0.9129	428.8	98.0	13.6	84.4	212508.6	0.9211	230.3
UK100_04-B	6.3	6.2	0.1	6.8	196099.0	0.9091	321.8	52.9	10.6	42.3	202401.9	0.9171	125.4
UK100_05-B	7.4	6.7	0.7	7.8	221772.5	0.8965	368.7	69.9	12.6	57.3	230489.6	0.9057	194.3
UK100_06-B	4.5	4.4	0.1	4.8	213304.8	0.9114	216.4	60.1	11.3	48.8	226259.5	0.9296	196.8
UK100_07-B	4.1	4.1	0.0	4.6	223105.0	0.9147	203.7	49.0	11.1	37.9	234897.4	0.9282	136.0
UK100_08-B	7.0	6.6	0.4	7.2	230346.7	0.9289	324.1	73.3	13.0	60.3	234991.1	0.9392	139.7
UK100_09-B	5.8	5.8	0.0	6.2	218667.3	0.9269	265.8	69.1	12.4	56.7	226787.6	0.9409	234.8
UK100_10-B	8.5	7.6	0.9	8.6	205521.6	0.9261	370.4	77.4	13.1	64.3	210447.0	0.9345	141.8
UK100_11-B	7.6	7.1	0.5	8.0	219813.5	0.9202	429.2	71.1	12.4	58.7	226803.7	0.9299	186.1
UK100_12-B	4.8	4.3	0.5	5.0	146086.3	0.9053	240.2	62.5	10.5	52.0	155418.0	0.9255	172.2
UK100_13-B	5.6	5.6	0.0	5.8	241400.6	0.9162	300.2	55.2	9.8	45.4	250030.6	0.9258	191.3
UK100_14-B	4.8	4.8	0.0	5.2	205538.3	0.9198	258.0	55.3	11.7	43.6	216549.3	0.9352	158.9
UK100_15-B	6.7	6.6	0.1	6.8	184101.4	0.9131	364.6	46.9	12.0	34.9	190124.9	0.9248	88.5
UK100_16-B	5.5	5.5	0.0	6.0	184777.6	0.9207	297.1	54.6	10.6	44.0	193615.0	0.9299	137.9
UK100_17-B	7.6	7.0	0.6	8.2	182544.9	0.9054	420.7	71.3	11.1	60.2	193250.2	0.9168	141.7
UK100_18-B	5.2	4.8	0.4	5.4	190524.1	0.9075	246.6	59.7	10.4	49.3	200926.6	0.9215	146.3
UK100_19-B	5.9	5.6	0.3	6.2	255855.4	0.9247	280.0	65.0	13.4	51.6	266002.1	0.9366	167.9
UK100_20-B	6.1	6.1	0.0	6.6	199861.3	0.9196	290.1	52.0	11.8	40.2	209712.5	0.9285	86.0
Average	6.3	6.0	0.3	6.6	203030.8	0.9145	314.7	63.6	11.7	51.9	211809.8	0.9265	157.7

TABLE 9 Characterization of the solutions obtained by 2PPLS in each phase of the algorithm - Instances C

	First phase							Second phase					
Instance	**#Sol**	**#St.**	**#NSt.**	**#C.**	$\mathcal{H}$	$\mathcal{R}$	***T*(s)**	**#Sol**	**#St.**	**#NSt.**	$\mathcal{H}$	$\mathcal{R}$	***T*(s)**
UK100_01-C	7.5	7.0	0.5	7.8	155257.5	0.9055	317.9	91.3	15.0	76.3	162461.7	0.9199	130.7
UK100_02-C	7.3	6.6	0.7	8.2	170925.5	0.9125	324.0	88.7	12.9	75.8	183414.7	0.9242	118.3
UK100_03-C	7.5	6.9	0.6	7.8	133884.5	0.9093	318.2	89.8	14.7	75.1	139473.5	0.9242	215.5
UK100_04-C	7.7	6.8	0.9	8.2	196122.4	0.9257	319.7	83.5	13.4	70.1	203208.2	0.9391	210.5
UK100_05-C	8.0	6.7	1.3	8.6	153161.5	0.9201	330.1	90.4	13.4	77.0	157937.2	0.9325	160.4
UK100_06-C	8.6	8.1	0.5	9.0	164855.7	0.9172	354.6	99.8	15.8	84.0	170646.0	0.9264	109.5
UK100_07-C	6.5	5.9	0.6	7.0	148690.5	0.9130	261.7	82.1	13.6	68.5	157590.5	0.9281	139.4
UK100_08-C	7.1	6.5	0.6	7.6	198801.8	0.9267	285.6	74.8	12.4	62.4	207032.6	0.9381	113.3
UK100_09-C	5.3	5.2	0.1	5.6	178483.2	0.9028	185.8	84.3	14.4	69.9	190909.1	0.9270	229.4
UK100_10-C	9.2	7.4	1.8	10.2	138056.8	0.9151	377.2	76.8	13.4	63.4	141106.8	0.9224	50.9
UK100_11-C	6.5	6.1	0.4	7.0	218888.7	0.9194	285.2	75.5	12.4	63.1	229173.0	0.9376	233.7
UK100_12-C	7.9	7.1	0.8	9.0	98566.2	0.9149	314.5	75.0	12.8	62.2	101181.4	0.9255	73.8
UK100_13-C	8.1	7.7	0.4	8.4	175513.5	0.9167	347.8	93.4	14.7	78.7	180856.1	0.9268	126.2
UK100_14-C	7.9	7.5	0.4	8.0	178913.3	0.9086	326.5	85.1	13.0	72.1	185454.4	0.9258	117.6
UK100_15-C	7.2	6.5	0.7	8.0	167282.9	0.9064	338.3	93.3	14.2	79.1	180755.8	0.9278	122.9
UK100_16-C	6.9	5.6	1.3	8.0	124246.0	0.9171	282.7	76.4	14.0	62.4	129825.3	0.9352	140.2
UK100_17-C	6.9	6.2	0.7	7.2	215452.6	0.9212	308.6	85.9	13.5	72.4	225831.9	0.9372	132.0
UK100_18-C	7.4	6.1	1.3	8.2	142504.0	0.9234	320.7	67.3	10.5	56.8	150931.3	0.9401	86.3
UK100_19-C	8.4	7.5	0.9	8.8	158845.2	0.9152	346.5	93.8	13.5	80.3	163494.9	0.9287	174.3
UK100_20-C	7.7	7.4	0.3	7.8	143986.1	0.9076	314.1	102.4	14.5	87.9	148553.0	0.9198	111.0
Average	7.5	6.7	0.7	8.0	163121.9	0.9149	313.0	85.5	13.6	71.9	170491.9	0.9293	139.8

