A Parallel Hybrid Genetic Algorithm on Cloud Computing for the Vehicle Routing Problem with Time Windows

André Siqueira Ruela*
and Frederico Gadelha Guimarães†
Programa de Pós-Graduação em Engenharia Elétrica
Universidade Federal de Minas Gerais
Belo Horizonte, MG, Brasil
*andre.siqueira.ruela@gmail.com
†fredericoguimaraes@ufmg.br

Ricardo Augusto Rabelo Oliveira‡,
Brayan Neves,
Vicente Peixoto Amorim
and Larissa Maiara Fraga
Programa de Pós-Graduação em Ciência da Computação
Universidade Federal de Ouro Preto
Ouro Preto, MG, Brasil
‡rrabelo@gmail.com

***Abstract*—This paper proposes a new Parallel Hybrid Genetic Algorithm approach for Vehicle Routing Problem with Time Windows. The algorithm was developed to be executed on cloud computing web services and serves as an online application for real world problems. A new parallel scheme was proposed with shared resources of candidate solutions accessed by many asynchronous tasks. The algorithm was tested over the classical well-known benchmark and presented excellent results for some instances in a low computational time. The algorithm reaches the best-known solutions for many instances and found high competitive solutions. The excellent performance of the proposed approach indicates its potential to be applied in real world applications, running on cloud computing servers.**

***Index Terms*—Parallel Hybrid Metaheuristics, Optimization on Cloud Computing, VRPTW.**

I. Introduction

Applications of routing and scheduling models are usually large-scale in real world scenario. Efficient routing and scheduling of vehicles can save the public and private sectors a large and considerable amount of money or other resources. In nature, the complexity of transportation logistics systems may arise from many different sources, such as customers, vehicles, shipments and physical infrastructure, all interacting in various ways. Moreover, due to increasing collaboration between transport companies and logistics partners, interesting and practical instances are becoming larger in size and complexity. The Vehicle Routing Problem with Time Windows (VRPTW) arises in retail distribution, school bus routing, mail and newspaper delivery, municipal waste collection, fuel oil delivery, dial-a-ride service and airline and railway fleet routing and scheduling.

Most of early research within parallelizing techniques in combinatorial optimization focuses on multi-core architectures with relatively few processors. This paper proposes a new Parallel Hybrid Genetic Algorithm (PHGA) for VRPTW, designed to run as a web service in cloud computing. The main objective is the design of an algorithm capable to handle real world problems with low computational time, which can be accessed anytime and anywhere, as soon as a new need emerges. The PHGA was tested on Amazon Web Services and the results obtained in computational experiments show that our major goal is achieved.

This Section presented a brief introduction to this paper. Section II covers the nature of the problem, the related problems, its concepts and definitions. Section III presents the proposed algorithm in a detailed view. It also describes in III-E how the tasks interact with shared resources in an asynchronous way. Section IV presents the parameters set, the computational results for running the algorithm on a cloud. Finally, Section V presents the conclusions and nearly future works.

II. Problem Definition

The Vehicle Routing Problem (VRP) is a generalization of the Traveling Salesman Problem (TSP) and consists in determining a set of vehicle routes of minimum total cost, such that each vehicle starts and ends at the depot (warehouse), each customer is visited exactly once, and the total demand handled by any vehicle does not exceed its capacity [1], [2]. In the Vehicle Routing Problem with Time Windows (VRPTW) these issues must be addressed under the added constraints on time windows, because there are customers imposing an earliest arrival time and a latest arrival time constraints [3]. In this case, a problem instance consists of a central depot, assuming homogeneous fleet, and a set of customers, each with a location, a demand, a service time, and a time window.

From a computational complexity view, VRP are difficult to handle. Since the VRP is NP-hard, by restriction, the VRPTW is NP-hard. Finding a feasible solution to the Traveling Salesman Problem with Time Windows (TSPTW) is an NP-complete problem. Even finding a feasible solution to the VRPTW when the number of vehicles is fixed is itself an NP-complete problem [4]. Therefore VRPTW is more complex as it involves servicing customers with time windows. Thus obtaining optimal solutions to VRPTW by the use of exact

methods requires unacceptable computational time for some instances [5]. Heuristic methods are by far more attractive, stemming from the fact that they often produce optimal or near optimal solutions in a reasonable amount of computer time. Several heuristic approaches have been proposed to this problem, making it a well-known problem in literature. However, there is still a considerable interest in the design of new heuristics for solving large scale practical VRPTW.

