A metaheuristic approach to solve Dynamic Vehicle Routing Problem in continuous search space

Michał Okulewicz[a,*], Jacek Mańdziuk[a]

[a] *Warsaw University of Technology*
Faculty of Mathematics and Information Science
Koszykowa 75, 00-662 Warsaw POLAND

Abstract

This paper investigates a hypothesis that in case of hard optimization problems, selection of the proper search space is more relevant than the choice of the solving algorithm. Dynamic Vehicle Routing Problem is used as as a test problem and the hypothesis is verified experimentally on the well-known set of benchmark instances. The paper compares Particle Swarm Optimization (PSO) and Differential Evolution (DE) operating in two continuous search spaces (giving in total four distinctive approaches) and a state-of-the art discrete encoding utilizing Genetic Algorithm (GA).

The advantage of selected continuous search space over a discrete one is verified on the basis of quality of the final solution and stability of intermediate partial solutions. During stability analysis it has been observed that practical level of uncertainty, resulting from the dynamic nature of the problem, is lower than could be expected from a popular *degree of dynamism* measure. In order to challenge that issue, benchmark problems have been solved also with a higher level of dynamism and an *empirical degree of dynamism* has been proposed as an additional measure of the amount of uncertainty of the problem.

The results obtained by both continuous algorithms outperform those of state-of-the-art algorithms utilizing discrete problem representation, while the performance differences between them (PSO and DE) are minute.

Apart from higher numerical efficiency in solving DVRP, the use of continuous problem encoding proved its advantage over discrete encoding also in terms of intermediate solutions stability. Requests-to-vehicles assignment sequences generated by the proposed approach were approximately 40% more accurate with respect to the final solution than their counterparts generated with a discrete encoding. Analysis of these sequences of intermediate solutions for several benchmark sets resulted in designing a penalty term taking into account a specific dynamic nature of the optimized problem. An addition of this penalty term improved the above-mentioned stability of partial solutions by another 10%.

*Corresponding author
Email addresses: M.Okulewicz@mini.pw.edu.pl (Michał Okulewicz), J.Mandziuk@mini.pw.edu.pl (Jacek Mańdziuk)

Preprint submitted to Swarm and Evolutionary Computation, DOI: 10.1016/j.swevo.2019.03.008 March 19, 2019

Keywords: Dynamic Vehicle Routing Problem, dynamic optimization, continuous search space, Vehicle Routing Problem, Particle Swarm Optimization, Differential Evolution

1. Introduction

Vehicle Routing Problem (VRP) has been introduced by Dantzig and Ramser in 1959 [1] as a mathematical model for transport management. Its dynamic version (DVRP) was proposed by Psaraftis in 1988 [2]. DVRP gained wide recognition as a suitable platform for studyimg application of metaheuristic optimization methods after presentation of the set of benchmark problems by Kilby et al. in 1998 [3] and subsequent publication of GRASP and Ant Colony System (ACS) algorithms by Montemanni et al. in 2005 [4]. Since then, one of the most prominent lines of DVRP research relied on development of Computational Intelligence (CI) metaheuristic algorithms, including Ant Colony Optimization (e.g. [5, 6]), genetic and evolutionary approaches (e.g. [7, 8, 9, 10]) or Particle Swarm Optimization (PSO)-based solutions [11, 12]. Other recent approaches to DVRP include, among others, market-based optimization [13], look-ahead approaches [14] or generic dynamic optimization methods (e.g. [15, 16, 17]). A detailed overview of recent CI-based approaches to various VRP formulations can be found in a dedicated survey paper [18].

DVRP, as an example of a discrete dynamic problem, offers a great opportunity for enhancing optimization metaheuristic methods with prediction based techniques. The problem is worth studying, as it combines complexity of solving NP-hard static VRP with an additional challenge of making on-line decisions against the unknown final state of the problem, i.e. with no knowledge about time, space and volume distributions of future requests. Studying dynamic problems of discrete nature may potentially help in making methodological improvement in existing metaheuristic approaches. To a certain extent prediction-based methods have been neglected in the field of discrete problems [19] with much more attention being paid to techniques of tracking changing optima while solving continuous dynamic problems [15]. It should also be noted that the ideas related to Worst Case Optimization [20] or Robust Optimization [21] have been utilized within metaheuristic approach only to a small degree [22, 23].

Another related research topic within the field of discrete dynamic problems is the impact of problem representation [19] on both the algorithm performance and the dynamic characteristics of the problem. While it is quite typical to analyze performance differences between algorithms [24], the impact of the utilized problem encoding on the results is somewhat neglected [25]. Moreover, since dynamic nature of an optimization problem being solved poses additional challenges for the design of the problem encoding, it should be worthwhile to observe how changes in the problem state actually affect the achieved solution [26, 27, 28].

In our previous work the 2-Phase Multi-Swarm Particle Swarm Optimization (2MPSO) algorithm for solving DVRP [11, 29] has been proposed and tested,

excelling the (then) state-of-the-art discrete algorithms. In this paper we introduce a generalized version of the 2MPSO and apply it to solve DVRP with **the main focus on discussing, in a general perspective, the consequences of using continuous encodings for discrete dynamic problems.** It is shown in the paper that while DVRP is a discrete optimization problem, using its graphical representation and solving it as a clustering problem on a continuous $\mathbb{R}^2$ plane proves to be beneficial over applying discrete encodings. Apart from the improvement of results obtained for widely-used Kilby et al.'s benchmark set [3], the motivation for using continuous representation is twofold:

- First of all, continuous encoding allows usage of a population-based continuous optimization algorithms of any kind, what makes the proposed approach truly general.
- Furthermore, solving requests-to-vehicles assignment as a continuous clustering problem naturally leads to inducing a division of the fleet operating area into subareas. This way the method mimics manual requests-to-vehicles assignment made by domain experts in their everyday professional routine.

This paper is intended as the continuation of the studies reported in [11] and introduces generalization of the aforementioned 2MPSO algorithm. The research concentrates on the impact of selecting a problem encoding, a metaheuristic optimization algorithm and an additional penalty term on the quality of results, while treating a solution initialization and helper heuristics as given (described in full detail in [11]). 2MPSO approach is in this work generalized mainly in the two following aspects:

- A continuous population-based optimization algorithm and the solution encoding utilized by that algorithm are considered to be the method's parameters, thus converting the 2MPSO into a general, *algorithm-free* continuous optimization approach.
- Both continuous and discrete encoding based optimization algorithms can be applied in the proposed generalized parallel optimization framework.

This new general optimization framework will be referred to as ***Parallel Services***, while any form of continuous DVRP encoding considered in this framework will be referred to as ***ContDVRP***.

1.1. Related work

The review of current taxonomy and solution methods for DVRP, although currently slightly outdated, was provided by Pillac et al. [30]. An updated survey including various aspects of DVRPs has been conducted by Psaraftis et al. [31] and concentrated on the advancements made during the 30 years prior to its publication in 2016. That survey discussed the areas of technology (available localization and communication systems, computational power), problem taxonomy and solution methods. A very recent survey by Mavrovouniotis et al.

[32] considers DVRP as one of many Dynamic Optimization Problems (DOP) and discusses various Swarm Intelligence approaches within the DOP domain. Finally, Marinakis et al. [33] present a review of applications of PSO across the whole VRP domain, including the DVRP variant considered in this paper.

1.1.1. Continuous search spaces for VRPs

Except for our previous papers which considered continuous search space for DVRP [34, 29, 11], other continuous encodings have been proposed for the related VRP models, most notably Stochastic VRP [35] and Capacitated VRP [36].

An approach described in [35] concentrates on presentation of the VRP as a giant TSP route problem [37] and focuses on changes in the PSO position update formula imposed by the chosen encoding. A distinctive feature of the continuous encoding presented in [36] is simultaneous coding of requests order and requests assignment. The order of requests is coded in the form of a rank vector, while assignment of requests is derived from vectors of cluster centers corresponding to the respective vehicles.

In this work we integrate the idea of simultaneous order and assignment encoding [36] with the flexibility of assignment stemming from a multicluster approach [29]. A detailed description of the integrated multicluster encoding is presented in section 4.

1.1.2. Other geometric approaches

Although we are not aware of any other continuous encodings for VRP, there are several approaches that take into account geometric properties of the problem. For instance, the method proposed in [38] selects vehicle routes from the set of convex route propositions (called *petals*), while algorithms presented in [39, 40] utilize Voronoi tessellation based on request locations.

1.1.3. State-of-the-art algorithms for DVRP

Previous state-of-the-art approaches to DVRP (before introduction of our 2MPSO continuous algorithm) included: (a) discrete and distributed PSO (called MEMSO) [12] and (b) Genetic Algorithm (GA) [7]. Since the latter was tested in [7] under time limit constraints on Pentium IV processor its efficacy cannot be directly compared with other methods using contemporary computing resources. For this reason we reimplemented the GA algorithm and tested it with the stopping criterion relying on the number of fitness function evaluations. In such a setting the method proved to be competitive to MEMSO (cf. Table 6). Therefore, GA has been chosen in this paper as a baseline discrete search space algorithm for solving DVRP. For a detailed comparison under both computation time limit and number of fitness function evaluation limit, between the original GA, MEMSO and our continuous 2MPSO method, please refer to [11].

1.1.4. Recent works on DVRP

This section briefly mentions recent attempts at solving DVRP with a particular focus on papers that utilized Kilby et al.'s benchmark instances [3], so as

to make them comparable with 2MPSO [11] and MEMSO [12] algorithms, which still hold together the largest number of best average results for this benchmark set.

Seemingly, the most prominent approaches are based on the Monarch Butterfly Optimization [41] and Genetic Algorithm [42], which respectively achieved the results only 4.6% and 5.3% worse on average then 2MPSO, with better average results in the case of 6 and 2 (out of 21) benchmark instances, resp. Still, the results cannot be compared directly, as the authors of both [41] and [42] had not normalized computational power with the state-of-the-art GA approach [7], yet used computational time as stopping criterion.

In terms of the average solution quality, 5.6% worse average results than 2MPSO were reported in [43], with better average result for 5 (out of 21) benchmarks, but with significantly smaller number of fitness function evaluations. Again, a direct comparison of those results with our approach is not possible, since in [43] an unknown amount of additional computation time is utilized during solution learning phase.

Less promising approaches were proposed in [44] and [45], utilizing respectively Variable Neighborhood Search and another type of GA implementation. Those results have been worse, on average, by 6.7% and 7.4%, resp. than those of 2MPSO. Additionally, Hybrid GA [46] and Enhanced Ant Colony Optimization [47] approaches were proposed, but with the average solution quality more than 10% worse compared to 2MPSO.

It is also worth to mention a GPU implementation of the GA method [48] which reported around 70 times speedup over original CPU implementation [7] together with some new best results. Unfortunately, the average results are not reported in [48] what hinders a thorough comparison with other methods.

1.2. Main contribution

The main contribution of this paper is fivefold:

- Proposition of a continuous optimization approach – ContDVRP. On a general note, ContDVRP can be parametrized with any type of continuous optimization method, eg. PSO or DE.
- Experimental verification of a high degree of independence of the quality of DVRP results from the utilized optimization method (PSO or DE) within the ContDVRP framework (section 6.2).
- Optimization of obtained solutions by means of handling dynamic features of a DVRP instance in the form of a specifically designed penalty term. This term estimates the total number of vehicles to be utilized during the whole working day (section 5.1.2)
- General discussion on the degree of dynamism of the popular and widely utilized Kilby et al.'s benchmark sets [3] and providing results for their more dynamic customization (section 6.5)

- Combining priorities and single-cluster encoding [36] with a multi-cluster encoding [11] and investigating its performance against a sole multi-cluster approach section 6.2).

The rest of the paper is organized as follows. Section 2 briefly presents PSO and DE algorithms utilized independently in the proposed ContDVRP framework. Section 3 discusses the properties of DVRP along with its main operational parameters. The next section introduces various types of continuous DVRP encodings and section 5 presents the ContDVRP algorithm and Parallel Services environment in which ContDVRP is utilized. The following section provides ContDVRP results on a set of benchmark instances and discusses their stability. The last section concludes the paper.

2. Population based optimization algorithms for solving DVRP

Population based approaches have proven to be successful in solving hard optimization problems such as Vehicle Routing Problem or Job Shop Scheduling Problem [24]. Two of the most popular and successful algorithms in that area, PSO and DE, are briefly discussed in the remainder of this section.

