A multioperator genetic algorithm for the traveling salesman problem with job-time

Pablo Gutiérrez-Aguirre
Departamento de Ingeniería Industrial
Universidad de Concepción
Concepción, Chile
pgutierrez2018@udec.cl

Carlos Contreras-Bolton
Departamento de Ingeniería Industrial
Universidad de Concepción
Concepción, Chile
carlos.contreras.b@udec.cl

***Abstract*—The traveling salesman problem consists of finding the optimal route for a given set of nodes to minimize the time traveled. This problem has many variants, one of them was recently proposed by Mosayebi et al. (2021), called Traveling Salesman Problem with Job-Time (TSPJ). TSPJ considers two sets of equal size, a set of tasks and a set of nodes, where each task is assigned to only one node, and each of these tasks is completed with a job time that depends on each node. The objective is to minimize the time of the last task performed. Among its applications are equipment maintenance, highly automated manufacturing, agricultural harvesting, disaster recovery, among others. Existing algorithms do not optimally solve the larger instances of the TSPJ due to its NP-hardness. Therefore, we propose an approach based on a multioperator genetic algorithm that works with several crossover and mutation operators. Our approach works with three crossover operators, which are the classical operators for permutation, and five mutation operators are considered, where two of them are local searches. The results are evaluated on four sets of instances ranging in size from 17 to 1000 nodes: classical, small, medium, and large. Our approach outperforms the best algorithm in the quality of the solutions with a short computing time in the first set. Meanwhile, the multioperator genetic algorithm obtains a good performance in small instances. Finally, for the medium and large instances, our approach achieves promising results for both instances sets, both the quality of the solutions and the computing times.**

***Index Terms*—Traveling salesman problem, Scheduling problem, Genetic algorithm, Multioperator**

I. Introduction

The traveling salesman problem consists of finding the optimal route for a given set of nodes to minimize the time traveled. This problem has many variants, one of them was recently proposed by Mosayebi et al. (2021) [1], called traveling Salesman Problem with Job-Time (TSPJ). TSPJ considers a set of n tasks and a set of n nodes, where each task is assigned to only one node, and each of these tasks is completed with a job time that depends on each node. The objective is determine the schedule that obtains the minimization of the maximum completion time of the schedule, that is known as makespan or C_{max}. Among its applications are equipment maintenance, highly automated manufacturing, agricultural harvesting, disaster recovery, among others. Existing algorithms do not optimally solve the larger instances of the TSPJ due to its NP-hardness. Therefore, we propose an approach based on a multioperator genetic algorithm (MGA) that works with several crossover and mutation operators. The proposed algorithm is evaluated on four sets of instances ranging in size from 17 to 1000 nodes.

The remainder of this work is organized as follows: Section II describes the details of the proposed algorithm. Next, the computational results of our approach are presented in Section III. Finally, the conclusions of the study are presented in Section IV.

II. Multioperator genetic algorithm

In the MGA, each individual corresponds to a double chromosome with a Hamiltonian path and a sequence of tasks. Thus, both chromosomes are represented by a permutation of nodes and tasks, respectively. MGA works as a simple genetic algorithm [2], described as follows:

Initially, a population with feasible solutions is built, and in each new generation, new solutions are created using the selection, crossover, and mutation operators. The selection is made in a tournament of four individuals. The crossover considers three operators, and five operators participate in the mutation. Elitism is also considered, in which the best current parents replace 10% of the worst individuals generated in each generation. The above process is repeated until a given maximum number of generations is reached.

