Optimizing a GPU-Accelerated Genetic Algorithm for the Vehicle Routing Problem

Marwan Abdelatti
Department of Industrial Engineering
University of Rhode Island
Kingston, Rhode Island
mabdelrazik@uri.edu

Abdeltawab Hendawi
Department of Computer Science
University of Rhode Island
Kingston, Rhode Island
hendawi@uri.edu

Manbir Sodhi
Department of Industrial Engineering
University of Rhode Island
Kingston, Rhode Island
sodhi@uri.edu

ABSTRACT

The capacitated vehicle routing problem (CVRP) is an NP-hard optimization problem with many applications. Genetic algorithms (GAs) are often used to solve CVRPs but require many parameters and operators to tune. Incorrect settings can result in poor solutions. In this work, a design of experiments (DOE) approach is used to determine the best settings for GA parameters. The GA runs entirely on an NVIDIA RTX 3090 GPU. The GPU execution for a 200-node benchmark shows a speed by a factor of 1700 compared to that on an octa-core i7 CPU with 64 GB RAM. The tuned GA achieved a solution for a 400-node benchmark that is 72% better than that of an arbitrarily tuned GA after only 263 generations. New best-known values for several benchmarks are also obtained. A comparison between the performance of the algorithm with different hardware and tuning sets is also reported.

CCS CONCEPTS

• **Computing methodologies** → **Massively parallel algorithms**; • **Applied computing** → **Operations research**; • **Mathematics of computing** → **Combinatorics**; **Solvers**; *Multivariate statistics*;

KEYWORDS

Vehicle routing problem, genetic algorithms, design of experiment, factorial design, local search, GPU, parallel computation

ACM Reference Format:
Marwan Abdelatti, Abdeltawab Hendawi, and Manbir Sodhi. 2021. Optimizing a GPU-Accelerated Genetic Algorithm for the Vehicle Routing Problem. In *2021 Genetic and Evolutionary Computation Conference Companion (GECCO '21 Companion), July 10–14, 2021, Lille, France.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3449726.3459458

1 INTRODUCTION

Genetic algorithms (GAs) are commonly used for the capacitated vehicle routing problems (CVRP) to find near-optimal delivery/pick-up routes through a set of locations [2, 4, 8]. Because of the large number of parameters and operators used in the GA, and the limited knowledge about the effect of parameter interactions [7], a two-level 2^k factorial design of experiment (DOE) is used to determine the best settings for the GA parameters. Pilot runs are performed on a small problem settings for the parameters are determined. The tuned parameters are then applied to larger problems. The developed GA (GPU-GA) runs entirely on graphics processing units (GPUs) to capitalize their parallel execution capabilities [1]. The tuned algorithm obtained an 8.89% gap from the best-known solution for a 200-node benchmark problem *M-n200-k17* [3] after 50,000 generations, A 72% improvement in the gap for a 400-node benchmark compared to arbitrary parameter sets within 263 generations only, and new best-known values for other problems. Compared to sequential executions on CPUs, GPUs achieved an improvement factor of 1700 in the execution speed.

2 GPU-ACCELERATED GA FOR CVRP

GPU-GA is a GA algorithm with a 2-opt local search. A clone-restricting procedure preserves the diversity of population throughout the evolution process. The GPU grid is arranged with a total of $(B * T)^2$ threads running in parallel where $B \times B$ is the 2D array arrangement of the GPU blocks and $T \times T$ are the thread/block arrangement, B and T are integers. A block diagram of the algorithm is shown in Figure 1 [1]. The algorithm starts by using the CPU for reading problem data, and then creates and copies an array of nodes to the GPU. The GPU creates the cost table array between nodes. Evolutionary processes like random initialization, selection, crossover, and mutation as well as the 2-opt local search run on the

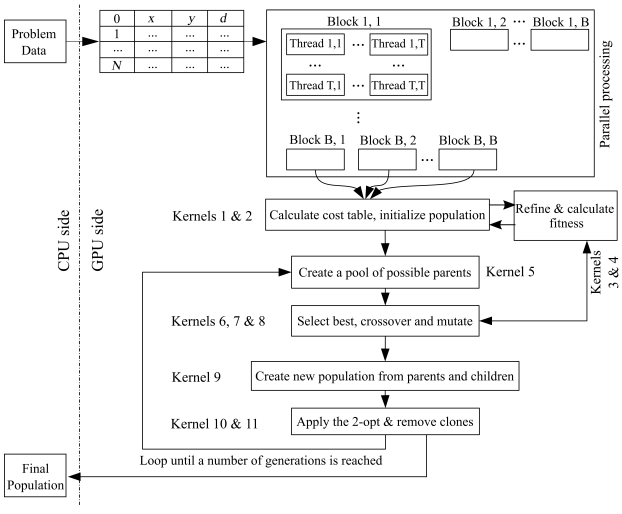

Figure 1: Algorithm implementation on GPU.