The heuristic techniques for the VRPTW can be divided into three groups: improvement heuristics, construction heuristics and metaheuristics. Improvement heuristics are essentially local search procedures, which search iteratively for better neighboring solutions [6], [7]. Constructive heuristics attempt to build a feasible solution by inserting customers into routes, iteratively, under some insertion criteria [3], [8]. Metaheuristic techniques attempt to perform a wide exploration over the objective function landscape. Metaheuristics are also known to be good methods for avoiding local mimina, often allowing infeasible solutions or worsening moves during it's search [9], [10].

Evolutionary computation has been successfully applied to solve this and many other combinatorial optimization problems. Mester [11], [12] designed evolution strategies specifically for large-scale problems. Two evolution strategies for solving the VRPTW were proposed by Homberger [13]. Co-evolution of two populations is applied by [14] in a parallel hybrid genetic algorithm. The genetic algorithm proposed in this paper is strongly based on the work of Prins [15]–[17] and Chang [18]. This paper is also related to the recent work of Błocho and Czech [19], which have developed a parallel heuristic that explores the scalability of the Message Passing Interface (MPI). Another similar work is [20] that uses Hadoop for a parallel large neighborhood search over a set of large scale VRPTW instances. For more information about the current state-of-the-art solutions for VRPTW, we suggest the read of surveys [21] and [22]. For more information about evolutionary computation running on cloud structures, we suggest the read of the recent work of Wilson [23].

A. Problem Formulation

The VRPTW is described as the graph theoretic problem: Let $G = (V, E)$ be a complete and undirected graph where $V = \{0, \ldots, n\}$ is the vertex set and E is the edge set. Vertex set $V_c = \{1, \ldots, n\}$ corresponds to n customers, whereas vertex 0 corresponds to the depot. The constraints consist of a set of homogeneous vehicles, a depot or warehouse node, a set of customers and a network connecting the warehouse and customers. There are $N + 1$ customers and K vehicles. The warehouse is denoted as customer 0. Each arc in the network represents a connection between two nodes. Each route starts from the depot. The number of routes in the network is equal to the number of vehicles used. One vehicle is dedicated to one route. A cost c_{ij} and a travel time t_{ij} are associated with each arc of the network. Every vehicle has the same capacity q_k and every customer has a varying demand m_i. The capacity q_k must be greater than or equal to the sum of all demands on the route traveled by the vehicle k, which means that no vehicle can be overloaded.

The objective is to design a network that satisfies all constraints and minimizes the total travel cost. The model is mathematically formulated below:

T_i arrival time at customer i
w_i wait time at customer i
x_{ijk} is 1 if vehicle k travels from customer i to customer j, 0 otherwise, such that $i \neq j$; $i, j \in \{0, 1, \ldots, N\}$
K total number of vehicles
N total number of customers
c_{ij} cost incurred on arc from customer i to j
t_{ij} travel time between customer i and j
m_i demand at customer i
q_k capacity of vehicle k
e_i earliest arrival time at customer i
l_i latest arrival time at customer i
f_i service time at customer i
r_k maximum route time allowed for vehicle k
p_i polar coordinate angle for customer i

Minimize:

$$\sum_{i=0}^{N} \sum_{j=0}^{N} \sum_{i \neq j, k=1}^{K} c_{ij} x_{ijk} \tag{1}$$

subject to:

$$\sum_{k=1}^{K} \sum_{j=1}^{N} x_{ijk} \leq K, i = 0 \tag{2}$$

$$\sum_{j=1}^{N} x_{ijk} = 1, i = 0, k \in \{1, \ldots, K\} \tag{3}$$

$$\sum_{j=1}^{N} x_{jik} = 1, i = 0, k \in \{1, \ldots, K\} \tag{4}$$

$$\sum_{k=1}^{K} \sum_{j=0, i \neq j}^{N} x_{ijk} = 1, i \in \{1, \ldots, N\} \tag{5}$$

$$\sum_{k=1}^{K} \sum_{i=0, i \neq j}^{N} x_{ijk} = 1, j \in \{1, \ldots, N\} \tag{6}$$

$$\sum_{i=1}^{N} m_i \sum_{j=0, i \neq j}^{N} x_{ijk} \leq q_k, k \in \{1, \ldots, K\} \tag{7}$$

