Optimal Solving of Permutation-based Optimization Problems on Heterogeneous CPU/GPU Clusters

Jan Gmys
Mathematics and Operations Research Department
University of Mons
Mons, Belgium
jan.gmys@umons.ac.be

EXTENDED ABSTRACT

We revisit the parallelization of Branch-and-Bound (B&B) algorithms on massively parallel and heterogeneous architectures integrating multi-core processors and GPU accelerators. We provide a comprehensive description of the incremental development of a B&B-algorithm for the exact resolution of large permutation-based combinatorial optimization problems (COP). In order to highlight the challenges related to the different levels of the algorithm, we proceed incrementally by increasing the complexity of the targeted hardware step-by-step (as shown in Fig. 1) from sequential to multi-core, to GPUs and multi-GPU systems and finally to heterogeneous clusters combining multi-core CPUs and accelerator devices. On a cluster composed of 9 GPU-accelerated compute nodes (130 000 CUDA cores) the described methodology reduces the resolution time for a hard instance of the flow shop scheduling problem (FSP) from 600 hours (using 300 CPUs) to 9 hours[1]. Three permutation problems with different computational characteristics are used for experimental evaluation: FSP, quadratic assignment (QAP) and the N-Queens problem.

A. *Parallel B&B for permutation problems*

Many COPs that arise in industrial and economic applications can be modeled by using permutations to represent candidate solutions (e.g. scheduling, assignment or routing problems). As most of these problems are NP-hard, approximate optimization methods like (meta)heuristics are often used in practice. While these methods find high-quality solutions within a reasonable amount of time, they fail in general to find optimal solutions and to provide error quantification. Conversely, exact methods find the optimal solution(s) with a proof of optimality, but require huge computational effort.

B&B is the most frequently used exact method to solve COPs. The algorithm recursively decomposes the initial problem by dynamically constructing and exploring a search-tree. This is done using four operators: branching, bounding, selection and pruning. The branching operator divides the initial problem into smaller subproblems and a bounding function computes lower bounds on the minimal cost of subproblems. Using these lower bounds, the pruning operator eliminates subproblems from the search that cannot lead to an improvement of the best solution found so far. The tree-traversal is guided by the selection operator, which returns the next subproblem to be processed according to a predefined strategy (e.g. depth-first search).

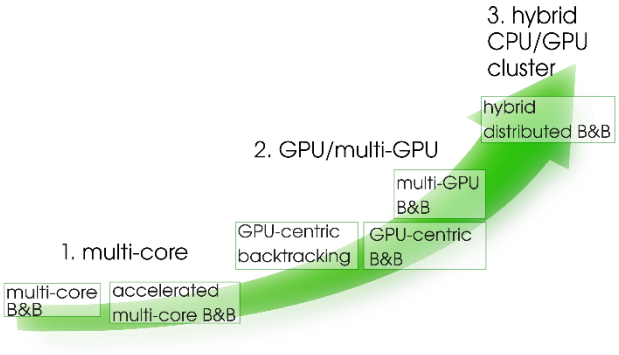

Figure 1. Outline

B&B generates huge search trees, often containing billions of nodes. For example, Taillard's FSP instance *Ta056* [1], which consists in scheduling 50 jobs on 20 machines, requires 22 years (aggregate time) to be solved on a single CPU processing core [2]. Therefore, ultra-scale computing is required to solve similar instances in a reasonable amount of time. For that, it is essential to define an efficient data structure for the storage and management of the "tsunami" of subproblems, dynamically generated at runtime. In [3] an innovative data structure called IVM (for Integer-Vector-Matrix) was proposed for the storage of search trees of subproblems (partial permutations). Extensive experimental evaluation has shown that IVM is effective in terms of memory footprint and pool management time. The IVM data structure is closely related to the encoding of permutations using Lehmer codes and an interval-based decomposition of the search space. In this approach, the search tree is encoded as a

1 22 years (aggregate time) using a single CPU core.

factoradic interval that is decomposed into sub-intervals, representing distinct portions of the search space that can be explored in parallel. However, as the amount of work (subproblems to decompose) in different parts of the search space is highly variable and unpredictable, a static decomposition of the search space is inefficient. Therefore, the work stealing paradigm is used to dynamically balance the work load between B&B processes.