2.1. Particle Swarm Optimization

PSO is an iterative global optimization metaheuristic method proposed in [49] and further studied and developed by many other researchers, e.g., [50, 51, 52]. The underlying idea of the method consists in maintaining the swarm of particles moving in the search space. For each particle the set of neighboring particles which communicate their positions and function values to this particle is defined. Furthermore, each particle keeps track of its current position and velocity, as well as remembers its historically best (in terms of solution quality) visited location. More precisely, in each iteration t, each particle i updates its position x_t^i and velocity v_t^i according to the following formulas [53, 50]:

$$x_{t+1}^i = x_t^i + v_t^i. \tag{1}$$

$$v_{t+1}^i = u_{U[0;g]}^{(1)}(x_{best}^{neighbors_i} - x_t^i) + u_{U[0;l]}^{(2)}(x_{best}^i - x_t^i) + a \cdot v_t^i \tag{2}$$

where g is a neighborhood attraction factor, $x_{best}^{neighbors_i}$ represents the best position (in terms of optimization) found hitherto by the particles belonging to the neighborhood of the ith particle, l is a local attraction factor, x_{best}^i represents the best position (in terms of optimization) found hitherto by particle i, a is an inertia coefficient, $u_{U[0;g]}^{(1)}$, $u_{U[0;l]}^{(2)}$ are random vectors with uniform distribution from the intervals $[0, g]$ and $[0, l]$, respectively.

In this study we use the Standard Particle Swarm Optimization 2007 (SPSO-07) [53] with random star neighborhood topology, in which, for each particle,

we randomly assign its neighbors, each of them independently, with a given probability[1].

2.2. Differential Evolution

DE is an iterative global optimization algorithm introduced in [54]. In DE, the population is moving in the search space of the objective function by means of testing new locations for each specimen created by cross-over operation. More precisely, we use a standard DE/rand/1/bin configuration in which in each iteration t and for each specimen x_t^i in the population a random specimen $x_t^{(3)}$ is chosen and mutated with a difference vector between random specimens $x_t^{(1)}$ and $x_t^{(2)}$ scaled by $F \in \mathbb{R}$:

$$y_t^{(3)} = x_t^{(3)} + F \times (x_t^{(2)} - x_t^{(1)}) \tag{3}$$

All randomly selected specimens $x_t^{(1)}$, $x_t^{(2)}$, $x_t^{(3)}$ are different from each other and from x_t^i. Subsequently, the mutant $y_t^{(3)}$ is crossed-over with x_t^i by binomial recombination with probability p:

$$y_t^i = Bin_p(x_t^i, y_t^{(3)}) \tag{4}$$

Finally, the new location y_t^i replaces the original x_t^i location *iff* it provides a better solution in terms of the objective function f:

$$x_{t+1}^i = \begin{cases} y_t^i & \text{if } f(y_t^i) < f(x_t^i) \\ x_t^i & \text{otherwise} \end{cases} \tag{5}$$

3. Dynamic Vehicle Routing Problem

Vehicle Routing Problem (VRP), a static version of the problem discussed in this paper was introduced in [1] as a problem of finding a set of routes for a fleet of gasoline delivery trucks, thus generalizing the Traveling Salesman Problem (TSP). Both TSP and VRP are NP-hard problems [55].

Dynamic VRP (DVRP) considered in this paper, unlike its static version, deals with a subset of requests to be served by the vehicles, which is not fully known *a priori*. Some requests are revealed when the optimization process is already in operation [30]. In this section we formally define the DVRP and look at this problem from the operations research perspective.

3.1. Problem formulation

In DVRP one considers:

- a fleet V of n vehicles,

[1] Please, note that the "neighboring" relation is not symmetrical, i.e. the fact that particle y is a neighbor of particle x, does not imply that x is a neighbor of y.

- a series C of m clients (requests) to be served, and
- a depot from which vehicles start their routes.

The fleet V is homogeneous, i.e. vehicles have identical capacity $cap \in \mathbb{R}$ and the same $speed \in \mathbb{R}$. The depot d is defined by:

- its location $l_0 \in \mathbb{R}^2$ and
- working hours (t_{start}, t_{end}), where $0 \leq t_{start} < t_{end}$.

Each client $c_l \in C$ $(l = 1, \ldots, m)$ is defined by the following set of attributes:

- location $l_l \in \mathbb{R}^2$,
- time $t_l \in \mathbb{R}$, which is a point in time when their request becomes available $(t_{start} \leq t_l \leq t_{end})$,
- unload time $u_l \in \mathbb{R}$, which is the time required to unload the cargo at the client's site,
- size $s_l \in \mathbb{R}$, which is the requests size $(s_l \leq cap)$.

A travel distance $\rho(i, j)$ is the Euclidean distance between l_i and l_j in $\mathbb{R}^2$, $i, j = 0, 1, \ldots, m$.

During the solving process, for each vehicle v_i, $r_i = (r_{i,1}, r_{i,2}, \ldots, r_{i,m_i})$ is a sequence of m_i indices of requests assigned to this vehicle, with depot being the first and the last elements of r_i. Therefore, r_i defines the route of the ith vehicle beginning and ending in the depot. The $arv_{r_{i,j}}$ is the time of arrival to the jth location assigned to the ith vehicle. $arv_{r_{i,j}}$ is induced by the permutation r_i, by the time when requests become available - see eqs. (7) and (8), and by the time $arv_{r_{i,j-1}}$ at which vehicle leaves the previous location.

The optimization goal is to serve all clients with their requested demands, with minimal total cost (travel distance) within the time constraints imposed by the working hours of the depots. In other words, the goal is to find such a set $R = \{r_1^*, r_2^*, \ldots, r_n^*\}$ of vehicles' routes that minimizes the following cost function:

$$COST(r_1, r_2, \ldots, r_n) = \sum_{i=1}^{n} \sum_{j=2}^{m_i} \rho(r_{i,j-1}, r_{i,j}) \tag{6}$$

under the following constraints (7) - (12).

Vehicle $v_i, i = 1, 2, \ldots, n$ cannot arrive at location $l_{r_{i,j}}$ until the time required for traveling from the last visited location $l_{r_{i,j-1}}$ (after receiving an information about the new request) is completed:

$$\forall_{i \in \{1,2,\ldots n\}} \forall_{j \in \{2,3,\ldots m_i\}} \quad arv_{r_{i,j}} \geq t_{r_{i,j}} + \rho(r_{i,j-1}, r_{i,j}) \tag{7}$$

Please recall that for $j = 2$, $l_{r_{i,j-1}}$ denotes the location of the initial depot.

A vehicle cannot arrive at location $l_{r_{i,j}}$ before serving the request $c_{r_{i,j-1}}$ and traveling to the next location:

$$\begin{aligned} &\forall_{i\in\{1,2,\ldots n\}}\forall_{j\in\{2,3\ldots m_i\}} \quad arv_{r_{i,j}} \\ &\geq arv_{r_{i,j-1}} + u_{r_{i,j-1}} + \rho(r_{i,j-1}, r_{i,j}) \end{aligned} \tag{8}$$

Each of the vehicles must not leave the depot before its opening and must return to the depot before its closing:

$$\forall_{i\in\{1,2,\ldots n\}} \quad arv_{r_{i,1}} \geq t_{start_{r_{i,1}}} \tag{9}$$

$$\forall_{i\in\{1,2,\ldots n\}} \quad arv_{r_{i,m_i}} \leq t_{end_{r_{i,m_i}}} \tag{10}$$

A sum of requests' sizes between consecutive visits to the depots must not exceed vehicle's capacity:

$$\begin{aligned} &\forall_{i\in\{1,2,\ldots n\}}\forall_{j_1<j_2\in\{1,2\ldots m_i\}} \quad (r_{i,j_1} \text{ and } \mathrm{r_{i,j_2}} \text{ are two} \\ &\text{subsequent depots in route } r_i) \quad \Rightarrow \quad (\sum_{j=j_1+1}^{j_2-1} s_{r_{i,j}} \leq cap) \end{aligned} \tag{11}$$

Each client must be assigned to exactly one vehicle:

$$\forall_{j\in\{1,\ldots m\}}\exists!_{i\in\{1,2,\ldots n\}} \quad j \in r_i \tag{12}$$

Note 1. *Without loss of generality, in all benchmarks used in this paper speed is defined as one distance unit per one time unit.*

3.2. Dynamic features

According to [15], a truly dynamic optimization problem is the one in which decisions regarding the optimized solution need to be made during the optimization process. These decisions have an impact on the possible future states of the problem and values of the cost function. The DVRP is not only a dynamic optimization problem in the above-described sense, but has also two distinctive additional features:

- the number of requests available for optimization is decremented with each ultimate commitment of any vehicle to any request,
- the number of requests available for optimization may grow until a certain time threshold - the so-called *cut-off time* - discussed below is reached.

The fact that some requests became unavailable for optimization does not usually need to be addressed in any other way than blocking any changes concerning them in the candidate solution. However, the possibility of additional requests appearance makes the DVRP an interesting and particularly challenging example of a dynamic optimization problem.

3.3. Measuring a degree of problem dynamism

In order to compare various approaches and choose suitable optimization techniques for a dynamic problem it is useful to observe and measure its degree of dynamism. In practice, the most critical operation performed on an intermediate DVRP solution R is reassignment of requests between vehicles, which leads to structural changes in the currently planned routes. Therefore, optimization processes which generate fewer such changes during the working day (with respect to subsequently developed solutions) are generally preferable in practice and considered more stable and robust.

For a given DVRP instance its *degree of dynamism* is measured according to the following definition.

Definition 1. ***Degree of dynamism (DoD)*** *[56] of a given DVRP instance is the ratio of the number of unknown requests m_u at the beginning of the solution process to the total number of requests m of that instance.*

$$dod = \frac{m_u(t_{start})}{m} \tag{13}$$

3.4. Operational parameters influencing the DoD

In a typical approach to solve DVRP, regardless of particular optimization method used, a vehicles' dispatcher (event scheduler) module is usually utilized, which is responsible for communication issues. In particular, the event scheduler collects information about new clients' requests, generates the current problem instance and sends it to the optimization module and, afterwards, uses the obtained solution to commit vehicles. A technical description of such an information technology system can be found, for instance, in [57].

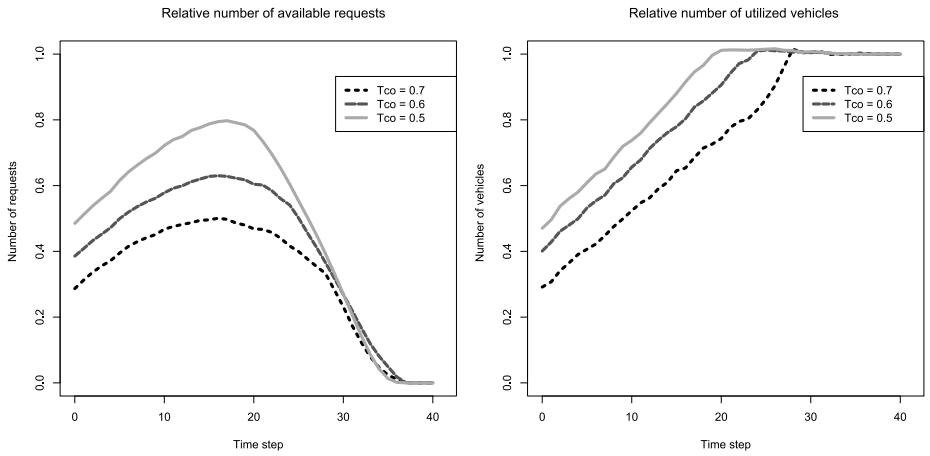
The event scheduler maintains the following three parameters:

- T_{co} - *cut-off time*,
- n_{ts} - number of *time slices*,
- T_{ac} - *advanced commitment* time,

which affect the *DoD* of a given problem instance and control the latest possible dispatch time of the vehicles.

3.4.1. Cut-off time

The *cut-off time* (T_{co}), in real business situations, can be interpreted as a time threshold for not accepting any new requests that arrive after T_{co} and treating them as the next-day's requests, available at the beginning of the next working day. In a one-day simulation horizon considered in this paper, likewise in the referenced works [58, 59, 12, 7, 34, 29, 11], the requests that arrive after the T_{co} are treated as being known at the beginning of the *current* day, i.e. they actually compose the initial problem instance. In benchmark sets the *cut-off time* value is typically set at half of the working day [3], i.e. $T_{co} = 0.5$.



(a) Dynamism of available requests - their variation in time.

(b) Dynamism of utilized vehicles - their variation in time.

Figure 1: Dynamism of two main DVRP parameters with respect to various *cut-off time* values ($T_{CO} = \{0.5, 0.6, 0.7\}$).

Figure 1 presents the influence of the *cut-off time* on the number of available requests and required vehicles in a sample algorithm run. Please observe the peak of the available requests' number curve in Figure 1a. For a standard setting of $T_{co} = 0.5$, almost 80% of requests are available for rescheduling at that time. This observation inspired us to devise yet another metric of dynamism (discussed in detail in Section 6.4) and perform experiments for higher values of T_{co} (presented in Section 6.5).