$$\sum_{i=0}^{N} \sum_{j=0, i \neq j}^{N} x_{ijk}(t_{ij} + f_i + w_i) \leq r_k, k \in \{1, \ldots, K\} \tag{8}$$

$$T_0 = w_0 = f_0 = 0 \tag{9}$$

$$\sum_{k=1}^{K} \sum_{i=0, i \neq j}^{N} x_{ijk}(T_i + t_{ij} + f_i + w_i) \leq r_k, k \in \{1, \ldots, K\} \tag{10}$$

$$e_i \leq (T_i + w_i) \leq l_i, i \in \{1, \ldots, N\} \tag{11}$$

The objective function (1) minimizes the total traveling cost. Constraint (2) guarantees that the number of tours is not greater than K. Constraints (3) and (4) guarantees that all the customers must be partitioned disjoint, the depot is included in all routes. Constraints (5) and (6) guarantees that every customer is visited only once. Constraint (7) prevents the overload of vehicles. The constraint set (9) sets the arrival time, waiting time and service time of each vehicle at the warehouse to zero. And finally, the restrictions (10) and (11) take care of time constraints. It is important to notice that minimizing the number of vehicles is not the real objective for the original VRPTW. Some authors attempt to minimize both vehicles and distance, but this is just a heuristic approach. In this paper, the focus is minimizing the traveling cost only.

III. Parallel Hybrid Genetic Algorithm

The Parallel Hybrid Genetic Algorithm (PHGA) proposed in this work is based on the algorithm presented in [15] and [18]. The major difference is the parallel approach, dividing the problem into few smaller independent tasks, and some heuristics variations.

A. Chromosome Codification and Evaluation

All solutions are coded to a sequence or permutation S of n customers without route delimiters. It can be interpreted as the order in which a vehicle must visit all customers. This simple encoding is appealing because there obviously exists one optimal sequence. Furthermore, classical well-known operators for solving TSP can be applied without complications. So the intrinsic parallelism of the GA is designed to find this sequence and a splitting procedure is used to split optimally any sequence into trips and recover the corresponding VRPTW solution, using an auxiliary directed, circuitless and weighted graph $H = (X, A, W)$. A min-cost path μ from node 0 (depot) to node n can be computed using Bellman's algorithm. For more details about the splitting procedure, check [15] and [18]. The PHGA uses this procedure to evaluate each chromosome and the fitness is simply the total cost of the resulting VRPTW solution. The evaluation is reasonably fast because H and μ can be built in $O(n^2)$.

B. Initialization

The population is implemented as an array Π of σ chromosomes, always sorted in increasing order of fitness value. In this way, the best solution is always coded by the first chromosome in Π or Π_0. For constructing the initial population, there are three different methods. The first method generates a completely random sequence. The idea behind the random generation is to feed the algorithm with diverse solutions allowing the exploration of a large space of solutions. The second method uses a rondomized version of the construction process proposed by Russell [24]. The third method is a randomized version of the Push-Forward Insertion Heuristic (PFIH) detailed on [25]. The PFIH starts a new route by selecting an initial customer and then inserting customers into the current route until no more feasibility is possible. The unrouted customer with the lowest cost is selected as the first customer to be visited. For more details about the PFIH, read [25]. To generate each chromosome in the initial population, one of these three methods is randomly chosen, considering an uniform distribution.

C. Selection and Reproduction

The chromosomes are selected for mating by the rules of a binary tournament. In a binary tournament, two chromosomes are randomly selected and the one that has the best fitness value is the winner. This process is repeated for selecting a second winner. Then the winners are submitted to the crossover operator.

The chromosome codification allows the application of classical permutation crossover operators, like the order crossover (OX) or linear order crossover (LOX) [26]. A given VRPTW solution can be encoded as different chromosomes, depending on which order its routes are concatenated. So there is no reason to distinguish a first or last client in the sequence, as LOX does. Additionally, [15] confirms the OX superiority. The OX operator generates two chromosomes, but only one is randomly selected to be the resulting child. There is no crossover rate evaluation, i.e. selected parents have 100% chance of reproduction.