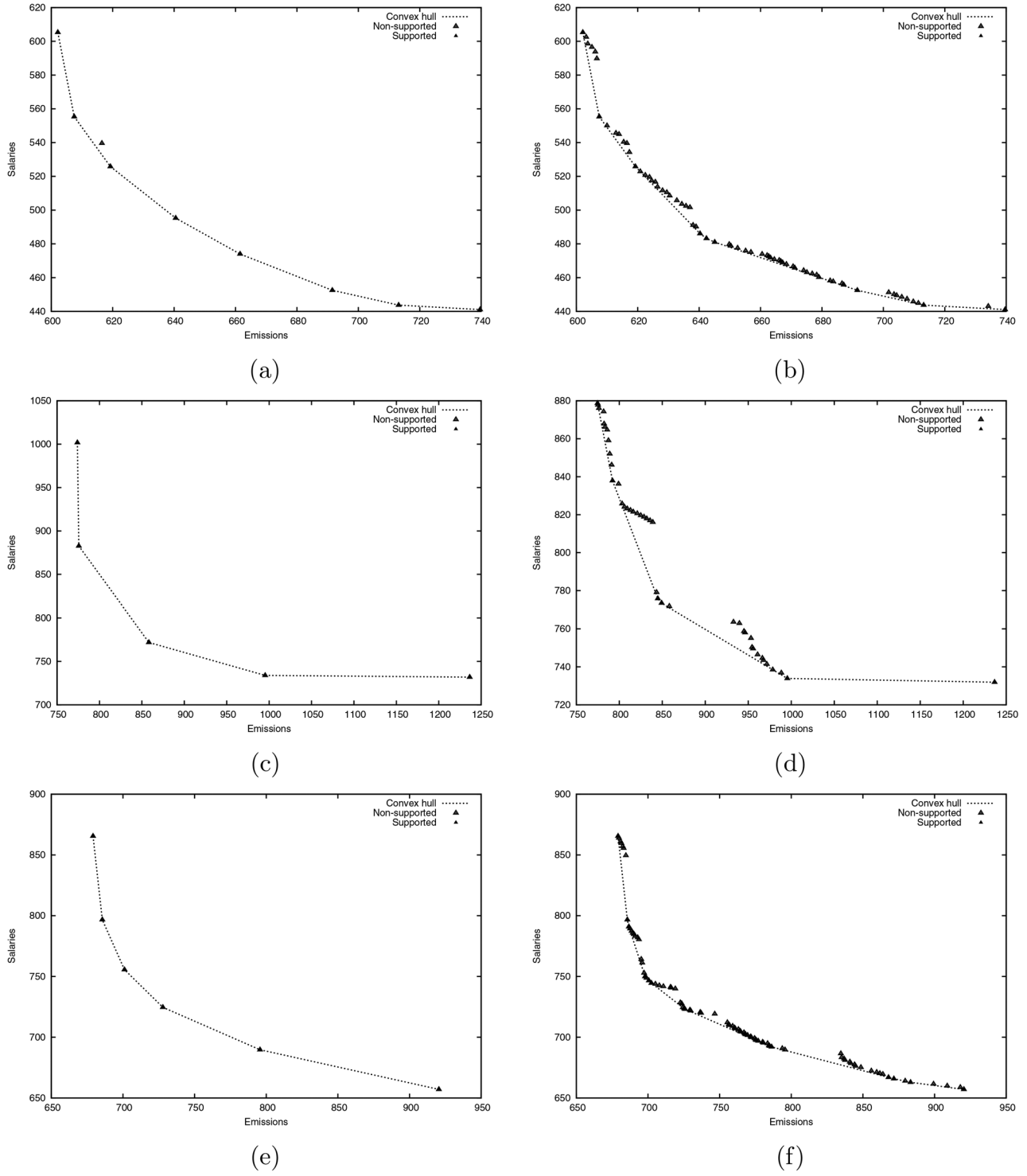

FIGURE 3 Convex hull for the Pareto front obtained during the two phases of 2PPLS. (a) Instance UK100_13-A - phase 1. (b) Instance UK100_13-A - phase 2. (c) Instance UK100_13-B - phase 1. (d) Instance UK100_13-B - phase 2. (e) Instance UK100_13-C - phase 1. (f) Instance UK100_13-C - phase 2

after each phase. The convex hulls of the fronts, which are required to identify the supported solutions, were obtained by employing Graham's algorithm [24]. A representation of the convex hull associated with some instances is depicted in Figure 3. Note that as we use a heuristic in the first phase to solve the weighted sum problems, potentially nonsupported efficient solutions can also be generated during the first phase. We also present in column **#C.** the number of calls to the mono-objective solver during the first phase.