3.4.2. Number of time slices

The number of *time slices* (n_{ts}) decides how often the dispatcher sends a new version of the problem to the optimization module. The initial research presented in [3] suggested setting this value to 50, while [60] proposed division into 25 time slices (which was adopted as a standard value in subsequent approaches), claiming the optimal trade-off between the quality of solutions and computation time. Parameter tuning for 2MPSO algorithm [11] resulted in choosing $n_{ts} = 40$. Generally speaking, dividing the day into greater number of time slices allows optimization module to react faster to the newly-arrived requests since it is informed sooner about the introduced changes. On the other hand, with the *fitness function evaluation* (FFE) budget fixed, the chances for optimizing the solution within each time slice decrease proportionally.

3.4.3. Advanced commitment time

The *advanced commitment* time (T_{ac}) influences the set of *vehicles-to-be-dispatched*. Formally, a set of vehicles that need to be dispatched in the closest

time slice (denoted V_{tbd}) is defined in the following way:

$$V_{tbd} = \left\{ v_i : arv_{r_{i,m_i}} \geq t_{end_{r_{i,m_i}}} - (T_{ac} + \frac{1}{n_{ts}}) \left(t_{end_{r_{i,m_i}}} - t_{start_{r_{i,1}}} \right) \right\} \quad (14)$$

Consequently, requests scheduled to be served by any vehicle from V_{tbd} are treated as ultimately approved and cannot be rescheduled to another vehicle. During the 2MPSO parameter tuning process [11] it has been observed that appropriate choice of T_{ac} allows greater flexibility in assigning requests to vehicles in the phase of a day just before the T_{co}, when appropriate handling of potential arrival of a big-size request is a critical issue.

3.4.4. Summary

Despite extensive research efforts, the problem of varying in time instance characteristics (cf. Fig. 1) is still a challenging issue and, to the best of our knowledge, not handled efficiently by any of the existing approaches. Some recent studies in this area include the look-ahead approach [14] and the use of Monte Carlo simulations [61].

4. VRP in continuous search space

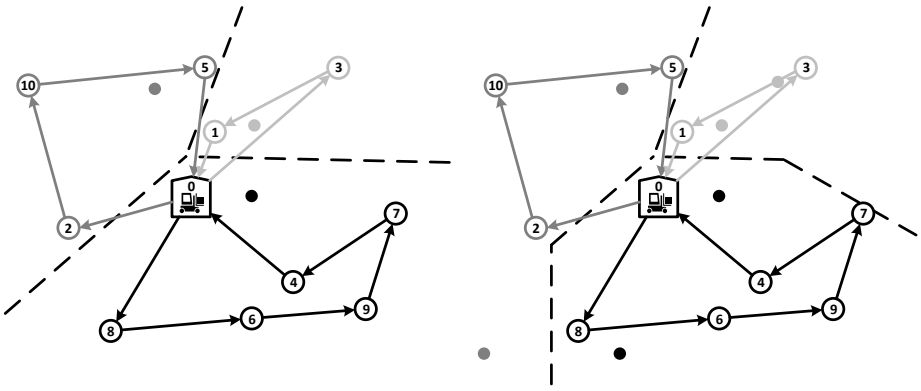
After introduction to the continuous global optimization algorithms presented in section 2 and description of a general characteristics of the DVRP provided in section 3, this section addresses the issue of handling VRP (and DVRP in particular) by a continuous optimization algorithm. Please note that while the research presented in this paper is focused on dynamic version of the VRP, proposed problem encodings are universal across the VRP domain and applicable to vast majority of possible VRP variants discussed in the literature.

4.1. Continuous VRP encodings

The following three types of continuous encodings are utilized in the VRP literature: *joint requests priorities and cluster centers* - originally proposed in the context of the Capacited VRP (CVRP) and VRP with Time Windows (VRPTW) [36, 62], *giant TSP tours (requests priorities)* - introduced in [35] for solving Stochastic VRP (SVRP), and *separate multi-cluster centers and requests priorities* - proposed in our previous research regarding DVRP [29, 11].

4.1.1. Priorities and clusters centers

Each candidate solution in the search space proposed in [36] is in the form of a vector of m request priorities and a vector of request cluster centers (2D coordinates) assigned to each of the estimated number of vehicles ($\hat{n}$). Therefore, candidate solutions belong to $\mathbb{R}^{m+2\hat{n}}$ search space. This type of candidate solution is unambiguously transformed into a discrete VRP solution R. Requests belonging to the same cluster are assigned to a common vehicle and the route is formed by ordering them according to their priorities. Please note that there exist route sets R which cannot be represented in this search space.



(a) A rough division of the working area imposed by one cluster of requests per vehicle [36].

(b) A fine-grained division of the working area enabled by using two clusters of requests per vehicle [11].

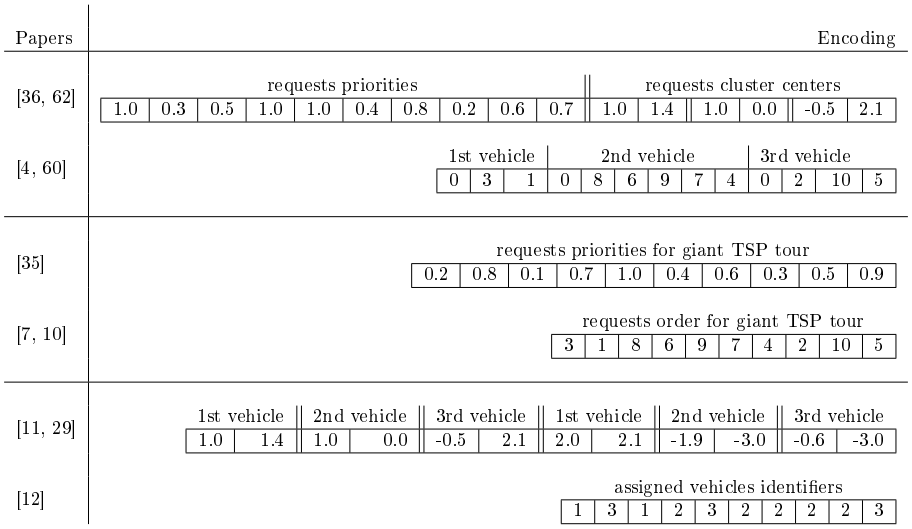


Figure 2: Three types of continuous VRP encodings and their discrete counterparts illustrated by a common example composed of 3 vehicles and 10 customers $\{1, \ldots, 10\}$. Location indicated by 0 represents the depot. Black and grey dots denote coordinates of the request cluster centers, and therefore generate the Voronoi diagram (denoted by dashed lines) of the service area. The resultant Voronoi cells generate the requests-to-vehicles assignment.

4.1.2. Priorities only

A candidate solution in the search space proposed in [35] is represented as a vector of m requests' priorities. Therefore, candidate solutions belong to $\mathbb{R}^m$ search space. Such a vector encodes the order of requests in the form of a giant TSP tour by sorting the vector indices according to the values of the respective elements. In order to create a VRP solution R from a giant tour representation, consistent parts of that tour are divided among the vehicles in a way that

complies with the time and capacity constraints. A discrete counterpart of such an encoding has been initially proposed in [7] and, more recently, utilized in [10].

4.1.3. Multicluster centers

In order to overcome the limitation of representing various solutions by means of a single cluster-per-vehicle encoding in the request-to-vehicles assignment [36] discussed in point 4.1.1, a multicluster approach has been proposed in our previous works [11, 29]. In the search space, a candidate solution consist of $k > 1$ request cluster centers per each of the $\hat{n}$ estimated vehicles. Therefore, candidate solutions belong to $\mathbb{R}^{2k\hat{n}}$ search space. In order to construct a VRP solution, the route for each vehicle is generated at random and improved with the 2–OPT algorithm [63]. A discrete counterpart to a multicluster requests-to-vehicles assignment was explored in [12].

4.1.4. Complexity of solutions decoding

A discussion of various continuous encodings must also take into account computational complexity of decoding representation from a given search space into the actual problem solution. While direct discrete encodings bear a decoding cost of $O(m)$, for continuous priorities and multiclusters it could be as high as $O(m \log m + mk\hat{n})$. Nevertheless, even with a larger computational cost of solution decoding, the continuous approach proved beneficial, even in time bounded experiments performed in [11]: "*In the timebounded experiment 2MPSO outperforms the average length of the GA's [7] routes by 7.1% and ACOLNS's [5] by 10.4%*". This phenomenon can possibly be attributed to the fact that having a request clustering heuristic embedded in the optimization process creates an implicit upper bound on the possible values of a solution cost. Meanwhile, algorithms utilizing discrete encodings might suffer from the risk of sampling solutions of low quality, due to lesser restrictions imposed on possible problem solutions to be encoded.

4.1.5. Summary of encoding schemes

Figure 2 presents an example of a VRP solution composed of a set of requests $\{1, \ldots, 10\}$ served by 3 vehicles stationed in a centrally located depot (denoted by 0). The figure graphically illustrates the solution (top part of the figure) and its possible encodings in the three above-discussed cases (in each of them both continuous and discrete solution representations are listed). Please observe the increasing flexibility in subareas definition along with the increasing number of request clusters per vehicle.

4.1.6. Final remarks

On a general note, the main difference between application of continuous and discrete encodings to discrete problems lies in *different focus* of these methods. Continuous approaches mainly concentrate on deriving a **suitable model of the optimization problem**, while discrete approaches rely more on proposing **efficient search operators**. Since our aim in this study is the application of

generic continuous optimization metaheuristics to solving DVRP, we shall focus on the problem encodings proposed in [36] and [11, 29].

5. The *Parallel Services* approach and the *ContDVRP* algorithm

This section presents a brief technical summary of the implemented optimization system - Parallel Services (PS). This system is a general optimization framework, while any type of continuous DVRP encoding within PS is referred to as ***ContDVRP***. A .NET Framework implementation of the Parallel Services, together with ContDVRP and GA code is available at `https://sourceforge.net/projects/continuous-dvrp/`

Figure 4 presents the overall view of the generic optimization process within the PS environment. Particular activities within a single optimization process are depicted in Figure 5. The main activities in PS are described in Figure 3

1. Load the initial state of the DVRP
2. Spawn as many parallel processes as there are available processing units (VCPUs)[2]
3. **For each** of the spawned processes:
   (a) Estimate the number of required vehicles
   (b) Create the initial population for the optimization algorithm (incorporate heuristic solution, historic solution and random solutions)
   (c) Perform requests-to-vehicles assignment optimization[3] with one of the following algorithms *(the same in all parallel processes)*:
   - PSO with multiclusters encoding *(= previous 2MPSO algorithm)*
   - DE with multiclusters encoding
   - PSO with priorities and multiclusters encoding
   - DE with priorities and multiclusters encoding
   - GA with discrete giant TSP route encoding
4. Select the best solution
5. Assign requests
6. Update problem state with new requests
7. Go back to step 3 if there are new requests available or previous requests are not yet ultimately assigned

Figure 3: Overall activities of the Parallel Process during optimization of a DVRP instance.

Observe that each of the parallel processes in PS is identically parametrized, **utilizes the same problem encoding and optimization algorithm.** The details of activities of a single optimization processes in a single time step are presented in Figure 5.

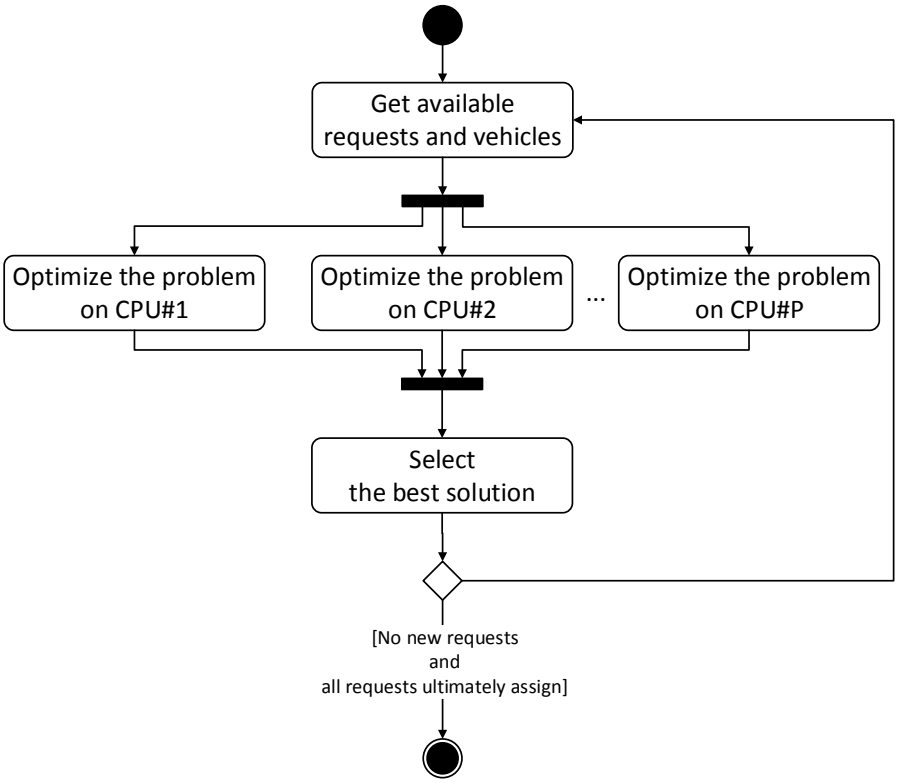


Figure 4: An activity diagram presenting a high level view of the DVRP optimization process within the Parallel Services framework.