After crossover, the child is submitted to the mutation procedure. There are two mutation procedures. The first mutation is a Local Search (LS) procedure under the rate ρ_{ls}. LS needs not be sophisticated because a relative weakness can be compensated by the intrinsic parallelism of the GA. The child is then converted to a VRPTW solution. Each iteration of LS scans in $O(n^2)$ all possible pairs of distinct nodes (u, v). Nodes u and v can be customers or the warehouse and belong to the same route or to different routes. For each pair (u, v), the following moves are tested in turn (x is the successor of u and y the successor of v along their respective routes):

- Move 1: move u after v;
- Move 2: move (u, x) after v;
- Move 3: move (x, u) after v;
- Move 4: permute u and v;
- Move 5: permute (u, x) with v;
- Move 6: permute (u, x) and (v, y);
- Move 7: if (u, x) and (v, y) are non adjacent in the same route, replace them by (u, v) and (x, y);
- Move 8: if (u, x) and (v, y) are in distinct routes, replace them by (u, v) and (x, y);
- Move 9: if (u, x) and (v, y) are in distinct routes, replace them by (u, y) and (x, v).

Move 7 corresponds to 2-opt, while moves 8 and 9 generalize 2-opt to different routes. Each iteration stops at the first improving move. The process is repeated until no further improvement can be found in the neighborhood. The solution of LS is coded back to a chromosome by concatenating its routes in a single sequence. In each LS execution, the moviments are chosen in a random order.

The second mutation procedure performs simple elementary perturbation over the child under the rate ρ_m. Random values

for u and v are chosen and an arbitrary and suitable move from the moves 1 to 9, is selected. No improvement is necessary in this case, i.e. the child just accepts any random suitable move, jumping to a neighboring solution.

D. Replacement Method and Stopping Criteria

The PHGA is also steady-state GA, which means that the best chromosomes are maintained in the population and only a portion of it is replaced by the children resulting from the breeding. In this work, the first half of population is maintained, i.e. the first $\sigma/2$ individuals are not replaced by new forthcoming breeds. So, the replaced individual is randomly chosen in the second half of Π.

The replacement is also subject to the $\Delta - property$ rule. Clones (identical solutions) are forbidden in Π to ensure a better dispersal of solutions. To avoid comparing chromosomes in details and to speed-up clone detection, a stricter condition is imposed: the fitness values (total traveled distance) of any two solutions generated by crossover or mutation must be spaced at least by a constant $\Delta > 0$. The $\Delta - property$ can be formalized as follows:

$$\forall P_1, P_2 \in \Pi : P_1 \neq P_2 \Rightarrow |F(P_1) - F(P_2)| \geq \Delta,$$

where P_1 and P_2 are parents selected from the current population, and $F(P_1)$ and $F(P_2)$ are the fitness values of the parents P_1 and P_2, respectively. In this work, we set $\Delta = 2$. The only exception to this rule occours when a new best solution is found. The population is re-sorted after each replacement.

The PHGA runs until it reaches the maximum number of productive breeds, i.e. generated children that follows the $\Delta - property$ rule or a maximal number of iterations without improving the best solution.

E. Breaking the Algorithm into Smaller Tasks

To exploit the maximum resource capabilities of multi-processor computers and the advantages of the cloud computation, the proposed hybrid GA is parallelized, divided into many smaller and asynchronous tasks. The idea behind the choosing of a steady-state GA for this work is to avoid the concept of generation. In a generational GA, all the offspring have to be completely generated to allow the beginning of the next generation. This waiting can be the bottleneck of the gerational approach. On the other hand, in a steady-state GA, newly generated children can participate in reproduction as soon as they enter the current population. The $\Delta - property$ described before ensures diversity of the individuals.

The basic genetic operators that compose the Hybrid Genetic Algorithm were divided into parallel procedures, named Tasks. As the population array Π, let P be the parents array, selected for breeding, but not submitted to OX crossover yet and X be the children array, produced by reproduction, but not submitted to the replacement method yet. Π, P and X are considered as shared resources. Despite the asynchronous nature of the tasks, the access to elements inside the resources is performed in a synchronized manner to avoid bad concurrency issues. Moreover, if a task requires the access to an element on a shared resource that is not ready or does not exist yet, the task waits until being notified by that resource. Hence, the resource notifies the waiting tasks if there is any modification in its elements array that allows the tasks to proceed in its operations. For example, the Replacement Manager (task) takes a child from X and attempts to replace it in Π. Suppose that child does not exist yet in X. So the task waits until X notifies it about the arrival of a new child.

