Solving multi-depot vehicle routing problem with particle swarm optimization

Matic Pintarič
University of Maribor, Faculty of Electrical Engineering and Computer Science
Koroška cesta 46
Maribor, Slovenia
matic.pintaric@student.um.si

Sašo Karakatič
University of Maribor, Faculty of Electrical Engineering and Computer Science
Koroška cesta 46
Maribor, Slovenia
saso.karakatic@um.si

ABSTRACT

Multi-depot vehicle routing problem (MDVRP) is an optimization problem with practical real-world applications in the commercial transportation sector. It deals with the optimization of the time and cost of the transportation of goods from and to customers from numerous predefined serving depots. The *multi-depot* variant adds constraints of multiple serving depots and variable customer serving capacity and is a NP hard problem. In this paper, we present an application of Particle Swarm Optimization (PSO) for continuous optimization to MDVRP, where nature-inspired optimization framework NiaPy is used. As MDVRP is a discreet optimization problem, but NiaPy is suited to work only with continuous optimization problems, a transformation with the repairing mechanism must be used. Our proposed approach is presented in detail and is tested on several standard MDVRP benchmark sets to provide a sufficient evidence of usability of the approach.

Keywords

Multi-depot Vehicle Routing Problem, Continuous optimization, Particle Swarm Optimization

1. INTRODUCTION

Optimization is a daily problem we face with in an increasing number of different logistics services, such as for example mail delivery, passenger transportation and other transportation of goods [1], [2], [4]. Because solving such problems is - due to restrictions on the route often quite difficult, we mostly rely on computer intelligence. By this we mean different algorithms that have some heuristics rules, which result in good solutions. They are classified as metaheuristic algorithms and often also referred to as nature-inspired or evolutionary algorithms (EA) [6], [9].

In order to execute the experiment of a comparison between different EA, we firstly developed a system that allows application of any EA to the vehicle routing problem (VRP). Then we tackled the VRP using five different evolutionary algorithms, which are genetic algorithm (GA), evolution strategy (ES), differential evolution (DE), particle swarm optimization (PSO) and harmony search (HS). Considering the obtained results, we decided to pay more attention to the PSO algorithm.

PSO is a popular algorithm for solving many complex optimization problems, including routing problems. In 2008 Mohemmed et al. [16] used PSO for simple routing problem with custom priority-based encoding and heuristic operator to prevent loops in the path construction. Next in 2009, Ai and Kachitvichyanukul [17] presented PSO with multiple social structures to solve VRP with simultaneous pickup and delivery. Yao et al. [18] proposed custom particle swarm optimization algorithm for carton heterogeneous vehicle routing problem. Kumar et al. [19] proposed a PSO for vehicle routing problem with time window constraint. More recently, Norouzi et al. [20] extended VRP with time window problem with additional fuel consumption constraint and solved the problem also with PSO. All these approaches treat routing problem as a discreet problem. As we used optimization framework, which only works with continuous optimization problem, the approaches from the referenced papers could not be used and a transformation was necessary.

The remaining of the paper is structured as follows. Second section presents and formulates MDVRP and PSO. Next section presents our proposed approach and its implementation with NiaPy framework. Fourth section presents the results of the experiment of the proposed approach on the standard benchmark sets. Last section discusses the results of the experiments and finishes with the concluding remarks.

2. SOLVING MDVRP

Logistic companies and other carriers, tasked with transporting or picking up shipments, face with route optimization on a daily level. The well-optimized route, taken by their fleets of vehicles, means saving fuel and thus reducing overall daily cost. From this we can see that similar scenarios are present in in everyday life and present a problem worth solving well.

The described problem is called VRP and was first addressed by George Dantzig and John Ramser in 1959, as a solution to optimize fuel delivery. It is a NP hard combinatorial optimization problem, generalized from travelling salesman problem, so there is no polynomial time solution known to it [1], [3]. The result of the problem is the optimal set of routes for multiple vehicles, transporting goods to or from customers, subject to restrictions along the routes. We also need to keep in mind that only one vehicle can visit specific customer at a time [1], [7].