5.1. Key features of ContDVRP

The two key aspects of the ContDVRP algorithm are efficient transfer of knowledge (partial solutions) between subsequent time slices and a method of handling unknown clients' requests which are likely to appear in future time slices. Both these issues are discussed in the reminder of this section, respectively in points 5.1.1 and 5.1.2.

5.1.1. Solution transfer between problem states (time slices)

Already in the initial works on DVRP [60, 4, 7] some form of passing best solutions from previous time slices to the current one was used. While such a policy seems to be rather obvious and natural, what comes as a surprise is the lack of more sophisticated methods which take into account particular dynamic features of DVRP.

Discrete GA [7] and MEMSO [12] approaches to solving DVRP **migrate a solution from the previous time step and adapt it to a new state of the problem.** ContDVRP, on the contrary, mainly follows the path of Ant Colony System (ACS), where **the rules for creating the solution are**

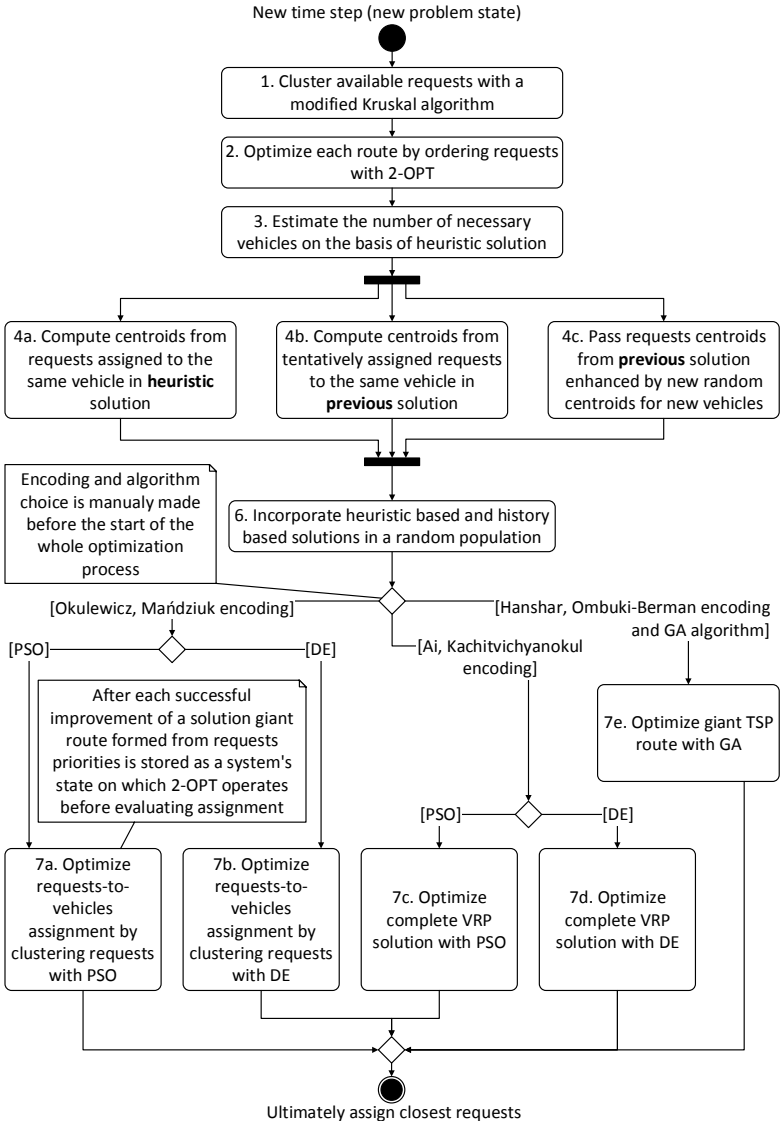
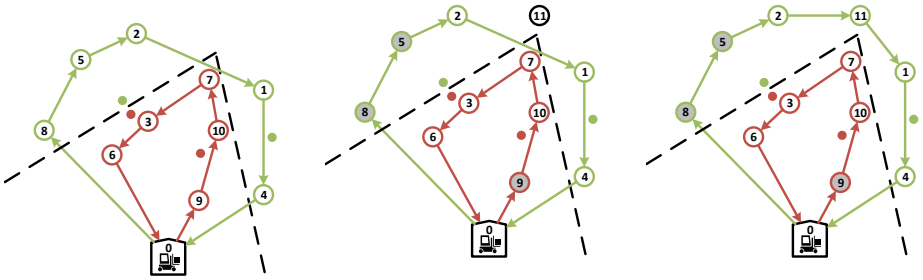


Figure 5: An activity diagram presenting a simplified view of the possible execution paths of a single optimization service process within a single time step within the PS optimization framework.

migrated and adapted. In ACS [60, 4] this idea is implemented by means of migrating and adapting pheromone levels of the edges. In ContDVRP [11] it consists in transferring the centers of request clusters assigned to vehicles during the optimization process. Continuous search space expands as necessary while the number of required vehicles increases during the course of the working day. Cluster centers representing newly-added vehicles are positioned randomly and



(a) A solution found in the ith time slice.

(b) An ultimate assignment of requests number "5", "8" and "9" and appearance of a new pending request ("11").

(c) Routes generated at the beginning of the $i+1$ time slice based on the solution transferred directly from the ith slice.

Figure 6: A high quality solution found immediately at the beginning of the next time slice **thanks to application of a *direct passing* knowledge transfer**. By preserving the request cluster centers found in previous time slices it is possible to accelerate the optimization process, as it does not have to search from scratch for complex, often non-linearly separable partial solutions. Dots denote the request cluster centers utilized as a continuous encoding of the problem, imposing the requests-to-vehicles assignment by means of Voronoi tessellation of the vehicles' operation area.

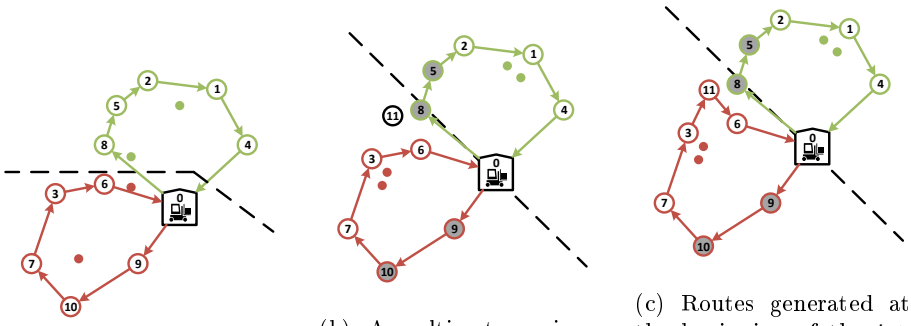
their location is concatenated with the previous solution vector. The number of required vehicles is estimated by the number of clusters obtained by solving the capacitated clustering problem over the set of known requests with a fast heuristic algorithm [61, 11].

ContDVRP utilizes two types of cluster centers transfers: direct passing and the approximation retrieval. Therefore, two distinct solutions, being the output of each of those procedures, are passed to the initial population of the subsequent step.

Direct passing creates the initial solution for a new state of the problem directly from the best **continuous solution** obtained in the previous state of the problem, supplemented with additional vehicles if necessary.

Approximation retrieval creates a continuous representation on the basis of a **discrete solution**. For each cluster center assigned to a given vehicle, the average coordinates of all pending requests assigned to this vehicle are computed and disturbed by adding a small random variable. These disturbed cluster centers attempt to reflect expected future locations of the vehicles.

Both methods have certain advantages, which make them useful as complementary approaches. *Direct passing* preserves the linearly non-separable request clusters (cf. Figure 6) while *approximation retrieval* pushes the algorithm out of the local optima, allowing for a better accommodation to future requests (cf. Figure 7.). Our previous studies [11] have shown, that while both methods are



(a) A solution found in the ith time slice.

(b) An ultimate assignment of requests number "5", "8", "9" and "10". Computation of the cluster centers of pending requests and appearance of a new request (number "11").

(c) Routes generated at the beginning of the $i+1$ time slice by adaptation of the transferred solution, which takes into account partial routes already committed to vehicles.

Figure 7: A high quality solution found immediately at the beginning of the next time slice **thanks to application of an *approximation retrieval* knowledge transfer**. Cluster centers generated by the approximation retrieval method are located in the average coordinates of all pending requests assigned to a given vehicle. Therefore, a new request number "11" is assigned to a vehicle that will be operating in the proximity of its location, instead of the one that has already served requests "8" and "5" and would have to return to serve request "11" if assigned to. Dots denote the requests cluster centers utilized as the continuous encoding of the problem, imposing the request-to-vehicles assignment by means of Voronoi tessellation of the vehicles' operation area.

beneficial for the cost of obtained solutions, direct passing has a generally higher impact on the solutions quality.

5.1.2. Taking into account unknown future requests

In [61] an initial method of estimating the expected number of unknown requests and their possible locations was proposed. A simplified version of this approach is proposed in this paper. It has fewer underlying assumptions regarding the future distribution of requests. In consequence, it impacts the computational cost of the whole method only by a factor of $\frac{{T_{CO}}^2}{2}$.

The approach is inspired by a robust optimization methodology and results in adding a penalty term to the solution cost function (eq. (6)). The role of the designed penalty term is to encourage the optimization process to provide solutions utilizing more vehicles than are actually necessary for the set of currently known requests. That way, future requests can be easily accommodated into existing solutions (cf. Figure 8). The method estimates the total number of vehicles $\hat{n}_{t_{end}}$ required in the final solution. The estimate is based on the requests appearance frequency and their size:

- the average appearance frequency of new requests remains at the same

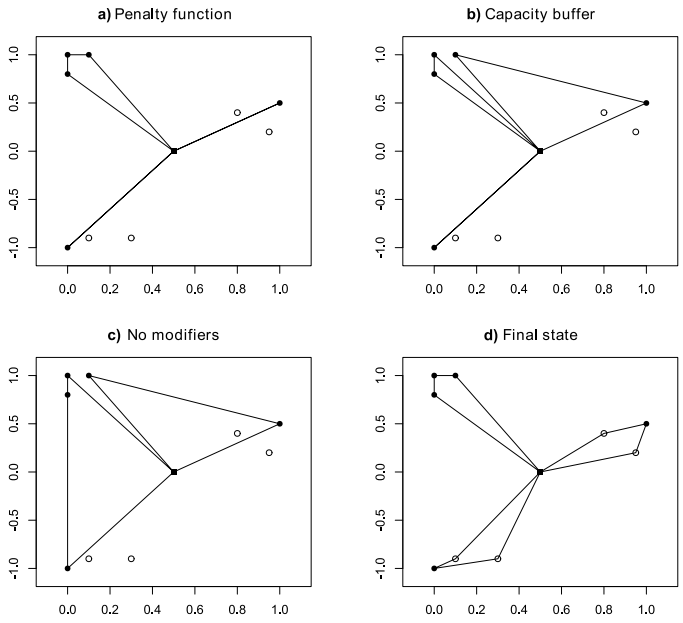


Figure 8: Comparison of partial solutions for the intermediate problem states (a)-(c) with the shape of the final solution when all requests are already known (d). Known requests are marked as discs and unknown ones as circles. The depot is marked in the square center. Subplots (a) - (c) present solutions obtained, respectively, when a penalty term is applied (a), when a capacity buffer or finish time buffer is applied to each of the vehicles (b), and when no changes are made to the problem or cost function (c).