The implemented tasks are detailed below:

- Selection Manager (SM): this task performs the binary tournament and has access to Π and P. The SM copies the winner parents from Π into P respecting the maximum storage capacity of P. If P is full, SM waits until new slots are released.
- Massive Breeder (MB): performs the OX crossover and mutation procedures, removing two parents from P, breeding them and storing the resulting child in X, respecting its storage capacity. If there is no parent in P, it waits for the work of SM. If X is full, it waits for slot releasing. MB operates with a $\rho_{ls} = 0.0001$ and $\rho_m = 0.025$ mutation rates.
- Exploit Breeder (EB): same thing of MB, but has $\rho_{ls} = 0.025$ and $\rho_m = 0$ mutation rates. So EB performs a large number of LS in comparison to MB. In this work, a large number of EB's tasks are instantiated during the execution of the algorithm.
- Replacement Manager (RM): removes one child from X and attempts to insert it into Π accordingly to the $\Delta - property$. If X is empty, it waits for the work of the Breeders. Sorts the elements of Π after each successful replacement.
- Execution Manager (EM): the first and the last working task, checks the running conditions of the PHGA and, if some stopping criteria was reached, determines the end of the execution finalizing the resources and waiting the ending of all other tasks. Also stores and returns the best solution to the main process.

The Massive Breeder has this name because its very low ρ_{ls} rate allows it to breed a large number of children in a small number of seconds. On average, MB performs the LS only 3 times over the entire execution. The idea of MB is to fast explore the space of solutions, generating new possible candidate individuals, while the EB performs the LS improving that individuals. Breeding a large number of low quality solutions is not a promising approach. So it is interesting to MB to wait for improved solutions generated by EB's. But if good solutions are the point, it is more useful to spend time improving its own children than waiting for other breeders to complete. So that's why MB has a low ρ_{ls} rate, instead zero or higher values.

Figure 1 illustrates the parallel scheme proposed for this work. As shown in Figure 1, the PHGA can instantiate a neb number of EB tasks. Although there are no limitations about the number of other tasks, except the obvious $n > 0$, in this paper all tasks, except EB, have just 1 running instance.

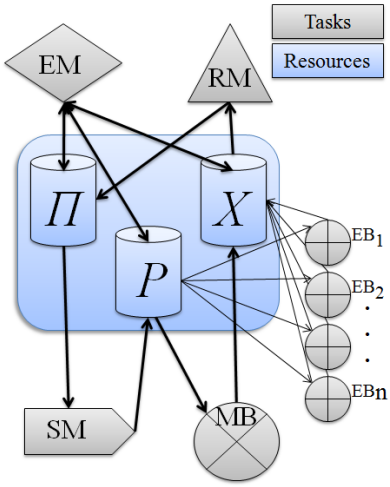

Fig. 1. PHGA schem.

IV. Computational Results

All algorithms were implemented in Java 1.6. The PHGA was tested on the Amazon Web Services (AWS). The computational experiments were assigned to run over an Amazon Linux AMI 2012.09.1 64bit operational system. A C1 High-CPU Extra Large (c1.xlarge) instance was allocated for the tests. The c1.xlarge machine has 20 ECUs units, 8 CPU Cores and 7GiB RAM. The instance was launched in EC2-Classic zones. All the parallel background, used in tasks and resources implementations, were built using Java native libraries. All math operations were handled by the Parallel Colt library.

We set the population size $\sigma = 30$, the number of EB tasks $neb = 11$ (the number of all other tasks was set to 1), the maximum number of breeds $mnb = 30000$, the maximum number of breeds without improving the best solution $mwi = 10000$, the maximum P size to $\sigma_P = 2neb$, the maximum X size to $\sigma_X = \sigma$.

Tables I, II, III show the performance of the PHGA in comparison to the current state-of-art solutions for the Solomon's [3] 100 customers instances. The column "Best-Known" refers to the best-known heuristic solution according to [27], while the column "Author" refers to the initials of the authors that reached the respecting result. We suggest the read of [27] for a full reference of all these works. The column "PHGA" presents the best result found by this paper. Let K be the number of the vehicles and TD be the total traveled distance of the best solution found. It is valid to remember that the approach proposed in this work attempts to minimize only the total traveled distance. Although K being shown on the tables, it is not the focus of this paper.