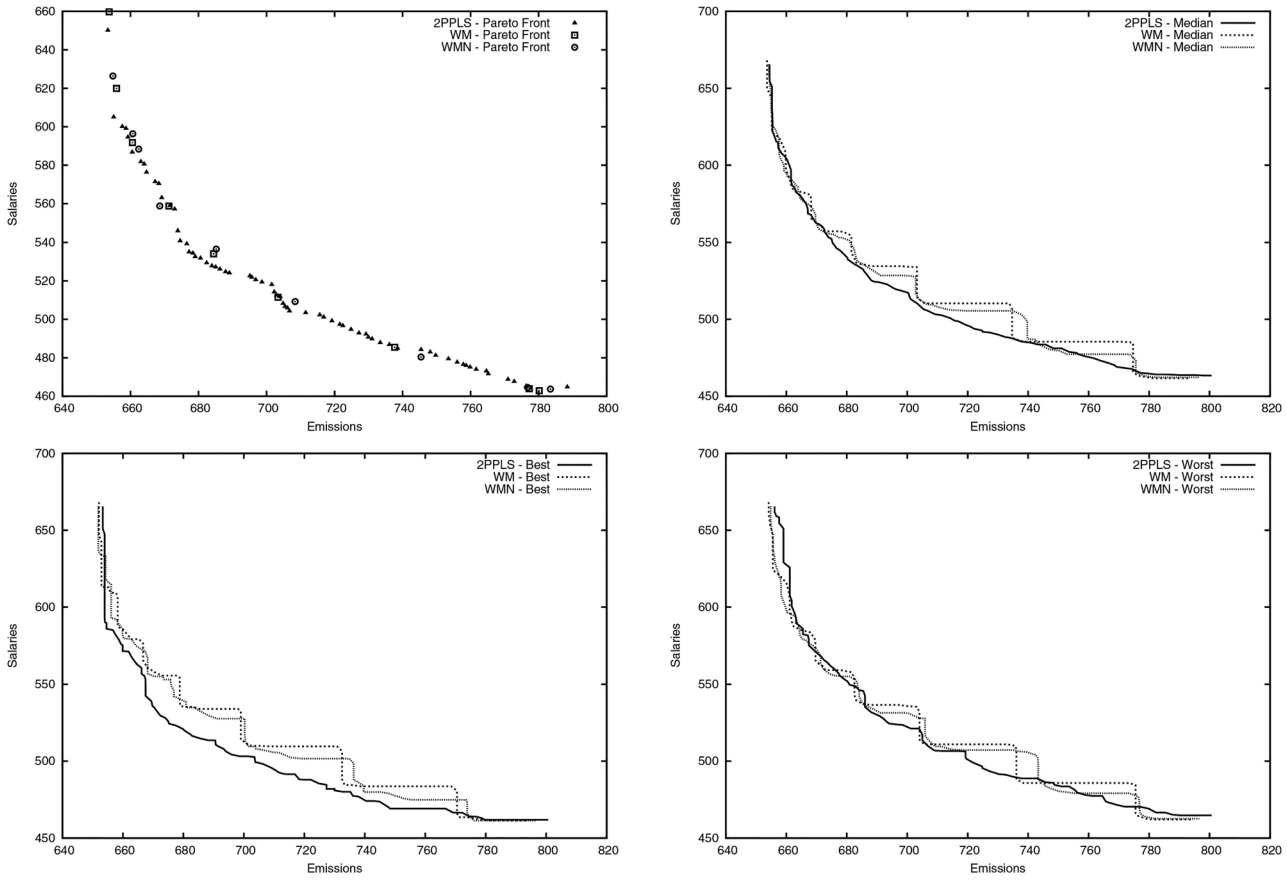

FIGURE 4 Comparison of the Pareto front (first run) an attainment surfaces of 2PPLS vs WM vs WMN for instance 1 of type A

We can observe that the second phase substantially improves the results found in the first phase. The first phase is quite time consuming: solving one single-objective PRP takes on average at least 30 seconds. It is thus important to find a good compromise between the first phase and PLS. The second phase is much more faster than the first one, especially for the instances of Set A. On average, PLS spends, for the instances of Sets A, B and C, 6.90%, 33.59%, and 30.51% of the total CPU time, respectively. The reason for the larger CPU time required by the second phase when solving instances from sets B and C is the same as explained previously, that is, the existence of tight time windows. As a result, the quality of the fronts obtained during the first phase is not as good as the one associated with the instances of Set A. It is thus more challenging for PLS to improve the fronts during the second phase. Nonetheless, the total CPU time of 2PPLS still remains competitive when compared to WM and WMN (Tables 4 and 5).

In addition, the results also confirm what has been already hinted in [43]: the computational effort of generating a good estimation of the Pareto front in the first phase can be extremely useful to enhance the performance of the second phase.

5.3.3 | Attainment surfaces

This section compares the worst, median and best summary attainment surfaces obtained by 2PPLS, WM, and WMN. However, to limit the number of figures, we only expose the results obtained for the first instance of each type (that is UK100-01-A, UK100_ 01-B and UK100_ 01-C). Similar results have been obtained for the other instances. We also represent the approximations of Pareto front obtained in the first run of each algorithm. The results for the instances of types A, B, and C are provided in Figures 4, 5, and 6, respectively.

As already observed in Tables 3–5, we remark that the approximations of Pareto fronts of 2PPLS contains many more points that WM and WMN. The points of WM and WMN are, however, well located and well spread. We also remark that the median and best summary attainment surfaces of 2PPLS are clearly better than WM and WMN for all types of instances. Nevertheless, for instances B and C, the worst summary attainment surfaces of 2PPLS are globally worse than those of WM and WMN. We can explain that by the fact that for instances B and C, the number of weighted sum generation in phase 1 of 2PPLS is lower

