Journal of Global Optimization manuscript No.
(will be inserted by the editor)

Cooperative versus non-cooperative parallel variable neighborhood search strategies: a case study on the capacitated vehicle routing problem

Panagiotis Kalatzantonakis · Angelo Sifaleras · Nikolaos Samaras

Received: date / Accepted: date

Abstract The Capacitated Vehicle Routing Problem (CVRP) is a well-known NP-hard combinatorial optimization problem with numerous real-world applications in logistics. In this work, we present a literature review with recent successful parallel implementations of Variable Neighborhood Search regarding different variants of vehicle routing problems. We conduct an experimental study for the CVRP using well-known benchmark instances, and we present and investigate three parallelization strategies that coordinate the communication of the multiple processors. We experimentally evaluate a non-cooperative and two novel cooperation models, the managed cooperative and the parameterized cooperative strategies. Our results constitute a first proof-of-concept for the benefits of this new self-adaptive parameterized cooperative approach, especially in computationally hard instances.

Keywords Variable Neighborhood Search, Parallel Computing, Vehicle Routing Problem, Self-Adaptive Mechanism

P. Kalatzantonakis
Department of Applied Informatics
University of Macedonia, School of Information Sciences
156 Egnatias Str., 54636 Thessaloniki, Greece
E-mail: mai18019@uom.edu.gr

A. Sifaleras
Department of Applied Informatics
University of Macedonia, School of Information Sciences
156 Egnatias Str., 54636 Thessaloniki, Greece
Tel.: +30-2310-891884 / Fax: +30-2310-891881
E-mail: sifalera@uom.gr

N. Samaras
Department of Applied Informatics
University of Macedonia, School of Information Sciences
156 Egnatias Str., 54636 Thessaloniki, Greece
Tel.: +30-2310-891866 / Fax: +30-2310-891879
E-mail: samaras@uom.gr

1 Introduction

Combinatorial or discrete optimization problems have significant importance to many industrial applications, with a vast number of uses and can be described as the effort to find an optimal solution from a finite number of alternative solutions. The Vehicle Routing Problem (VRP) is one of the most well-studied combinatorial optimization problems, and it emerges when one seeks an optimal route for a fleet of vehicles to accommodate a set of clients, given a set of constraints.

The CVRP, which was initially introduced by Dantzig and Ramser in 1959 [14] is a variation of VRP, in which a fleet of homogenous delivery vehicles of limited carrying capacity must service known client demands from a depot, at a minimum transit cost. The CVRP is an NP-hard problem with significant impact on the fields of transportation, distribution, and logistics since transportation is usually a significant component of the cost of a product. One of the major concerns of the industry has always been the minimization of the product cost, essential both for the achievement of a more substantial profit as for the maintenance of a vantage over the competition. In addition, a growing interest in reducing the environmental impact of their products and services is also cultivated among companies, thus creating a trend towards a greener management of the modern supply chain [32].

Finding an optimal solution for the CVRP is generally a computationally difficult problem. Exact algorithms exist, but those are not considered efficient due to the computational nature of all NP-hard problems, which leads to markedly long periods of computation time when in the process of solving problems with many clients. Many different exact algorithms are proposed in the literature for the CVRP [3]. Laporte and Nobert [22] gave a survey covering early exact methods for the CVRP. Toth and Vigo [35] delivered a complete overview of exact methods for the CVRP. For more insight into the literature of exact methods, the interested reader is referred to the work by Cordeau et al. [8].

Metaheuristics can find applications in complex problem solving in a wide array of fields ranging from finance to production management and engineering and are part of a very diverse group of algorithms including tabu search, genetic algorithms, greedy randomized adaptive search, ant colony optimization, simulated annealing, Variable Neighborhood Search (VNS), scatter search, and others [30]. As suggested by the Greek prefix “meta”, metaheuristics are higher-level heuristics, in contrast with problem-specific heuristics.

VNS is a metaheuristic proposed by Mladenović and Hansen in 1997 [25]. VNS and its variants are widely recognized as a very efficient approach for many hard optimization problems. Its strategy consists in the exploration of dynamically changing neighborhoods for a given solution. A systematic change of neighborhood happens both within a descent phase in order to find a local optimum, as well as in a perturbation phase so as to get out of the corresponding valley [20].

The purpose of this paper is threefold: Firstly, to survey recent successful parallel implementations of VNS regarding different variants of the VRP. Secondly, to present three parallel VNS methods using the General VNS variant to tackle the CVRP, in which we examine different approaches of exchanging solutions among parallel executions. Thirdly, to study how the level of cooperation can affect the performance.

The remainder of this paper is organized as follows: We illustrate some basic concepts and define a CVRP formulation in Section 2. In Section 3, we present related works in parallel approaches of VNS on VRP variants. Next, in Sections 4 and 5 we present the GVNS method and three parallel GVNS models for the solution of the CVRP and discuss their benefits and drawbacks. The results of this study are presented in 6. Finally, the conclusions and prospects are summarized in Section 7.

2 CVRP mathematical formulation

Several mathematical formulations have been proposed for the VRP [34], such as the Vehicle Flow Formulation (VFF), the Commodity Flow Formulation (CFF), and the Set Partitioning Formulation (SPF).

The simplest version of the VRP is the CVRP. In this work, we use the VFF to formulate the CVRP under the assumption that, the number of vehicles is unlimited and the goal is to obtain a solution that minimizes the total travel cost. VFF uses discrete variables associated with each arc that, count the number of times that the edge is traversed by a vehicle. The number of variables is polynomially bounded, and the number of constraints is exponential [35].

Let $G = (V, E)$ be an undirected weighted graph where $V = \{0, n\}$ is the set of vertices ($i = 0$ the depot, and $i = 1, \ldots, n$ the clients). By E we denote the set of edges (i.e., the roadways between the clients) and K is the available number of vehicles. Each edge (i, j) is associated with a non-negative cost c_{ij} and a binary variable x_{ij} (whether it is traversed or not). The CVRP formulation is as follows:

$$\min \sum_{i=0}^{n} \sum_{j=0}^{n} c_{ij} x_{ij}$$

subject to:

$$\sum_{i \in V} x_{ij} = 1 \quad \forall j \in V \backslash \{0\} \tag{1}$$

$$\sum_{j \in V} x_{ij} = 1 \quad \forall i \in V \backslash \{0\} \tag{2}$$

$$\sum_{i \in V} x_{i0} = K \tag{3}$$

$$\sum_{j \in V} x_{0j} = K \tag{4}$$

$$\sum_{i \notin S} \sum_{j \in S} x_{ij} \geq r(S) \quad \forall S \subseteq V \backslash \{0\}, S \neq 0 \tag{5}$$

$$x_{ij} \in (0, 1) \quad \forall i, j \in V \tag{6}$$

Constraints 1 and 2 impose the in-degree and out-degree constraints so that precisely one edge enters and leaves each vertex associated with a client, respectively.

Constraints 3 and 4 impose client degree requirements for the depot. Constraint 5 is called the Capacity-Cut Constraint (CCC) and imposes both the connectivity of the solution and the vehicle capacity requirements. Capacity-Cut Constraints (CCCs) specify that each cut specified by vertex sets (S, S') is crossed by a number of edges not less than $r(S)$, where $r(S)$ stands for the minimum number of vehicles needed to serve set S. Constraint 6 describes the binary nature of variables x_{ij}, in other words, value 1 is assigned if edge $(i, j) \in E$ belongs to the optimal solution and 0 otherwise.

3 Related Work in Parallel Strategies of VNS for VRP variants

Recently, the rise of multi-core processors along with cluster and grid computing popularity attracted several researchers to design and develop new parallel metaheuristics. Parallel processing is considered to be a cost-effective method for the fast solution of computationally hard problems. Except for the availability of computing resources, the use of parallel metaheuristics has also increased since, as it has been demonstrated by Bouthillier and Crainic in [23], such parallel algorithms are capable of both speeding-up the search and improving the robustness and the quality of the solutions obtained.

