A neighborhood search-based heuristic for the dynamic vehicle routing problem

Wilton Gustavo Gomes da Costa[1], Ricardo Santos[2], Willy Alves de Oliveira Soler [3] , Bianca de Almeida Dantas[4]*

[1] wilton_costa@trimble.com, [2] ricardo.santos@ufms.br, [3] willy.oliveira@ufms.br, [4] bianca.dantas@ufms.br. Federal University of Mato Grosso do Sul (UFMS), Federal University of Mato Grosso do Sul (UFMS), Campo Grande, MS, Brazil
* Corresponding author

Abstract
The Vehicle Routing Problem (VRP) is a classical optimization problem that aims to find routes for a fleet of vehicles to meet the demands of a set of customers. Due to its wide applicability, especially in the logistics sector, numerous VRP variants exist. One such variant is the Dynamic Vehicle Routing Problem (DVRP), where new customer demands may arise after the initial routing has been defined. The combinatorial nature of the DVRP makes it challenging to find high-quality feasible solutions within a reasonable runtime for real-world large-sized problems, motivating the development of heuristic approaches to address this issue. This paper presents a new heuristic algorithm, Dynamic Search per Neighbors Routes (DSNR), which uses neighborhood search approaches based on the 2-Opt* operator to allocate new delivery packages to existing routes. The DSNR algorithm was evaluated on instances from the Loggibud benchmark, representing real data from three different Brazilian cities. Regarding distance and number of vehicles, the proposed approach outperforms two other techniques from the literature: the QRP Sweep (QRPS) and the K-Means Greedy (KG). The DSNR achieved gains ranging from 15% to 17% in distance and final delivery costs.
Keywords: 2-Opt*; DSNR; dynamic vehicle routing; local search heuristic; k-means greedy; QRP sweep.

Uma heurística baseada em busca local para o problema de roteamento dinâmico de veículos

Resumo
O Problema de Roteamento de Veículos (VRP) é um problema clássico de otimização no qual se objetiva determinar as rotas mais econômicas para uma frota de veículos, visando atender às demandas de um conjunto de clientes. Devido à sua aplicabilidade, principalmente na área de logística, existem inúmeras variantes do VRP na literatura. Em particular, o Problema de Roteamento Dinâmico de Veículos (DVRP) é uma variante do VRP em que algumas demandas de clientes surgem após as rotas dos veículos já terem sido determinadas. A natureza combinatória do problema torna difícil encontrar soluções de boa qualidade, em tempo computacional aceitável, para instâncias de grande porte oriundas de cenários reais. Esse fato motiva o desenvolvimento de abordagens heurísticas para a solução do DVRP. Nesse contexto, o presente artigo propõe um novo algoritmo para o DVRP , intitulado "Dynamic Search per Neighbors Routes (DSNR)", em que estratégias de busca em vizinhança, baseadas no operador 2-Opt, são aplicadas para alocar novos pacotes de entrega às rotas existentes. O algoritmo DSNR foi avaliado em um conjunto de instâncias do benchmark Loggibud, representando dados reais de pontos de entrega referentes a três cidades brasileiras. Considerando a distância e o número de veículos utilizados, a abordagem proposta supera o desempenho de duas outras técnicas existentes na literatura, a saber, o QRP Sweep (QRPS) e o K-Means Greedy (KG). Os ganhos em distância e custos de entrega final variam de 15% a 17%.*
Palavras-chave: *2-Opt*;* DSNR; *heurísticas de busca local;* K-Means Greedy; QRP Sweep; *roteamento dinâmico de veículos.*

1 Introduction

The Vehicle Routing Problem (VRP) is a classical optimization problem, introduced in the literature by Dantzig and Ramser (1959). According to Pillac et al. (2013), the VRP is usually formulated considering a graph $G = (V, E, D)$, where $V = \{0, 1, \ldots, N\}$ is the set of vertices, with vertex 0 denoting a depot that is the starting point for a vehicle fleet and nodes $1, \ldots, N$, denoting the customers to be visited. $E = \{(i, j) \in V^2 \mid i \neq j\}$ is the set of arcs; and $D = (d_{ij})_{(i,j) \in E}$ is the cost matrix defined over the arc set, usually representing the costs, times, or distances to move between different vertices. The classical VRP consists of finding routes for identical vehicles, ensuring each customer is visited exactly once while minimizing the incurred routing costs, times, or distances.

The VRP finds applicability in various logistic problems arising in different real-world contexts. For example, Ralphs et al. (2003) addressed a VRP considering capacitated vehicles; Bräysy and Gendreau (2005) considered delivery time windows to visit the customers; and Koç et al. (2016) considered a heterogeneous vehicle fleet. This paper focuses mainly on the Dynamic Vehicle Routing Problem (DVRP), a particular VRP in which some packages (customers) to be delivered may not be available in advance, being revealed dynamically during the execution of the routes. Specifically, this scenario considers that new customer orders may be received after the routes have been chosen, requiring some routes to be adjusted to meet these new demands.

Psaraftis (1988) first characterized the DVRP as a service in which vehicles must satisfy demands that evolve in real time. According to Larsen (2000), the DVRP is more challenging to solve than the traditional VRP, which is NP-hard as stated by Lenstra and Kan (1981). This complexity necessitates the development of specialized solution methods capable of finding high-quality feasible solutions in a reasonable runtime. In this context, this paper presents a new neighborhood search heuristic approach to address the DVRP, aiming to provide feasible solutions for large-sized real-world test instances.

The proposed heuristic, Dynamic Search per Neighbors Routes (DSNR), examines neighboring routes to schedule dynamic packages to vehicles while considering time, distance, and cost constraints. The approach is evaluated using the dataset proposed in LOGGIBUD (2021), which consists of thousands of georeferenced delivery points covering capitals from three states (PA, DF, and RJ) of Brazil. The algorithm has been compared to the DVRP algorithms proposed by Bertsimas and Van Ryzin (1993), Gillett and Miller (1974) (QRP Sweep – QRPS), and Comert et al. (2018) (K-means Greedy - KG). DSNR outperforms both QRPS and KG in terms of distance and the number of vehicles used, achieving up to 17% less distance than QRPS and KG. The main contributions of this work are:

- A new heuristic approach to address the DVRP using local search strategies to schedule new dynamic packages;
- A comprehensive evaluation and statistical analysis of the proposed technique on real-world and large-scale dataset of deliveries.

The rest of this paper is organized as follows: Section 2 introduces the DVRP and the theoretical concepts; Section 3 presents the related work; Section 4 describes the DSNR algorithm; Section 5 discusses the results and findings; and Section 6 concludes the paper and proposes directions for future work.

2 Dynamic Vehicle Routing Problem

The DVRP, also known as real-time or online VRP, involves input data that is only partially available at the beginning of the routes planning. This means the total number of packages to be delivered is not known a priori and is revealed over time.

In the literature, the DVRP is categorized on the type of dynamic information considered, such as new customer requests, demands, service times, and travel times. The DVRP was first explored by Psaraftis (1988), and since then, researchers have proposed various methods to evaluate the dynamic degree of the problem, as suggested by Larsen, Madsen, and Solomon (2002).

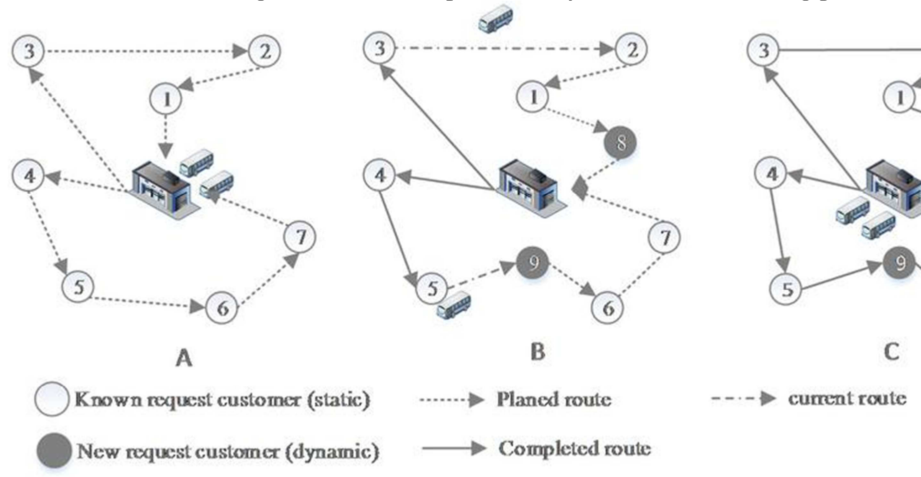
A solution for the DVRP consists of policies to schedule new packages arriving at the depot to the existing routes. Figure 1 illustrates an example of the DVRP:

- There are seven known customers (vertices 1-7) in the first phase (A);
- The vehicles have already started their deliveries in the second phase (B). One vehicle is at customer 3 and the other at customer 5. Two new customer demands (8 and 9) are received

and should be added to the routes. The initial routes are recalculated to accommodate the new customer demands given the current positions of the vehicles;

- The third phase (C) shows that all the customers were served and the vehicles returned to the depot.