Over time, different classifications of the problem have formed due to differences in constraints. The most common versions of the problem are VRP with capacity constraints (CVRP), multi-depot VRP (MDVRP) and VRP with time windows constraints (VRPTW). In addition, just about every problem classification also has a certain distance limit of the individual vehicle [1], [4].

We focused on solving MDVRP classification of the problem, which is also less frequently referred to as multi-depot capacitated vehicle routing problem (MDCVRP) [4]. The version of the problem, in addition to multiple customers, consists of multiple depots and thus more predefined vehicles - each vehicle can carry

DOI: https://doi.org/10.26493/978-961-7055-82-5.53-56

a certain payload, travel a certain distance and must eventually return to its starting depot [3], [11]. If any of the restrictions is violated, penalty function is used to punish the vehicle with a certain mark-up value. Because MDVRP is represented as a directed graph, every node of customer and depot has its own x and y coordinate pair [4], [5]. When looking for a solution to the problem, we usually divide it into three phases, collectively referred to as decision making or decision hierarchy in MDVRP. We call them the merging phase, the routing phase and the scheduling phase. In the first phase, we try to allocate customers to the individual depots according to the distance, which is present between them. The second phase, with previously divided customers, draws up several routes, which vehicles will take. After that, each path is sequenced in the third phase [3], [11].

Figure 1 is an example of MDVRP problem, where letters A and B represent depots or vehicles with a maximum capacity of 10 units of weight, and circles with numbers represent different customers. There are four paths between depots and customers, based on genotype numbers that are converted to phenotype numbers by the first conversion method. The routes were determined according to the previously mentioned MDVRPs decision hierarchy.

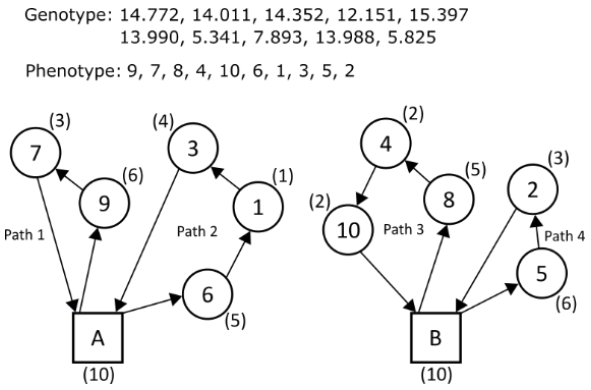

Figure 1. MDVRP example.

2.1 Particle Swarm Optimization for MDVRP

The system we developed for experiment purposes, supports importing any EA from the library or microframework called NiaPy. It was developed by researches from the Faculty of Electrical Engineering and Computer Science and the Faculty of Economics and Business from University of Maribor. Main purpose for developing the framework was lack of easy and fast use of EA, since the own implementation of a single algorithm is often difficult and time-consuming. The library architecture is divided into two parts, which we call algorithms and benchmarks. In addition to the developed normal EA versions, we can also find hybrid variants, such as hybrid bat algorithm and self-adaptive differential evolution algorithm. The framework supports EA startup and testing with predefined and generated comparisons, while also allowing the export of results in three different formats, which are LaTeX, JSON and Excel [13].

Although some similar frameworks for managing nature-inspired algorithms already exist, NiaPy differs mainly in minimalism and ease of use. Its main functions are weak coupling, good documentation, user friendliness, fair comparison of algorithms, quick overview of results and friendly support community. NiaPy project is designated as open source and licensed under an MIT license. Because the framework is developed in Python programming language, installation is possible on all systems, which have the support for the language and installed PIP package manager. Due to further development, new algorithms are being added to the framework and the previously implemented algorithms are being further improved [13].

One of the algorithms implemented in NiaPy is also PSO, which is a population stochastic optimization algorithm and belongs to the EA group. The algorithm, which is based on the behavior of the swarms of animals such as fish and birds was first developed by James Kennedy and Russel Eberhart in the mid-90s of the 20th century. It was created as a by-product of the desire to graphically represent various flight patterns of animals [6], [8], [14], [15].