The VNS algorithm is a trajectory-based metaheuristic. The evaluation of constraints and objective components for each solution in the neighborhood in these kinds of algorithms is an embarrassingly parallel task [31]. Thus, several strategies for parallelizing a VNS algorithm have been already proposed and analyzed in the literature [10,15,16,27].

Generally, contributions to the VRP using parallel metaheuristics published before the year 2000 are not as numerous as for other combinatorial optimization problems [11]. Besides the fact that, VNS consists an embarrassingly parallel task, one is surprised to realize that even fewer works have targeted the VRP using VNS. For more insight into the literature of solving hte VRP using additional metaheuristics methods, the interested reader is referred to the work by Crainic [9].

Polacek et al. in 2008 [28] worked on producing parallel algorithms for the multi-depot Vehicle Routing Problem with Time Windows (VRPTW). The authors provided two approaches for a parallel VNS by changing the configuration between cooperation and asynchronous exchanges through a central memory. In the first approach, the VNS threads communicated their best solutions to the central memory (serving as a “master”) at preset regular intervals (number of iterations). The objective was to reproduce the way the sequential method works, only faster. The second approach used full VNS threads searching through a limited number of neighborhoods. The VNS threads collaborated through exchanges of best solutions through the central memory. Each VNS thread sent its best solution, and as soon as the overall best was improved, it was communicated to all threads. The performance was satisfying, but the second, full cooperation method performed best, notably due to its higher adaptability to the problem instance. Both cooperation algorithms displayed satisfying run-time scalability. Specifically, utilizing 32 search threads, the runtime was reduced from 48.7 to 1.7 hours, also for all 20 MDVRFTW instances the best-known solution was obtained, and in 11 cases new best solutions were found.

Coelho et al. in 2012 [7] proposed an approach for the Single Vehicle Routing Problem with Deliveries and Selective Pickups based on a Hybrid General VNS (HGVNS). The parallel solution is heterogeneous since it is implemented in CPU and GPU and is called HP-HGVNS. The incentive behind this approach is based on the decomposition and distribution of the HGVNS workload to the CPU and the GPU. Since the most expensive part of GVNS is to perform the best improvement local search in the VND method, the GPU is utilized in order to take advantage of the fine-grain parallelism a GPU can offer. The CPU, on the other hand, is used for creating an initial solution, choosing the operation to be performed, and checking for the best solution generated by the GPU. Concerning performance, the 1Or-Opt and 2Or-Opt local search scored a 27.26 and 43.75 speedup respectively, while the Swap local search had the lowest speedup values, from 1.91 to 14.36. HP-HGVNS achieved an average speedup from 2.73 to 16.23. Dissecting the quality of the results, HGVNS produced better solutions than the best results obtained in literature. Other recent works with hybrid CPU-GPU parallel VNS methods include [1].

Polat in 2017 [29], proposed a parallel metaheuristic algorithm for the Vehicle Routing Problem with Divisible Deliveries and Pickups. In this approach, the author used asynchronous cooperation with a centralized information exchange strategy for the parallelization of the VNS (CVNS). Based on the Crainic and Hail taxonomy [11], this approach was classified as pC/C/MPSS. In this strategy, medium-grained parallelization was used. Data exchange was only performed after all local search procedures were completed. Examining the quality of the results, solutions were improved or reached the best-known solutions for 179 out of 220 benchmark instances while using less computational time.

4 The proposed GVNS for solving CVRP

There are three main steps in parallel algorithm design: the decomposition of the workload, the distribution of the tasks to the available processors and perhaps the most crucial step, the coordination strategy of the components of the parallel program. In our approach, we materialize three scenarios. Our intent is to study how the level of coordination between the components of the parallel VNS affects performance.

To describe parallel metaheuristic strategies, we adopt the classification of Crainic and Hail [11] that generalizes that of Crainic et al. [12]. The classification consists of three dimensions that indicate how the global process is controlled (search control cardinality), how information is exchanged among processes (search control and communication), and the variety of solution methods involved in the search for solutions (search strategies) [9].

4.1 Initial Solution

The Savings Algorithm of Clarke and Wright [6] was used to construct the initial solution for all models, as it is one of the best approaches to solve the VRP. The algorithm starts with one route per client, and in each iteration, the number of routes is reduced by merging two routes that give max savings (S) (Figures 1). The classical Savings formula (S_{ij}) is depicted in Equation 7.

$$S_{ij} = dist(D, i) + dist(D, j) - dist(i, j) \tag{7}$$

In this equation, i and j represent a pair of clients, D is the depot, and the function $dist(x, y)$ calculates the distance between two given points.

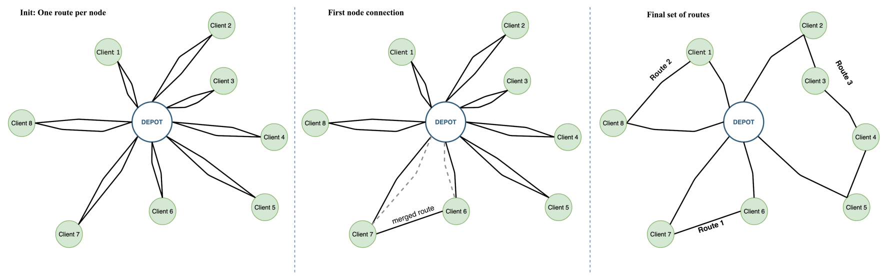

Fig. 1 Savings algorithm. Start, first iteration, and last iteration.

4.2 General Variable Neighborhood Search (GVNS)

All models are using an identical neighborhood structure, consisting of three widely used inter and intra-route operators (Figure 2):

- 2-opt (Intra-route) exchanging two edges with other two edges, in order to form a tour again.
- Swap (Inter-route) exchanging nodes between routes.
- Relocate (Inter-route) moving nodes to other routes.

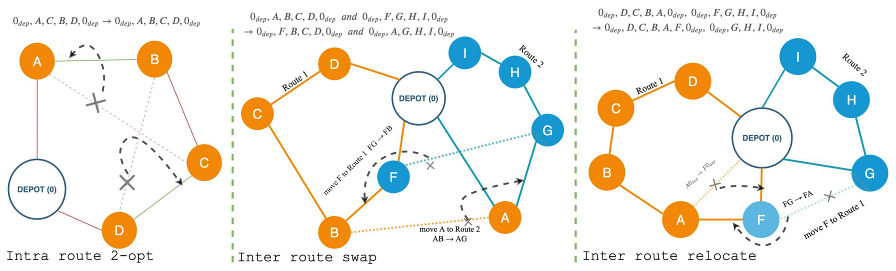
$0_{dep}, A, C, B, D, 0_{dep} \rightarrow 0_{dep}, A, B, C, D, 0_{dep}$
$0_{dep}, A, B, C, D, 0_{dep}$ and $0_{dep}, F, G, H, I, 0_{dep}$
$\rightarrow 0_{dep}, F, B, C, D, 0_{dep}$ and $0_{dep}, A, G, H, I, 0_{dep}$
$0_{dep}, D, C, B, A, 0_{dep}, 0_{dep}, F, G, H, I, 0_{dep}$
$\rightarrow 0_{dep}, D, C, B, A, F, 0_{dep}, 0_{dep}, G, H, I, 0_{dep}$
DEPOT (0)
Route 1
Route 2
move F to Route 1 FG → FB
move A to Route 2
AB → AG
FG → FA
move F to Route 1
Intra route 2-opt
Inter route swap
Inter route relocate

Fig. 2 Neighborhood structures.