The PHGA had an average execution time around 200 seconds. The task that requires the major computational effort is by far the EB, because its LS mutation procedure demands checking iteratively all neighboring solutions for an improvement. The current parameters set allows, on average, the execution of 30 LS procedures for EB's tasks and 3 LS for MB task. In other words MB imposes the rhythm of the search process in PGHA due to the chosen parameters, such as the stopping criteria and mutation rates. Usually, when the MB finishes its last breed procedure, the EB's tasks are often performing their third or fourth LS.

In this way, over 95% of children generated during the PHGA search process came from MB operators, but the major impacting changes in the quality of Π is due to EB's LS. The idea behind the maintenance of one MB is to promote diversity and not only quality.

For C-type instances, the PHGA demonstrated to be a very competitive approach, reaching the best-known solutions for 8 of the 17 instances tested and finding at least a near-best solution for other instances. For R1-type and RC1-type instances the PHGA had a bad performance in comparison to the best-known solutions, but its low computational time can compensate this. On the other hand, for R2-type and RC2-type instances, specifically the R201, R202, RC201, RC202 and RC205 instances, the PHGA reaches excellent solutions, close to the best-known. These instances have larger vehicles capacity, customer demands and time windows, which means that R2-types and RC2-types are more similar to real world problems. The good performance of PHGA for these instances indicates its potential to be applied in real world applications, running on cloud computing servers.

TABLE I
PHGA results for Solomon's [3] 100 customers C-type instances

P C	Best-Known K	Best-Known TD	Author	PHGA K	PHGA TD	P C	Best-Known K	Best-Known TD	Author	PHGA K	PHGA TD
101	10	**828.94**	RT	10	**828.94**	201	3	**591.56**	RT	3	**591.56**
102	10	**828.94**	RT	10	853.61	202	3	**591.56**	RT	3	**591.56**
103	10	**828.06**	RT	11	974.54	203	3	**591.17**	RT	3	605.44
104	10	**824.78**	RT	11	1050.17	204	3	**590.60**	RT	4	730.72
105	10	**828.94**	RT	10	**828.94**	205	3	**588.88**	RT	4	618.57
106	10	**828.94**	RT	10	**828.94**	206	3	**588.49**	RT	3	**588.49**
107	10	**828.94**	RT	10	**828.94**	207	3	**588.29**	RT	3	**588.29**
108	10	**828.94**	RT	10	829.69	208	3	**588.32**	RT	3	588.49
109	10	**828.94**	RT	11	898.45						

TABLE II
PHGA results for Solomon's [3] 100 customers R-type instances

P R	Best-Known K	Best-Known TD	Author	PHGA K	PHGA TD	P R	Best-Known K	Best-Known TD	Author	PHGA K	PHGA TD
101	19	**1645.79**	H	22	1713.27	201	4	1252.37	HG	10	**1198.45**
102	17	**1486.12**	RT	19	1565.55	202	3	1191.70	RGP	7	**1133.54**
103	13	**1292.68**	LLH	17	1411.22	203	3	**939.50**	WL	8	1051.99
104	9	**1007.24**	MBD	14	1168.48	204	2	**825.52**	BVH	5	914.63
105	14	**1377.11**	RT	18	1461.61	205	3	**994.42**	RGP	7	1049.54
106	12	**1251.98**	MBD	16	1380.53	206	3	**906.14**	SSSD	6	1056.32
107	10	**1104.66**	S97	13	1266.03	207	2	**890.61**	RP	4	972.37
108	9	**960.88**	BBB	12	1111.41	208	2	**726.75**	MBD	5	840.65
109	11	**1194.73**	HG	15	1306.59	209	3	**909.16**	H	5	985.65
110	10	**1118.59**	MBD	16	1340.09	210	3	**939.34**	MBD	7	1019.74
111	10	**1096.72**	RGP	15	1242.82	211	2	**885.71**	WL	6	938.31
112	9	**982.14**	GTA	14	1263.18						