Figure 1 – An example of the dynamic vehicle routing problem



A
B
C
Known request customer (static)
New request customer (dynamic)
Planed route
Completed route
current route

Source: Chen, Chen and Gao (2017)

The DVRP can be characterized by the dynamic degree (Equation 1), represented by variable Ω. As an example, Figure 1 has two dynamic packages (two new customers) and a total of nine packages, so that:

$$\Omega = \frac{Number\ of\ dynamic\ packages}{Total\ of\ packages} = \frac{2}{9} = 22.22\% \qquad (1)$$

Given Ω, one can evaluate if the system is dynamic ($\Omega > 0$) or static ($\Omega = 0$). According to Larsen, Madsen, and Solomon (2002), there are three categories of dynamic systems:

- Weakly dynamic: $0 < \Omega \leq 20\%$;
- Moderately dynamic: $20\% \leq \Omega < 80\%$;
- Strongly dynamic: $80\% \leq \Omega < 100\%$.

This work assumes that static packages may be included in the routing. More specifically, we consider that initial routes are created considering a set of known static customer demands. Subsequently, new customer demands arrive at the depot, requiring adjustments in the initial routes. In this paper, we proposed a heuristic solution that handles batches of up to 75 packages in the warehouse and requires redefining the routes of the vehicles.

To conclude this section, we present the mathematical model used to define the addressed DVRP. Suppose that N customer demands are known and K identical vehicles are used to meet these customers. Moreover, consider the graph $G = (V, E, D)$ introduced in Section 1, in which $D = (d_{ij})$ represents the distance matrix, and the following decision variables:

- $x_{ijk} := 1$ if the vehicle k travels from i to j, and 0 otherwise;
- $z_{jk} := 1$ if the vehicle k serves customer j, and 0 otherwise;
- $y_{jk} := 1$ if the customer j is the last visited by vehicle k, and 0 otherwise;
- $v_k := 1$ if the vehicle k is used to serve at least one customer, and 0 otherwise;
- u_{jk}: representing the order in which the customer j is visited by vehicle k.

The mixed integer programming model consists of minimizing the objective function (Equation 2), subject to constraints, Equations 3 to 14.

$$\sum_{i}^{N}\sum_{j}^{N}\sum_{k}^{K} d_{ij}x_{ijk} \tag{2}$$

subject to:

$$\sum_{k}^{K} z_{jk} = 1, \forall j \tag{3}$$

$$\sum_{j}^{N} z_{jk} \leq N v_k, \forall k \tag{4}$$

$$\sum_{i}^{N} x_{ijk} = z_{jk}, \forall k \tag{5}$$

$$\sum_{i}^{N} x_{ijk} = \sum_{i}^{N} x_{jik} + y_{jk}, \forall j, k \tag{6}$$

$$\sum_{j}^{N} x_{0jk} = v_k, \forall k \tag{7}$$

$$u_{jk} \geq u_{ik} + 1 - N\left(1 - x_{ijk}\right), \forall i, j, k \tag{8}$$

$$v_{k+1} \leq v_k, \forall k < K \tag{9}$$

$$v_k \in \{0,1\}, \forall k \tag{10}$$

$$x_{ijk} \in \{0,1\}, \forall i, j, k \tag{11}$$

$$z_{jk} \in \{0,1\}, \forall j, k \tag{12}$$

$$y_{jk} \in \{0,1\}, \forall j, k \tag{13}$$

$$u_{jk} \geq 0, \forall j, k \tag{14}$$

The objective function (2) aims to minimize the sum of distances traveled to serve the customers. Constraints (3) ensure that each customer is visited by a single vehicle, while constraints (4) ensure that customers can only be assigned to utilized vehicles. Constraints (5) guarantee that the customers assigned to each vehicle appear in their respective routes. Constraints (6) are flow constraints, whereas constraints (7) ensure that the utilized vehicles leave the depot. Constraints (8) are the traditional Miller, Tucker, and Zemlin – MTZ- constraints (Miller; Tucker; Zemlin, 1960) to eliminate subroutes. Constraints (9) are valid constraints traditionally used for VRP problems with homogeneous vehicle fleets, to avoid multiple solutions with the same objective function value. More specifically, these constraints ensure that the vehicles are used in ascending order of indexes. Finally, constraints (10) to (14) define the domain of the decision variables.

3 Related work

As the VRP and its variants belong to the NP-hard class of problems, heuristic methods are often used to tackle them. The literature comprises various approaches, including ant colony systems, genetic algorithms, particle swarm optimization algorithms, and combinations of data mining, classification, or ranking techniques. Some recent scientific work is presented in this section, but for a more detailed review, refer to Rios et al. (2021).

The papers in this section were selected based on their proposals of heuristics and techniques to solve the DVRP. Another criterion was the application of the proposed solution to instances or datasets to evaluate and compare it with other algorithms.

Many research works have applied Ant Colony Systems (ACS) to the VRP and its variants. Montemanni et al. (2005) described a strategy based on ACS to solve the DVRP. Their strategy splits the working day into time slices of equal length and processes the arrivals of each time slice only at its end; this approach allows them to deal with a VRP instance at each time slice. The authors assessed the quality of their algorithm results using the dataset originally proposed by Kilby, Prosser, and Shaw (1998) and a real-case scenario in Lugano, Switzerland. The instances proposed by Kilby, Prosser, and Shaw (1998) are problems with 50 to 199 clients. The authors compared their results (minimum, maximum, and average travel time) with those obtained from the original GRASP (Greedy Randomized Adaptive Search Procedure) proposed by Resende and Ribeiro (2003). They found that ACS could achieve better quality results over five runs of both algorithms. They also claimed that their approach was suitable for the proposed real-case scenario, emphasizing that performance was highly dependent on the parameter choices of the ACS, mainly the number of considered time slices.

The Multiple Ant Colony System for Vehicle Routing Problem with Time Windows (MACS-VRPTW) was proposed by Gambardella, Taillard, and Agazzi (1999) to optimize the number of vehicles and the total travel time. The authors proposed a colony system for each objective and implemented cooperation between both colonies through pheromone updating. They evaluated their approach on 56 instances of Solomon's benchmark for VRPTW (Solomon, 1987), divided into six different problem types with 100 nodes each. The results were competitive with previous work, with MACS-VRPTW achieving the best-known results in most instances.

The research conducted by Barán and Schaerer (2003) designed a multi-objective ant colony system to obtain a set of Pareto optimal solutions, considering three objectives simultaneously: the number of vehicles, the total traveling time, and the total delivery time. The proposed algorithm is an extension of the MACS-VRPTW approach (Gambardella; Taillard; Agazzi, 1999), with the difference that no objective had precedence over the others. Their strategy was also evaluated using Solomon's instances, and the results outperformed those of the MACS-VRPTW. However, the authors only presented the results for a specific instance, C101.

Van Veen et al. (2013) proposed MACS-DVRPTW, a strategy inspired by MACS that considered the dynamic version of VRPTW. They created a benchmark based on Solomon's benchmark for VRPTW, using the same instances but with some clients being revealed throughout the day. Their approach is a hybrid algorithm that combines an ACS with specific construction and local search methods. Besides these new dynamic instances, the authors extended their work and evaluated their strategy in a real-case scenario of a working day in a Dutch security company (Yang et al., 2017).

Necula, Breaban, and Raschip (2017) also considered DVRPTW and proposed a novel approach with a single ant colony system to optimize the number of vehicles and the total distance traveled. The algorithm they developed, DVRPTW-ACS, was based on the approach of Van Veen et al. (2013), which involved dividing a working day into equal-sized time slices. The dynamic problem was then viewed as a series of consecutive instances of the static problem. DVRPTW-ACS was evaluated using the same instances proposed by Van Veen et al. (2013) and found that DVRPTW-ACS outperformed MACS-DVRPTW as the dynamicity of the problem increased. Although their approach did not surpass MACS-DVRPTW in most instances with dynamicity levels between 0% and 10%, it achieved better results with 50% to 100% dynamicity levels on both objectives.