The GVNS method (Hansen et al. [19]), described in Algorithm 1, is used to improve the initial solution given by the Clarke and Wright algorithm for all the proposed methods. The Variable Neighborhood Descent (VND) in line 5 is described in Algorithm 2. By t_{max}, k_{max}, and l_{max} we denote the maximum CPU time allowed before termination, the maximum number of shaking iterations, and the number of neighborhood structures, respectively.

Algorithm 1: GVNS

```
input  : $x$, $l_{max}$, $k_{max}$, $t_{max}$
output: $x$
1 while $t < t_{max}$ do
2     $k \leftarrow 1$;
3     while $k < k_{max}$ do
4         $x' \leftarrow Shake(x, k)$;
5         $x'' \leftarrow VND(x', l_{max})$;
6         $x, k \leftarrow NeighborhoodChange(x, x'', k)$;
7     end
8     $t \leftarrow CpuTime()$;
9 end
```

Algorithm 2: VND

```
input  : $x$, $l_{max}$
output: $x$
1 $l \leftarrow 1$;
2 while $l < l_{max}$ do
3     $x' \leftarrow \arg\min_{y \in N_{l(x)}} f(y)$;
4     $x, l \leftarrow NeighborhoodChange(x, x'', l)$;
5 end
```

4.3 Stopping conditions

Two stopping criteria have been adopted in the proposed methods. The first is the total execution time (3,600 seconds = 1 hour) and the second is the completion of the desired GVNS iterations. Only in special cases where an optimum value existed, the achievement of the optimum value was an additional stopping criterion.

5 Parallel GVNS models

5.1 Parallel GVNS - The Non-Cooperative Model (NCM)

The proposed implementation employs an island-based scheme where, every thread runs the GVNS completely isolated. Once the initial solution is generated using the Clarke and Wright algorithm, it is given to each thread. All threads are using the same search strategy (GVNS). Soon after, the GVNS algorithm begins executing without further communication between threads. As threads work independently and possible solutions are improved in each one of them, their paths diverge as the shaking procedure takes place. As the time limit is reached or the max allowed iteration count is reached, all threads terminate, and the best solution among all is reported.

Based on the classification of Crainic and Hail [11], the non-cooperative model fits into the pC/RS/SPSS classification. The notation pC stands for "poly-Control", corresponds to the Search Control cardinality and indicates that, the search control is distributed across many processes. In this model, each single thread has its own search control. The notation RS stands for "Rigid Synchronization" and it means that, little or no information exchange takes place in order to improve solutions at the same level of the communication hierarchy. The notation SPSS stands for "Same initial Point, Same search Strategy" and it indicates that, all threads had the same initial solution as a starting point and followed the same search strategy. Figure 3, depicts the execution timeline of this non-cooperative model.

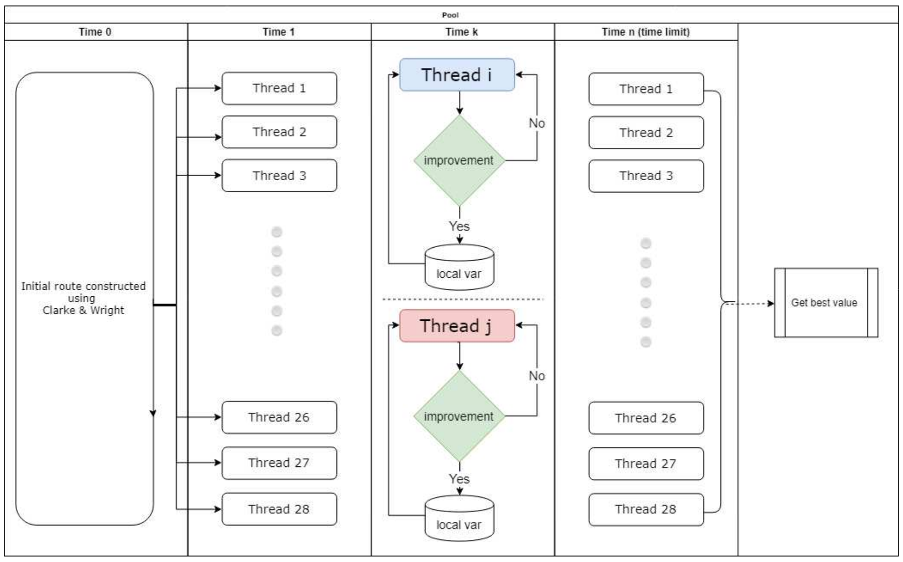

Fig. 3 The non-cooperative model execution timeline.

5.2 Parallel GVNS - Managed Cooperative Model (MCM)

The initial solution produced by the Clarke and Wright algorithm is passed to all threads, except for one, which will take up the role of the server manager, essentially being a solution warehouse. Once a single thread improves a solution previously acquired, it communicates with the server manager in order to ask if a better solution exists. In the case that, the server manager holds a better solution, the thread rejects the inferior one and continues its work with the one provided. Otherwise (i.e., if the thread holds a better solution than the server manager), the new and improved solution is stored in the server manager and the thread continues with the one it found. The server manager stores and manages the best-found solution without broadcasting any information to the other threads. Communication between the threads and the server manager is very dense at the start of the GVNS algorithm, promoting intensification, and sparse while the process continues, favoring diversification.

The cooperative model fits into the pC/C/SPSS classification. Each dimension in this taxonomy is identical to the non-cooperative model except for the C class, which stands for “Collegial” and indicates that, we extract and use only good solutions from the Manager.

The cooperative model exchanges information asynchronously, at irregular intervals dynamically determined by each process. The search process may change its current solution using a solution that is received by the Manager, and for this reason, the intensification level dynamically changes. Since no broadcasting occurs, information will not spread any further.

5.3 Parallel GVNS - Parameterized Cooperative Model (PCM)

The previous models were each at the other end of the spectrum of the communication strategy. The cooperative model favors intensification without any insight of the search space, while the non-cooperative model promotes exploration. In order to bridge the gap, a parameterized cooperative GVNS model is proposed. This time, the cooperative model will accept a solution from the server manager only if the normalized distance between two solutions is greater than a parameter θ ($distance > \theta$). The Damerau – Levenshtein distance [13] is used to compute the distance between two solutions.

This metric can be described as a search space distance rather than a solution space distance due to the fact that, the edit operations applied in the solution representation can be considered to be neighborhood operations. Figure 4 illustrates the normalized Damerau – Levenshtein distance calculation between two solutions for a CVRP instance.

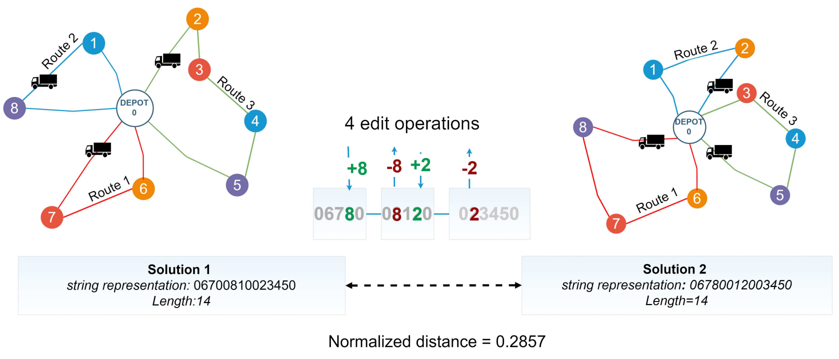

Fig. 4 Levenshtein distance for solutions that are two inter-route relocations apart.