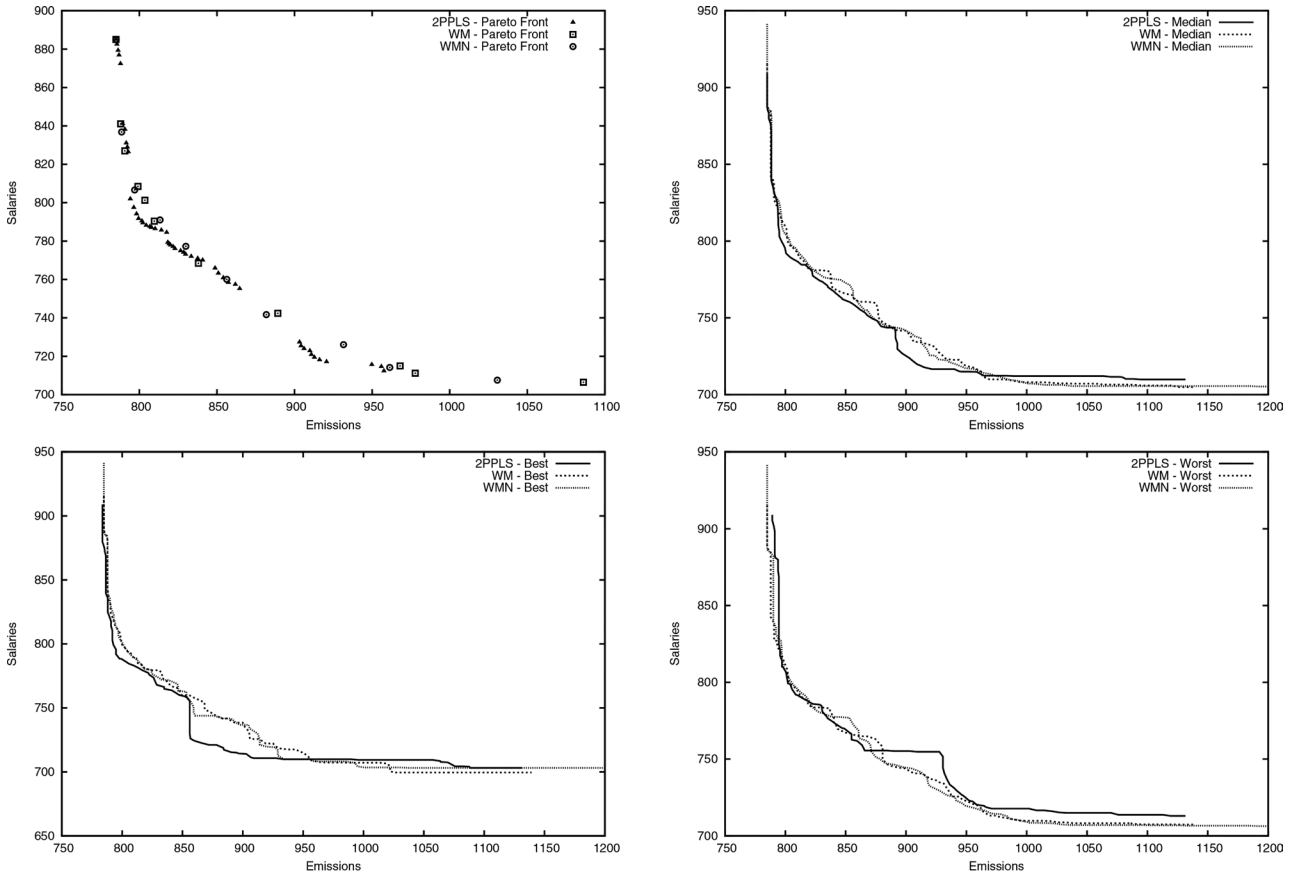

FIGURE 5 Comparison of the Pareto front (first run) an attainment surfaces of 2PPLS vs WM vs WMN for instance 1 of type B

than the number of weighted sum generation of WM and WMN. Therefore, the chances that all the weighted sum generations of 2PPLS give low-quality results are higher than with WM or WMN, which make these algorithms more robust for instances B and C. For instances of type A, the worst summary attainment surfaces of 2PPLS are better.

6 | CONCLUDING REMARKS

This article proposed a two-phase Pareto local search (2PPLS) for the bi-objective pollution-routing problem (bPRP). The proposed algorithm combines the efficient ideas presented in [37] for the single-objective PRP with those developed in [43] for general MO problems. We showed that using PLS allows to limit the number of computationally demanding weighted sum problems solved in the first phase while keeping high-quality results, which is better than preceding state-of-the-art methods. The method remains, however, simple; the same neighborhood operators are used in PLS and in the single-objective method. Furthermore, the method proposed in this work could be easily adapted to solve other complex MO problems, for which an efficient single-objective heuristic is known. The crucial part of such adaptation is to find the good compromise between the number of weighted sum problems solved in the first phase and the size of the neighborhood used in PLS. As for future work, we intend to propose a full automatic way to adapt a heuristic single-objective solver to obtain high-quality results for MO versions of the same problem. Moreover, richer PRP variants, such as the time-dependent PRP [18, 19] and the heterogeneous fleet PRP [33], might be addressed under a multi-objective paradigm.

ACKNOWLEDGMENTS

This research was partially supported by Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq/Brazil), grants 132610/2014-0, 132789/2015-9, 305223/2015-1, 428549/2016-0, and GDE 201222/2014-0.