The PSO population is referred to as a swarm and instances inside of it flying particles, which are constantly moving inside of a hyperdimensional search space. The position between particles is determined with social-psychology tendency to be better and to imitate other, closer instances [6], [8]. Each particle is moving with its own speed, knows its previous locations and never goes extinct, because there is no selection, mutation or recombination [10]. Velocity is changed in every generation/iteration. For changing velocity, we have three different parts, namely previous velocity, cognitive component and social component

First component represents the memory of the previous direction motion and prevents the current flight direction from drastically changing. Second component expresses the performance of the current particle with respect to past results. Lastly, third component expresses the performance of the current particle with respect to some group of particles or neighbors around it [8].

During the operation, we need to store some other information like information about the best found location of each particle (pbest), information about the best found location of the currently selected part of the swarm (lbest) and information about the best found location of the particle of any swarm (gbest). When finding new top locations, current values need to be updated [6], [8]. The PSO algorithm uses fitness function for guidance of the search over the search space. When the stopping condition is meet, algorithms returns global best solution found [10].

3. IMPLEMENTATING PSO FOR MDVRP

For the purpose of the experiment we used programming language Python to develop a system, which allows application of any EA from NiaPy library to the different MDVRP examples. The system can handle CSV cases of VRP made by Cordeau [12].

The system consists of several different classes, one of which is class Evaluation, which is responsible for solving the problem. In the class we firstly convert given genotype from imported EA to an appropriate phenotype, which can be then used to tackle the problem. First genotype to phenotype conversion assigns ascending index number to each gene, according to its value in the array. Second conversion assigns genes, representing nodes or customers, to specific vehicles based on the value scale, made of number of depots.

The fitness function of the program itself is quite simple, since it only adds up all the paths made to the total distance. This then represents the fitness value of the current instance. It is also important to add penalty to the result, if solution violated any of the limitations of the MDVRP. This is done by checking the limits during the program operation and in case of a violation, send the current result into the penalty function, which then adds a certain distance unit to the distance already completed.

Another important class is Graph, which is responsible for drawing different graphs and connecting nodes of customers and depots on it with paths. Each customer and depot object have its own coordinate pair x and y, which is then plotted on a graph and connected to paths, obtained from the current result object. The final graph thus illustrates all the paths made by vehicles between all the customers and depots. The class can also draw a final bar graph of all fitness values across generations, received through the objects of all solutions. Image Class on the other hand, captures an image from a drawn graph and saves it to the appropriate directory. It can also use previously stored images to generate animated gifs that show the composition of the found route.

The program itself starts in its main function, where we specify desired parameters, such as population size, number of generations to be made, number of instances inside one generation, seed value and genotype to phenotype conversion. Results are displayed on the console at the end of the solving.

4. RESULTS OF THE EXPERIMENT

The experiment we conducted was performed on five MDVRP cases and with five competitive evolutionary algorithms (Particle Swarm Optimization **PSO**, Evolutionary Strategy **ES**, Genetic Algorithm **GA**, Differential Evolution **DE** and Harmony Search **HS**), using settings of 10 generations, 5 instances and 20 units of distance as the penalty value. Each of five test cases was run with a random seed value between 1.000 and 10.000 and with first genotype to phenotype conversion. Unfortunately, the testing was only performed once due to poor hardware capabilities, which were processor Intel Core i5-6267U 3.300Ghz, graphic card Intel Iris Graphics 550, Kingston 8Gb RAM and Liteon 250Gb SSD. The experiment was performed on the Linux operating system Ubuntu 18.04.

The exact NiaPy algorithms, which were used in the testing are GeneticAlgorithm for GA, EvolutionStrategyMpL for ES, DifferentialEvolution for DE, ParticleSwarmAlgorithm for PSO and HarmonySearch for HS. Settings of individual evolutionary algorithm were left at default values from the NiaPy framework.

When testing on the first and simplest example of the problem *pr00 (2 depots/vehicles and 10 customers)*, PSO fund the fourth best route, which is not exactly good. It was overtaken by all the algorithms except the ES, which achieved an even worse fitness result. Comparing on the execution time of the algorithms, the PSO achieved the best value, although it was quite like the DE and HS values. All the results described are shown in Table 1.