B. Parallel B&B for multi-core processors

According to this interval-based decomposition/work stealing approach each thread in the multi-core B&B-algorithm uses its private IVM structure to locally explore a different portion of the search space (encoded as an interval). When a thread finishes the exploration of its interval it attempts to steal a portion of another threads interval. In [4] we investigate several work stealing strategies using different victim selection and granularity policies. Experimental results show that the approach achieves near-linear speed-ups on multi-core systems with up to 28 cores and on Xeon Phi MIC architectures with 60 cores, even for fine-grained problems with computationally inexpensive node evaluation functions (e.g. N-Queens). When the bounding operation is moderately time-consuming or when the latter is coprocessor-accelerated, the efficient management of subproblems becomes indeed a critical component of B&B. In these cases, experimental results show that IVM-based B&B algorithms can significantly outperform their counterparts based on conventional data structures (e.g. stacks, deques).

C. GPU-centric Parallel B&B

Existing GPU-accelerated B&B algorithms in the literature can be divided into two mainly two approaches. The first consists in offloading active nodes for parallel evaluation to the device while maintaining one or several work pools on the CPU. This offloading approach requires careful design and tuning of data transfers between host and accelerator [5]-[7]. Moreover, especially if node evaluation costs are low, it may not be possible to completely overlap communication and computations. The second approach consists in exploring the B&B tree up to a certain depth on the host processor and using the open nodes at that depth as roots of subtrees explored in parallel on GPU [8]-[9]. However, these approaches generally suffer from load imbalance, as they use a static decomposition of the search space without load balancing after the parallel GPU-based searches are launched.

The IVM data structure, thanks to its small and constant memory footprint allows to implement the entire B&B algorithm on the GPU, including GPU-based dynamic load balancing mechanisms. As shown in [10,11] such a GPU-centric approach solves the issue of host-device data movements and load imbalance inside the GPU. Two variants of GPU-centric B&B are proposed, using different parallelization models for problems with different granularities. Both variants are extended to multi-GPU systems. We present and discuss design choices and mapping approaches, as well as different GPU-based work stealing strategies. For a class of ten 20-jobs-on-20-machines FSP instances with sequential (CPU) execution times between 15 minutes and 22 hours, the resolution times using four gaming GPUs (GTX 980) range from 1 second to 1 minute, i.e. the GPU-centric B&B achieves speedups of up to three orders of magnitude compared to a single CPU core execution, using hundreds of thousands of parallel B&B searches efficiently.

D. Heterogeneous CPU/GPU cluster

As failure probability increases with each new generation of supercomputers, fault tolerance becomes a critical issue to the scalability and correctness of large-scale parallel programs. Using B&B@Grid [2] (a fault-tolerant master-worker approach designed for computational grids) as a starting point, we integrate the previously presented multi-core and GPU-centric algorithms into a distributed B&B for CPU-GPU clusters. For the sake of locality of reference, the approach is hierarchical, as multi-core processors and GPUs are seen as single workers on the global level. Work units on the inter-node level are redefined and load balancing, checkpointing and communication mechanisms are revisited consequently.
Fig. 2 reports the experimental results obtained using Grid'5000 in 2006 [2], a GPU-accelerated compute node in 2015 and two hybrid GPU-based clusters in 2017. On a cluster of 9 GPU-accelerated compute nodes (130 000 CUDA cores) our approach reduces the resolution time for the FSP instance *Ta056* mentioned above from 600 hours (using 300 CPUs) to 9 hours.

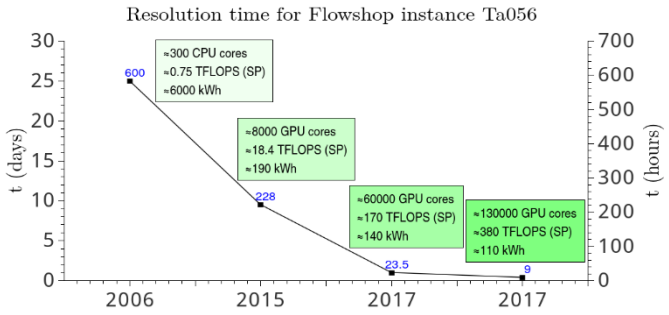

Figure 2. Resolutions of FSP instance *Ta056* [1] on different systems. From left to right: (2006) B&B@Grid [2] – (2015) multi-GPU-B&B (4 GTX980) – (2017/1) 8 nodes (2 Broadwell Xeon/2 GTX1080Ti) of Grid'5000/Lille – (2017/2) 9 OpenPower nodes (2 Power8+/4 Pascal P100) of *Ouessant* (IDRIS)

E. Future work

For the implementation of the presented algorithms, we used C++ in combination with the Pthreads library, CUDA and socket programming for the inter-node communication layer. In order to improve the scalability (and maintainability) we plan to investigate the use of programming interfaces using the PGAS programming model. We also plan to hybridize our B&B approach with metaheuristics in order to find high-quality suboptimal solutions of hard instances of permutation problems.