A. Crossover

For the crossover operators, the classic operators are used, where each one has a probability of execution, these operators are:

- Ordered crossover (OX): It generates holes in the input individuals. A hole is created when an attribute of an individual is between the two crossover points of the other individual. Then, it rotates the element so that all holes are between the crossover points and fills them with the removed elements in order [3].
- Partially matched crossover (PMX): It generates two children by matching pairs of values in a certain range of the two parents and swapping the values of those indexes [4].
- Uniform partially matched crossover (UPMX): It generates two children by matching pairs of values chosen at

TABLE I
RESULTS

Instance	C_{max}						Time [s]				
	BKS	I	II	III	IV	AG	I	II	III	IV	AG
Classic	3732.84	3823.31	3897.5	3819.61	3838.22	3802.01	2.33	2.45	3.51	4.26	4.93
Small	-	289.93	292.74	289.34	289.39	297.62	0.98	0.97	1.57	1.54	6.25
Medium	-	3551.98	3553.12	3544.75	3544.88	3832.26	105.33	114.88	208.51	186.94	62.09
Large	-	13908.57	13921.98	13892.40	13882.74	15093.08	518.38	575.30	1040.25	957.40	178.80

random with a probability of γ in the two parents and swapping the values of those indexes [5], we set $\gamma = 0.5$.

B. Mutation

Three of the mutation operators work on diversifying the solution, and the remaining two intensify the solution through local searches. The mutation operators proposed are:

1) This operator performs a random exchange of two nodes or jobs.
2) This operator reverses the order between two randomly chosen nodes.
3) This operator performs an exchange between a random node and its successor.
4) The first one tries to improve the route and is based on a 2-opt [6], which uses a stopping criterion that when a better route is found, it stops.
5) Iteratively traverses the jobs in such a way that at the node in which the current C_{max} is found, it is exchanged with the jobs of the rest of the nodes, this considering that the times change for each node, stopping when it finds an improvement in the C_{max}.

III. RESULTS

To test the computational performance of the MGA, we use the instances proposed by Mosayebi et al. [1]. Each problem instance consists of a list of nodes and a list of jobs such that the number of nodes and jobs are equal. In addition, the distance between nodes and the time of jobs are specified. There are four sets of instances, where the first one consists of a set based on TSPLIB data for TSP. The rest of the sets correspond in small, medium, and large instances, with the number of nodes and jobs ranging from 44 to 1000.

Table I contains 12 columns. The first column corresponds to the name of the set instance. The second column shows the best known solution (BKS) delivered by the mixed integer programming model. The next five columns show the results of each algorithm proposed by Mosayebi et al. [1], the last one corresponds to the MGA. The last five columns show the computing times of each algorithm.

In the first set, MGA outperforms the best algorithm in the quality of the solutions with a short computing time. Meanwhile, the MGA obtains a good performance in small instances. Finally, for the medium and large instances, our approach achieves promising results for both instances sets, both the quality of the solutions and the computing times.

IV. CONCLUSION

This work proposed a multioperator genetic algorithm with several crossover and mutation operators. Our algorithm used three crossover operators, classical operators for permutation, and five mutation operators considered, two of which are local searches. The results are evaluated on four sets of instances ranging in size from 17 to 1000 nodes. The performance of our approach obtains good results in the classical and small instances. While for the medium and large instances, the performance achieves promising results for both instances sets, both the quality of the solutions and the computing times.

ACKNOWLEDGMENT

This research was partially supported by VRID – INICIACIÓN 220.097.016-INI, Vicerrectoría de Investigación y Desarrollo (VRID), Universidad de Concepción.

REFERENCES

[1] M. Mosayebi, M. Sodhi, and T. A. Wettergren, "The traveling salesman problem with job-times (tspj)," *Computers & Operations Research*, vol. 129, p. 105226, 2021.

[2] A. Eiben and J. Smith, *Introduction to Evolutionary Computing*, ser. Natural Computing Series. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015.

[3] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, 1st ed. USA: Addison-Wesley Longman Publishing Co., Inc., 1989.

[4] D. E. Goldberg and R. Lingle, "Alleleslociand the traveling salesman problem," in *Proceedings of the 1st International Conference on Genetic Algorithms*. USA: L. Erlbaum Associates Inc., 1985, p. 154–159.

[5] V. A. Cicirello and S. F. Smith, "Modeling ga performance for control parameter optimization," in *Proceedings of the 2nd Annual Conference on Genetic and Evolutionary Computation*, ser. GECCO'00. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2000, p. 235–242.

[6] G. A. Croes, "A method for solving traveling-salesman problems," *Operations Research*, vol. 6, no. 6, pp. 791–812, 1958.