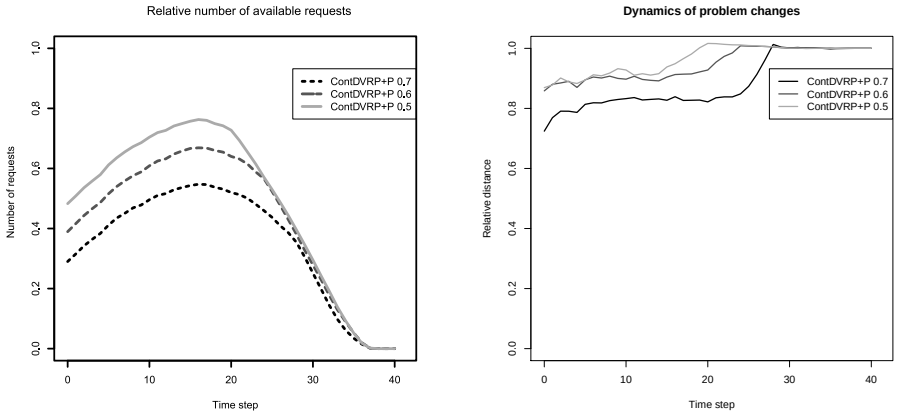
level during the entire optimization period,

- the requests' size distribution of already known requests properly estimates the final distribution (in the entire problem instance).

$$\hat{n}_{t_{end}}(t) = \left\lceil \frac{T_{CO}(t_{end} - t_{start}) + t_{start} - t}{T_{CO}(t_{end} - t_{start}) - t_{start} + t} * \sum_{i=1}^{m_t} \frac{s_i}{cap} \right\rceil \tag{15}$$

The estimate computed in (15) is (a) utilized by the heuristic clustering algorithm and (b) in the additional penalty term of the cost function. In the former case it is used as an additional stopping criterion in the modified Kruskal algorithm solving the capacitated clustering problem [11]. If the number of clusters reaches $\hat{n}_{t_{end}}(t)$ the clustering process is stopped. In the latter case, the penalty term (16) is added to the cost function (6) in time slice t, if the current number of used vehicles $\hat{n}_R(t)$ is smaller than that estimated for the final solution ($\hat{n}_{t_{end}}(t)$):

$$Penalty(R,t) = \left(\hat{n}_{t_{end}(t)} - \hat{n}_R\right) * \frac{Cost(R)}{\hat{n}_R} \tag{16}$$



(a) Dynamics of available requests. (b) Dynamics of utilized vehicles.

Figure 9: DVRP dynamics for various *cut-off time* settings ($T_{CO} = \{0.5, 0.6, 0.7\}$) after applying the penalty term.

where R is the currently best solution. The impact of using the penalty term on the problem dynamism and the obtained intermediate solutions is illustrated in Figures 8 and 9, respectively.

6. Results

This section presents experimental results of *ContDVRP* application to Kilby et al.'s benchmark instances [3]. The properties of continuous approach are assessed in three different experiments. The first one verifies the impact of selection of an optimization algorithm and a type of continuous problem encoding on the quality of obtained results (section 6.2). The second one is focused on the role of the penalty term in stabilization of intermediate solutions during the ContDVRP optimization process (section 6.3). The last one (section 6.5) provides new insights regarding various *cut-off time* settings, which is a consequence of our discovery that commonly used benchmark instances have relatively low empirical dynamism (*em.dod*), a measure introduced in section 6.4.

6.1. Experimental setup

All the experiments were conducted on 21 widely-known benchmark instances [3]. Values of the main steering parameters (common for all experiments) are presented in Table 1, and those of specific, experiment-related parameters are listed in Tables 3, 5 and 7, respectively. All values were either carefully tuned (see Appendix A in [11]) or taken from the source literature [7]. For each parameter setting, 30 algorithm runs were performed, each of them with a limited number of fitness function evaluations (FFEs), in order to assure comparability with the existing literature results.

Table 1: Values of the main parameters in the Parallel Services based experiments.

Parameter	Value
Parallel Processes	
#parallel optimization processes	8
n_{ts}	40
PSO	
g	0.60
l	2.20
a	0.63
$P(X$ is a neighbor of $Y)$	0.50
#iterations	140
#particles	22
DE	
c	0.9
F	0.5
#iterations	195
#specimen	16
GA	
P(mutate X)	0.15
tournament size	2
P(selecting lower quality solution)	0.2
elite size	2
#iterations	140
#specimen	22

In what follows, the ContDVRP variants are denoted by the type of optimization algorithm used (DE or PSO), the number of request clusters per vehicle ($k = 2$ in all cases) and the possible usage of the penalty term ($+P$, if applied). Whenever appropriate, the *cut-off time* value ($T_{CO} \in \{0.5, 0.6, 0.7\}$) and the type of encoding (*clusters only* or *clusters and priorities (ranks)*) will also be listed.

6.1.1. Results comparison

Results of various methods were compared based on the Student t-test, which validated statistical significance of their differences. In the case of comparison of literature results (external implementations) with our approach, a single sample t-test was applied, independently for each benchmark problem. A comparison between literature algorithms which relied on our own re-implementation with our approach was performed using two-sample t-test.

Table 2: A comprehensive comparison of state-of-the art CI approaches to DVRP. Statistical significance of the differences in the average results has been measured by t-test with Bonferroni correction. Overall minimum and best average results for each benchmark problem are resented in bold. Grey background denotes high quality results (either a significantly better performance of a literature algorithms (external implementation) compared to our best approach, or all the approaches with insignificant differences with our best approach, in the case where no external algorithm yielded a significantly better average result (cf. section 6.1.1)).

	MEMSO [12] 10^6 FFE GRID 5000 cluster		$ES-L$ [43] $2.25*10^4$ FFE *unknown*		$H-GA$ [46] $2.45*10^5$ FFE *unknown*		*GA* [7] 10^6 FFE Intel Core i7 3.4 GHz		$ContDVRP_{PSO}^{k=2}$ [11] 10^6 FFE Intel Core i7 3.4 GHz		$ContDVRP_{PSO}^{k=2}+P$ 10^6 FFE Intel Core i7 3.4 GHz	
	Min	Avg	Min	Avg	Min	Avg	Min	Avg	Min	Avg	Min	Avg
c50	577.60	592.95	595.43	614.44	618.36	654.41	574.71	600.56	**544.11**	**578.31**	551.34	580.56
c75	928.53	962.54	949.48	979.19	975.46	1038.53	**875.34**	938.20	884.43	903.72	886.42	**901.05**
c100b	864.19	878.81	879.39	891.19	882.29	950.68	844.69	919.17	**819.56**	845.80	**819.56**	**844.56**
c100	949.83	968.92	932.06	963.33	966.14	1062.97	924.42	965.65	902.00	933.46	**873.77**	**920.69**
c120	1164.63	1284.62	1283.15	1381.07	1271.06	1403.78	1208.36	1282.52	**1053.18**	**1071.38**	1056.70	1116.08
c150	1274.33	1327.24	1260.35	1306.26	1339.25	1514.65	1168.37	1228.99	1098.03	1134.20	**1097.27**	**1127.04**
c199	1600.57	1649.17	1545.96	1627.48	1778.55	1875.81	1476.06	1524.69	**1362.65**	**1408.70**	1374.47	1418.71
f71	283.43	294.85	273.28	278.26	295.72	335.40	286.07	307.87	274.16	298.50	**270.20**	**275.27**
f134	14814.10	16083.82	14811.60	15516.56	15079.61	16155.45	11936.42	13055.75	11746.40	11892.00	**11713.20**	**11810.04**
tai75a	1785.11	1837.00	1859.96	1887.35	1719.87	1815.51	1705.21	1787.79	**1685.23**	1805.03	1691.95	**1782.79**
tai75b	1398.68	1425.80	1376.14	1485.08	1444.63	1536.46	1365.21	1420.97	1365.36	1422.60	**1356.50**	**1401.38**
tai75c	1490.32	1532.45	1439.64	1494.18	1511.16	1644.64	1439.21	1532.23	1439.02	1510.00	**1424.91**	**1486.78**
tai75d	**1342.26**	1448.19	1396.55	**1423.80**	1448.06	1538.87	1428.75	1466.43	1408.79	1433.25	1403.85	1428.91
tai100a	2170.54	2213.75	2151.69	**2203.86**	2217.44	2368.73	2141.29	2230.84	**2137.30**	2216.23	2147.07	2226.48
tai100b	2093.54	2190.01	2088.07	2153.57	2177.04	2323.06	2110.11	2211.16	2060.65	2136.80	**2041.96**	**2125.19**
tai100c	1491.13	1553.55	1492.98	1570.38	1519.22	1609.33	1476.91	1537.92	1458.81	1494.72	**1446.98**	**1485.47**
tai100d	1732.38	1895.42	2002.09	2028.87	1806.18	1927.21	1680.29	1799.04	1663.87	1727.95	**1658.48**	**1718.18**
tai150a	**3253.77**	3369.48	3281.28	**3369.25**	3543.58	3665.33	3314.22	3471.67	3338.71	3530.82	3396.49	3526.10
tai150b	2865.17	2959.15	**2857.65**	**2880.87**	2991.52	3196.82	2919.36	3071.33	2910.06	3026.89	2931.16	3034.65
tai150c	2510.13	2644.69	2528.06	2703.04	2656.99	2843.49	2544.24	2729.26	**2497.65**	**2603.53**	2523.53	2642.82
tai150d	2872.80	3006.88	2880.68	2976.71	3035.93	3206.45	**2845.01**	**2973.14**	2869.79	3009.01	2929.91	3023.48

	GA [7] 750 seconds Intel Pentium IV 2.80 GHz		*ACOLNS* [5] 1500 seconds Intel Core i5 2.4 GHz		*MBO* [41] 750 seconds Intel Xeon 2.40 GHz		$E-ACO$ [47] *unknown* Intel Core i5 3.2 GHz		*VNS* [44] 125 seconds Core2 Quad 2.66 GHz		$ContDVRP_{PSO}^{k=2}$ [11] 75 seconds Intel Core i7 3.4 GHz	
	Min	Avg	Min	Avg	Min	Avg	Min	Avg	Min	Avg	Min	Avg
c50	570.89	593.42	601.78	623.09	590.73	**570.89**	607.21	647.21	592.60	618.44	**562.70**	581.46
c75	981.57	1013.45	1003.20	1013.47	909.56	1013.45	924.71	1045.44	940.25	994.89	**874.08**	**905.95**
c100b	881.92	900.94	932.35	943.05	839.93	**841.44**	869.22	950.17	883.91	910.67	**819.56**	844.90
c100	961.10	987.59	987.65	1012.30	930.38	987.59	973.40	1044.96	943.87	985.94	**882.96**	**930.95**
c120	1303.59	1390.58	1272.65	1451.60	1112.58	1153.29	1108.15	1197.68	1219.73	1319.39	**1066.15**	**1085.46**
c150	1348.88	1386.93	1370.33	1394.77	**1135.85**	1386.93	1378.63	1472.40	1253.66	1323.40	1147.50	**1195.95**
c199	1654.51	1758.51	1717.31	1757.02	**1414.40**	1758.51	1561.12	1836.86	1538.87	1608.40	1434.70	**1503.94**
f71	301.79	309.94	311.33	320.00	277.00	306.33	**259.71**	297.08	273.08	291.21	270.35	**290.62**
f134	15528.81	15986.84	15557.82	16030.53	11853.29	15528.81	20000.00	20000.00	14559.58	15423.87	**11773.74**	**12038.02**
tai75a	1782.91	1856.66	1832.84	1880.87	1799.15	**1782.91**	**1690.91**	1983.92	1833.18	1913.51	1767.64	1825.87
tai75b	1464.56	1527.77	1456.97	1477.15	1385.45	1452.26	1509.56	1647.78	1460.65	1496.77	**1366.80**	**1419.66**
tai75c	1440.54	1501.91	1612.10	1692.00	1483.41	**1441.91**	**1329.42**	1470.60	1558.05	1616.87	1427.76	1487.39
tai75d	**1399.83**	**1422.27**	1470.52	1491.84	1435.62	**1422.27**	1409.14	1661.73	1428.74	1452.73	1404.75	1442.45
tai100a	2232.71	2295.61	2257.05	2331.28	2212.08	**2232.71**	2281.70	2550.64	**2134.95**	2246.88	2196.91	2261.66
tai100b	2147.70	2215.39	2203.63	2317.30	2105.15	2182.61	2255.83	2500.72	2126.68	2204.05	**2060.46**	**2151.73**
tai100c	1541.28	1622.66	1660.48	1717.61	1474.32	1541.25	**1442.45**	1743.07	1517.57	1650.93	1476.24	**1512.13**
tai100d	1834.60	1912.43	1952.15	2087.96	1722.88	1912.43	**1581.36**	1843.82	1807.86	1950.62	1676.10	**1746.44**
tai150a	3328.85	3501.83	3436.40	3595.40	3539.61	**3185.73**	3307.63	3684.03	**3274.78**	3408.89	3476.48	3777.98
tai150b	2933.40	3115.39	3060.02	3095.61	3038.38	**2880.57**	3128.00	3439.38	**2819.46**	2902.42	2978.30	3120.09
tai150c	2612.68	2743.55	2735.39	2840.69	2641.62	2743.55	2583.36	2729.15	**2491.58**	**2643.11**	2532.23	2678.16
tai150d	2950.61	3045.16	3138.70	3233.39	3047.30	3045.16	**2808.99**	3186.08	2927.21	**3006.04**	2958.75	3141.63

6.2. Relevance of the optimization algorithm selection

First experiment tests the impact of the optimization algorithm and the type of a problem encoding on the quality of results. Test is performed for each of the five configurations of the optimization algorithms available within the PS environment (cf. Figure 5). Test runtime parameters are listed in Tables 1 and 3, while its results are presented in Table 4.

Please note, that the search space of the *clusters only* encoding is exactly the one discussed in section 4.1.3 and its dimension is equal to $2k\hat{n}$. Meanwhile, *clusters and priorities* search space is a combination of the search spaces discussed in sections 4.1.1 and 4.1.3, and as a result its dimension is equal to $m + 2k\hat{n}$.