Different authors, such as Hong (2012), Pillac, Guéret, and Medaglia (2012), and Silva Júnior, Leal, and Reimann (2021), proposed strategies to solve the DVRPTW that considered the possibility of rejecting a customer. Hong (2012) proposed a strategy that decomposes the dynamic problem into a series of static instances based on the arrival of a new client and uses an improved Large Neighborhood Search (LNS) algorithm to solve the static problem. Pillac, Guéret, and Medaglia (2012) proposed a Parallel Adaptive Large Neighborhood Search (pALNS) algorithm to compute an initial solution and update the solution each time a new customer arrives. Silva Júnior, Leal, and Reimann (2021) described a framework to solve DVRPTW using seven algorithmic variants based on an insertion heuristic, ant colony systems, variable neighborhood descent, and random variable neighborhood descent. All three works evaluated their algorithms using instances proposed by Lackner

(2004) that consist of 56 instances with 100 customers from Solomon's dataset. Silva Júnior, Leal, and Reimann (2021) evaluated the influence of the variability of each heuristic's hyperparameters on the quality of the results and found that their algorithm effectively minimized the number of unserved clients. In situations where the rejection of clients is less complex, the approach of Pillac, Guéret, and Medaglia (2012) achieved better results.

The work of Marinakis and Marinaki (2010) proposed a solution to DVRP with three phases. The solution was based on the combination of a genetic algorithm with their Multiple Phase Neighborhood Search-Greedy Randomized Adaptive Search Procedure (MPNS-GRASP) (Marinakis; Migdalas; Pardalos, 2009) and a Particle Swarm Optimization (PSO) algorithm. The approach was evaluated using the dataset instances proposed by Solomon (1987).

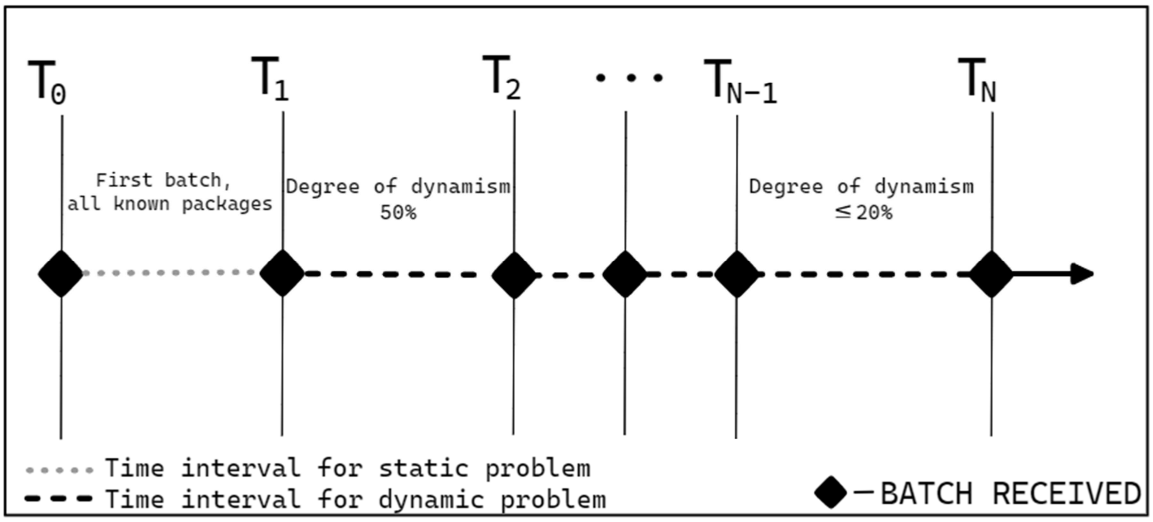
In Fonseca-Galindo et al. (2022), a heuristic strategy for DVRP is described, which takes into account limited capacity vehicles and stochastic clients. The proposed heuristic combines a multi-agent system with trajectory data mining techniques to obtain territorial patterns. The heuristic uses the FP-Growth algorithm, allowing distributed computing to mine large databases. A pre-processing phase, mapping a sequence of points (clients' locations) into a sequence of geographic areas called cells, is proposed to reduce the granularity of information processed by the FP-Growth algorithm. The Google S2 geometry library is then used to represent these points and cells. The algorithm was evaluated using a real-world dataset with 136,000 packages delivered in 2019 from three expedition centers in Belo Horizonte, Brazil.

This section presented a set of different approaches proposing solutions to the DVRP. Most of the solutions are applied to small instances, thus mitigating the scalability of the approach. Fonseca-Galindo et al. (2022) ran their heuristic on a large set of instances of short distances and did not compare the results to other algorithms. One may observe a gap in high scalability proposals that could work on large instances of long distances and, mainly, compare the results to other DVRP algorithms. This work focuses on this gap, showing that the proposed heuristic outperforms classical algorithms when looking at the traveled distance and the number of vehicles allocated to the delivery.

4 Dynamic search per neighbor route

The proposed algorithm processes package batches in specific periods, as shown in Figure 2. The first batch received at the depot at time T_0 can be considered a static problem (CVRP) since all the packages in the batch are known a priori. From the second batch (time T_1), the problem becomes dynamic (dynamic degree ≤ 50%), since half of the packages already have routes from the first batch. At time T_{N-1}, the N_{th} batch is inserted into the problem.

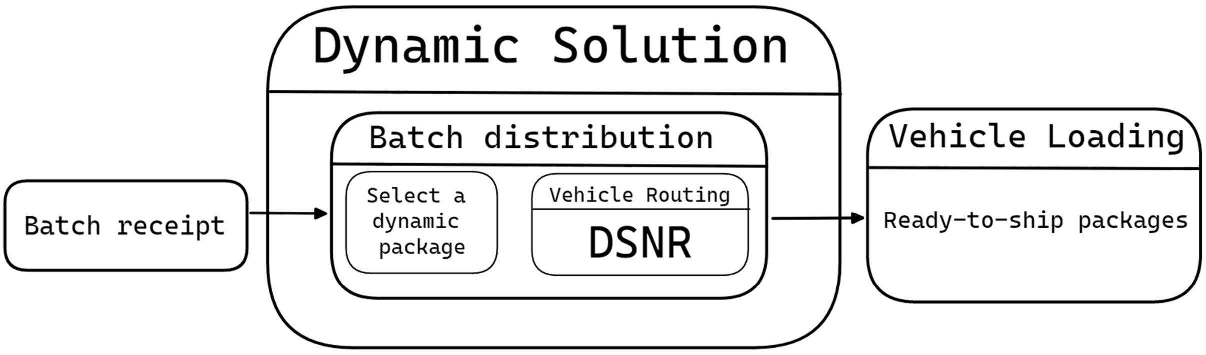
Figure 2 – Batch packages routing in T_N periods


T_0
T_1
T_2
...
T_{N-1}
T_N
First batch, all known packages
Degree of dynamism 50%
Degree of dynamism ≤ 20%
Time interval for static problem
Time interval for dynamic problem
BATCH RECEIVED

Source: authors

Figure 3 shows the steps of applying the algorithm to a new package batch. First, a dynamic package is selected from the batch and Algorithm 1 is executed. The algorithm selects whether to create a new route or insert the package into a neighboring route. If a new route is created, it must be balanced with the nearest existing one, generating a new route solution. Delivery can start if the vehicle responsible for that route is filled.

Figure 3 – Steps for routing dynamic packages



Source: authors.

