On the Effect of Using Multiple GPUs in Solving QAPs with CUDA

Shigeyoshi Tsutsui
Hannan University
5-4-33, Amami-higashi, Matsubara
Osaka 580-8502, Japan
+81-72-332-1224
tsutsui@hannan-u.ac.jp

Noriyuki Fujimoto
Osaka Prefecture University
1-1 Gakuen-cho, Nakaku, Sakai
Osaka 599-8531, Japan
+81-72-252-1161
fujimoto@mi.s.osakafu-u.ac.jp

ABSTRACT
In this paper, we implement ACO algorithms on a PC which has 4 GTX 480 GPUs. We implement two types of ACO models; the island model, and the other is the master/slave model. When we compare the island model and the master/slave model, the island model shows promising speedup values on class (iv) QAP instances. On the other hand, the master/slave model showed promising speedup values both on classes (i) and (iv) with large-size QAP instances.

Categories and Subject Descriptors
I.2.8 [Artificial Intelligence]: Problem Solving, Control Methods, and Search Heuristic Methods; D.1.3 [Programming Techniques]: Concurrent Programming Distributed programming, Parallel programming

General Terms
Algorithms

Keywords
Graphics Processing Units, GPU, Multiple GPUs, QAP, ACO, Tabu Search

1. INTRODUCTION
Recently, Parallel EAs using GPU has become popular. In this paper, we implement the ACO algorithm on a PC which has 4 GTX 480 GPUs to solve the quadratic assignment problem (QAP) fast. We implement two types of ACO models using multiple GPUs.

One is the island model, and the other is the master/slave model. In the island model, we implement one colony on each GPU, agents (solutions) are exchanged among colonies at defined intervals using several types of topologies. In the master/slave model, we have only one colony of ACO in the CPU, and only local search (:Tabu Search, TS) processes are distributed to each GPU. The results of the models are discussed through experiments.

GECCO'12 Companion, July 7–11, 2012, Philadelphia, PA, USA.
ACM 978-1-4503-1178-6/12/07.

2. A REVIEW OF AN ACO WITH TABU SEARCH ON A GPU WITH MATA
MATA was proposed for efficient threads assignment method in solving QAP on a GPU [1]. As is well known, in TS we need to check all solutions neighboring the current solution to obtain the best move. This move-cost calculation is costly. Let $N(\phi)$ be the set of neighbors of the current solution ϕ of a QAP. A neighbor, $\phi' \in N(\phi)$, is obtained by exchanging a pair of elements (i, j) of ϕ. Then, we need to compute move costs $\Delta(\phi, i, j)=cost(\phi')-cost(\phi)$ for all the neighboring solutions where $cost(\phi)$ is the cost function of a solution ϕ of a QAP. When we exchange r-th and s-th elements of ϕ (i.e., $\phi(r)$ and $\phi(s)$), $\Delta(\phi, r, s)$ can be calculated in computing cost $O(n)$ where n is problem size. Let ϕ' be obtained from ϕ by exchanging r-th and s-th elements of ϕ, then fast computation of $\Delta(\phi', u, v)$ is obtained in computing cost $O(1)$ if u and v satisfy the condition $\{u, v\} \cap \{r, s\} = \emptyset$ [2].

Threads in a block in CUDA are executed through a mode called *SIMT* [nvidia2010-programming]. In SIMT, each MP executes threads in groups of 32 parallel threads called *warps*. A warp executes one common instruction at a time, so full efficiency is realized when all 32 threads of a warp agree on their execution path. In MATA, we assign move cost computations of a solution ϕ which are in $O(1)$ and in $O(n)$ to threads which belong to different warps in a block to reduce performance degradation caused by SIMT.

Table 1. Results of ACO on a single GPU

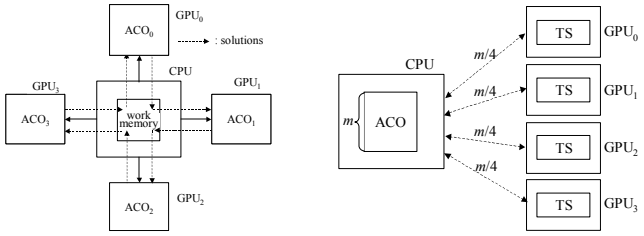
	QAP instances	GPU Computation (GTX 480) T_{avg} (sec) MATA	non-MATA	non-MATA / MATA	CPU (i7 965) T_{avg} (sec)	Speedup in T_{avg} CPU / MATA
class (i)	tai40a	9.2	70.8	7.7	191.4	20.8
	tai50a	17.8	126.6	7.1	464.0	26.1
	tai60a	34.8	283.9	8.2	963.8	27.7
	tai80a	153.9	728.6	4.7	3131.8	20.3
	tai100a	431.5	2357.3	5.5	7907.0	18.3
class (iv)	tai50b	0.2	1.5	6.3	6.0	24.9
	tai60b	0.4	2.8	7.7	12.7	35.5
	tai80b	6.6	33.9	5.1	141.6	21.4
	tai100b	10.1	55.2	5.5	296.5	29.5
	tai150b	2888.4	16348.8	5.7	48953.8	16.9
average		–	–	6.4	–	24.2

Table 1 shows revised results of previous ACO with TS on a GPU (GTX 480) in solving QAPs [1]. CPU is Core i7 965 (3.2 GHz). As for ACO, the agent size m was set to n. In this revised experiment, if a total length of TS in the algorithm IT_{TOTAL} reaches $m \times n \times 3200$ or a known optimal solution is found, the algorithm terminates. 25 runs were performed. In the table, we showed results with MATA, without MATA (non-MATA) on the GPU, and CPU computation. Please note these results are slightly

different from those of in [1] because we revised parameter settings for this study.