Table 3: Settings of the ContDVRP realizations used in the experiment aimed at verification of the impact of an optimization algorithm selection.

Encoding	Algorithm	Stopping criterion	Penalty	k	T_{CO}	T_{AC}
Clusters	PSO	#FFE	No	2	0.5	0.04
Clusters and priorities	PSO	#FFE	No	2	0.5	0.04
Clusters	DE	#FFE	No	2	0.5	0.04
Clusters and priorities	DE	#FFE	No	2	0.5	0.04

On a general note, two main conclusions can be easily drawn from the results. First of all, each of the continuous realizations of the optimization algorithm outperformed a discrete GA-based approach which yielded only 1 best average result and was significantly worse in 16 problem instances. Second of all, the PSO-based realizations are generally better than their DE counterparts.

On a more detailed level, however, it can be observed that none of the four configurations of ContDVRP has a clear advantage over the remaining ones. The baseline version, $ContDVRP^{k=2}_{PSO} (= 2MPSO$ [11]) without requests priorities, gained 7 best average results, but at the same time yielded statistically significantly worse results than the best of the competitive approaches for 5 problem instances. The same metrics were equal to 1 and 5 for $ContDVRP^{k=2}_{DE}$ without requests priorities, 7 and 5 for $ContDVRP^{k=2}_{PSO}$ with requests priorities, and 5 and 3 for $ContDVRP^{k=2}_{DE}$ with requests priorities, respectively.

Finally, it is worth to note that DE-based realizations generally converged faster than the PSO-based ones, with the average speed advantage reaching 21% in the extreme case.

6.3. Impact of the penalty term

The second experiment aimed at testing the efficacy of using the penalty term (16) within the ContDVRP approach. Tables 1 and 5 provide parameter settings in this experiment and Table 6 compares the results of ContDVRP approach with and without penalty term with MEMSO and GA methods. The first conclusion from the presented results is the comparable performance of the two discrete approaches: GA and MEMSO. Furthermore, addition of the penalty term, although improves the average results of ContDVRP, does not

Table 4: The average results obtained by all versions of the algorithm tested in the PS framework. For each problem instance, the best result is bolded. Grey background denotes results which are not significantly worse than the best one, according to a one-sided t-test with $\alpha = 0.05$.

	Discrete	Continuous - clusters only		Continuous - clusters and priorities	
	GA	$ContDVRP_{PSO}^{k=2}$	$ContDVRP_{DE}^{k=2}$	$ContDVRP_{PSO}^{k=2}$	$ContDVRP_{DE}^{k=2}$
Name	Avg	Avg	Avg	Avg	Avg
c50	600.56	**578.31**	580.02	583.49	581.24
c75	938.2	903.72	908.28	**901.63**	904.11
c100	965.65	933.46	937.07	**932.76**	936.92
c100b	919.17	845.8	843.83	842.85	**826.82**
c120	1282.52	**1071.38**	1104.61	1082.02	1079.75
c150	1228.99	**1134.2**	1140.14	1145.89	1147.87
c199	1524.69	1408.7	1414.85	1412.28	**1407.8**
f71	307.87	298.5	295.58	292.92	**289.48**
f134	13055.75	**11892**	11916.7	11969.68	12006
tai75a	**1787.79**	1805.03	1807.49	1817.41	1812.93
tai75b	1420.97	1422.6	1412.85	**1411.7**	1419.86
tai75c	1532.23	1510	**1501.64**	1501.91	1523.06
tai75d	1466.43	**1433.25**	1451.04	1442.31	1444.14
tai100a	2230.84	**2216.23**	2247.21	2251.18	2229.71
tai100b	2211.16	2136.8	2145.85	**2126.98**	2157.35
tai100c	1537.92	1494.72	1495.87	**1494.31**	1502.06
tai100d	1799.04	**1727.95**	1736.33	1742.27	1729.63
tai150a	3471.67	3530.82	3489.65	3528.1	**3453.73**
tai150b	3071.33	3026.89	3038.63	**3014.45**	3023.5
tai150c	2729.26	2603.53	2563.01	2574.42	**2562.45**
tai150d	2973.14	3009.01	2999.43	**2955.65**	2987.38
sum	47055.18	**44982.9**	45030.08	45024.21	45025.79

have a statistically significant impact. On the other hand, plots of dynamic features presented in Figure 10 show that application of continuous encoding results in greater stability of intermediate solutions, and adding a penalty term enhances that property even further.

Additionally, Table 2 presents a detailed comparison of our method with selected external approaches discussed in section 1.1.4. For the sake of clarity, results are presented separately for the FFE bounded experiments and computation time bounded experiments. The set of comparative methods consists of the state-of-the-art algorithms utilizing various CI techniques, namely: MEMSO [12], Variable Neighbourhood Search (VNS) [44], Monarch Butterfly Optimization (MBO) [41], Enhanced Ant Colony Optimization (E-ACO) [47], Evolution Strategy with Learning Capabilities (ES-LC) [43], Hybrid Genetic Algorithm (H-GA) [46], and Ant Colony Optimization with Large Neighborhood Search (ACOLNS) [5]. In principle, incomparable stopping criteria reported in the literature (various running times and different numbers od FFEs) hinder drawing definite conclusions about the relative performance of discussed algorithms. At the same time, please observe that since computation times of our method in the experiments limited by number of FFEs was between 37 and 429 seconds for baseline ContDVRP, and between 53 and 602 seconds for ContDVRP with applied penalty, it is safe to conclude that our continuous approach

Table 5: Settings of the PS algorithms used in the experiment verifying the impact of the penalty term.

Encoding	Algorithm	Stopping criterion	Penalty	k	T_{CO}	T_{AC}
Clusters	PSO	#FFE	No	2	0.5	0.04
Clusters	PSO	#FFE	Yes	2	0.5	0.20
Giant TSP tour	GA	#FFE	N/A	N/A	0.5	0.15

Table 6: Comparison of *ContDVRP* and MEMSO [12] (one of the state-of-the-art DVRP algorithms with the constrained budget of FFEs) with our implementation of GA-based method [7]. For each algorithm the number of FFEs is presented in parenthesis and broken down into three components: the number of time slices, the number of optimizers (parallel solving instances) and the number of FFEs for each optimizer, in each time slice. For each problem instance, the best (minimum) and the average values are bolded. Grey background denotes average results which are not significantly worse than the best average result, according to a one-sided t-test with $\alpha = 0.05$.

	$MEMSO$ [12] $(25 * 8 * (0.5 * 10^4))$		GA [7] $(40 * 8 * (0.31 * 10^4))$		$ContDVRP^{k=2}_{PSO}$ $(40 * 8 * (0.31 * 10^4))$		$ContDVRP^{k=2}_{PSO} + P$ $(40 * 8 * (0.31 * 10^4))$	
	Min	Avg	Min	Avg	Min	Avg	Min	Avg
c50	577.60	592.95	574.71	600.56	**544.11**	**578.31**	551.34	580.56
c75	928.53	962.54	**875.34**	938.20	884.43	903.72	886.42	**901.05**
c100b	864.19	878.81	844.69	919.17	**819.56**	845.80	**819.56**	**844.56**
c100	949.83	968.92	924.42	965.65	902.00	933.46	**873.77**	**920.69**
c120	1164.63	1284.62	1208.36	1282.52	**1053.18**	**1071.38**	1056.70	1116.08
c150	1274.33	1327.24	1168.37	1228.99	1098.03	1134.20	**1097.27**	**1127.04**
c199	1600.57	1649.17	1476.06	1524.69	**1362.65**	**1408.70**	1374.47	1418.71
f71	283.43	294.85	286.07	307.87	274.16	298.50	**270.20**	**275.27**
f134	14814.10	16083.82	11936.42	13055.75	11746.40	11892.00	**11713.20**	**11810.04**
tai75a	1785.11	1837.00	1705.21	1787.79	**1685.23**	1805.03	1691.95	**1782.79**
tai75b	1398.68	1425.80	1365.21	1420.97	1365.36	1422.60	**1356.50**	**1401.38**
tai75c	1490.32	1532.45	1439.21	1532.23	1439.02	1510.00	**1424.91**	**1486.78**
tai75d	**1342.26**	1448.19	1428.75	1466.43	1408.79	1433.25	1403.85	**1428.91**
tai100a	2170.54	**2213.75**	2141.29	2230.84	**2137.30**	2216.23	2147.07	2226.48
tai100b	2093.54	2190.01	2110.11	2211.16	2060.65	2136.80	**2041.96**	**2125.19**
tai100c	1491.13	1553.55	1476.91	1537.92	1458.81	1494.72	**1446.98**	**1485.47**
tai100d	1732.38	1895.42	1680.29	1799.04	1663.87	1727.95	**1658.48**	**1718.18**
tai150a	**3253.77**	**3369.48**	3314.22	3471.67	3338.71	3530.82	3396.49	3526.10
tai150b	**2865.17**	**2959.15**	2919.36	3071.33	2910.06	3026.89	2931.16	3034.65
tai150c	2510.13	2644.69	2544.24	2729.26	**2497.65**	**2603.53**	2523.53	2642.82
tai150d	2872.80	3006.88	**2845.01**	**2973.14**	2869.79	3009.01	2929.91	3023.48

remains a state-of-the-art method (as was initially concluded in [11]). The other general conclusion is that none of the tested external algorithms should be considered as the clearly preferred one, although the learning procedure of ES-LC [43] seems to be worth further investigations as it has potential to lower the amount of the number of FFE required to obtain high quality solutions.

6.4. Measuring stability of solutions

Observations stemming from the above experiment are concluded with a proposal of two new metrics for measuring the level of dynamism of a given DVRP instance: *empirical degree of dynamism* and *relative solutions distance*.

First of all, observe that *degree of dynamism* (13) defined in section 3.3 captures only the initial number of available requests. An alternative measure is the *effective degree of dynamism* introduced in [64], which measures the average time of request's availability during the whole optimization process. However, in the considered DVRP instances we have observed (cf. Section 3.4) that just before the middle of the working day multiple requests are still available for reassignment in the standard setup of $T_{CO} = 0.5$. Considering the size of benchmark instances and currently available computational power, it makes the problem far less dynamic then the *degree of dynamism* or *effective degree of dynamism* would suggest. Therefore, we propose an *empirical degree of dynamism* (*em.dod*) as an alternative measure, which we believe is more adequate for capturing this phenomenon than the two aforementioned metrics.

Definition 2. *For a given DVRP instance, an* ***empirical degree of dynamism*** *(em.dod) is the minimal ratio (computed over all of the time slices) of the sum of the number of unknown (m_u) and the number of ultimately assigned (m_a) requests to the total number of requests m:*

$$em.dod = \min_{i \in \{0,1,\ldots,N_{TS}\}} \frac{m_u(t_i) + m_a(t_i)}{m} \tag{17}$$

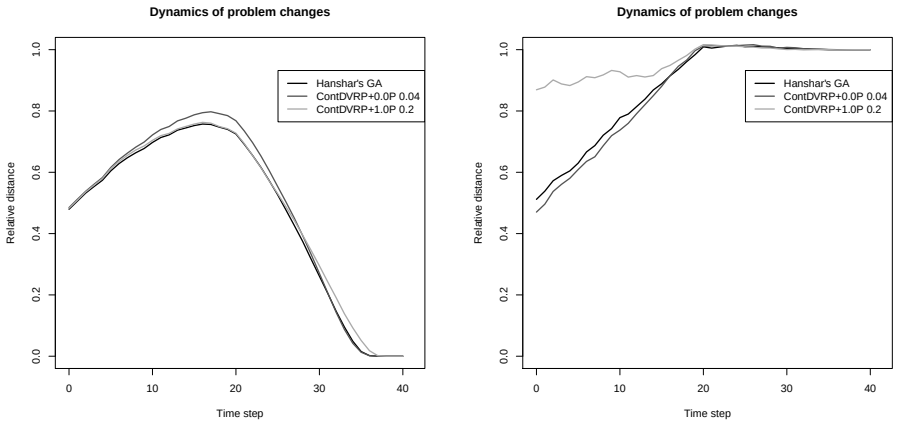
Figure 10 presents the relative number of pending requests and relative number of vehicles needed to serve them. It can be concluded from the plots, that for Kilby et al.'s benchmark set the average value of *dod* is close to the value of T_{CO}, while the average *em.dod* values are much smaller, around 0.2, 0.4 and 0.55 for T_{CO} equal to 0.5, 0.6 and 0.7 respectively. In order to measure stability of partial (intermediate) solutions obtained during optimization process of a given problem instance a *relative solution distance* is introduced.