The "Dynamic Solution" step presented in Figure 3 has two main components:

a. Select a dynamic package: a batch has a set of packages that can be sorted before being sent to the final distribution center. Equation 15 calculates the weight of each package. $weight_i$ is the average distance (d_{ij}) that a given packet i has to the already routed packages j, including the depot. The set S includes all packages (including the depot):

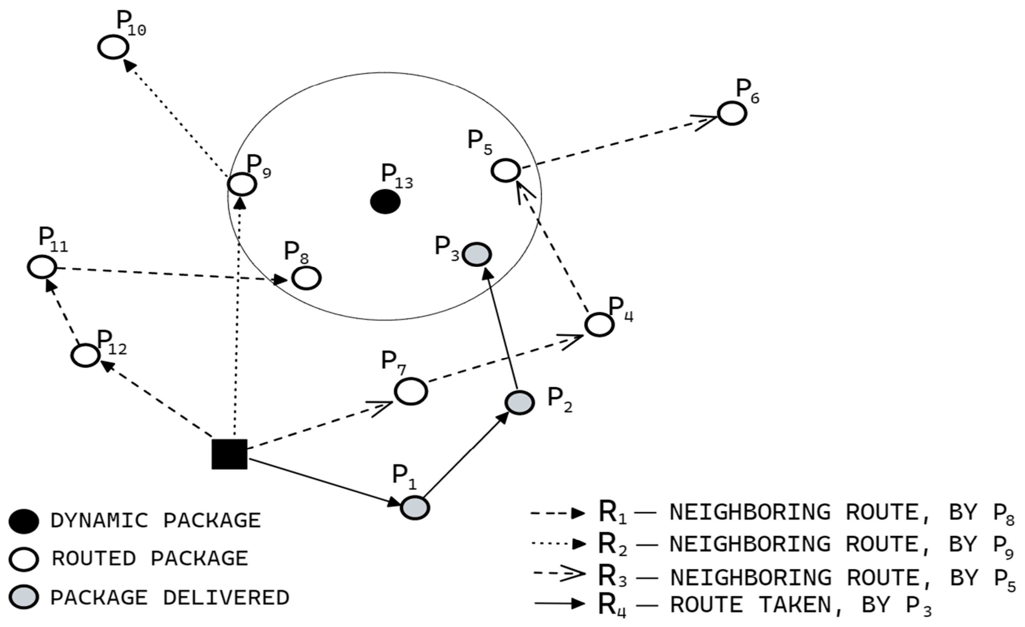
$$weight_i = \frac{\sum_j^N d_{ij}}{|S|} \quad (15)$$

The weight can be used to sort the packages in ascending or descending order.

The application context for the proposed algorithm involves new package batches and available routes that can be reused. Consider the scenery in Figure 4, where there are dynamic packages (black circles), routed packages (white circles), and delivered packages (gray circles). Dynamic packages are those in the current batch. Routed packages have been scheduled to a vehicle that has not left the depot yet. Delivered packages have been allocated to a vehicle that has left the depot and cannot have their routes modified.

b. Vehicle routing (DSNR): once a dynamic package has been selected, the next challenge is to find a neighboring route to insert it. The routed packages will be listed in ascending order of their distance to the dynamic package. After sorting the package, their routes are defined as neighboring routes of the dynamic package. There may be R neighboring routes, where $1 \leq R \leq N$, with N being the number of routed packages neighboring the dynamic package. Figure 4 presents an example with $N = 3$. Packages P_5, P_8, and P_9 are selected. P_3 is in a closed route (the vehicle started the delivery).

Figure 4 – A scenario with one new dynamic package (P_{13}) to be allocated to a new or an existing route



Source: authors

Given the neighboring routes of the dynamic packets, the first route R_3 can be selected according to Algorithm 1.

Algorithm 1 – Insert dynamic package into a neighboring route

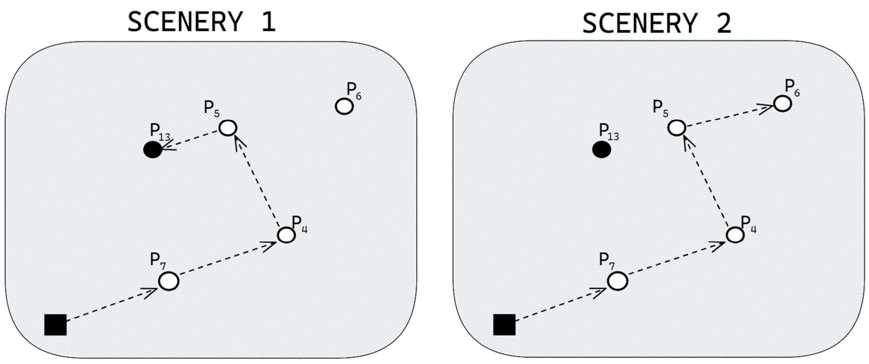
```
function InsertPackage(p, route, routing_packages) {package p insert to route, routing packages is a list of packages that are in a route.}

auxroute = PackageInserted(p, route)
if CapacityExceeded(auxroute) then
        scenery1 = RouteWithDynamicNoWorst(route)
        scenery2 = RouteNoDynamicWithWorst(route)
    if dist(scenery1) < dist(scenery2) then
            EjectsTheWorstPackage(route, routing_packages)
            InsertPackage(p, route, routing_packages)
            return scenery1
    else
             return scenery2 {Scenery 2 = Route without dynamic route}
    end if
 else
    return auxroute
end if
end function
```

The first step of the algorithm is to generate an auxiliary route by inserting the dynamic package into the best position. Once the insertion is done, the vehicle capacity of the auxiliary route is evaluated:

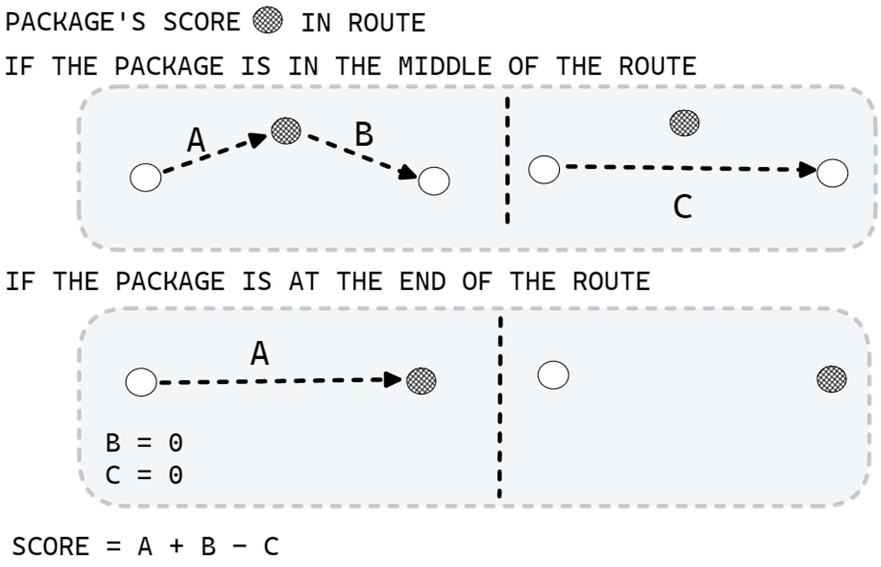
- If the capacity is exceeded, two scenarios are generated, as shown in Figure 5. The first scenario assumes that the dynamic package replaces the worst package in the route. The worst package is the one with the highest score (highest distance). Removing this package will improve the route, as shown in Figure 6. The second scenario occurs when the dynamic package is not inserted into the route.

Figure 5 – Scenarios to insert a new package



Source: authors

Figure 6 – Score of a package on a route


PACKAGE'S SCORE ● IN ROUTE
IF THE PACKAGE IS IN THE MIDDLE OF THE ROUTE
A
B
C
IF THE PACKAGE IS AT THE END OF THE ROUTE
A
B = 0
C = 0
SCORE = A + B − C

Source: authors

Given both scenarios, if scenario 1 has a smaller distance, the worst package is extracted from the route, becoming a dynamic package inserted at the end of the batch. Otherwise, the dynamic package looks for the next neighbor route; if no more neighbor routes exist, a new route is generated for the dynamic package.

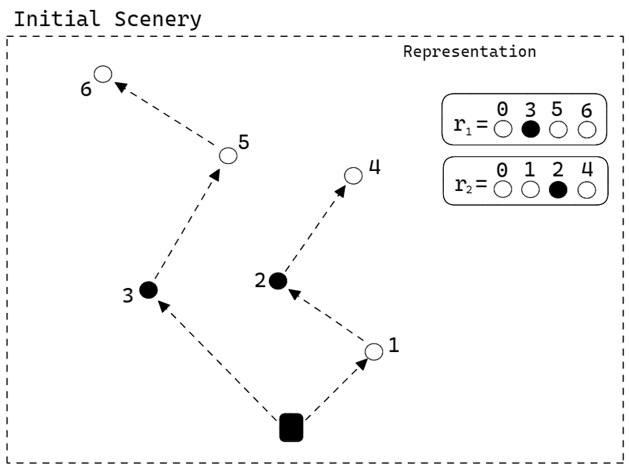
- If the route can accommodate the new dynamic package, the auxiliary route is returned, allowing for the selection of a new package from the batch.

After inserting the dynamic package, if a new route is created, an optimization is performed between the new route and the nearest existing route. The optimization uses a new exchange operator based on the 2-Opt* optimization, called 2-Opt**. The operator is applied until no further improvements in the total distance of the routes can be made.