The normalized Damerau – Levenshtein algorithm returns a float (between 0.0 and 1.0) by evaluating Equation 8.

$$Normalized\ distance = \frac{Levenshtein\ distance(sol_1, sol_2)}{maxLength(Len_1, Len_2)} \tag{8}$$

As seen in the above example of Figure 4 the solution of the CVRP is represented as a string of size $n + 2 \times k$, where $n,\ k$ are the numbers of clients and vehicles, respectively. Thus, the string length of both the first solution Len_1(“06700810023450′′) and the second solution Len_2(“06700810023450′′) is equal to $8 + 2 \times 3 = 14$, and the maximum length $max(Len_1, Len_2) = 14$. The Levenshtein distance is equal to four, since only four edit operations are needed to get Solution 2 from Solution 1. Hence, by using the above values we get the normalized distance equal to $\frac{4}{14} = 0.2857$.

The parameter θ ranges from value 1.0 (non-cooperative) to 0.0 (cooperative). In the beginning, it promotes exploration by ignoring solutions stored in the solution warehouse ($distance \leq \theta = 1$), and as the time passes, θ value gradually decreases and the parameterized cooperative method starts to accept values even with small error distances. In our tests, a cut off was selected either at the 20% of the max GVNS iterations ($niter_{\max}$) or the maximum time ($time_{\max}$) allowed. The optimal cut-off value was found by performing several tests, spanning from 10% to 40% of the max iterations or the max time. The best results for all instances were obtained around 20%. Thus, the self-adaptation of the parameter θ is based on the remaining time and GVNS iterations as depicted in Algorithm 3:

Algorithm 3: Parameter θ self-adaptation

```
input  : Current time (time), current number of iterations (niter)
output: $\theta$
1 $\theta = 1$ ;
2 if ($niter_i > 0.2 \times niter_{\max}$) $\vee$ ($currenttime > 0.2 \times time_{\max}$) then
3     $\theta = \min\big((1 - \frac{niter}{niter_{max}}), (1 - \frac{time}{time_{max}})\big)$;
4 end
```

The parameterized cooperative model fits into the pC/KC/SPSS classification. KC class stands for “Knowledge Collegial” and indicates an asynchronous communication scheme in which the contents of communications are analyzed to infer good solutions. Figure 5 describes the execution timeline of both cooperative schemes, MCM and PCM, where each horizontal line represents the execution of a single event critical for the execution flow.

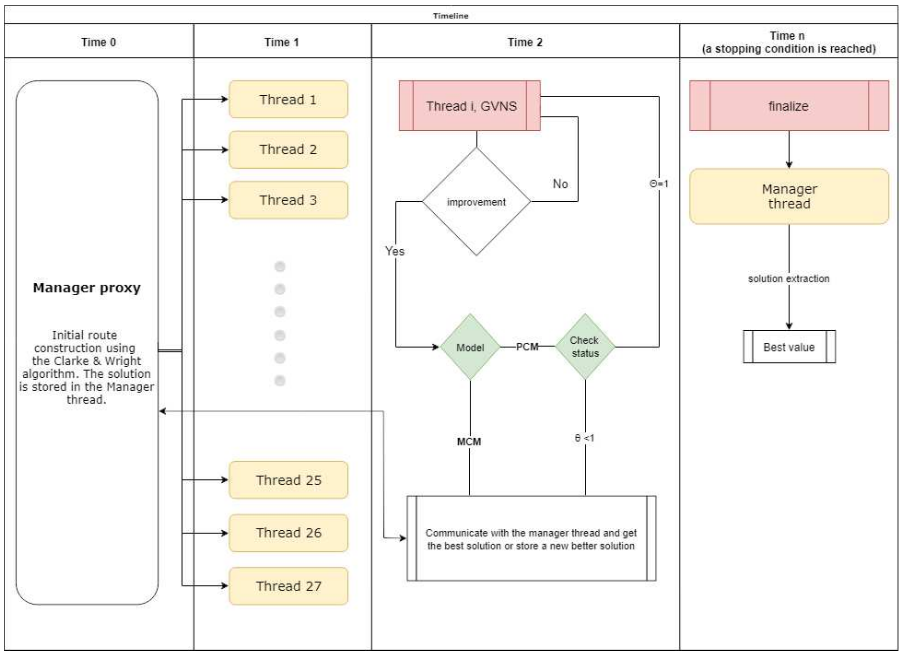
Timeline
Time 0
Time 1
Time 2
Time n
(a stopping condition is reached)
Manager proxy
Initial route construction using the Clarke & Wright algorithm. The solution is stored in the Manager thread.
Thread 1
Thread 2
Thread 3
Thread 25
Thread 26
Thread 27
Thread i, GVNS
improvement
No
Yes
Θ=1
Model
PCM
Check status
MCM
θ <1
Communicate with the manager thread and get the best solution or store a new better solution
finalize
Manager thread
solution extraction
Best value

Fig. 5 Managed and parameterized cooperation model execution timeline.

5.4 Cooperation strategies

For the needs of the proposed cooperation models, asynchronous cooperation was selected since synchronous methods have a more significant computational cost and are unaware of the findings during the search process. Since no broadcasting takes place, each thread can dynamically guide the diffusion of information, depending on its state. The connection pattern in both the proposed communication models is undetermined and adopts a less obtrusive method, where no thread gets interrupted so to display the clear benefits of communication.

In order to better manipulate the interplay between exploration and exploitation, a parameterized cooperation is proposed. This is a fine-grained cooperation method, where a process may discard the best solution collected by the Manager based on the iterations it has performed and its current status.

The asynchronous cooperation strategy followed in the managed cooperation model is novel compared to the previous works reported by Coelho et al. [7], Polacek [28], and Polat [29]. Furthermore, the strategy that follows the principle of controlled diffusion of information at the process level in the parameterized cooperative model is a completely new cooperation strategy with promising results.

5.5 Parallel Technologies

The parallelization of the GVNS method for the proposed models utilized the standard multiprocessing library of Python 3.7. In the cooperative models, instead of using shared memory, a Manager class was used. Manager objects control a server process which handles shared objects and enables other processes to manage them using proxies. Server process managers are more flexible than shared memory objects since they support arbitrary objects and direct manipulation from processes on the same or different computers over a network.

6 Computational experiments

This section reports and analyzes the computational experiments related to the efficiency of the three proposed parallel GVNS models for solving the CVRP. All the algorithms were implemented using Python 3.7. An additional speedup (~35%) was achieved using Cython. The experiments were conducted using an Intel Core i9 7940X CPU (28 threads at 3.50 GHz) and 32GB RAM. All computational tests were carried out on 76 instances of the CVRP library [36] available at `http://vrp.atd-lab.inf.puc-rio.br`.

In order to evaluate the performance of the proposed models, we have focused our analysis in two parts. Firstly, we analyze the performance based on the overall quality of solutions of all instances, and secondly, we select some representative, computationally difficult instances and check the quality of solutions in relation to this difficulty.

In the first part of this analysis, the effect of the cooperation strategy of each algorithm is studied on 65 instances from the set A and set B [2], set E [4], a subset of set M (M-n101-k10, M-n200-k16) [5] and 11 instances from the Golden set [17].

The selected 76 CVRP instances have a number of clients ranging from 32 to 360, known optimum number of vehicles, fixed vehicle capacity, and known optimal solution. Since all the 65 instances from the sets A, B, E, and M have optimal solutions, reaching the optimum value consists an additional stopping criterion. In order to have a fair comparison and evaluate the performance and the quality of the solutions, all the experiments were repeated ten times, and an average value was reported.

In the results shown in Table 1 (on sets A, B, E, and M), essential differences among the compared methods can be observed. The first row of the Table 1 refers to the average error of each parallel model. The parameterized model surfaces as the best strategy, producing better solutions. In the second row, we report the average CPU time needed to complete 300 GVNS iterations. The non-communicative model is the fastest method, completing the task after 76.880 seconds. A theoretical explanation of this speedup in the non-cooperative process is that communication deprives the search procedure of some CPU cycles. In the third row, we list the number of instances (with known optimal solution) in which the model found the optimal solution in less than 300 iterations. The full benefits of communication are not particularly evident as the non-cooperative model produces comparable results.