Definition 3. ***Relative solutions distance*** *($\rho(R_{t_j}, R_{t_k})$) is the number of requests known in solutions R_{t_j} and R_{t_k} assigned to different vehicles divided by the total number of requests known in both solutions.*

$$\rho(R_{t_j}, R_{t_k}) = \frac{\sum\limits_{i \notin C_U(t_j) \wedge i \notin C_U(t_k)} \sum\limits_{r_l(t_j) \in R_{t_j}} I_{(i \in r_l(t_j) \wedge i \notin r_l(t_k))}}{|\{i : i \notin C_U(t_j) \wedge i \notin C_U(t_k)\}|}, \tag{18}$$

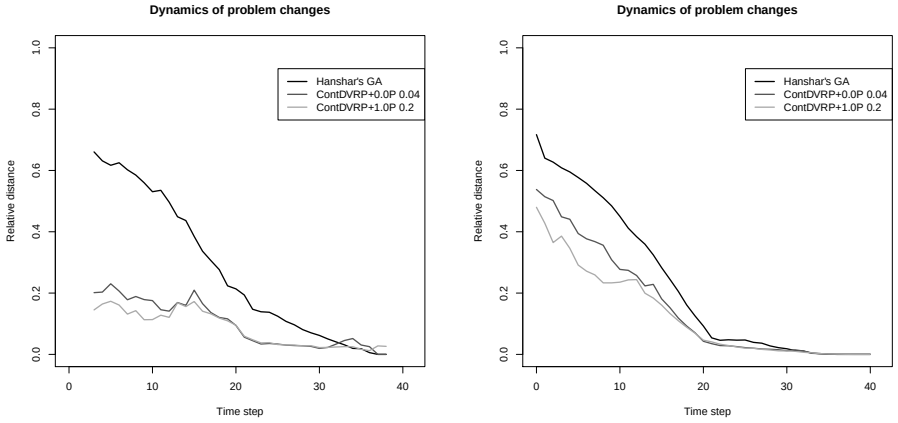
where I is an indicator function and $C_U(t)$ is a set of requests unknown at time t.

As stated above, the average performance of ContDVRP with and without penalty term is roughly similar, albeit the results of $ContDVRP + P$ are more stable during the optimization process (cf. Figure 10). However, it should also be noted that increased stability comes with the cost of longer computation time. Introducing the penalty term extends the running time by 23% on average, due to higher number of estimated vehicles $\hat{n}$ in intermediate solutions (therefore, adequately higher search space dimensionality).



(a) Average relative number of available requests.

(b) Average relative number of vehicles.



(c) Average relative number of request reassignments.

(d) Average distance of intermediate solutions to the final solution.

Figure 10: DVRP dynamism in various optimization approaches, aggregated over all benchmark datasets.

6.5. Algorithm behavior for various cut-off times

The final experimental evaluation was conducted in a way similar to experiments presented in [47, 58] and aimed at verifying the impact of the higher degree of problem dynamism (in the form of greater T_{CO} values). The respective parameter settings are presented in Tables 1 and 7.

Table 8 presents the results achieved for different *cut-off time* values by ContVRP with penalty term. A more fine-grained results, in the relation to the best known values for $T_{CO} = 0.5$, are presented in Figure 11. The plots

Table 7: Settings of the PS algorithms used in the experiment verifying the impact of the *cut-off time* selection.

Encoding	Algorithm	Stopping criterion	Penalty	k	T_{CO}	T_{AC}
Clusters	PSO	#FFE	No	2	0.5	0.04
Clusters	PSO	#FFE	Yes	2	0.5	0.20
Giant TSP tour	GA	#FFE	N/A	N/A	0.5	0.15
Clusters	PSO	#FFE	No	2	0.6	0.20
Clusters	PSO	#FFE	Yes	2	0.6	0.25
Giant TSP tour	GA	#FFE	N/A	N/A	0.6	0.20
Clusters	PSO	#FFE	No	2	0.7	0.30
Clusters	PSO	#FFE	Yes	2	0.7	0.35
Giant TSP tour	GA	#FFE	N/A	N/A	0.7	0.35

Table 8: Summary of results of ContDVRP with penalty term, for various *cut-off times*.

	$ContDVRP^{k=2}_{PSO}+P$ $T_{CO}=0.5$		$ContDVRP^{k=2}_{PSO}+P$ $T_{CO}=0.6$		$ContDVRP^{k=2}_{PSO}+P$ $T_{CO}=0.7$	
Name	Min	Avg	Min	Avg	Min	Avg
c50	**551.34**	**580.56**	579.78	602.23	714.38	766.67
c75	886.42	**901.05**	**886.17**	944.82	950.96	1004.69
c100	**873.77**	**920.69**	934.57	988.91	1012.24	1091.49
c100b	**819.56**	844.56	824.38	**839.56**	865.18	1006.01
c120	**1056.7**	**1116.08**	1206.7	1255.44	1326.5	1416.88
c150	**1097.27**	**1127.04**	1121.12	1184.9	1194.96	1272.62
c199	**1374.47**	**1418.71**	1400.53	1445.99	1525.32	1610.45
f71	**270.2**	**275.27**	286.7	318.54	354.6	387.25
f134	**11713.2**	**11810.04**	11804.33	11963.52	11731.95	11898.52
tai75a	**1691.95**	**1782.79**	1738.86	1859.9	1847.61	1981.86
tai75b	**1356.5**	**1401.38**	1375.46	1413.49	1483.51	1566.71
tai75c	**1424.91**	**1486.78**	1453.87	1524.47	1451.14	1581.5
tai75d	**1403.85**	**1428.91**	1476.65	1520.23	1500.56	1565.57
tai100a	**2147.07**	**2226.48**	2177.51	2231.35	2315.86	2477.76
tai100b	**2041.96**	**2125.19**	2071.83	2165.51	2117.79	2302.81
tai100c	**1446.98**	**1485.47**	1551.75	1676.66	1733.95	1846.85
tai100d	**1658.48**	**1718.18**	1673.19	1719.4	1899.32	2043.63
tai150a	3396.49	**3526.1**	**3387.75**	3601.21	4048.82	4472.62
tai150b	**2931.16**	**3034.65**	2976.8	3079.57	3269.54	3632.86
tai150c	2523.53	2642.82	**2464.72**	**2626.94**	2523.54	2633.84
tai150d	2929.91	**3023.48**	**2927.44**	3133.48	3032.81	3189.79
sum	**43595.72**	**44876.23**	44320.11	46096.12	46900.54	49750.38

illustrate the impact of growing degree of problem dynamism which can be observed regardless of the optimization algorithm used. Generally speaking, the performance of ContDVRP exceeds that of GA by a constant margin within the tested range of T_{CO} values.

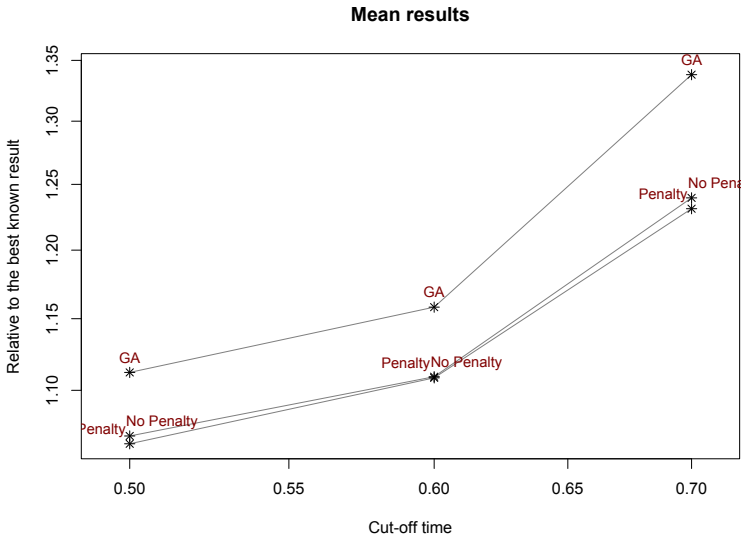


Figure 11: Average results relative to the best known values for $T_{CO} = 0.5$ for various *cut-off times*. The figure compares performance of GA run in Parallel Services environment (GA), $ContDVRP^{k=2}_{PSO}$ ($NoPenalty$) and $ContDVRP^{k=2}_{PSO} + P$ ($Penalty$).

7. Conclusions

Research results presented in this paper confirm that discrete Dynamic Vehicle Routing Problem can be efficiently solved in a continuous search space. It has also been presented that the average quality of ContDVRP solutions is, to a large degree, independent of the optimization algorithm, as both PSO and DE accomplished similar results for both types of continuous encodings (*clusters only* and *clusters and priorities*). Additionally, the study revealed that GA method run with the constrained budget of FFE yields results competitive to those of MEMSO algorithm, and therefore can be used on equal terms as a reference algorithm utilizing discrete search space. As for the reasons behind the apparent advantage of the continuous encodings over the discrete one we may offer a following explanation. Continuous clustering approach implicitly includes a heuristic assumption that requests with locations close to one another should be served by the same vehicle. Meanwhile, discrete encodings offer no such restriction and the algorithms utilizing them are theoretically prone to exploring even the worst possible solutions. Investigation of the level of difference between the worst possible solutions achievable in each search space is an interesting topic for future research.

Since standard degree of dynamism *dod* does not fully capture dynamic nature of DVRP, we proposed the empirical degree of dynamism *em.dod*. This proposed *em.dod* measure refers to the minimal ratio of fixed and unknown requests during the optimization process (see Figure 10 for the differences between

dod and *em.dod*). Following the observation of persisting high availability of requests (at the level of 80% of all problem requests) for the *cut-off time* set in the middle of the working day, new results for higher values of T_{CO} have been computed. Those experiments have been performed with ContDVRP and GA on the same set of benchmark instances [3] as in the rest of article and presented in Table 8 and Figure 11.

Using continuous encoding with a representation size proportional to the number of estimated vehicles allows for stabilization of the optimization process (cf. Figure 10) by means of applying a robust approach to design of the penalty term. It can also be observed that ContDVRP generates more stable sequence of intermediate solutions than discrete GA approach, even without application of the penalty term.

Finally, proposed continuous encoding induces a division of the operational area in a way familiar to human vehicles dispatchers, which can be further tuned using higher number of request clusters per vehicle, albeit at the cost of longer computation time.

The most salient feature of the proposed continuous encoding scheme is its high degree of independence of the optimization method used to solve the DVRP. Consequently, practically any general-purpose continuous optimization metaheuristics can be employed for solving this problem.

Acknowledgments

This work was supported by the National Science Centre, Poland, grants number 2012/07/B/ST6/01527 and 2017/25/B/ST6/02061.

References

[1] G. B. Dantzing, J. H. Ramser, The Truck Dispatching Problem, Management Science 6 (1) (1959) 80–91. doi:10.1287/mnsc.6.1.80.

[2] H. N. Psaraftis, Dynamic Vehicle Routing Problems, in: Vehicle Routing: Methods and Studies, Elsevier, 1988, pp. 223–248.

[3] P. Kilby, P. Prosser, P. Shaw, Dynamic VRPs: A Study of Scenarios (1998). URL https://www.researchgate.net/publication/2944001_Dynamic_VRPs_A_Study_of_Scenarios

[4] R. Montemanni, L. M. Gambardella, A. E. Rizzoli, A. V. Donati, Ant colony system for a dynamic vehicle routing problem, Journal of Combinatorial Optimization 10 (4) (2005) 327–343.

[5] M. J. Elhassania, B. Jaouad, E. A. Ahmed, A new hybrid algorithm to solve the vehicle routing problem in the dynamic environment, International Journal of Soft Computing 8 (5) (2013) 327–334.

[6] J. Euchi, A. Yassine, H. Chabchoub, The dynamic vehicle routing problem: Solution with hybrid metaheuristic approach, Swarm and Evolutionary Computation 21 (2015) 41–53. doi:10.1016/J.SWEVO.2014.12.003.

[7] F. T. Hanshar, B. M. Ombuki-Berman, Dynamic vehicle routing using genetic algorithms, Applied Intelligence 27 (1) (2007) 89–99. doi:10.1007/s10489-006-0033-z.

[8] M. Elhassania, B. Jaouad, E. A. Ahmed, Solving the dynamic Vehicle Routing Problem using genetic algorithms, in: Logistics and Operations Management (GOL), 2014 International Conference on, IEEE, 2014, pp. 62–69.

[9] P. Garrido, M. C. Riff, DVRP: a hard dynamic combinatorial optimisation problem tackled by an evolutionary hyper-heuristic, Journal of Heuristics 16 (6) (2010) 795–834.

[10] J. Mańdziuk, A. Żychowski, A memetic approach to vehicle routing problem with dynamic requests, Applied Soft Computing 48 (2016) 522–534. doi:http://dx.doi.org/10.1016/j.asoc.2016.06.032.

[11] M. Okulewicz, J. Mańdziuk, The impact of particular components of the PSO-based algorithm solving the Dynamic Vehicle Routing Problem, Applied Soft Computing 58 (2017) 586–604. doi:https://doi.org/10.1016/j.asoc.2017.04.070.