Table 1. Test results of solving the example *pr00*

Algorithm	Fitness (unit of distance)	Run time (seconds)
GA	688.02	168.59
ES	774.52	155.93
DE	711.88	148.61
PSO	749.02	148.43
HS	679.10	148.78

GA achieved the best fitness value in solving the second MDVRP case *pr04 (4 depots/vehicles and 192 customers)*. Again, the PSO found the fourth best route with 13603.86 units of distance, and the worst path was found by the ES. This time, PSO solved the problem with the second fastest time, as it was overtaken by the DE for only one second. Results are shown in Table 2.

Table 2. Test results of solving the example *pr04*

Algorithm	Fitness (unit of distance)	Run time (seconds)
GA	13282.01	3906.96
ES	13640.05	3808.14
DE	13476.95	3713.49
PSO	13603.86	3714.37
HS	13573.96	3865.54

PSO achieved the second-best fitness value when solving the third MDVRP case - *pr08 (6 depots/vehicles and 144 customers)*, which is interesting because the example was a bit easier that he previous one. It also solved the problem as the fastest optimization algorithm of all tested. Overall, the algorithm proved to be a good choice for solving the specific case. Test results are recorded in Table 3.

Table 3. Test results of solving the example *pr08*

Algorithm	Fitness (unit of distance)	Run time (seconds)
GA	13282.01	3906.96
ES	13640.05	3808.14
DE	13476.95	3713.49
PSO	13603.86	3714.37
HS	13573.96	3865.54

The fourth MDVRP case *pr14 (4 depots/vehicles and 192 customers)* was similar in complexity to the second, but with one depot less. PSO solved the problem well again, with its fitness result reaching second place, and ES reaching last place. With 3709.32 seconds, PSO solved the problem fastest once again. All the results can be seen in Table 4.

Table 4. Test results of solving the example *pr14*

Algorithm	Fitness (unit of distance)	Run time (seconds)
GA	13723.88	3896.00
ES	13885.56	3732.49
DE	13494.85	3755.60
PSO	13500.67	3709.32
HS	13873.96	3950.66

The toughest test case *pr20 (6 depots/vehicles and 288 customers)* was with 21085.43 units of distance best resolved by PSO – the rest of the algorithms were left behind by about 200 units of distance or more. It achieved the third-best computing time, and interestingly GA achieved first, although its fitness value wasn't very good. All the test results are shown in Table 5.

Table 5. Test results of solving the example *pr20*

Algorithm	Fitness (unit of distance)	Run time (seconds)
GA	21610.55	7341.09
ES	21397.89	7345.13
DE	21224.04	7487.69

PSO	21085.43	7461.45
HS	21293.09	7656.94

If we look at Table 6, we can see the rankings of the PSO algorithm in fitness scores and runtime relative to the other algorithms. The PSO reached an average of 2.6 for fitness rankings and 1.6 for the runtime, and thus solved MDVRP cases the best of all the tested algorithms. It handled more difficult cases better but achieved the fastest resolution times for all of the MDVRP examples.

We ran the test cases again with second conversion of genotype to phenotype but did not get any different results – all fitness scores and running times have overall deteriorated.

Table 6. The PSO algorithm ranks

Order of place	pr00	pr04	pr08	pr14	pr20
Fitness	4	4	2	2	1
Run time	1	2	1	1	3

5. CONCLUSIONS

In this paper we present the application of particle swarm optimization algorithm with the usage of NiaPy optimization framework on the multi-depot capacitated vehicle routing problem. Our approach differs from the similar relevant approaches in the way we represent the optimization problem. In the literature it is normal to treat routing problems as discreet optimization problems. As this was not possible with the usage of NiaPy optimization framework, we presented a method on how to solve MDVRP as the continuous optimization problem.

The proposed method was tested on several standard MDVRP benchmark sets and the results of PSO were compared with several evolutionary algorithms. The results of the experiment show that PSO of continuous optimization is a viable and competitive method for solving MDVRP, especially in the speed of the optimization – it was the fastest in three cases out of five sets. Our proposed PSO reached the best (shortest) route in only one case and it resulted with second shortest routes in two other cases. Thus, we can conclude that PSO can effectively solve MDVRP problem with competitive solutions in fastest run times out of all five included algorithms.

Future work includes the implementation of advanced PSO operators from the referenced literature and customizing them for the continuous optimization. Also, there are numerous other VRP variants, which should be tested with our proposed approach.