Keywords– Combinatorial optimization; heterogeneous computing; GPU computing; load balancing

ACKNOWLEDGMENT

Experiments presented in this paper were carried out using the OpenPower prototype *Ouessant* hosted by the **I**nstitute for **D**evelopment and **R**esources in **I**ntensive

Scientific Computing (see http://www.idris.fr/) and the Grid'5000 testbed, supported by a scientific interest group hosted by Inria and including CNRS, RENATER and several Universities as well as other organizations (see https://www.grid5000.fr).

BIOGRAPHY

JAN GMYS has received the Master's degree (2014) in Advanced Scientific Computing from the University of Lille and Ph.D's in Engineering Sciences/Computer Science (2017) from the Universities of Mons/Lille. He is currently a research assistant at the University of Mons. His research interests include parallel/distributed/GPU computing and combinatorial optimization. He is the co-author of 5 journal papers and 3 papers in international workshops/conferences.

REFERENCES

[1] E. Taillard, "Benchmarks for basic scheduling problems", *Journal of Operational Research*, vol. 64, pp. 278–285, 1993.

[2] M. Mezmaz, N. Melab and E. G. Talbi, "A grid-enabled branch and bound algorithm for solving challenging combinatorial optimization problems", in *2007 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pp. 1–9, Mar. 2007.

[3] M. Mezmaz, R. Leroy, N. Melab, and D. Tuyttens, "A multi-core parallel branch-and-bound algorithm using factorial number system", in *28th IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pp. 1203–1212, May 2014.

[4] J. Gmys, R. Leroy, M. Mezmaz, Melab N. and D. Tuyttens, "Work stealing with private integer-vector-matrix data structure for multi-core branch-and-bound algorithms, *Concurrency & Computation : Practice & Experience*, 28, 18, 4463-4484, 2016.

[5] I. Chakroun, N. Melab, M. Mezmaz and D. Tuyttens, "Combining multi-core and gpu computing for solving combinatorial optimization problems*", Journal of Parallel and Distributed Computing*, vol. 73, no. 12, pp. 1563–1577, 2013.

[6] T. Vu and B. Derbel, "Parallel Branch-and-Bound in multi-core multi-GPU heterogeneous environments", *Future Generation Comp. Syst.*, 56:95-109, 2016.

[7] J. Shen, K. Shigeoka, F. Ino and K. Hagihara, "An out-of-core branch and bound method for solving the 0-1 knapsack problem on a GPU, in *Algorithms and Architectures for Parallel Processing (ICA3PP 2017)*, Lecture Notes in Computer Science, vol 10393. Springer, Cham, 2017.

[8] T. Carneiro, A. Muritiba, M. Negreiros and G. Lima de Campos, "A new parallel schema for branch-and-bound algorithms using GPGPU", in *23rd International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD)*, pp. 41–47, 2011.

[9] A. Borisenko, M. Haidl and S. Gorlatch, "A GPU parallelization of branch-and-bound for multiproduct batch plants optimization", *J. Supercomput.* 73, 2, pp. 639-651, 2017.

[10] J. Gmys, M. Mezmaz, N. Melab and D. Tuyttens, "IVM-based work stealing for parallel branch-and-bound on GPU", in *Parallel Processing and Applied Mathematics (PPAM 2015)*, Lecture Notes in Computer Science, vol 9573. Springer, Cham, 2015.

[11] T. Carneiro, J. Gmys, N. Melab, F.H. de Carvalho Junior and D. Tuyttens, "A GPU-based backtracking algorithm for permutation combinatorial problems", in *Algorithms and Architectures for Parallel Processing* (ICA3PP 2016), Lecture Notes in Computer Science, vol 10048. Springer, Cham, 2016.