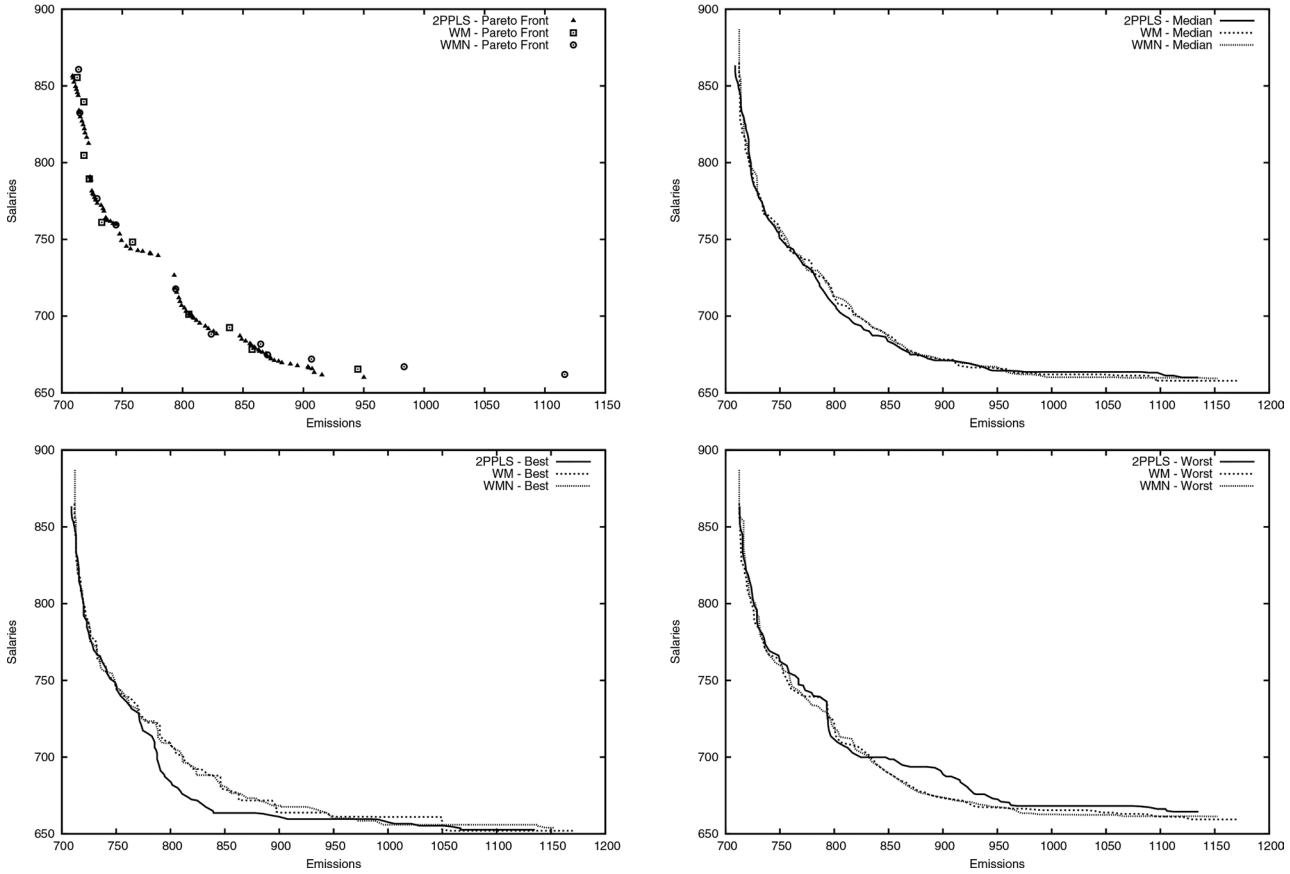

FIGURE 6 Comparison of the Pareto front (first run) an attainment surfaces of 2PPLS vs WM vs WMN for instance 1 of type C

ORCID

Luciano Costa http://orcid.org/0000-0002-6324-6556
Thibaut Lust http://orcid.org/0000-0001-8433-518X

REFERENCES

[1] Y.P. Aneja and K.P.K. Nair, *Bicriteria transportation problem*, Manag. Sci. **25** (1979), 73–78.

[2] E. Angel, E. Bampis, and L. Gourvès, "*A dynasearch neighborhood for the dicriteria traveling salesman problem*," *Metaheuristics for Multi-objective Optimization*, Volume 535 of *Lecture Notes in Economics and Mathematical Systems*, X. Gandibleux et al. (eds.), Springer, Berlin, 2004, pp. 153–176.

[3] M. Barth and K. Boriboonsomsin, *Real-world carbon dioxide impacts of traffic congestion*, Transp. Res. Rec. **1** (2008), 163–171.

[4] M. Barth, T. Younglove, and G. Scora, Development of a heavy-duty diesel modal emissions and fuel consumption model, Technical Report, California Partners for Advanced Transit and Highways (PATH), UC Berkeley (2005).

[5] T. Bektaş and G. Laporte, *The pollution-routing problem*, Transp. Res. B Methodol. **45** (2011), 1232–1250.

[6] S. Chand and M. Wagner, *Evolutionary many-objective optimization: A quick-start guide*, Surv. Oper. Res. Manag. Sci. **20** (2015), 35–42.

[7] S. Dabia, E. Demir, and T. Van Woensel, *An exact approach for a variant of the pollution-routing problem*, Transp. Sci. **51** (2017), 607–628.

[8] S. Dabia et al., *Approximating multi-objective scheduling problems*, Comput. Opera. Res. **40** (2013), 1165–1175.

[9] K. Deb et al., *A fast and elitist multiobjective genetic algorithm: NSGA-II*, IEEE Trans. Evolut. Comput. **6** (2002), 182–197.

[10] E. Demir, T. Bektaş, and G. Laporte, *A comparative analysis of several vehicle emission models for road freight transportation*, Transp. Res. D Transp. Environ. **16** (2011), 347–357.

[11] E. Demir, T. Bektaş, and G. Laporte, *An adaptive large neighborhood search heuristic for the pollution-routing problem*, Eur. J. Oper. Res. **223** (2012), 346–359.

[12] E. Demir, T. Bektaş, and G. Laporte, *The bi-objective pollution-routing problem*, Eur. J. Oper. Res. **232** (2014), 464–478.

[13] E. Demir, T. Bektaş, and G. Laporte, *A review of recent research on green road freight transportation*, Eur. J. Oper. Res. **237** (2014), 775–793.

[14] G. Desaulniers, O.B.G. Madsen, and S. Ropke, "*The vehicle routing problem with time windows*," *Vehicle Routing: Problems, Methods and Applications*, 2nd ed., chapter 5, P. Toth and D. Vigo (eds.), Society for Industrial and Applied Mathematics, Philadelphia, PA, 2014, pp. 119–159.

[15] M. Ehrgott, *Multicriteria Optimization*, 2nd ed., Springer, Berlin, 2005.

[16] M. Figliozzi, *Vehicle routing problem for emissions minimization*, Transp. Res. Rec. **2197** (2010), 1–7.

[17] C.M. Fonseca and P.J. Fleming, "*On the performance assessment and comparison of stochastic multiobjective optimizers*," *The 4th International Conference on Parallel Problem Solving from Nature Berlin*, Germany, September 22–26, 1996 Proceedings, H.M. Voigt et al. (eds.) Springer, Berlin, 1996, pp. 584–593.

[18] A. Franceschetti et al., *A metaheuristic for the time-dependent pollution-routing problem*, Eur. J. Oper. Res. **259** (2017), 972–991.

[19] A. Franceschetti et al., *The time-dependent pollution-routing problem*, Transp. Res. B Methodol. **56** (2013), 265–293.

[20] R. Fukasawa et al., A joint routing and speed optimization problem. Technical Report. Optimization-Online, 2017, www.optimization-online.org/DB_FILE/2016/02/5344.pdf.

[21] R. Fukasawa, Q. He, and Y. Song, *A branch-cut-and-price algorithm for the energy minimization vehicle routing problem*, Transp. Sci. **50** (2016), 23–34.

[22] R. Fukasawa, Q. He, and Y. Song, *A disjunctive convex programming approach to the pollution-routing problem*, Transp. Res. B Methodol. **94** (2016), 61–79.

[23] X. Gandibleux et al., (eds.), *Metaheuristics for Multiobjective Optimisation. Lecture Notes in Economics and Mathematical Systems*, 1st ed., Springer, Berlin, 2004.

[24] R.L. Graham, *An efficient algorithm for determining the convex hull of a finite planar set*, Inf. Process. Lett. **1** (1972), 132–133.

[25] S. Holm, *A simple sequentially rejective multiple test procedure*, Scand. J. Stat. **6** (1979), 65–70.

[26] IEA, CO_2 emissions from fuel combustion: Overview. Technical Report. International Energy Agency, 2017.