6. ACKNOWLEDGMENTS

The authors acknowledge financial support from the Slovenian Research Agency (Research Core Funding No. P2-0057).

7. REFERENCES

[1] W. Cao and W. Yang, “A Survey of Vehicle Routing Problem,” MATEC Web Conf., vol. 100, pp. 1–6, 2017.

[2] I.-M. Chao, E. Wasil, and B. L. Golden, “A new heuristic for the multi-depot vehicle routing problem that improves upon best-known solutions,” Am. J. Math. Manag. Sci., vol. 13, no. 3–4, pp. 371–406, 1993.

[3] W. Ho, G. T.S. Ho, P. Ji, and H. C.W. Lau, “A hybrid genetic algorithm for the multi-depot open vehicle routing problem,” Eng. Appl. Artif. Intell., vol. 21, pp. 401–421, 2008.

[4] S. Karakatič and V. Podgorelec, “A survey of genetic algorithms for solving multi depot vehicle routing problem,” Appl. Soft Comput. J., vol. 27, pp. 519–532, 2015.

[5] G. Laporte, M. Gendreau, J.-Y. Potvin, and F. Semet, “Classical and modern heuristics for the vehicle routing problem,” Int. Trans. Oper. Res., vol. 7, pp. 285–300, 2000.

[6] S. Luke, Essentials of Metaheuristics, Second Edi. 2013.

[7] B. M. Baker and M. A. Ayechew, “A genetic algorithm for the vehicle routing problem,” Comput. Oper. Res., vol. 30, no. 5, pp. 787–800, 2003.

[8] A. P Engelbrecht, “Computational Intelligence.” p. 597, 2007.

[9] A. Shukla, R. Tiwari, and R. Kala, Real Life Applications of Soft Computing. 2012.

[10] R. Storn and K. Price, “Differential Evolution – A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces,” J. Glob. Optim., pp. 341–359, 1997.

[11] P. Surekha, Sumathi, and Dr.S., “Solution To Multi-Depot Vehicle Routing Problem Using Genetic Algorithms,” World Appl. Program., vol. 1, no. 3, pp. 118–131, 2011.

[12] N. and E. O. G. University of Málaga, “Multiple Depot VRP with Time Windows Instances,” 2013. [Online]. Available: http://neo.lcc.uma.es/vrp/vrp-instances/multiple-depot-vrp-with-time-windows-instances. [Accessed: 31-Aug-2019].

[13] G. Vrbančič, L. Brezočnik, U. Mlakar, D. Fister, and I. Fister Jr., “NiaPy: Python microframework for building nature-inspired algorithms,” J. Open Source Softw., vol. 3, p. 613, 2018.

[14] X.-S. Yang, “Firefly algorithms for multimodal optimization,” Springer-Verlag Berlin Heidelb., vol. 5792 LNCS, pp. 169–178, 2009.

[15] X.-S. Yang and X. He, “Firefly algorithm: recent advances and applications,” Int. J. Swarm Intell., vol. 1, no. 1, pp. 36–50, 2013.

[16] Mohemmed, A.W., Sahoo, N.C. and Geok, T.K., 2008. Solving shortest path problem using particle swarm optimization. Applied Soft Computing, 8(4), pp.1643-1653.

[17] Ai, T.J. and Kachitvichyanukul, V., 2009. A particle swarm optimization for the vehicle routing problem with simultaneous pickup and delivery. Computers & Operations Research, 36(5), pp.1693-1702.

[18] Yao, B., Yu, B., Hu, P., Gao, J. and Zhang, M., 2016. An improved particle swarm optimization for carton heterogeneous vehicle routing problem with a collection depot. Annals of Operations Research, 242(2), pp.303-320.

[19] Kumar, R.S., Kondapaneni, K., Dixit, V., Goswami, A., Thakur, L.S. and Tiwari, M.K., 2016. Multi-objective modeling of production and pollution routing problem with time window: A self-learning particle swarm optimization approach. Computers & Industrial Engineering, 99, pp.29-40.

[20] Norouzi, N., Sadegh-Amalnick, M. and Tavakkoli-Moghaddam, R., 2017. Modified particle swarm optimization in a time-dependent vehicle routing problem: minimizing fuel consumption. Optimization Letters, 11(1), pp.121-134.