Table 1 Comparison of the three GVNS parallel variants at 300 GVNS iterations

	NCM	MCM	PCM
Average error	0.837%	1.260%	0.729%
Average CPU time (secs)	76.880	92.100	88.640
Number of optimal solutions	21	16	21

At first glance, even though the parameterized model produces better results, the complete absence of communication between processes appears to yield promising results. The non-cooperative model is probing a much more significant portion of the search space with the hope of finding promising solutions; thus having a lower probability of being trapped in a local optimum.

In Figure 6, we plot the average error of each parallel model on instances with smaller size and known optimal value. The instances are grouped by their set (sets A, B, E, and M). It should be noted that, all algorithms achieved the optimum value for 14 out of 65 instances; thus, these instances have been omitted. An emerging pattern from this data visualization indicates that, the gap between the three models closes, especially when dealing with computationally harder instances.

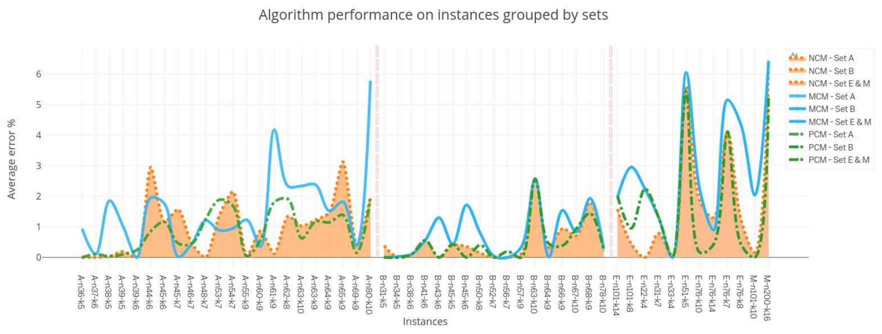

Fig. 6 Algorithm performance on instances grouped by sets. Methods supporting communication close the performance gap on computationally difficult instances.

The assumption of normality was tested with the Kolmogorov-Smirnov test. The results indicate that, the distribution of the data deviates significantly from the normal distribution; therefore, the analysis proceeded with non-parametric tests. For inferential statistics, the Friedman test was applied to the performance results collected by the execution of the three parallel models. The obtained p-value was almost zero ($3 \times * 10^{-11}$), thus showing that there is enough statistical evidence to consider the three algorithms different. A typical significance level of $\alpha = 0.05$ was regarded as the threshold for rejecting the null hypothesis.

Having established the diversity between the models, the observed trend was further analyzed. The instances were classified into four categories (i.e., easy, medium, hard, and very hard) mainly according to the size of each instance (number of clients) and the number of routes (tracks). However, other important factors (i.e., depot positioning, high coefficient of variation in demands, and formation of clusters due to the clients' positions) also play an essential role in this classification. Thus, the remaining 51 instances, based on their computational difficulty, formed the following four categories as follows:

– Easy (A-n36-k5, A-n37-k6, A-n38-k5, A-n39-k5, A-n39-k6, A-n55-k9, B-n31-k5, B-n34-k5, B-n45-k6, B-n52-k7, B-n57-k9, E-n22-k4, E-n33-k4),
– Medium (A-n44-k6, A-n45-k6, A-n45-k7, A-n46-k7, A-n48-k7, A-n53-k7, A-n54-k7, A-n60-k9, A-n61-k9, A-n62-k8, A-n63-k10, A-n63-k9, A-n69-k9, B-n38-k6, B-n41-k6, B-n43-k6, B-n45-k5, B-n50-k8, B-n56-k7, B-n64-k9, B-n66-k9, B-n78-k10, E-n31-k7, E-n51-k5),
– Hard (A-n64-k9, A-n65-k9, A-n80-k10, B-n63-k10, B-n67-k10, B-n68-k9, E-n76-k7, E-n76-k8, E-n76-k10, E-n76-k14, E-n101-k14, E-n101-k8, M-n101-k10, M-n200-k16)
– Very hard, with unknown optimum values (Golden 1, Golden 5, Golden 9, Golden 10, Golden 11, Golden 13, Golden 14, Golden 15, Golden 17, Golden 18, Golden 19)

Table 2 summarizes the results and makes it easier for one to observe the effect of communication on instances with greater difficulty.

Table 2 Comparison of the three parallel GVNS algorithms on all instances classified by difficulty.

Instance Class	Model	Avg. Error %	GVNS iterations
Easy	NCM	0.042%	300
	MCM	0.351%	300
	PCM	0.103%	300
Medium	NCM	0.990%	300
	MCM	1.731%	300
	PCM	1.059%	300
Hard	NCM	2.043%	300
	MCM	3.386%	300
	PCM	1.978%	300
Very hard	NCM	9.620%	500
	MCM	9.380%	500
	PCM	8.270%	500

The well-known Wilcoxon non-parametric test was performed for pairwise comparisons within each category, in order to analyze whether the four parallel models produce different results or not. In order to have a fair comparison and evaluate the performance and the quality of the solutions, each parallel algorithm was executed ten times, with each execution thread reaching either 300 GVNS iterations (for sets A, B, E, and M) or 500 GVNS iterations (for the Golden set). The results are presented in Table 3.

Table 3 Results from pairwise Wilcoxon tests, where the groups of algorithms that differ significantly are underlined.

Parallel GVNS models	Easy p-value	Medium p-value	Hard p-value	Golden p-value
NCM vs MCM	0.014	0.089	0.021	0.678
NCM vs PCM	0.944	0.409	0.002	0.000
MCM vs PCM	0.032	0.014	0.032	0.009
Best method	NCM	PCM, NCM	PCM	PCM

The resulting p-values obtained from the Easy class analysis demonstrate that, the managed cooperative model differs significantly from both the non-cooperative and the parameterized model when solving easier instances. The non-cooperative method delivers better results in this category. The unfiltered cooperation method performed poorly while on the other hand, the parameterized cooperation yielded better results but still didn't exceed the solution quality of the non-cooperative method, and according to the Wilcoxon test, the solutions obtained from the parameterized cooperation and the non-cooperative method do not represent two different populations.

When solving instances characterized as of medium difficulty, the parameterized cooperation method managed to produce almost the same quality of solutions as the non-cooperative method. All models fail to differentiate between one another according to the Wilcoxon test.

In the more difficult instance category classified as Hard, the parameterized cooperation model produced superior solutions regarding quality. This time, the non-cooperative algorithm performed worse than the managed cooperative model. At this point, all algorithms produced solutions that represent different populations according to the Wilcoxon test.

Furthermore, when analyzing the most difficult instances from the Golden set, unfiltered communication seems to have no statistically significant differences when compared to the non-cooperative model, even though it generally leads to better solutions. In particular, after ten repeats the managed cooperation model had an average improvement of 2.494%, but the Wilcoxon test returned a p-value of 0.678 which indicates that, there are no statistically significant differences between the two methods.

The PCM had an average improvement of 14.033% at the Golden subset, and 3.181% at the Hard class when compared to the NCM (Table 2). The performance of the three models on very hard instances, is depicted in Figure 7. According to the Wilcoxon test, there are statistically significant differences between the other models (Table 3).