TABLE III
PHGA results for Solomon's [3] 100 customers RC-type instances

P RC	Best-Known K	Best-Known TD	Author	PHGA K	PHGA TD	P RC	Best-Known K	Best-Known TD	Author	PHGA K	PHGA TD
101	14	**1696.94**	TBGGP	18	1788.76	201	4	1406.91	MBD	8	**1296.22**
102	12	**1554.75**	TBGGP	16	1618.74	202	3	1365.65	GCC	8	**1162.57**
103	13	**1261.67**	S98	14	1496.07	203	3	**1049.62**	CC	7	1066.30
104	10	**1135.48**	CLM	14	1473.03	204	3	**798.41**	MBD	5	942.71
105	13	**1629.44**	BBB	17	1703.97	205	4	1297.19	MBD	8	**1250.34**
106	11	**1424.73**	BBB	16	1550.73	206	3	**1146.32**	H	7	1202.69
107	11	**1230.48**	S97	14	1440.88	207	3	**1061.14**	BVH	7	1067.73
108	10	**1139.82**	TBGGP	13	1511.22	208	3	**828.14**	IKMUY	5	917.10

V. Conclusion

This paper proposes a new Parallel Hybrid Genetic Algorithm (PHGA) approach for VRPTW. The algorithm was developed to be executed on cloud computing web services providing an online application for real world problems. A new parallel scheme was proposed with shared resources of candidate solutions accessed by many asynchronous tasks. The algorithm was tested over the classical well-known benchmark [3], [27] and presented excellent results for some instances in a low computational time. The algorithm reaches the best-known solutions for 8 of the C-type instances and found good for 5 instances of R2-type and RC2-types.

The approach sounds promising for running over a large set of benchmark instances. Also, speed-up, scalability and parallel efficiency tests must be done to check the performance of the algorithm in other physical architectures. Also related to future works is the development of a application prototype of the practical web service running the PHGA. Moreover, the algorithm can be extended to the Dynamic-VRPTW which is more suitable to handle a large set of real world problems.

VI. Acknowledgements

This work has been supported by the Brazilian agencies CAPES, CNPq, and FAPEMIG; and the Marie Curie International Research Staff Exchange Scheme Fellowship within the 7th European Community Framework Programme. The authors also would like to thank SEVA Engenharia Eletrônica S/A with the project KITT, supported by the informatic laws 8248, 10.176 e 11.077, Brazil, and Prof. Dr. Joubert Lima.

References

[1] L. D. Bodin, "A taxonomic structure for vehicle routing and scheduling problems," *Computers & Urban Society*, vol. 1, no. 1, pp. 11 – 29, 1975. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0305709775900034

[2] B. Eksioglu, A. V. Vural, and A. Reisman, "The vehicle routing problem: A taxonomic review," *Computers & Industrial Engineering*, vol. 57, no. 4, pp. 1472 – 1483, 2009. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0360835209001405

[3] M. M. Solomon, "Algorithms for the vehicle routing and scheduling problems with time window constraints," *Operations Research*, vol. 35, pp. 254–265, 1987.

[4] M. Savelsbergh, "Local search in routing problems with time windows," *Annals of Operations Research*, vol. 4, pp. 285–305, 1985. [Online]. Available: http://dx.doi.org/10.1007/BF02022044

[5] M. Desrochers, J. Desrosiers, and M. Solomon, "A new optimization algorithm for the vehicle routing problem with time windows," *Oper. Res.*, vol. 40, no. 2, pp. 342–354, Mar. 1992. [Online]. Available: http://dx.doi.org/10.1287/opre.40.2.342

[6] J.-Y. Potvin and J.-M. Rousseau, "An exchange heuristic for routeing problems with time windows," *Journal of the Operational Research Society*, vol. 45, pp. 1433–1446, 1995.

[7] O. Bräysy, G. Hasle, and W. Dullaert, "A multi-start local search algorithm for the vehicle routing problem with time windows," *European Journal of Operational Research*, vol. 159, no. 3, pp. 586 – 605, 2004. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0377221703004351

[8] J.-Y. Potvin and J.-M. Rousseau, "A parallel route building algorithm for the vehicle routing and scheduling problem with time windows," *European Journal of Operational Research*, vol. 66, no. 3, pp. 331 – 340, 1993. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0377221793902218

[9] Y. Rochat and E. Taillard, "Probabilistic diversification and intensification in local search for vehicle routing," *Journal of Heuristics*, vol. 1, pp. 147–167, 1995. [Online]. Available: http://dx.doi.org/10.1007/BF02430370

[10] Z. J. Czech and P. Czarnas, "A parallel simulated annealing for the vehicle routing problem with time windows," in *10th Euromicro Workshop on Parallel, Distributed and Network-basedProcessing*, Canary Islands, Spain, January 2002, pp. 376–383.