[27] O. Jabali, T. Van Woensel, and A.G. de Kok, *Analysis of travel times and CO2 emissions in time-dependent vehicle routing*, Prod. Oper. Manag. **21** (2012), 1060–1074.

[28] A. Jaszkiewicz, *Genetic local search for multi-objective combinatorial optimization*, Eur. J. Oper. Res. **137** (2002), 50–71.

[29] A. Jaszkiewicz, *On the performance of multiple-objective genetic local search on the 0/1 knapsack problem – a comparative experiment*, IEEE Trans. Evolut. Comput. **6** (2002), 402–412.

[30] J. Jemai, M. Zekri, and K. Mellouli, "*An NSGA-II algorithm for the green vehicle routing problem*," *Evolutionary Computation in Combinatorial Optimization*, Springer, Berlin, 2012, pp. 37–48.

[31] D. Joshua, L.T. Knowles, and E. Zitzler, A Tutorial on the performance assessment of stochastic multiobjective optimizers. Technical Report. Computer Engineering and Networks Laboratory (TIK), ETH Zurich, 2006.

[32] J. Knowles, *A summary-attainment-surface plotting method for visualizing the performance of stochastic multiobjective optimizers*, 5th International Conference on Intelligent Systems Design and Applications (ISDA'05), 2005, pp. 552–557.

[33] Ç. Koç et al., *The fleet size and mix pollution-routing problem*, Transp. Res. B Methodol. **70** (2014), 239–254.

[34] H. Kopfer and H. Kopfer, "*Emissions minimization vehicle routing problem in dependence of different vehicle classes*," *Dynamics in Logistics, Lecture Notes in Logistics*, H.J. Kreowski, B. Scholz-Reiter, and K.D. Thoben (eds.), Springer, Berlin, 2013, pp. 49–58.

[35] H.W. Kopfer, J. Schönberger, and H. Kopfer, *Reducing greenhouse gas emissions of a heterogeneous vehicle fleet*, Flex. Serv. Manuf. J. **26** (2014), 221–248.

[36] R. Kramer et al., *A speed and departure time optimization algorithm for the pollution-routing problem*, Eur. J. Oper. Res. **247** (2015), 782–787.

[37] R. Kramer et al., *A matheuristic approach for the pollution-routing problem*, Eur. J. Oper. Res. **243** (2015), 523–539.

[38] R.S. Kumar et al., *Multi-objective modeling of production and pollution routing problem with time window: A self-learning particle swarm optimization approach*, Comput. Ind. Eng. **99** (2016), 29–40.

[39] Y. Kuo, *Using simulated annealing to minimize fuel consumption for the time-dependent vehicle routing problem*, Comput. Ind. Eng. **59** (2010), 157–165.

[40] C. Lin et al., *Survey of green vehicle routing problem: Past and future trends*, Exp. Syst. Appl. **41** (2014), 1118–1138.

[41] H.R. Lourenço, O.C. Martin, and T. Stützle, "*Iterated local search: Framework and applications*," *Handbook of Metaheuristics*, Volume 146 of *International Series in Operations Research & Management Science*, M. Gendreau and J.Y. Potvin (eds.), Springer, New York, 2010, pp. 363–397.

[42] T. Lust, New metaheuristics for solving MOCO problems: Application to the knapsack problem, the traveling salesman problem and IMRT optimization. Ph.D. thesis. Université de Mons, 2009.

[43] T. Lust and J. Teghem, *Two-phase Pareto local search for the biobjective traveling salesman problem*, J. Heuristics **16** (2009), 475–510.

[44] T. Lust and J. Teghem, *The multiobjective multidimensional knapsack problem: A survey and a new approach*, Int. Trans. Oper. Res. **19** (2012), 495–520.

[45] T. Lust and D. Tuyttens, *Variable and large neighborhood search to solve the multiobjective set covering problem*, J. Heuristics **20** (2014), 165–188.

[46] E.Q.V. Martins, *On a multicriteria shortest path problem*, Eur. J. Oper. Res. **16** (1984), 236–245.

[47] J.C. Molina et al., *Multi-objective vehicle routing problem with cost and emission functions*, Procedia Soc. Behav. Sci. **160** (2014), 254–263.

[48] L. Paquete, M. Chiarandini, and T. Stützle, "*Pareto local optimum sets in the biobjective traveling salesman problem: An experimental study*," *Metaheuristics for multiobjective optimization*, Volume 535 of *Lecture Notes in Economics and Mathematical Systems*, X. Gandibleux et al. (eds.), Springer, Berlin, 2004, pp. 177–199.

[49] L. Paquete and T. Stützle, "*A two-phase local search for the biobjective traveling salesman problem*," *Evolutionary Multi-Criterion Optimization*, Volume 2632 of *Lecture Notes in Computer Science*, C.M. Fonseca et al. (eds.), Springer, Berlin, 2003, pp. 479–493.

[50] P.H.V. Penna, A. Subramanian, and L.S. Ochi, *An iterated local search heuristic for the heterogeneous fleet vehicle routing problem*, J. Heuristics **19** (2013), 201–232.

[51] A. Przybylski, X. Gandibleux, and M. Ehrgott, *Two phase algorithms for the bi-objective assignment problem*, Eur. J. Oper. Res. **185** (2008), 509–533.

[52] R Core Team, R: A language and environment for statistical computing. R Foundation for Statistical Computing. Vienna, Austria. URL: http://www.R-project.org/. ISBN 3-900051-07-0, 2015.

[53] A. Sbihi and R.W. Eglese, *Combinatorial optimization and green logistics*, 4OR **5** (2007), 99–116.

[54] W.S.H. Siu, C.K. Chan, and H.C.B. Chan, *Green cargo routing using genetic algorithms*, Proceedings of the International Multiconference of Engineers and Compute Scientists, Hong Kong, 2012, pp. 12–17.

[55] M. Soysal, J.M. Bloemhof-Ruwaard, and T. Bektaş, *The time-dependent two-echelon capacitated vehicle routing problem with environmental considerations*, Int. J. Prod. Econ. **164** (2015), 366–378.

[56] P. Toth and D. Vigo, *The Vehicle Routing Problem*, 1st ed., SIAM, Philadelphia, 2002.

[57] T. Vidal et al., *A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows*, Comput. Oper. Res. **40** (2013), 475–489.