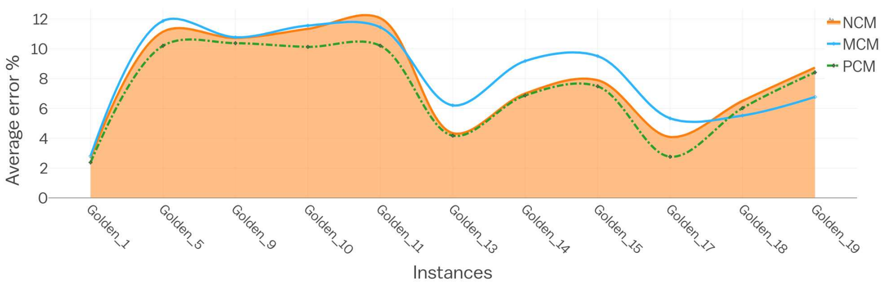

Fig. 7 Average error after ten repeats of 500 GVNS iterations in 11 Golden instances.

In conclusion, information sharing seems to outperform the independent search method and is a valuable strategy for tackling hard instances. The parameterized model is achieving better convergence rate and solution quality by reducing the level of interaction between the threads. Sparse information sharing in the parameterized cooperation model (Figure 8) leads to a better performance than dense solution exchange (Figure 9).

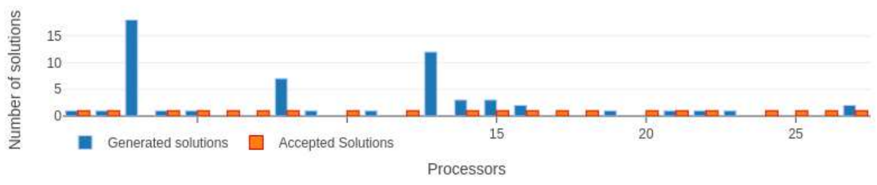

Fig. 8 Solution exchange in parameterized model is sparse. This leads to better exploration through partial thread isolation.

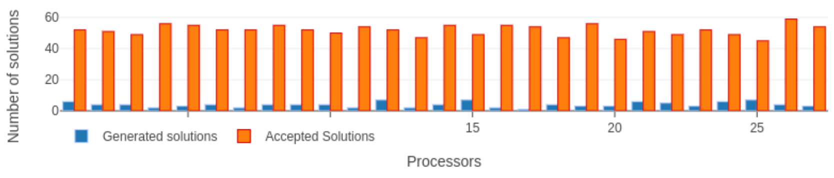

Fig. 9 Solution exchange in managed cooperation is dense. Generation of new global solutions stabilizes in a lower plateau.

In order to determine the communication overhead between processors, the total execution time was measured. The time window with the most intensive information sharing is located at the first GVNS iterations. Separately timing the messages passed from the processors to the solution warehouse at irregular intervals, dynamically determined by each process, consists a non-trivial task using Python multithreading library. Therefore, a total time execution measurement should reflect the communication overhead with better accuracy. To get this measurement, ten GVNS iterations for the three models were executed and this experiment was repeated 40 times. The additional, average, execution time of MCM compared to NCM (no communication overhead) is depicted in the third column of Table 4.

Table 4 MCM model performance and communication overhead

Set	Avg. relative error	Avg. additional time due to communication overhead
Set A	1.289%	11.460%
Set B	3.135%	15.138%
Set E	5.101%	5.077%
Set M	15.269%	10.713%
Golden	11.959%	9.623%

By comparing the MCM model with the NCM model we find that, on average, communication requires an extra 10% in overall execution time (Figure 10). For example, as seen in Table 4, upon completing the first ten GVNS iterations in the Golden set, the MCM model had a completion time of 227.969 seconds (9.623% more than NCM). Thus, it is safe to assume that, the communication overhead for the PCM model lies somewhere in between that of the NCM model and the MCM model, since the communication is far more sparse while delivering better results with less communication.

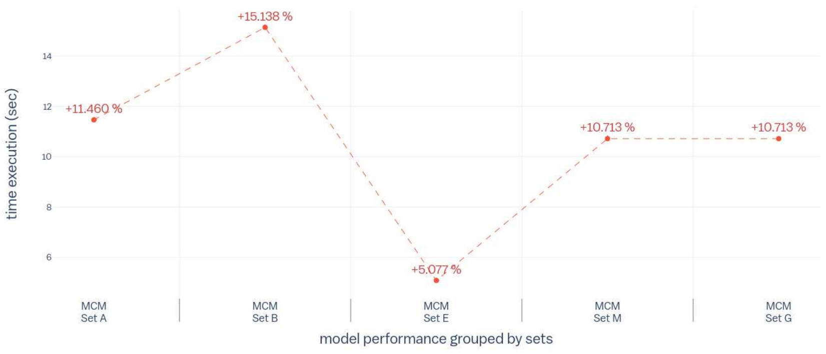

Fig. 10 Communication overhead expressed in total time execution

The best performance of the three models at 800 GVNS iterations after ten repeats are depicted in Table 5. For the Golden subset, 500 GVNS iterations were executed. All instances completed under 3,600 seconds.