[11] D. Mester, "An evolutionary strategies algorithm for large scale vehicle routing problem with capacitate and time windows restrictions," Working Paper, Institute of Evolution, University of Haifa, Israel, 2002.

[12] D. Mester and O. Bräysy, "Active guided evolution strategies for large-scale vehicle routing problems with time windows," *Computers & Operations Research*, vol. 32, no. 6, pp. 1593 – 1614, 2005. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0305054803003605

[13] J. Homberger, H. Gehring, F. Hagen, L. Wirtschaftsinformatik, D-Hagen, and B. Deutschland, "Two evolutionary metaheuristics for the vehicle routing problem with time windows," *INFOR*, vol. 37, pp. 297–318, 1999.

[14] J. Berger, M. Barkaoui, and O. Bräysy, "A parallel hybrid genetic algorithm for the vehicle routing problem with time windows," Working paper, Defense Research Establishment Valcartier, Canada,, 2001.

[15] C. Prins, "A simple and effective evolutionary algorithm for the vehicle routing problem," *Computers & Operations Research*, vol. 31, no. 12, pp. 1985 – 2002, 2004. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0305054803001588

[16] N. Labadi, P. Lacomme, and C. Prins, "A memetic algorithm for the heterogeneous fleet vrp," in *Proceedings of Odysseus 2006*, E. B. et al. (Eds.), Ed. Univ. of Valencia, 2006, pp. 209–213.

[17] C. Prins, "Two memetic algorithms for heterogeneous fleet vehicle routing problems," *Engineering Applications of Artificial Intelligence*, vol. 22, no. 6, pp. 916 – 928, 2009, ¡ce:title¿Artificial Intelligence Techniques for Supply Chain Management¡/ce:title¿. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0952197608001693

[18] Y. Chang and L. Chen, "Solve the vehicle routing problem with time windows via a genetic algorithm," in *Proceedings of the 6th AIMS International Conference*, Poitiers, France, 2007, pp. 240–249.

[19] M. Bł ocho and Z. J. Czech, "A parallel algorithm for minimizing the number of routes in the vehicle routing problem with time windows," in *Parallel Processing and Applied Mathematics*, ser. Lecture Notes in Computer Science, R. Wyrzykowski, J. Dongarra, K. Karczewski, and J. Waśniewski, Eds. Springer Berlin Heidelberg, 2012, vol. 7203, pp. 255–265. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-31464-3_26

[20] S. Ryza, "Solving hard problems with lots of computers," 2012, undergraduate Honors Theses, Brown Computer Science. [Online]. Available: http://cs.brown.edu/research/pubs/theses/ugrad/2012/ryza.pdf

[21] S. Kumar and R. Panneerselvam, "A survey on the vehicle routing problem and its variants," *Intelligent Information Management*, vol. 4, no. 3, pp. 66–74, 2012.

[22] M. Gendreau and C. D. Tarantilis, *Solving large-scale vehicle routing problems with time windows: The state-of-the-art*. CIRRELT, 2010.

[23] D. Wilson, K. Veeramachaneni, and U.-M. O'Reilly, "Cloud scale distributed evolutionary strategies for high dimensional problems," in *Applications of Evolutionary Computation*, ser. Lecture Notes in Computer Science, A. Esparcia-Alcázar, Ed. Springer Berlin Heidelberg, 2013, vol. 7835, pp. 519–528. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-37192-9_52

[24] W.-C. Chiang and R. A. Russell, "Simulated annealing metaheuristics for the vehicle routing problem with time windows," *Annals of Operations Research*, vol. 63, no. 1, pp. 3–27, 1996.

[25] S. R. Thangiah, "A hybrid genetic algorithms, simulated annealing and tabu search heuristic for vehicle routing problems with time windows," *Practical handbook of genetic algorithms*, vol. 3, pp. 347–381, 1999.

[26] P. Moscato *et al.*, "On genetic crossover operators for relative order preservation," *C3P Report*, vol. 778, 1989.

[27] sintef. Top/vrptw: Best known solutions for the 100 customer instances of solomon's vrptw benchmark problems from 1987. SINTEF. [Online]. Available: http://www.sintef.no/Projectweb/TOP/VRPTW/Solomon-benchmark/100-customers/