4.1 4C 2-Opt** Optimization

After package routing, two local search operators are applied to find better solutions. The first operator, 2-Opt* (Fahrion; Wrede, 1990), checks if exchanging two packages, each in a different route, improves the traveled distance. The 2-Opt** operator finds the best placement for a new package in a route. Figure 7 shows an example where r_1 is the first route and r_2 the second route, with packages 2 and 3 selected to explore new possibilities for inserting the new dynamic package.

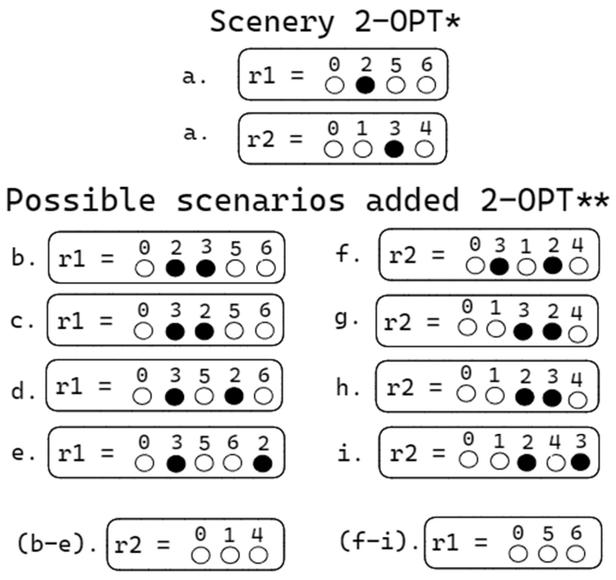
Figure 7 – Initial route for applying 2-Opt*



Source: authors

In 2-Opt*, a single scenario is possible: the exchange between 2 and 3 (a. in Figure 8). 2-Opt** increases the possibility of allocating a package into a route, as both selected packages can be routed into only one route. The heuristic evaluates the best position where one of the selected packages could be allocated to the other route.

Figure 8 – 2-Opt** example



Source: authors

5 Dataset and Heuristics Evaluation

In this work, the routing solutions were tested and evaluated using the Loggibud urban routes benchmark proposed in Loggibud (2021). The Loggibud repository provides historical delivery data from three different cities in Brazil: Rio de Janeiro (RJ), Belém (PA), and Brasília (DF). Each package varies between 1-10 load units, and each available vehicle is capable of carrying up to 180 load units. Additionally, each package has its latitude and longitude location, and distances are calculated using the Open Search Routing Machine (OSRM) server. The deliveries are organized into instances delivered in one business day, and there is no specific order in which the packages arrive at the warehouse. However, the designer can sort the packages into batches, assuming a maximum of 75 packages per batch. Each batch is assumed to arrive at the warehouse with this quantity.

The repository provides two algorithms to solve the dynamic approach, both requiring a parameter *I*, which denotes the number of initial regions for training.

In the Loggibud repository, there are two types of files:

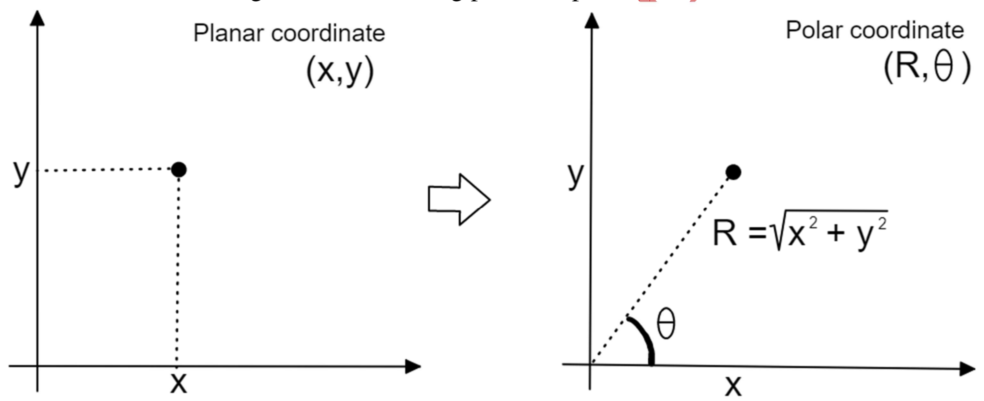
- Input file (CVRPInstance): this file contains information about the warehouse (depot) location and the georeferenced coordinates of each package that should be delivered within one business day;
- Output file (CVRPSolution): this file has the sequence of packages that will be delivered by each vehicle departing from the depot.

The Loggibud repository provides two dynamic routing algorithms: QRP Sweep (QRPS) and K-means Greedy (KG).

5.1 QRP Sweep

This algorithm separates enabled regions with a scan method as discussed in Bertsimas and Van Ryzin (1993) and Gillett and Miller (1974). First, there is a training phase based on historical data, where delivery demands have their planar coordinates (x, y) converted to polar coordinates (R, Θ), with the angles Θ in the range of [–180°, 180°] and $R = \sqrt{x^2 + y^2}$, as shown in Figure 9. A *K-means* algorithm is applied to the training set with polar coordinates, with the depot as the center. After generating the regions, each package is assigned to a region. Once a vehicle reaches its maximum package capacity, or there are no more packages to be shipped in that region, the vehicle responsible for the region is dispatched. Before each vehicle is dispatched, a TSP (Traveling Salesman Problem) is solved using the Branch-and-Cut algorithm from a commercial high-performance MIP (Mixed-Integer Programming) solver, as detailed in Section 6.

Figure 9 – Converting planar to polar coordinates



Source: authors

5.2 Kmeans Greedy

The KG algorithm also uses a training set based on historical data. After the training, the algorithm assigns each package to the region with the closest centroid, with each region having its own vehicle. When the region reaches a limit (i.e., the vehicle capacity), the vehicle from that region is dispatched. As with QRPS, a Branch-and-Cut algorithm from a commercial MIP solver is used to route the packages for each dispatched vehicle within the generated region.

6 Results and discussions

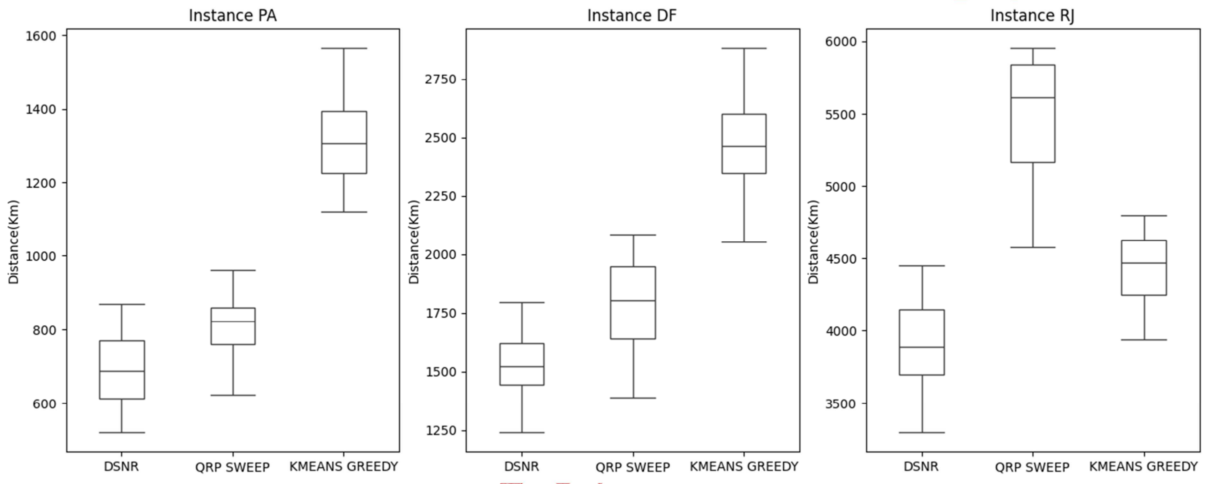
In this section, we present the experiments evaluating the DSNR algorithm. The proposed algorithm was analyzed and compared to the QRPS and KG implementations in the Loggibud repository. All algorithms were executed on a computer with a 6-core AMD Ryzen 5600x processor, 12 threads, GPU RTX 2080 Super, and 16 GB of RAM at 3200MHz. We used the Branch-and-Cut algorithm of the commercial MIP solver IBM ILOG Cplex 20.1 with default settings to solve the necessary mixed integer programming (MIP) sub-problems. The motivation for comparing DSNR to QRPS and KG is that all algorithms can use the same datasets (from the Loggibud benchmark) as inputs to solve the DVRP. Additionally, they all provide the number of vehicles and distance (in kilometers) as outcomes of their solutions.

