V-ACO: A vectorization approach for high-performance Ant Colony Optimization

José M. Cecilia[1] Antonio Llanes[1], Li-Wen Chang[2], José M. García[3], Nacho Navarro[4], and Wen-Mei Hwu[2]

[1] BIOHPC group, Computer Science Department, Universidad Católica San Antonio de Murcia (Spain)
{jmcecilia, allanes}@ucam.edu
[2] University of Illinois at Urbana-Champign, Center for Reliable and High-Performance Computing, Illinois (USA)
{lchang20,w-hwu@illinois.edu}
[3] GACOP group, DITEC, Universidad de Murcia (Spain)
jmgarcia@ditec.um.es
[4] Barcelona Supercomputing Center, Universitat politecnica de Catalunya, Barcelona (Spain)
nacho@ac.upc.edu

1 Introduction

Ant colony optimization (ACO) [1–3] is a metaheuristic based on foraging behavior observed in colonies of real ants, and has been applied to a wide variety of problems, including vehicle routing [4], feature selection [5] and autonomous robot navigation [6]. The method generally uses simulated "ants" (i.e., mobile agents), which first construct tours or paths on a network structure (corresponding to solutions to a problem), and then deposit "pheromone" (i.e., signalling chemicals) according to the quality of the solution generated. The algorithm takes advantage of emergent properties of the multi-agent system, in that positive feedback (facilitated by pheromone deposition) quickly drives the population to high-quality solutions.

Parallel versions of ACO have been developed [7–10] (see also [11] for a survey), and, in recent work, we present a graphics processor unit (GPU)-based version of ACO that, for the first time, parallelizes *both* main phases of the algorithm (that is, tour construction and pheromone deposition), placing more emphasis on *data parallelism* [12, 13]. However, the parallel designs for the ACO algorithm proposed so far on these architectures are either: task-based approaches [11] which compromise the GPU parallelism or highly computationally demanding to avoid serialization [12, 13]. As mentioned in [14], there are some issues with the latter approach:

– The reduction is performed on the entire thread block and this prohibits synchronization between warps so that idle warps result.
– The random numbers generated can decrease the influence of the heuristic and the pheromone information, and thus it can lead to a reduction in the quality of the tours generated.
– A random number must be generated for each thread for n iterations in order to build a complete tour.

We rethink the parallelization strategies for the ACO algorithm that solves the Travelling Salesman Problem (TSP) on massively parallel architectures. Of particular relevance to us are attempts to parallelize the *tour construction* stage as it is the most challenging and irregular part of ACO-based algorithms. In this work, we propose a vectorization-based design which is based on basic parallel primitives such as prefix scan, stencil and reduction to leverage the GPU's massively parallel architecture. Our design identifies an ant as a 32-width vector (or warp in CUDA), evenly distributing them among thread blocks. This design enhances the application parallelism, but also centralizes all operations performed by each ant at warp level, and thus the algorithm can benefit from new *shuffle instructions* that allow threads within the same warp to communicate with each other. This kind of instructions are available in the last generation of Nvidia GPUs and proposed in the new upcoming OpenCL standard 2.0. Moreover, we introduce an *SS-Roulette* method (Scan-Stencil Roulette) as a new way to implement the classic Roulette Wheel selection while improving GPU parallelism. SS-Roulette consists of reproducing the traditional roulette wheel selection procedure, but now in parallel. SS-Rouletter only generates a random number per each run of ACO algorithm to feed into the stochastic simulation. Thus, the probabilities and tabu values (a value showing whether the city has been visited or not) are multiplied and stored in the shared memory array before prefix scan is developed on this in-place array. Then, a stencil checking is performed on the prefix scan array to decide which city is the next city to go.