[58] Y. Xiao et al., *Development of a fuel consumption optimization model for the capacitated vehicle routing problem*, Comput. Oper. Res. **39** (2012), 1419–1431.

[59] E. Zitzler, Evolutionary algorithms for multiobjective optimization: Methods and applications. Ph.D. thesis. Swiss Federal Institute of Technology Zurich. Switzerland, 1999

[60] E. Zitzler, K. Deb, and L. Thiele, *Comparison of multiobjective evolutionary algorithms: Empirical results*, Evol. Comput. **8** (2000), 173–195.

[61] E. Zitzler et al., *Performance assessment of multiobjective optimizers: An analysis and review*, IEEE Trans. Evolut. Comput. **7** (2003), 117–132.

How to cite this article: Costa L, Lust T, Kramer R, Subramanian A. A two-phase Pareto local search heuristic for the bi-objective pollution-routing problem. *Networks.* 2018;72:311–336. https://doi.org/10.1002/net.21827

APPENDIX: COMPARISON BETWEEN THE PRP ALGORITHMS

In this section, we compare the performance of five algorithms for solving the mono-objective PRP, namely: the ALNS algorithm by Demir et al. [11] (`DBL12`), the hybrid evolutionary algorithm by Koç et al. [33] (`KBJL14`), the ALNS algorithm by Franceschetti et al. [18] (`FDHWLS17`), the matheuristic ILS-SP-SOA by Kramer et al. [37] (`KSVC15`), and a simpler version of the latter that does not include the SP component (`KSVC15-basic`). This experiment has been performed in order to select a fast yet efficient method to be integrated into our 2PPLS algorithm.

The computational environments where each algorithm was executed is provided in Table 9. Only the results for `KSVC15-basic` have been obtained from our experiments. The ones for `DBL12`, `KBJL14`, `KSVC15`, and `FDHWLS17` have been collected from the references given in the second column of Table 9.

The results of each method are reported in Tables 2, 4. The first two columns contain the name of each instance and their corresponding best-known solution (BKS). The remaining columns report the solution costs (**Cost**), the percentage gaps with respect to BKS (**Gap (%)**), and the CPU time (**T(s)**) required by each method. The results for `DBL12`, `KBJL14`, and `FDHWLS17` refer to the best solutions of 10 runs, while those for `KSVC15` and `KSVC15-basic` refer to the average of 10 runs. Among the state-of-the-art algorithms, only Kramer et al. [37] reported results for the instance sets B and C.

TABLE A1 Computational environments

Algorithm	Reference	Machine	Lang. Prog.
KSVC15	Kramer et al. [37]	Intel(R) i7-3440 CPU @ 3.40 GHz with 16Gb	C++
KSVC15-basic	Kramer et al. [37] Adapt.	Intel(R) i7-6700 CPU @ 3.40 GHz with 16Gb	C++
DBL12	Demir et al. [11]	Intel Xeon(R) @3.00 GHz with 1Gb	C++
KBJL14	Koç et al. [33]	Intel Xeon(R) @2.60 GHz with 1Gb	C++
FDHWLS17	Franceschetti et al. [18]	Intel Xeon(R) X5675 @3.07 GHz with 96Gb	Java

Although the algorithms have been tested in different computational environments, KSVC15-basic appears to be the fastest algorithm. Regarding the solution costs, KSVC15 was capable of finding better results for most of the instances. Nevertheless, when comparing the average costs of KSVC15-basic with those of DBL12, KBJL14, and FDHWLS17 (Table 9), we can observe that KSVC15-basic still seems highly competitive, clearly outperforming DBL12. Based on these results, we decided to use KSVC15-basic in our 2PPLS algorithm for solving the bPRP.

TABLE A2 Comparing algorithms performance for solving the mono-objective PRP – Set A instances

		KSVC15-basic			KSVC15			DBL12			KBJL14			FDHWLS17		
Instance	**BKS**	**Cost**	**Gap(%)**	***T*(s)**	**Cost**	**Gap(%)**	***T*(s)**	**Cost**	**Gap(%)**	***T*(s)**	**Cost**	**Gap(%)**	***T*(s)**	**Cost**	**Gap(%)**	***T*(s)**
UK100_01-A	1209.11	1217.23	0.67	17.99	1211.34	0.18	34.65	1240.79	2.62	92.10	1212.72	0.30	262.20	1216.18	0.58	-
UK100_02-A	1146.55	1150.99	0.39	16.73	1148.79	0.20	33.20	1168.17	1.89	98.20	1149.16	0.23	280.20	1146.55	0.00	-
UK100_03-A	1078.75	1082.19	0.32	17.21	1080.11	0.13	33.28	1092.73	1.30	207.90	1080.87	0.20	317.40	1089.74	1.02	-
UK100_04-A	1075.29	1083.32	0.75	16.61	1077.42	0.20	35.79	1106.48	2.90	149.70	1085.66	0.96	307.80	1086.95	1.08	-
UK100_05-A	1028.86	1038.62	0.95	18.68	1036.51	0.74	33.63	1043.41	1.41	159.00	1033.19	0.42	295.80	1041.57	1.24	-
UK100_06-A	1192.67	1196.29	0.30	16.23	1195.60	0.25	29.37	1213.61	1.76	133.80	1192.67	0.00	289.80	1194.73	0.17	-
UK100_07-A	1044.58	1051.51	0.66	14.70	1047.66	0.29	28.63	1060.08	1.48	102.60	1044.58	0.00	270.60	1051.99	0.71	-
UK100_08-A	1089.84	1093.42	0.33	13.31	1092.70	0.26	26.63	1106.78	1.55	209.50	1092.67	0.26	340.20	1090.29	0.04	-
UK100_09-A	988.41	992.18	0.38	16.69	991.18	0.28	30.47	1015.46	2.74	154.00	992.36	0.40	298.20	991.38	0.30	-
UK100_10-A	1059.95	1063.97	0.38	15.12	1061.45	0.14	29.73	1076.56	1.57	199.00	1063.05	0.29	338.40	1067.70	0.73	-
UK100_11-A	1196.50	1205.91	0.79	19.30	1202.36	0.49	36.26	1210.25	1.15	107.10	1200.53	0.34	246.60	1204.70	0.69	-
UK100_12-A	1027.38	1036.87	0.92	15.06	1029.86	0.24	31.91	1053.02	2.50	206.40	1030.17	0.27	338.40	1039.43	1.17	-
UK100_13-A	1129.73	1134.48	0.42	15.90	1134.15	0.39	27.46	1154.83	2.22	87.90	1132.02	0.20	209.40	1129.73	0.00	-
UK100_14-A	1241.31	1243.66	0.19	17.29	1243.67	0.19	31.45	1264.50	1.87	91.80	1241.31	0.00	257.40	1245.03	0.30	-
UK100_15-A	1300.13	1306.64	0.50	18.29	1303.81	0.28	36.24	1315.50	1.18	110.90	1311.36	0.86	232.20	1306.31	0.48	-
UK100_16-A	980.46	986.05	0.57	14.30	984.52	0.41	28.14	1005.03	2.51	254.70	986.57	0.62	358.20	980.46	0.00	-
UK100_17-A	1257.44	1262.04	0.37	19.88	1259.27	0.15	38.88	1284.81	2.18	152.80	1257.44	0.00	251.40	1267.22	0.78	-
UK100_18-A	1073.38	1083.70	0.96	16.81	1081.40	0.75	33.15	1106.00	3.04	92.60	1088.89	1.44	252.60	1086.44	1.22	-
UK100_19-A	1015.95	1020.15	0.41	16.33	1018.71	0.27	30.67	1044.71	2.83	91.00	1024.17	0.81	251.40	1016.82	0.09	-
UK100_20-A	1237.87	1246.49	0.70	16.91	1244.41	0.53	30.05	1263.06	2.03	204.40	1249.84	0.97	310.20	1237.87	0.00	-
Average	-	1124.79	0.55	16.67	1122.25	0.32	31.98	1141.29	2.04	145.27	1123.46	0.43	285.42	1124.55	0.53	-