Table 5 Best performance of the three models

Instance name CVRPLIB	GVNS iterations	Best-known solution	Initial solution by Clarke & Wright	Solution by GVNS	Relative error (%)	Model with the best solution
A-n32-k5	**800**	787.0819	1595.1053	787.0819	0	All models
A-n33-k5	**800**	662.1101	1160.8442	662.1101	0	All models
A-n33-k6	**800**	742.6932	1214.5044	742.6932	0	All models
A-n34-k5	**800**	780.9361	1750.1267	780.9361	0	All models
A-n36-k5	**800**	802.1318	1347.7531	802.1318	0	NCM,PCM
A-n37-k5	**800**	672.4652	1615.9513	672.4652	0	All models
A-n37-k6	**800**	950.8522	1467.1614	950.8522	0	All models
A-n38-k5	**800**	734.1846	1688.6356	734.1846	0	All models
A-n39-k5	**800**	828.9891	1328.892	829.4375	0	All models
A-n39-k6	**800**	833.2046	1665.7231	833.2046	0	All models
A-n44-k6	**800**	939.3346	1418.0345	939.3346	0	NCM, PCM
A-n45-k6	**800**	944.8763	1807.3193	944.8763	0	NCM, PCM
A-n45-k7	**800**	1146.9089	1683.5652	1146.9089	0	NCM
A-n46-k7	**800**	917.9073	1720.1586	917.9073	0	NCM, PCM
A-n48-k7	**800**	1074.3378	1805.8584	1074.3378	0	All models
A-n53-k7	**800**	1012.3255	2348.7395	1012.2490	0	NCM, PCM
A-n54-k7	**800**	1171.7843	2396.2949	1183.5745	0.3996	NCM
A-n55-k9	**800**	1074.4636	2218.6536	1074.4636	0	NCM, PCM
A-n60-k9	**800**	1355.7992	2839.9319	1355.7992	0	MCM,PCM
A-n61-k9	**800**	1039.0784	1989.6459	1040.3088	0.1182	MCM,PCM
A-n62-k8	**800**	1294.2821	2548.6326	1313.0479	0.7362	PCM
A-n63-k10	**800**	1313.7294	2656.6606	1316.8519	0.2371	PCM
A-n63-k9	**800**	1622.1446	2715.2231	1633.9357	0.7216	PCM
A-n64-k9	**800**	1400.832	2402.5803	1403.2783	0.1743	MCM
A-n65-k9	**800**	1181.6874	2743.033	1181.6874	0	PCM
A-n69-k9	**800**	1165.9945	2630.5024	1170.5383	0.3881	MCM,PCM
A-n80-k10	**800**	1766.4999	3167.9758	1790.4506	1.3376	PCM
B-n31-k5	**800**	676.0884	1870.3339	676.0884	0	All models
B-n34-k5	**800**	790.1838	1272.2225	790.1838	0	All models
B-n35-k5	**800**	956.2941	1918.5958	956.2941	0	All models
B-n38-k6	**800**	807.8788	1780.4048	807.8788	0	All models
B-n39-k5	**800**	553.1564	1165.3777	553.1564	0	All models
B-n41-k6	**800**	833.8063	1436.605	833.8063	0	All models
B-n43-k6	**800**	746.9838	1137.7297	746.9838	0	All models
B-n44-k7	**800**	914.9648	1484.5364	914.9648	0	All models
B-n45-k5	**800**	754.4388	1690.8717	754.4388	0	All models
B-n45-k6	**800**	680.4379	1449.5807	680.4379	0	NCM ,PCM
B-n50-k7	**800**	744.228	1687.9685	744.228	0	All models
B-n50-k8	**800**	1321.5236	1743.5549	1319.5033	0	PCM
B-n51-k7	**800**	1019.3251	2995.2715	1022.6471	0	All models
B-n52-k7	**800**	749.9697	1859.6626	749.9697	0	All models
B-n56-k7	**800**	712.9161	1714.6548	712.9161	0	NCM,PCM
B-n57-k7	**800**	1143.9458	2814.8279	1143.3284	0	All models
B-n57-k9	**800**	1603.3706	2676.2986	1603.3706	0	All models
B-n63-k10	**800**	1501.2707	2125.509	1506.8827	0.3724	MCM
B-n64-k9	**800**	869.3157	2688.2004	869.3157	0	All models
B-n66-k9	**800**	1325.355	3410.4368	1327.4408	0.1571	PCM
B-n67-k10	**800**	1039.3589	1727.8862	1039.3589	0	PCM
B-n68-k9	**800**	1278.2107	2303.6853	1283.8157	0.4365	MCM
B-n78-k10	**800**	1229.2734	3301.4128	1232.6011	0.2699	PCM
E-n13-k4	**800**	247	371	247.0000	0	All models
E-n22-k4	**800**	375.2798	660.4315	375.2798	0	All models
E-n23-k3	**800**	568.5625	1305.4011	568.5625	0	All models
E-n30-k3	**800**	538.7947	1353.3146	538.7947	0	All models
E-n31-k7	**800**	379	1036	379.0000	0	All models
E-n33-k4	**800**	837.6716	1234.9409	837.6716	0	All models
E-n51-k5	**800**	524.9442	1440.7513	524.9442	0	NCM
E-n76-k7	**800**	666.8325	2069.3508	697.7697	2.26	PCM
E-n76-k8	**800**	740.6554	2067.9832	740.6554	0	PCM
E-n76-k14	**800**	1026.7063	2355.0591	1026.706	0	PCM
E-n76-k10	**800**	837.3556	2431.7793	838.9785	0.1934	PCM
E-n101-k14	**800**	1082.6501	3000.6836	1094.6382	1.0951	PCM
E-n101-k8	**800**	826.908	2453.4512	832.7041	0.6960	PCM
M-n101-k10	**800**	819.5575	2482.2693	819.5575	0	PCM
M-n200-k16	**800**	1274	5038.9443	1331.2376	2.7471	PCM
Golden 1*	**500**	5627.54	23137.2109	5742.6367	2.0751	PCM
Golden 5*	**500**	6460.98	29397.4941	7194.6733	10.1977	PCM
Golden 9*	**500**	585.43	1614.6719	646.8176	10.375	PCM
Golden 10*	**500**	741.56	2034.4878	827.7824	11.1288	PCM
Golden 11*	**500**	918.45	2970.5544	1029.8816	11.4432	PCM
Golden 13	**500**	859.11	3563.9204	894.5257	4.1738	PCM
Golden 14	**500**	1081.31	5387.9766	1160.2756	6.8712	PCM
Golden 15*	**500**	1345.23	6892.6626	1446.0418	7.4805	PCM
Golden 17	**500**	707.79	2578.7881	727.8627	2.7618	PCM
Golden 18	**500**	997.52	3751.8257	1053.2850	5.5213	MCM
Golden 19	**500**	1366.86	5227.4507	1464.723	6.7674	MCM

In the first column, the instance name is shown. The second column refers to the number of GVNS iterations completed in each instance. The third column shows the best-known solution available in the recent literature. In the fourth and the fifth column, we report the first solution given by Clarke and Wright algorithm and the final solution provided by GVNS. The sixth column shows the average (relative) error compared to the best-known solution in the literature, and the last column shows the best method in respect to the solution quality. Instances with no known optimum are marked with (*) and the best-known values are taken from the solutions provided in [36]. In the above table, regarding the instances with known optimum solution, the stopping criterion for the instances with zero error was proved to be the achievement of the optimum value (rather than the maximum CPU time or the maximum number of iterations). Furthermore, the stopping criterion for the instances with non-zero error, was proved to be the maximum CPU time or the maximum number of iterations (rather than the achievement of the known optimum value). On the other hand, regarding the instances with not known optimum solution, the stopping criterion was obviously the maximum CPU time or the maximum number of iterations.

7 Conclusions and future work

In this paper, a literature review with recent successful parallel implementations of VNS regarding different variants of VRPs was presented. Also, three models for the parallelization of VNS for the efficient solution of CVRP were proposed. Several well known instances for the CVRP were used in order to compare and analyze the cooperation strategies between the three parallel metaheuristic models.

A strong indication that, the cooperation strategy can have a decisive influence on the quality of the solutions, seems to exist. Specifically, partial cooperation through filtered communication seems to provide better quality solutions on harder problems. Finally, the impact of communication appears to be a function of two factors: the time of communication and the size of solution space (the size of the neighborhood to explore). It appears that filtered or no communication near the end of the search yields better results and communication favors vast neighborhoods.

The findings presented in Section 6, are in line with previous findings by other researchers working with different solution methods and on different optimization problems. Thus, an observation supporting our findings was also made by Groër et al. [18]. The authors examined the gain provided by cooperation by studying solution quality within two versions of the same algorithm. Both versions used the same metaheuristic solution methods except that one variant did not use information sharing among the processors. The authors noticed that the cooperative algorithm did not perform well on small problems, specifically on the instances of [4,5]. Their cooperative algorithm produced solutions with almost two times better quality on the problem sets of Taillard [33], Golden et al. [17] and Li et al. [24].

In another study by Davidović et al. [16], the researchers examine several parallel local search strategies. The authors conclude that, a parallel approach which dynamically partitions the neighborhoods among the processors yields the best results. This method uses a supervisor that, broadcasts a new solution among the workers. A negative aspect of this approach is the significant increase in communication. Even though the best results were obtained using a modest number of processors due to less communication overhead, the authors noticed that, the use of a large number of processors is beneficial when the instance dimensions grow. Thus, the communication cost proves to be not an important factor in the solution of large-scale hard instances.

Munera et al. in their recent paper in 2016 [26] presented a cooperative parallel extremal optimization for the solution of the quadratic assignment problem. The authors mention that, they have performed the tuning process in a manual way without parameter self-adaptation. Therefore, the self-adaptive mechanism presented in this work constitutes a step towards this direction. To the best of the authors' knowledge, the proposed cooperative search mechanism presented in this paper is a novel approach.

Such cooperative parallelization strategies may help to compute solutions for even more complicated integrated problems, e.g., location-inventory-routing problems [21]. Future studies may include the use of machine learning and artificial intelligence in order to achieve even better algorithm configuration and parameter tuning. Also, the use of smarter memory-based strategies could provide better solutions.