Our major contributions include the following:

1. To the best of our knowledge, this is the first vectorization-based parallelism scheme on massively parallel architectures for the ACO tour construction stage. Our design identifies an ant as a 32-width vector (or warp in CUDA), evenly distributing them among thread blocks.
2. We introduce an SS-Roulette method (Scan-Stencil Roulette) as a new way to implement the classic Roulette Wheel selection while improving GPU parallelism.
3. We evaluate the shuffle instructions that allow threads within the same warp to communicate with each other, available in the last generation of Nvidia GPUs and proposed in the new upcoming OpenCL standard 2.0.
4. We offer an in-depth analysis of both stages of the ACO algorithm for different instances of the TSP problem. Several GPU parameters are tuned to reach a speed-up factor of up to 3x for the tour construction stage compared to the best GPU implementation previously published.
5. The solution accuracy obtained by our GPU algorithm is comparable to that of the sequential counterpart version using TSPLIB.

Acknowledgment

This work has been jointly supported by a collaboration grant from the EU FP7 NoE HiPEAC ICT-287759, the European Network of Excellence on High Performance and Embedded Architecture and Compilation, the Fundación Séneca (Agencia Regional de Ciencia y Tecnología de la Región de Murcia) under grant 15290/PI/2010, by the Spanish MINECO and the European Commission FEDER funds under grant TIN2012-31345, and by Universidad Católica Sant Antonio de Murcia (UCAM) under grant PMAFI/26/12. We also thank Nvidia for hardware donation under CUDA Teaching Center 2014.

References

1. Dorigo, M., Birattari, M., Stutzle, T.: Ant colony optimization. Computational Intelligence Magazine, IEEE **1** (2006) 28–39
2. Dorigo, M., Di Caro, G.: Ant colony optimization: A new meta-heuristic. In: Proceedings of the 1999 Congress on Evolutionary Computation (CEC'99), IEEE Press (1999) 1470–1477
3. Dorigo, M., Stützle, T.: Ant colony optimization: overview and recent advances. In: Handbook of metaheuristics. Springer (2010) 227–263
4. Yu, B., Yang, Z.Z., Yao, B.: An improved ant colony optimization for vehicle routing problem. European Journal of Operational Research **196** (2009) 171–176
5. Chen, Y., Miao, D., Wang, R.: A rough set approach to feature selection based on ant colony optimization. Pattern Recognition Letters **31** (2010) 226–233
6. García, M.P., Montiel, O., Castillo, O., Sepúlveda, R., Melin, P.: Path planning for autonomous mobile robot navigation with ant colony optimization and fuzzy cost function evaluation. Applied Soft Computing **9** (2009) 1102–1110
7. Delévacq, A., Delisle, P., Gravel, M., Krajecki, M.: Parallel ant colony optimization on graphics processing units. Journal of Parallel and Distributed Computing **73** (2013) 52–61
8. Manfrin, M., Birattari, M., Stützle, T., Dorigo, M.: Parallel ant colony optimization for the traveling salesman problem. In: Ant Colony Optimization and Swarm Intelligence. Springer (2006) 224–234
9. Stützle, T.: Parallelization strategies for ant colony optimization. In: Parallel Problem Solving from Nature (PPSN V), Springer (1998) 722–731
10. Zhu, W., Curry, J.: Parallel ant colony for nonlinear function optimization with graphics hardware acceleration. In: Systems, Man and Cybernetics, 2009. SMC 2009. IEEE International Conference on, IEEE (2009) 1803–1808
11. Pedemonte, M., Nesmachnow, S., Cancela, H.: A survey on parallel ant colony optimization. Applied Soft Computing **11** (2011) 5181–5197
12. Cecilia, J.M., García, J.M., Nisbet, A., Amos, M., Ujaldón, M.: Enhancing data parallelism for ant colony optimization on GPUs. Journal of Parallel and Distributed Computing **73** (2013) 42–51
13. Cecilia, J.M., García, J.M., Ujaldón, M., Nisbet, A., Amos, M.: Parallelization strategies for ant colony optimisation on GPUs. In: Proceedings of the 2011 IEEE International Symposium on Parallel and Distributed Processing, IEEE (2011) 339–346
14. Dawson, L., Stewart, I.: Improving ant colony optimization performance on the GPU using CUDA. In: Evolutionary Computation (CEC), 2013 IEEE Congress on, IEEE (2013) 1901–1908