Subsection 6.1 presents a characterization of each instance (DF, PA, and RJ) of the datasets, considering the distance calculated by each algorithm. Subsection 6.2 outlines how the DSNR algorithm works according to the ordering of the packages. Subsection 6.3 compares the three algorithms in the same instances, considering the number of vehicles and the distance. We also present the final costs of the deliveries based on the number of vehicles and distances reported by each algorithm.

6.1 Characterization of the Instances

Figure 10 presents a boxplot of distances calculated by each algorithm using the Loggibud benchmark. It can be observed that the distances calculated by DSNR are lower than those by KG in all scenarios. In the DF dataset, the median of DSNR is quite similar to that of QRPS but is clearly lower than both QPRS and KG in the PA and RJ datasets, indicating that DSNR achieves better results on instances with large distances.

Figure 10 – Distances calculated by each algorithm in the packages of PA, DF and RJ datasets



Source: https://www.kaggle.com/datasets/wiltoncosta7/pesquisa-por-rotas-vizinhas-dinamicas

6.2 Ordering of the dynamic search per neighbor route

The input of the DSNR approach is a package to be routed and delivered. Each package can be part of a set of packages. The packages in a set can be sorted according to their weights. A package weight is given by the average of the distances from the package to each other package (Equation 2). Once the packages in a set have their weights, they can be passed to the algorithm as follows:

- Non-ordering (N): the packages are inputs to the algorithm as they are in the set;
- Increasing ordering (I): the packages are passed to the algorithm in increasing order of weight;
- Decreasing ordering (D): the packages are passed to the algorithm in decreasing order of weight.

Those three ordering approaches were compared to check if the package ordering impacts the algorithm results. Each set has 30 packages. Tables 1, 2, and 3 present the distance and number of vehicles for each ordering technique (*N*, *I*, *D*) on 10 instances from three different cities (Belem – PA, Brasilia – DF, and Rio de Janeiro – RJ) in Brazil. The results for all 30 instances are available at https://www.kaggle.com/datasets/wiltoncosta7/pesquisa-por-rotas-vizinhas-dinamicas. For each instance, the best (smallest) results for distance and number of vehicles are highlighted in bold.

Table 1 – Distance and number of vehicles for the three ordering approaches on instances of Belem (PA)

Instance	**Distance**			**Vehicles**		
	DSNR-N	**DSNR-I**	**DSNR-D**	**DSNR-N**	**DSNR-I**	**DSNR-D**
pa-90	608	**594**	669	10	**10**	10

pa-91	706	**699**	769	10	**10**	10
pa-92	**520**	700	533	**10**	10	10
pa-93	552	593	**513**	10	10	**10**
pa-94	806	**733**	806	10	**10**	10
pa-95	**675**	854	804	**10**	10	10
pa-96	836	**784**	843	10	**10**	10
pa-97	**649**	721	698	**10**	10	10
pa-98	780	**696**	765	10	**10**	10
pa-99	**761**	908	806	**10**	10	10
Average	**689.3**	728.2	720.6	**10**	10	10

Source: https://www.kaggle.com/datasets/wiltoncosta7/pesquisa-por-rotas-vizinhas-dinamicas

Table 2 – Distance and number of vehicles for the three ordering approaches on instances of Distrito Federal (DF)

Instance	**Distance**			**Vehicles**		
	DSNR-N	**DSNR-I**	**DSNR-D**	**DSNR-N**	**DSNR-I**	**DSNR-D**
df-90	1,465	1,466	**1,448**	46	**44**	45
df-91	**1,293**	1,344	1,331	39	**37**	41
df-92	**1,394**	1,402	1,409	43	42	**40**
df-93	1,356	**1,255**	1,269	40	39	**38**
df-94	1,621	**1,442**	1,693	34	**31**	35
df-95	1,582	1,602	**1,540**	50	**48**	51
df-96	**1,508**	1,647	1,795	41	**39**	39
df-97	**1,303**	1,396	1,344	39	**34**	39
df-98	1,321	**1,261**	1,355	**37**	38	41
df-99	1,519	1,455	**1,441**	31	**30**	36
Average	1,436.2	**1,427**	1,462.5	40	**38.2**	40.5

Source: https://www.kaggle.com/datasets/wiltoncosta7/pesquisa-por-rotas-vizinhas-dinamicas

Table 3 – Distance and number of vehicles for the three ordering approaches on instances of Rio de Janeiro (RJ)

Instance	**Distance**			**Vehicles**		
	DSNR-N	**DSNR-I**	**DSNR-D**	**DSNR-N**	**DSNR-I**	**DSNR-D**
rj-90	**3,896**	3,919	4,007	**188**	193	207
rj-91	4,192	**3,978**	4,134	220	**205**	213
rj-92	4,077	**4,019**	4,185	**203**	207	213
rj-93	4,017	**3,963**	4,118	172	**158**	162
rj-94	**3,881**	3,887	3,998	**161**	164	171
rj-95	3,455	3,451	**3,432**	**113**	117	121
rj-96	3,980	**3,801**	3,994	157	147	**146**
rj-97	3,371	**3,227**	3,498	171	**156**	170
rj-98	3,985	**3,979**	4,096	**169**	172	183
rj-99	**4,221**	4,241	4,385	**215**	217	230
Average	3,907.5	**3,846.5**	3,984.7	176,9	**173,6**	181.6

Source: https://www.kaggle.com/datasets/wiltoncosta7/pesquisa-por-rotas-vizinhas-dinamicas

To calculate a cost for the trips of each instance, the following constraints are applied:

- Distances are measured in kilometers (km);
- Trips for an instance must be completed within one workday;
- Each vehicle incurs an average cost of R$ 1.92 per km;
- The minimum cost for a vehicle to be allocated to a delivery is R$ 13.90.

The vehicles are assumed to be homogeneous, meaning they have the same capacity and costs. Equation 16 is used to calculate the cost for a delivery:

$$Cost = R\$\ 1.92 \times Distance + R\$\ 13.90 \times Vehicles \quad (16)$$

Table 4 presents the costs (in thousands of reais, R$) for all instances of Belem, Distrito Federal, and Rio de Janeiro based on the data presented in Tables 1, 2 and 3.

Table 4 – Costs (R$ × 1,000) of the three ordering approaches

City - Nº samples	DSNR-N (COST)	DSNR-I(COST)	DSNR-D(COST)
PA-300	**R$ 43,954**	R$ 45,115	R$ 44,080
DF-1000	R$ 105,181	R$ 105,335	**R$ 104,850**
RJ-4000	R$ 295,354	**R$ 294,580**	R$ 299,676

Source: https://www.kaggle.com/datasets/wiltoncosta7/pesquisa-por-rotas-vizinhas-dinamicas

One can observe a difference in costs based on the ordering approach. Tests were conducted to determine if there was a statistical difference among these costs. The Shapiro-Wilk test for normality was applied to the costs for each instance in the three cities. Assuming a confidence level (α = 0.05) and a $p - value > \alpha$, the test results indicated no grounds to reject the normality hypothesis. Given that the normality hypothesis was not rejected, an analysis of variance (ANOVA) test was conducted to determine if there were significant differences among the three ordering approaches. The smallest $p - value$ from ANOVA was 0.41, which is greater than α, indicating no statistically significant effects from the orderings. Therefore, the null hypothesis of no effects among the approaches is accepted. Based on this analysis, the non-ordering approach was adopted to evaluate and compare the results with other dynamic techniques.

6.3 Comparison and data analysis

Tables 5, 6 and 7 present the results (distance and number of vehicles) of DSNR (non-ordering), QRPS and KG algorithms. QRPS and KG were executed following *task2* in the Loggibud benchmark repository (*https://github.com/loggi/loggibud*). All algorithms were run on instances PA, DF, and RJ, with each batch containing 75 packages.