TABLE A3 Comparing algorithms performance for solving the mono-objective PRP – Set B instances

		KSVC15-basic			KSVC15		
Instance	**BKS**	**Cost**	**Gap(%)**	***T*(s)**	**Cost**	**Gap(%)**	***T*(s)**
UK100_01-B	1591.20	1600.63	0.59	24.21	1594.84	0.23	83.29
UK100_02-B	1599.56	1610.68	0.70	21.53	1603.01	0.22	96.50
UK100_03-B	1500.40	1520.16	1.32	24.03	1506.67	0.42	229.98
UK100_04-B	1472.49	1478.86	0.43	22.41	1473.84	0.09	117.36
UK100_05-B	1488.73	1504.53	1.06	23.40	1493.42	0.32	108.81
UK100_06-B	1645.77	1658.09	0.75	22.17	1649.37	0.22	56.75
UK100_07-B	1508.05	1515.39	0.49	21.18	1510.73	0.18	80.10
UK100_08-B	1466.62	1493.19	1.81	20.37	1477.29	0.73	110.27
UK100_09-B	1377.64	1381.65	0.29	18.12	1380.30	0.19	60.76
UK100_10-B	1477.25	1487.24	0.68	20.34	1479.82	0.17	73.18
UK100_11-B	1618.94	1643.71	1.53	24.04	1630.02	0.68	71.22
UK100_12-B	1362.14	1384.53	1.64	20.94	1366.52	0.32	91.03
UK100_13-B	1605.99	1616.49	0.65	22.65	1606.47	0.03	68.06
UK100_14-B	1690.25	1694.06	0.23	22.20	1690.25	0.00	76.47
UK100_15-B	1734.92	1739.10	0.24	24.14	1735.18	0.01	56.36
UK100_16-B	1381.03	1395.53	1.05	20.81	1386.92	0.43	78.23
UK100_17-B	1678.04	1688.08	0.60	23.75	1678.78	0.04	118.82
UK100_18-B	1495.60	1519.16	1.58	20.84	1508.96	0.89	128.55
UK100_19-B	1400.28	1412.98	0.91	20.38	1401.65	0.10	86.72
UK100_20-B	1628.89	1632.23	0.21	19.85	1629.15	0.02	47.20
Average	-	1548.81	0.84	21.87	1540.16	0.26	91.98

TABLE A4 Comparing algorithms performance for solving the mono-objective PRP – Set C instances

		KSVC15-basic			KSVC15		
Instance	**BKS**	**Cost**	**Gap(%)**	***T*(s)**	**Cost**	**Gap(%)**	***T*(s)**
UK100_01-C	1486.34	1504.03	1.19	19.35	1492.63	0.42	49.58
UK100_02-C	1431.55	1435.73	0.29	18.48	1432.84	0.09	56.86
UK100_03-C	1322.73	1333.32	0.80	19.08	1326.02	0.25	65.95
UK100_04-C	1377.73	1394.99	1.25	18.56	1382.50	0.35	80.26
UK100_05-C	1306.04	1326.00	1.53	17.47	1310.80	0.36	55.27
UK100_06-C	1485.99	1496.72	0.72	17.94	1487.86	0.13	46.01
UK100_07-C	1331.67	1342.76	0.83	17.32	1338.78	0.53	74.42
UK100_08-C	1373.20	1392.97	1.44	16.93	1380.89	0.56	62.02
UK100_09-C	1270.84	1285.12	1.12	17.27	1275.60	0.37	52.63
UK100_10-C	1329.95	1344.91	1.12	16.86	1335.22	0.40	56.29
UK100_11-C	1499.15	1519.03	1.33	19.77	1500.60	0.10	44.47
UK100_12-C	1233.28	1243.68	0.84	16.13	1237.53	0.34	52.03
UK100_13-C	1442.65	1451.23	0.59	18.10	1443.21	0.04	56.31
UK100_14-C	1552.85	1560.44	0.49	18.30	1555.29	0.16	50.09
UK100_15-C	1625.66	1640.57	0.92	19.13	1627.87	0.14	64.80
UK100_16-C	1216.84	1233.15	1.34	16.86	1219.92	0.25	47.35
UK100_17-C	1553.50	1581.46	1.80	20.57	1563.23	0.63	109.41
UK100_18-C	1321.19	1348.61	2.08	18.58	1336.22	1.14	72.65
UK100_19-C	1272.96	1285.33	0.97	19.34	1275.42	0.19	83.30
UK100_20-C	1540.99	1551.29	0.67	17.67	1540.99	0.00	61.09
Average		1413.57	1.07	18.19	1403.17	0.32	62.04