3. IMPLEMENTATION OF ACO ON MULTIPLE GPUs

There are four PCIe x16 slots in our system. For this study, we added an additional 3 GTX 480 GPUs and constructed a multi-GPU environment with a total of 4 GTX 480 GPUs. In this section, we propose two types of multi-GPU for ACO with MATA, to attain fast computation speed in solving QAPs. For the parallel models, we chose the island model and the master/slave model, the most popular parallel EA models.



island model master/slave model

Figure 1. ACO on multiple GPUs

ACO uses a set on m agents. This set of agents is similar to a population in GAs. In our implementation, we exchange (i.e., immigrate) the solutions in the set of agents. In our implementation, one ACO model in a GPU in Section 2 composes one island.

Although there are many topologies for island models [3], in this study we implement the following 4 island models; **(1) Island model with independent runs (IM_INDP)**: Four GPUs are executed independently; **(2) Island model with elitist (IM_ELIT)**: At defined interval $I_{interval}$ the CPU collects the global best solution from the 4 GPUs, and then distributes it to all GPUs except the GPU that produced that best solution. In each GPU, the worst solution in each archive is replaced with the received solution; **(3) Island model with ring connected (IM_RING)**: The best solution in each GPU g (g=0, 1, 2, 3) is distributed to its neighbor GPU (g+1) *Mod* 4. In each GPU, the worst solution in each archive is replaced with the received solution if the received one is better than the worst one; **(4) Island model with elitist and massive ring connected (IM_ELMR)**: In this model, we first apply strategy of IM_ELIT. Then, in addition to this operation, randomly selected $m \times d_{rate}$ of solutions in the archive in each GPU are distributed to its neighbor. Received solutions in each GPU are compared with randomly selected, non-duplicate solutions. We use d_{rate} of 0.5 in this study.

In the master/slave model, the ACO is executed in the CPU. When new solutions are generated in the CPU, m/4 number of solutions are transferred to each GPU to improve them by TS as shown in Figure 2.

4. RESULTS

The CPU is the same as in Section 2. As for the GPU, we used 4 GTX 480 GPUs in PCIe. Termination criteria are slightly different from those in Section 2. In this experiment, we run the algorithms until predetermined acceptable solutions are obtained and effectiveness of using multiple GPUs is measured by average time ($T_{avg,4}$) to obtained the solutions. We obtain the speedup by $T_{avg,4}/T_{avg,1}$, where $T_{avg,1}$ is average time to obtain acceptable solutions by ACO using a single GPU. We determined acceptable solutions as follows. For the class (i) instances we used, it is difficult to obtain known optimal solutions 25 times in 25 runs with reasonable run time, so we set the their acceptable solution to be within 0.5% of the known optimal solutions. For the class (iv) instances, except tai150b, we set them to known optimal solutions. We set tai150b to be within 0.2% of the known optimal solution.

Figures 2 and 3 show the results of the island models and the master/slave model.

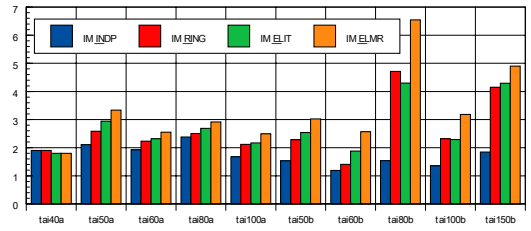


Figure 2. Results of the island models

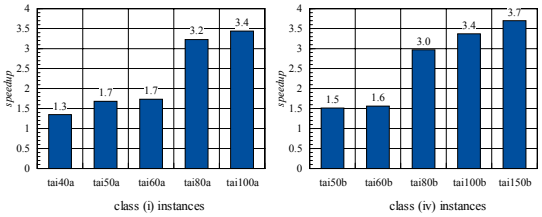


Figure 3. Results of the master/slave model

5. CONCLUDING REMARKS

As for the island model, we used 4 types of topologies. Although the results of speedup much depend on instances, we showed island model with a topology, which we call the island model with elitist and massive ring connected (IM_ELMR), has a good speedup feature. As for the master/slave model, we observed a reasonable speed-ups for large-size of instances, where we use large-number of agents.

When we compare the island model and the master/slave model, the island model showed promising speedup values on class (iv) instances of QAP. On the other hand, the master/slave model showed promising speedup values for large-size instances both on classes (i) and (iv) QAP. As regards to this comparison, a more intensive analytical study is an interesting future research direction.

6. REFERENCES

[1] Tsutsui, S., Fujimoto, N.: ACO with tabu search on a GPU for solving QAPs using move-cost adjusted thread assignment. In: GECCO 2011, ACM (2011) 1547–1554.

[2] Taillard, É.: Comparison of iterative searches for the quadratic assignment problem. Location Science 3(2) 1995, 87–105.

[3] Cantú-Paz, E.: Efficient and Accurate Parallel Genetic Algorithms. Kuwer Academic, Publishers 2000.