Table 5 – Results of distance (km) and number of vehicles of the dynamic algorithms on instances of Belem (PA)

Instance	Distance			Vehicles		
	DSNR	QRPS	KG	DSNR	QRPS	KG
pa-90	**608**	718	1,198	**10**	12	82
pa-91	**706**	876	1,255	**10**	12	81
pa-92	**520**	816	1,322	**10**	12	86
pa-93	**552**	622	1,182	**10**	12	81
pa-94	**806**	807	1,295	**10**	11	85
pa-95	**675**	917	1,548	**10**	13	88
pa-96	**836**	952	1,524	**10**	12	90
pa-97	**649**	754	1,135	**10**	13	76
pa-98	780	**772**	1,215	**10**	12	78
pa-99	**761**	912	1,564	**10**	12	88
Average	**689.3**	814.6	1,323.8	**10**	12.1	83.5

Source: https://www.kaggle.com/datasets/wiltoncosta7/pesquisa-por-rotas-vizinhas-dinamicas

Table 6 – Results of distance (km) and number of vehicles of the dynamic algorithms on instances of Distrito Federal (DF)

Instance	Distance			Vehicles		
	DSNR	QRPS	KG	DSNR	QRPS	KG
df-90	**1,465**	1,748	2,549	**46**	46	107

df-91	**1,293**	1,540	2,231	39	**37**	100
df-92	**1,394**	1,681	2,472	43	**41**	108
df-93	**1,356**	1,388	2,053	40	**36**	92
df-94	**1,621**	1,821	2,729	**34**	42	110
df-95	**1,582**	1,986	2,775	50	**45**	114
df-96	**1,508**	1,932	2,419	**41**	46	106
df-97	**1,303**	1,535	2,449	39	**36**	106
df-98	**1,321**	1,594	2,362	**37**	38	103
df-99	**1,519**	1,626	2,180	**31**	39	98
Average	**1,436.2**	1,685.1	2,421.9	**40**	40.6	104.4

Source: https://www.kaggle.com/datasets/wiltoncosta7/pesquisa-por-rotas-vizinhas-dinamicas

Table 7 – Results of distance (km) and number of vehicles of the dynamic algorithms on instances of Rio de Janeiro (RJ)

Instance	**Distance**			**Vehicles**		
	DSNR	**QRPS**	**KG**	**DSNR**	**QRPS**	**KG**
rj-90	**3,896**	3,919	4,007	**188**	193	207
rj-91	4,192	**3,978**	4,134	220	**205**	213
rj-92	4,077	**4,019**	4,185	**203**	207	213
rj-93	4,017	**3,963**	4,118	172	**158**	162
rj-94	**3,881**	3,887	3,998	**161**	164	171
rj-95	3,455	3,451	**3,432**	**113**	117	121
rj-96	3,980	**3,801**	3,994	157	147	**146**
rj-97	3,371	**3,227**	3,498	171	**156**	170
rj-98	3,985	**3,979**	4,096	**169**	172	183
rj-99	**4,221**	4,241	4,385	**215**	217	230
Average	3,907.5	**3,846.5**	3,984.7	176.9	**173.6**	181.6

Source: https://www.kaggle.com/datasets/wiltoncosta7/pesquisa-por-rotas-vizinhas-dinamicas

Table 8 presents the total delivery costs for each set of instances (DF, PA, RJ) using each dynamic algorithm. Parameters for QRPS and KG were optimized to find the best setup for each algorithm. Both algorithms require an initial number of clusters. Additionally, a case study was conducted for the KG algorithm to determine the optimal I parameter. The values that yielded the best results for each algorithm are presented below and were found empirically:

- For the PA instance, QRPS performed best with 10 clusters, while KG performed best with 8 clusters;
- For the DF instance, QRPS performed best with 40 clusters, while KG performed best with 10 clusters;
- For the RJ instance, the best performance was achieved when the number of clusters equaled the number of vehicles.

Even with the best cluster configurations, DSNR achieved the lowest cost for DF and RJ. There is a small, statistically insignificant difference between DSNR and KG using the PA instance.

Table 8 – Total cost of the dynamic approaches

City - Nº samples	**DSNR**	**QRPS**	**KG**
PA-300	R$ 43,954	R$ 48,604	**R$ 43,896**
DF-1000	**R$ 105,181**	R$ 120,947	R$ 107,731
RJ-4000	**R$ 295,354**	R$ 395,411	R$ 342,507

Source: https://www.kaggle.com/datasets/wiltoncosta7/pesquisa-por-rotas-vizinhas-dinamicas

The maximum cost reduction (in percentage) achieved by the DSNR algorithm compared to the best cost from the KG algorithm using the RJ instance is 13.76%. The Shapiro-Wilk test was also applied to the costs for each sample (Tables 9, 10, and 11). Most *p*-values were higher than $\alpha = 0.05$, indicating no significant effects to reject the null (normality) hypothesis.

Table 9 – Shapiro's normality test on the distance data of Belem (PA)

SHAPIRO	DSNR	QRPS	KG
W	0.9889	0.9770	0.9768
P-VALUE	0.7951	0.7425	0.7351

Source: https://www.kaggle.com/datasets/wiltoncosta7/pesquisa-por-rotas-vizinhas-dinamicas

Table 10 – Shapiro's normality test on the distance data of Distrito Federal (DF)

SHAPIRO	DSNR	QRPS	KG
W	0.9771	0.9610	0.9869
P-VALUE	0.7448	0.3278	0.9644

Source: https://www.kaggle.com/datasets/wiltoncosta7/pesquisa-por-rotas-vizinhas-dinamicas

Table 11 – Shapiro's normality test on the distance data of Rio de Janeiro (RJ)

SHAPIRO	DSNR	QRPS	KG
W	0.9572	0.8805	0.9768
P-VALUE	0.2622	**0.029**	0.1284

Source: https://www.kaggle.com/datasets/wiltoncosta7/pesquisa-por-rotas-vizinhas-dinamicas

The ANOVA test was applied to the same data to compare the differences in distance among the algorithms. All *p*-values were less than α (p-value $< 2.2e^{-16}$), indicating significant differences in distance among DSNR, QRPS, and KG. The results presented in this section are available online at *https://www.kaggle.com/datasets/wiltoncosta7/pesquisa-por-rotas-vizinhas-dinamicas*.

7 Conclusions

This paper presented the design and development of a new heuristic for the Dynamic Vehicle Routing Problem (DVRP). The proposed algorithm, Dynamic Search per Neighbors Routes (DSNR), creates routes based on delivery regions so that new package batches may reuse previously created routes.

The DSNR was evaluated using a set of routing instances from the Loggibud benchmark, representing delivery points from three different cities in Brazil. Our approach outperformed two other DVRP techniques, QRPS and KG, regarding distance and the number of vehicles used. Specifically, DSNR achieved distance savings ranging from 15% to 17%.

The DSNR heuristic's impacts can also be observed in the cost savings once less distance reduces the total delivery operational cost. Consequently, the proposed approach also has an original benefit on the sustainability of the delivery operation considering that less distance or fewer vehicles has a straight effect on greenhouse gas emissions.

One observable limitation of the DSNR approach relies on the package ordering pre-processing. The data in Table 4 revealed that the ordering may have a consequence in the routing final results (costs) so that this a challenge to adopt DSNR in real-world systems where the dynamic degree could be high and there is a dynamic behavior on the delivery distances.

Future work could involve using the K-means technique alongside DSNR to create regions based on historical data. Another potential direction would be to start the dynamic algorithm with a set of packages using a more efficient static approach before applying DSNR to subsequent packages.

These opportunities could lead to further advancements and improvements in the dynamic vehicle routing field.

Financial Support

The authors acknowledge the financial support provided by the Brazilian Research Agency – CNPq, under Process nº 133954/2020-0 and the Federal University of Mato Grosso do Sul for their financial support of this work.

Conflict of Interest

The authors declare that they have no conflict of interest.

References

BARÁN, B.; SCHAERER, M. A multiobjective ant colony system for vehicle routing problem with time windows. In: IASTED CONFERENCE APPLIED INFORMATICS, 21., 2003, Innsbruck. **Proceedings […]**. Innsbruck, 2003, p. 97-102, 2003. Available at: https://www.cnc.una.py/publicaciones/1_75.pdf. Accessed on: 11 june 2024.

BERTSIMAS, D. J.; VAN RYZIN, G. Stochastic and dynamic vehicle routing with general demand and interarrival time distributions. **Advances in Applied Probability,** v. 25, n. 4, p. 947-978, 1993. DOI: https://doi.org/10.2307/1427801.

BRÄYSY, O.; GENDREAU, M. Vehicle routing problem with time windows, Part II: Metaheuristics. **Transportation Science,** v. 39, n. 1, p. 119-139, 2005. DOI: https://doi.org/10.1287/trsc.1030.0057.

CHEN, S.; CHEN, R.; GAO, J. A monarch butterfly optimization for the dynamic vehicle routing problem. **Algorithms**, v. 10, n. 3, 107, 2017. DOI: https://doi.org/10.3390/a10030107.