References

1. Antoniadis, N., Sifaleras, A.: A hybrid CPU-GPU parallelization scheme of variable neighborhood search for inventory optimization problems. Electronic Notes in Discrete Mathematics **58**, 47–54 (2017)
2. Augerat, P., Belenguer, J., Benavent, E., Corberán, A., Naddef, D., Rinaldi, G.: Computational results with a branch and cut code for the capacitated vehicle routing problem. Tech. Rep. 495, Institute for Systems Analysis and Computer Science (IASI), Rome (1995)
3. Baldacci, R., Toth, P., Vigo, D.: Exact algorithms for routing problems under vehicle capacity constraints. Annals of Operations Research **175**(1), 213–245 (2010)
4. Christofides, N., Eilon, S.: An algorithm for the vehicle-dispatching problem. Journal of the Operational Research Society **20**(3), 309–318 (1969)
5. Christofides, N., Mingozzi, A., Toth, P.: The vehicle routing problem. In: N. Christofides, A. Mingozzi, P. Toth, C. Sandi (eds.) Combinatorial Optimization, vol. 1, chap. 11, pp. 315–338. Wiley Interscience, Chichester, UK (1979)
6. Clarke, G., Wright, J.W.: Scheduling of vehicles from a central depot to a number of delivery points. Operations Research **12**(4), 568–581 (1964)
7. Coelho, I.M., Ochi, L.S., Munhoz, P.L.A., Souza, M.J.F., Farias, R., Bentes, C.: The single vehicle routing problem with deliveries and selective pickups in a CPU-GPU heterogeneous environment. In: 14th IEEE International Conference on High Performance Computing and Communication & 9th IEEE International Conference on Embedded Software and Systems (HPCC-ICESS), pp. 1606–1611. IEEE (2012)
8. Cordeau, J.F., Laporte, G., Savelsbergh, M.W., Vigo, D.: Vehicle routing. In: C. Barnhart, G. Laporte (eds.) Transportation, *Handbooks in Operations Research and Management Science*, vol. 14, chap. 6, pp. 367–428. Elsevier (2007)
9. Crainic, T.G.: Parallel solution methods for vehicle routing problems. In: B. Golden, S. Raghavan, E. Wasil (eds.) The vehicle routing problem: Latest advances and new challenges, pp. 171–198. Springer US, Boston, MA (2008)
10. Crainic, T.G., Gendreau, M., Hansen, P., Mladenović, N.: Cooperative parallel variable neighborhood search for the p-median. Journal of Heuristics **10**(3), 293–314 (2004)
11. Crainic, T.G., Hail, N.: Parallel metaheuristics applications. In: E. Alba (ed.) Parallel Metaheuristics, chap. 19, pp. 447–494. John Wiley & Sons , Ltd (2005)
12. Crainic, T.G., Toulouse, M., Gendreau, M.: Toward a taxonomy of parallel tabu search heuristics. INFORMS Journal on Computing **9**(1), 61–72 (1997)

13. Damerau, F.J.: A technique for computer detection and correction of spelling errors. Communications of the ACM **7**(3), 171–176 (1964)
14. Dantzig, G.B., Ramser, J.H.: The truck dispatching problem. Management Science **6**(1), 80–91 (1959)
15. Davidović, T., Crainic, T.G.: MPI parallelization of variable neighborhood search. Electronic Notes in Discrete Mathematics **39**, 241–248 (2012)
16. Davidović, T., Crainic, T.G.: Parallel local search to schedule communicating tasks on identical processors. Parallel Computing **48**, 1–14 (2015)
17. Golden, B.L., Wasil, E.A., Kelly, J.P., Chao, I.M.: The impact of metaheuristics on solving the vehicle routing problem: algorithms, problem sets, and computational results. In: T.G. Crainic, G. Laporte (eds.) Fleet management and logistics, pp. 33–56. Springer US, Boston, MA (1998)
18. Groër, C., Golden, B., Wasil, E.: A parallel algorithm for the vehicle routing problem. INFORMS Journal on Computing **23**(2), 315–330 (2011)
19. Hansen, P., Mladenović, N., Brimberg, J., Pérez, J.A.M.: Variable neighborhood search. In: M. Gendreau, J.Y. Potvin (eds.) Handbook of Metaheuristics, pp. 57–97. Springer International Publishing, Cham (2019)
20. Hansen, P., Mladenović, N., Todosijević, R., Hanafi, S.: Variable neighborhood search: basics and variants. EURO Journal on Computational Optimization **5**(3), 423–454 (2017)
21. Karakostas, P., Sifaleras, A., Georgiadis, M.C.: A general variable neighborhood search-based solution approach for the location-inventory-routing problem with distribution outsourcing. Computers & Chemical Engineering **126**, 263–279 (2019)
22. Laporte, G., Nobert, Y.: Exact algorithms for the vehicle routing problem. In: S. Martello, G. Laporte, M. Minoux, C. Ribeiro (eds.) Surveys in Combinatorial Optimization, *North-Holland Mathematics Studies*, vol. 132, pp. 147–184. North-Holland (1987)
23. Le Bouthillier, A., Crainic, T.G.: A cooperative parallel meta-heuristic for the vehicle routing problem with time windows. Computers & Operations Research **32**(7), 1685–1708 (2005)
24. Li, F., Golden, B., Wasil, E.: Very large-scale vehicle routing: new test problems, algorithms, and results. Computers & Operations Research **32**(5), 1165–1179 (2005)
25. Mladenović, N., Hansen, P.: Variable neighborhood search. Computers & Operations Research **24**(11), 1097–1100 (1997)
26. Munera, D., Diaz, D., Abreu, S.: Solving the quadratic assignment problem with cooperative parallel extremal optimization. In: F. Chicano, B. Hu, P. Garca-Snchez (eds.) Evolutionary Computation in Combinatorial Optimization, pp. 251–266. Springer (2016)
27. Pérez, J.A.M., Hansen, P., Mladenović, N.: Parallel variable neighborhood search. In: E. Alba (ed.) Parallel Metaheuristics, chap. 11, pp. 247–266. John Wiley & Sons, Ltd. (2005)
28. Polacek, M., Benkner, S., Doerner, K.F., Hartl, R.F.: A cooperative and adaptive variable neighborhood search for the multi depot vehicle routing problem with time windows. Business Research **1**(2), 207–218 (2008)
29. Polat, O.: A parallel variable neighborhood search for the vehicle routing problem with divisible deliveries and pickups. Computers & Operations Research **85**, 71–86 (2017)
30. Ribeiro, C.C., Maniezzo, V., Stützle, T., Blum, C., Juan, A.A., Ramalhinho, H., Mladenović, N., Sifaleras, A., Sörensen, K., Souza, M.J.: Preface to the special issue on matheuristics and metaheuristics. International Transactions in Operational Research **27**(1), 5–8 (2020)
31. Schulz, C., Hasle, G., Brodtkorb, A.R., Hagen, T.R.: GPU computing in discrete optimization. Part II: Survey focused on routing problems. EURO Journal on Transportation and Logistics **2**(1-2), 159–186 (2013)
32. Skouri, K., Sifaleras, A., Konstantaras, I.: Open problems in green supply chain modeling and optimization with carbon emission targets. In: P.M. Pardalos, A. Migdalas (eds.) Open Problems in Optimization and Data Analysis, pp. 83–90. Springer Optimization and Its Applications (2018)
33. Taillard, É.: Parallel iterative search methods for vehicle routing problems. Networks **23**(8), 661–673 (1993)
34. Toro, O., Eliana, M., Escobar, Z., Antonio, H., Granada, E.: Literature review on the vehicle routing problem in the green transportation context. Revista Luna Azul (42), 362–387 (2016)
35. Toth, P., Vigo, D.: The Vehicle Routing Problem. Society for Industrial and Applied Mathematics (2002)
36. Uchoa, E., Pecin, D., Pessoa, A., Poggi, M., Vidal, T., Subramanian, A.: New benchmark instances for the capacitated vehicle routing problem. European Journal of Operational Research **257**(3), 845–858 (2017)