GECCO '21 Companion, July 10–14, 2021, Lille, France

ACM ISBN 978-1-4503-8351-6/21/07.
https://doi.org/10.1145/3449726.3459458

Table 1: Results of running the tuned GPU-GA

		on 2080 Ti		on 3090		on CPU	
Problem ID	Best Known	%Gap	Speed (s/loop)	%Gap	Speed (s/loop)	%Gap	Speed. (s/loop)
P-n16-k8	450	**0.0000**	0.1006	**0.0000**	0.0397	**0.0000**	0.7936
P-n20-k2	216	**0.0000**	0.2392	**0.0000**	0.0837	18.042	9.23
P-n22-k8	603	**0.0000**	0.1205	**-2.1559**	0.0479	3.0182	4.08
P-n23-k8	529	**0.0000**	0.178	**0.0000**	0.0524	3.4801	12.62
B-n31-k5	672	**0.0000**	0.2145	**0.0000**	0.0905	12.0074	20.86
P-n40-k5	458	3.0568	0.1878	0.2183	0.0831	40.0436	27.39
B-n50-k8	1312	1.6768	0.1992	1.1433	0.0827	34.0114	21.54
P-n55-k15	989	**-2.9323**	0.4489	**-3.4378**	0.1598	21.0141	36
P-n65-k10	792	1.7677	0.4566	5.0505	0.1823	52.1022	25.34
P-n70-k10	827	7.7388	0.4655	4.4740	0.1885	56.0785	42.03
A-n80-k10	1763	8.9052	1.5229	7.2603	0.1848	80.1497	63.67
E-n101-k14	1071	20.168	1.8299	7.5630	0.2386	100.1699	85.51
M-n200-k17	1373	99.4901	2.8063	8.3029	0.9387	201.9876	1592.14
Golden_3 (400 nodes)	11063.22	36.6774	12.3702	13.8616	4.7246	401.4459	9240

GPU until reaching a stopping criteria: either stagnating at a cost value for a certain number of generations, or, reaching a limit on the number of generations (in case of performance analysis). The final solution is sent back to the CPU for output. Interested readers can clone the GPU-GA from the GitHub repository[1].

3 DESIGN OF EXPERIMENT

To obtain high-quality solutions from the proposed algorithm, multiple GA parameters with different values have been collected from various studies [5, 6, 9]. A 2^k factorial design of experiment (DOE) is utilized on a 70-node *(P-n70-k10)* benchmark to analyze different parameter combinations and their interactions. Each combination is independently run five times. The parameters considered are: **crossover operator**: 1-point or 2-point crossover; **mutation operator**: either inversion mutation where a random range of genes is inverted, or swap mutation that switches two random genes; **population size**: 10× or 20× the number of nodes n. **elitism rate**: the percentage of parents transferred to the new generation along with the offspring; and **crossover and mutation rates**: the occurrence probabilities of crossover and mutation during the evolution process respectively.

It was found from the experiment that the elitism rate has no significant effect on the GA behavior.

4 EXECUTION RESULTS AND CONCLUSIONS

The tuned GPU-GA is used to solve a number of benchmark problems. The %Gap between the obtained and the best-known solution is the measure of solution quality where
$\%Gap = (obtained\ soln - best\ known)\ /best\ known.$
The stopping criteria is reaching or exceeding the best-known value or solution stagnation. The CPU is an Intel Octa-core i7 CPU @ 2.4 GHz and 64 GB RAM, and an NVIDIA RTX 3090 GPU with 24 GB of memory. The GA executes on 35×35 blocks with 20×20 threads per block. The obtained solution quality is compared with solutions obtained with random parameter settings. The optimal parameters for the GA were: 1-point crossover, inversion mutation, a $10 \times n$ population size, a crossover rate of 0.8, and a mutation rate of 0.1.

Numerical results show that the tuned parameter set results in faster convergence and no prmature termination. The optimized algorithm is run on different hardware (e.g., CPU, NVIDIA RTX 3090 GPU with 10,496 cores and 24 GB memory, and RTX 2080Ti with 4,352 cores and 11 GB memory). The results are shown in Table 1. It is found that the number of CUDA cores has a great impact on the GA performance - more than double the speed is achieved on the 3090 GPU compared to the 2080Ti. Worth mentioning is that for a 200-node problem, the algorithm executes at a speed of 26 min/generation on the CPU compared with 0.9 sec/generation on the RTX 3090 GPU, reducing the execution time from 2.4 years on the CPU to 12 hours on GPU for a similar result quality.

[1] https://github.com/MarwanAbdelatti/GA_VRP_mod

REFERENCES

[1] Marwan F Abdelatti and Manbir S Sodhi. 2020. An improved GPU-accelerated heuristic technique applied to the capacitated vehicle routing problem. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*. 663–671.

[2] Barrie M Baker and MA Ayechew. 2003. A genetic algorithm for the vehicle routing problem. *Computers & Operations Research* 30, 5 (2003), 787–800.

[3] Nicos Christofides and Samuel Eilon. 1969. An algorithm for the vehicle-dispatching problem. *Journal of the Operational Research Society* 20, 3 (1969), 309–318.

[4] Paul C Chu and John E Beasley. 1997. A genetic algorithm for the generalised assignment problem. *Computers & Operations Research* 24, 1 (1997), 17–23.

[5] Agoston Endre Eiben and Selmar K Smit. 2011. Evolutionary algorithm parameters and methods to tune them. In *Autonomous search*. Springer, 15–36.

[6] Giorgos Karafotias, Mark Hoogendoorn, and Ágoston E Eiben. 2014. Parameter control in evolutionary algorithms: Trends and challenges. *IEEE Transactions on Evolutionary Computation* 19, 2 (2014), 167–187.

[7] Mohsen Mosayebi and Manbir Sodhi. 2020. Tuning genetic algorithm parameters using design of experiments. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion*. 1937–1944.

[8] Christian Prins. 2004. A simple and effective evolutionary algorithm for the vehicle routing problem. *Computers & Operations Research* 31, 12 (2004), 1985–2002.

[9] Selmar Kagiso Smit. 2012. *Parameter tuning and scientific testing in evolutionary algorithms*. Vrije Universiteit.