COMERT, S. E.; YAZGAN H. R.; KIR S.; YENER F. A cluster first-route second approach for a capacitated vehicle routing problem: a case study. **International Journal of Procurement Management (IJPM),** v. 11, n. 4, p. 399-419, 2018. DOI: https://dx.doi.org/10.1504/IJPM.2018.092766.

DANTZIG, G. B.; RAMSER, J. H. The truck dispatching problem. **Management Science,** v. 6, n. 1, p. 80-91, 1959.

FAHRION, R.; WREDE, M. On a principle of chain-exchange for vehicle-routing problems (1-VRP). **Journal of the Operational Research Society,** v. 41, n. 9, p. 821-827, 1990.

FONSECA-GALINDO, J. C.; SURITA, G. C.; MAIA NETO, J.; CASTRO, C. L.; LEMOS, A. P. A multi-agent system for solving the dynamic capacitated vehicle routing problem with stochastic customers using trajectory data mining. **Expert Systems with Applications,** v. 195, 116602, 2022. DOI: https://doi.org/10.1016/j.eswa.2022.116602.

GAMBARDELLA, L. M.; TAILLARD, É.; AGAZZI, G. MACS-VRPTW: A multiple ant colony system for vehicle routing problems with time windows. In: CORNE, D.; DORIGO, M.; GLOVER, F. (ed.). **New ideas in optimization**. London: McGraw-Hill, 1999. p. 63-76.

GILLETT, B. E.; MILLER, L. R. A heuristic algorithm for the vehicle-dispatch problem. **Operations Research,** v. 22, n. 2, p. 340-349, 1974.

HONG, L. An improved LNS algorithm for real-time vehicle routing problem with time windows. **Computers & Operations Research,** v. 39, n. 2, p. 151-163, 2012. DOI: https://doi.org/10.1016/j.cor.2011.03.006.

KILBY, P.; PROSSER, P.; SHAW, P. **Dynamic VRPs**: a study of scenarios. Report APES-06-1998. University of Strathclyde Technical Report, v. 1, n. 11, 1998.

KOÇ, Ç.; BEKTAŞ, T.; JABALI, O.; LAPORTE, G. Thirty years of heterogeneous vehicle routing. **European Journal of Operational Research,** v. 249, n. 1, p. 1-21, 2016. DOI: https://doi.org/10.1016/j.ejor.2015.07.020.

LACKNER, A. ***Dynamische Tourenplanung mit ausgewählten metaheuristiken:*** *eine untersuchung am beispiel des kapazitätsrestriktiven dynamischen tourenplanungsproblems mit zeitfenstern. Göttingen: Cuvillier Verlag, 2004. (Göttinger Wirtschaftsinformatik).*

LARSEN, A. **The dynamic vehicle routing problem.** 2000. Ph.D Thesis (Doctorate in Mathematical Modelling) – Technical University of Denmark, Lyngby, 2000. Available at: https://backend.orbit.dtu.dk/ws/portalfiles/portal/5261816. Accessed on: 11 june 2024.

LARSEN, A.; MADSEN, O.; SOLOMON, M. Partially dynamic vehicle routing: models and algorithms. **Journal of the Operational Research Society,** v. 53, n. 6, p. 637-646, 2002. DOI: https://doi.org/10.1057/palgrave.jors.2601352.

LENSTRA, J. K.; KAN, A. H. G. R. Complexity of vehicle routing and scheduling problems. **Networks,** v. 11, p. 221-227, 1981. DOI: https://doi.org/10.1002/net.3230110211.

LOGGIBUD. **loggiBUD: Loggi Benchmark for Urban Deliveries.** 2021. GitHub repository. Available at: https://github.com/loggi/loggibud. Accessed on: 11 june 2024.

MARINAKIS, Y.; MARINAKI, M. A hybrid genetic–particle swarm optimization algorithm for the vehicle routing problem. **Expert Systems with Applications,** v. 37, n. 2, p. 1446-1455, 2010. DOI: https://doi.org/10.1016/j.eswa.2009.06.085.

MARINAKIS, Y.; MIGDALAS, A.; PARDALOS, P. M. Multiple phase neighborhood search: GRASP based on Lagrangean relaxation, random backtracking Lin-Kernighan and path relinking for the TSP. **Journal of Combinatorial Optimization,** v. 17, n. 2, p. 134-156, 2009. DOI: https://doi.org/10.1007/s10878-007-9104-2.

MILLER, C. E.; TUCKER, A. W.; ZEMLIN, R. A. Integer programming formulation of traveling salesman problems. **Journal of the ACM (JACM)**, v. 7, n. 4, p. 326-329, 1960. DOI: https://doi.org/10.1145/321043.321046.

MONTEMANNI, R.; GAMBARDELLA, L. M.; RIZZOLI, A. E.; DONATI, A. V. Ant colony system for a dynamic vehicle routing problem. **Journal of Combinatorial Optimization,** v. 10, n. 4, p. 327-343, 2005. DOI: https://doi.org/10.1007/s10878-005-4922-6.

NECULA, R.; BREABAN, M.; RASCHIP, M. Tackling dynamic vehicle routing problem with time windows by means of ant colony system. In: 2017 IEEE CONGRESS ON EVOLUTIONARY COMPUTATION (CEC), 2017, Donostia. **Proceedings […]**. Donostia: IEEE, 2017. p. 2480-2487. DOI: https://doi.org/10.1109/CEC.2017.7969606.

PILLAC, V.; GENDREAU, M.; GUÉRET, C.; MEDAGLIA, A. L. A review of dynamic vehicle routing problems. **European Journal of Operational Research,** v. 225, n. 1, p. 1-11, 2013. DOI: https://doi.org/10.1016/j.ejor.2012.08.015.

PILLAC, V.; GUÉRET, C.; MEDAGLIA, A. **A fast re-optimization approach for dynamic vehicle routing.** Research Report, Ecole des Mines de Nantes, 2012. Available at: https://hal.science/hal-00739782. Accessed on: 11 june 2024.

PSARAFTIS, H. N. Dynamic vehicle routing problems. In: GOLDEN, B. L.; ASSAD, A. A. (ed.). **Vehicle routing**: methods and studies. North Holland: Elsevier, v. 16, p. 223-248, 1988.

RALPHS, T. K.; KOPMAN, L.; PULLEYBLANK, W. R.; TROTTER, L. E. On the capacitated vehicle routing problem. **Mathematical Programming,** v. 94, n. 2, p. 343-359, 2003. DOI: https://doi.org/10.1007/s10107-002-0323-0.

RESENDE, M. G. C.; RIBEIRO, C. C. Greedy randomized adaptive search procedures. *In*: GLOVER, F.; KOCHENBERGER, G. A. (Eds.). **Handbook of metaheuristics.** Boston: Springer, 2003. p. 219-249. DOI: https://doi.org/10.1007/0-306-48056-5_8.

RIOS, B. H. O.; XAVIER, E. C.; MIYAZAWA, F. K.; AMORIM, P.; CURCIO, E.; SANTOS, M. J. Recent dynamic vehicle routing problems: a survey. **Computers & Industrial Engineering,** v. 160, 107604, 2021. DOI: https://doi.org/10.1016/j.cie.2021.107604.

SILVA JÚNIOR, O. S.; LEAL, J. E.; REIMANN, M. A multiple ant colony system with random variable neighborhood descent for the dynamic vehicle routing problem with time windows. **Soft Computing,** v. 25, n. 4, p. 2935-2948, 2021. DOI: https://doi.org/10.1007/s00500-020-05350-4.

SOLOMON, M. M. Algorithms for the vehicle routing and scheduling problems with time window constraints**. Operations Research,** v. 35, n. 2, p. 254-265, 1987.

VAN VEEN, B.; EMMERICH, M.; YANG, Z.; BÄCK, T.; KOK, J. Ant colony algorithms for the dynamic vehicle routing problem with time windows. *In*: VICENTE, J. M. F.; ÁLVAREZ SÁNCHEZ, J. R.; LÓPEZ, F. P.; MOREO, F. J. T. (eds.) **Natural and artificial computation in engineering and medical applications.** Berlin: Springer, 2013. p. 1-10. DOI: https://doi.org/10.1007/978-3-642-38622-0_1.

YANG, Z.; VAN OSTA, J.-P.; VAN VEEN, B.; VAN KREVELEN, R.; VAN KLAVEREN, R.; STAM, A.; KOK, J.; BÄCK, T.; EMMERICH, M. Dynamic vehicle routing with time windows in theory and practice. **Natural Computing,** v. 16, n. 1, p. 119-134, 2017. DOI: https://doi.org/10.1007/s11047-016-9550-9.