[12] M. R. Khouadjia, E.-G. Talbi, L. Jourdan, B. Sarasola, E. Alba, Multi-environmental cooperative parallel metaheuristics for solving dynamic optimization problems, Journal of Supercomputing 63 (3) (2013) 836–853.

[13] C. Bright, L. While, T. French, M. Reynolds, Using market-based optimisation to solve the dynamic vehicle routing problem, in: 2017 IEEE Symposium Series on Computational Intelligence, SSCI 2017, Honolulu, HI, USA, November 27 - Dec. 1, 2017, IEEE, 2017, pp. 1–8. doi:10.1109/SSCI.2017.8280869.

[14] H. Zou, M. M. Dessouky, A look-ahead partial routing framework for the stochastic and dynamic vehicle routing problem, Journal on Vehicle Routing Algorithms (2018) 1—16doi:10.1007/s41604-018-0006-5.

[15] T. T. Nguyen, S. Yang, J. Branke, Evolutionary dynamic optimization: A survey of the state of the art, Swarm and Evolutionary Computation 6 (2012) 1–24. doi:10.1016/j.swevo.2012.05.001.

[16] M. Mavrovouniotis, S. Yang, Ant algorithms with immigrants schemes for the dynamic vehicle routing problem, Information Sciences 294 (October) (2014) 456–477. doi:10.1016/j.ins.2014.10.002.

[17] T. Blackwell, Particle Swarm Optimization in Dynamic Environments, Evolutionary Computation in Dynamic and Uncertain Environments 51 (2007) 29–49. doi:10.1007/978-3-540-49774-5_2.

[18] J. Mańdziuk, New shades of the Vehicle Routing Problem: Emerging problem formulations and Computational Intelligence solution methods, IEEE Transactions on Emerging Topics in Computational Intelligence (in print) 1–15doi:10.1109/TETCI.2018.2886585.

[19] J. Branke, M. Orbayi, Ş. Uyar, The role of representations in dynamic knapsack problems, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 3907 LNCS (2006) 764–775. doi:10.1007/11732242_74.

[20] D. Pflugfelder, J. J. Wilkens, U. Oelfke, Worst case optimization: a method to account for uncertainties in the optimization of intensity modulated proton therapy., Physics in medicine and biology 53 (6) (2008) 1689–700. doi:10.1088/0031-9155/53/6/013.

[21] A. Ben-Tal, A. Nemirovski, Robust optimization - methodology and applications, Mathematical Programming 92 (3) (2002) 453–480. doi:10.1007/s101070100286.

[22] Y. Jin, J. Branke, Evolutionary Optimization in Uncertain Environments—A Survey, IEEE Transactions on Evolutionary Computation 9 (3) (2005) 1–15.

[23] H. G. Beyer, B. Sendhoff, Robust optimization - A comprehensive survey, Computer Methods in Applied Mechanics and Engineering 196 (33-34) (2007) 3190–3218. doi:10.1016/j.cma.2007.03.003.

[24] V. Kachitvichyanukul, Comparison of Three Evolutionary Algorithms: GA, PSO, and DE, Industrial Engineering and Management Systems 11 (3) (2012) 215–223. doi:10.7232/iems.2012.11.3.215.

[25] M. Okulewicz, J. Mańdziuk, Particle Swarm Optimization hyper-heuristic for the Dynamic Vehicle Routing Problem, in: 7th BIOMA Conference, 2016, pp. 215–227. doi:10.13140/RG.2.2.27509.58082.

[26] A. Younes, P. Calamai, O. Basir, Generalized benchmark generation for dynamic combinatorial problems, in: Proceedings of the 7th annual workshop on Genetic and evolutionary computation, ACM, 2005, pp. 25–31.

[27] P. Rohlfshagen, X. Yao, Attributes of dynamic combinatorial optimisation, in: Asia-Pacific Conference on Simulated Evolution and Learning, Springer, 2008, pp. 442–451.

[28] P. Rohlfshagen, X. Yao, Dynamic combinatorial optimisation problems: An analysis of the subset sum problem, Soft Computing 15 (9) (2011) 1723–1734. doi:10.1007/s00500-010-0616-9.

[29] M. Okulewicz, J. Mańdziuk, Two-Phase Multi-Swarm PSO and the Dynamic Vehicle Routing Problem, in: 2nd IEEE Symposium on Computational Intelligence for Human-like Intelligence, IEEE, Orlando, Fl, USA, 2014, pp. 86–93. doi:10.1109/CIHLI.2014.7013391.

[30] V. Pillac, M. Gendreau, C. Guéret, A. L. Medaglia, A review of dynamic vehicle routing problems., European Journal of Operational Research 225 (1) (2013) 1–11. doi:10.1016/j.ejor.2012.08.015.

[31] H. N. Psaraftis, M. Wen, C. A. Kontovas, Dynamic vehicle routing problems: Three decades and counting, Networks 67 (1) (2016) 3–31. doi:10.1002/net.21628.

[32] M. Mavrovouniotis, C. Li, S. Yang, A survey of swarm intelligence for dynamic optimization: Algorithms and applications, Swarm and Evolutionary Computation 33 (2017) 1–17. doi:10.1016/J.SWEVO.2016.12.005.

[33] Y. Marinakis, M. Marinaki, A. Migdalas, Particle Swarm Optimization for the Vehicle Routing Problem : A Survey and a Comparative Analysis, Springer International Publishing, Cham, 2017. doi:10.1007/978-3-319-07153-4_42-1.

[34] M. Okulewicz, J. Mańdziuk, Application of Particle Swarm Optimization Algorithm to Dynamic Vehicle Routing Problem, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 7895 (2) (2013) 547–558. doi:10.1007/978-3-642-38610-7_50.

[35] Y. Marinakis, G.-R. Iordanidou, M. Marinaki, Particle Swarm Optimization for the Vehicle Routing Problem with Stochastic Demands, Applied Soft Computing Journal 13 (4) (2013) 1693–1704. doi:10.1016/j.asoc.2013.01.007.

[36] T. J. Ai, V. Kachitvichyanukul, A particle swarm optimization for the vehicle routing problem with simultaneous pickup and delivery, Computers and Operations Research 36 (5) (2009) 1693–1702. arXiv:arXiv:1011.1669v3, doi:10.1016/j.cor.2008.04.003.

[37] J. E. Mendoza, J. G. Villegas, A space biased-sampling approach for the vehicle routing problem with stochastic demands, in: 9th Metaheuristics International Conference, 2011, pp. 643–645.

[38] D. M. Ryan, C. Hjorring, F. Glover, Extensions of the Petal Method for Vehicle Routeing, The Journal of the Operational Research Society 44 (3) (1993) 289–296. doi:10.2307/2584199.

[39] W. Tu, Z. Fang, Q. Li, S.-L. Shaw, B. Chen, A bi-level Voronoi diagram-based metaheuristic for a large-scale multi-depot vehicle routing problem, Transportation Research Part E: Logistics and Transportation Review 61 (2014) 84–97. doi:10.1016/j.tre.2013.11.003.

[40] W. Tu, Q. Li, Z. Fang, B. Zhou, A Novel Spatial-Temporal Voronoi Diagram-Based Heuristic Approach for Large-Scale Vehicle Routing Optimization with Time Constraints, ISPRS International Journal of Geo-Information 4 (4) (2015) 2019–2044. doi:10.3390/ijgi4042019.

[41] S. Chen, R. Chen, J. Gao, S. Chen, R. Chen, J. Gao, A Monarch Butterfly Optimization for the Dynamic Vehicle Routing Problem, Algorithms 10 (3) (2017) 107. doi:10.3390/a10030107.

[42] H. Xu, F. Duan, P. Pu, Solving Dynamic Vehicle Routing Problem using Enhanced Genetic Algorithm with Penalty Factors, International Journal of Performability Engineering 14 (4) (2018) 611–620. doi:10.23940/ijpe.18.04.p3.611620.

[43] L. Zhou, L. Feng, A. Gupta, Y.-S. Ong, K. Liu, C. Chen, E. Sha, B. Yang, B. W. Yan, Solving dynamic vehicle routing problem via evolutionary search with learning capability, in: 2017 IEEE Congress on Evolutionary Computation (CEC), IEEE, 2017, pp. 890–896. doi:10.1109/CEC.2017.7969403.

[44] B. Sarasola, K. F. Doerner, V. Schmid, E. Alba, Variable neighborhood search for the stochastic and dynamic vehicle routing problem, Annals of Operations Research 236 (2) (2016) 425–461. doi:10.1007/s10479-015-1949-7.

[45] A. M. F. AbdAllah, D. L. Essam, R. A. Sarker, On solving periodic re-optimization dynamic vehicle routing problems, Applied Soft Computing 55 (2017) 1–12. doi:10.1016/J.ASOC.2017.01.047.

[46] R. Yi, W. Luo, C. Bu, X. Lin, A hybrid genetic algorithm for vehicle routing problems with dynamic requests, in: 2017 IEEE Symposium Series on Computational Intelligence (SSCI), IEEE, 2017, pp. 1–8. doi:10.1109/SSCI.2017.8285301.

[47] H. Xu, P. Pu, F. Duan, Dynamic Vehicle Routing Problems with Enhanced Ant Colony Optimization, Discrete Dynamics in Nature and Society 2018 (2018) 1–13. doi:10.1155/2018/1295485.

[48] A. Benaini, A. Berrajaa, Genetic algorithm for large dynamic vehicle routing problem on GPU, in: 2018 4th International Conference on Logistics Operations Management (GOL), IEEE, 2018, pp. 1–9. doi:10.1109/GOL.2018.8378082.

[49] J. Kennedy, R. Eberhart, Particle Swarm Optimization, Proceedings of IEEE International Conference on Neural Networks. IV (1995) 1942–1948.

[50] Y. Shi, R. C. Eberhart, Parameter selection in particle swarm optimization, Proceedings of Evolutionary Programming VII (EP98) (1998) 591–600.

[51] Y. Shi, R. C. Eberhart, A modified particle swarm optimizer, Proceedings of IEEE International Conference on Evolutionary Computation (1998) 69–73.

[52] I. C. Trelea, The particle swarm optimization algorithm: convergence analysis and parameter selection, Information Processing Letters 85 (6) (2003) 317–325.

[53] M. Clerc, Standard PSO 2007 and 2011 (2012).
URL http://www.particleswarm.info/

[54] R. Storn, K. Price, Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces, Journal of Global Optimization 11 (4) (1997) 341–359. doi:10.1023/A:1008202821328.

[55] J. K. Lenstra, A. H. G. R. Kan, Complexity of vehicle routing and scheduling problems, Networks 11 (1981) 221–227.

[56] K. Lund, O. B. G. Madsen, J. M. Rygaard, Vehicle routing with varying degree of dynamism, technical report (1996).

[57] C. Lin, K. L. Choy, G. T. S. Ho, H. Y. Lam, G. K. H. Pang, K. S. Chin, A decision support system for optimizing dynamic courier routing operations, Expert Systems with Applications 41 (15) (2014) 6917–6933. doi:10.1016/j.eswa.2014.04.036.

[58] M. R. Khouadjia, B. Sarasola, E. Alba, L. Jourdan, E.-G. Talbi, A comparative study between dynamic adapted PSO and VNS for the vehicle routing problem with dynamic requests, Applied Soft Computing 12 (4) (2012) 1426–1439. doi:10.1016/j.asoc.2011.10.023.

[59] M. R. Khouadjia, E. Alba, L. Jourdan, E.-G. Talbi, Multi-Swarm Optimization for Dynamic Combinatorial Problems: A Case Study on Dynamic Vehicle Routing Problem, in: Swarm Intelligence, Vol. 6234 of Lecture Notes in Computer Science, Springer, Berlin / Heidelberg, 2010, pp. 227–238. doi:10.1007/978-3-642-15461-4_20.

[60] R. Montemanni, L. M. Gambardella, A. Rizzoli, A. Donati, A new algorithm for a dynamic vehicle routing problem based on ant colony system, Journal of Combinatorial Optimization 10 (2005) 327–343.

[61] M. Okulewicz, J. Mańdziuk, Dynamic Vehicle Routing Problem: A Monte Carlo approach, in: 2nd ITRIA Conference, 2015, pp. 119–138.
URL http://www.mini.pw.edu.pl/~mandziuk/dynamic/wp-content/uploads/ITRIA-2015.pdf

[62] T. J. Ai, V. Kachitvichyanukul, Particle swarm optimization and two solution representations for solving the capacitated vehicle routing problem, Computers & Industrial Engineering 56 (1) (2009) 380–387. doi:10.1016/j.cie.2008.06.012.

[63] G. A. Croes, A method for solving traveling salesman problems, Operations Res. 6 (1958) 791–812.

[64] A. Larsen, The Dynamic Vehicle Routing Problem, Ph.D. thesis, Technical University of Denmark (2000).
URL http://orbit.dtu.dk/fedora/objects/orbit:83345/datastreams/file_5261816/content