

Accelerated Local Search Procedures on Graph Cut Optimization Problems using CUDA

Antonio S. Montemayor*, Raúl Cabido, Abraham Duarte, Juan José Pantrigo
Universidad Rey Juan Carlos (Madrid, SPAIN)

1 Introduction

Graph-Cut optimization problems consist in finding a bipartition of a graph into two subsets, in such a way that a given objective function is optimized (maximized or minimized). Some Graph-Cut examples are Max-Cut, Mix-Cut, N-Cut or Max-Min-Cut, among others. Most of them, are NP-Hard optimization problems. Therefore, it is convenient to devise algorithms for finding an approximate solution to this problem in a reasonable time. Some of the most successful methods for solving this kind of problems are called metaheuristics [Blum and Roli 2003]. These procedures are high level strategies for exploring search spaces, where the dynamic balance between diversification and intensification is the key point. The term diversification generally refers to the exploration of the search space, whereas the term intensification refers to the exploitation of the accumulated search experience [Blum and Roli 2003]. Intensification strategies usually are implemented as local search methods, where a given neighbourhood of a solution is systematically examined, looking for the best solution.

In general, local search procedures are very computational demanding tasks, independent and quite parallelizable. Recently, [Harish and Narayanan 2007; Dixit et al. 2005] exploited the power of a GPU to accelerate graph algorithms although they represent the graph as a usual adjacency list or matrix in a node based approach. Instead of that we propose the cocycle matrix as an edge based representation, which could be efficiently exploited in certain cases.

2 Our edge-based representation proposal

In this paper we will drive the Max-Cut problem that consists of finding a bipartition where the sum of the edge weights having endpoints in different subsets is maximized. For the analysis, we have considered two different representations of a cut: node-based and edge-based representations. In the first one, each component of the solution represents a graph node, and its value (0 or 1) decides whether the node belongs to the considered cut solution or not. In the edge-based representation, each component of the solution represents a graph edge, and its value describes the weight of the edge that is cut by the considered solution or graph cut.

A node-based representation is mainly used to solve the Max-Cut problem in a traditional CPU implementation. Unfortunately, a node-based model is usually inefficient for the GPU because the evaluation of each solution involves direct accesses to the graph adjacency matrix. Also, using this representation, only one solution

is evaluated at a time before modifying it in case of improvement. After the movement, all the process is repeated until convergence.

Given a vertex v its cocycle is defined as the set of edges with origin in v and endpoints at any other vertex of the graph. Then, the cocycle matrix stores the cocycles defined by each vertex. This matrix can be used to describe the neighborhood structure explored by a Local Search procedure at each time step of the algorithm by performing an element-by-element integer-XOR operation between the binary cut solution and each row of the corresponding cocycle matrix. The edge-based representation is more adequate to the GPU computation model because we avoid direct and heterogeneous accesses to the graph structure. Only after the evaluation of the given neighborhood the whole process is repeated, starting from the best solution found previously until no new better solution is found.

3 Preliminary results

Our implementation is running on a 3.6 GHz Pentium 4 Dual Core, 1 GB RAM with Nvidia 8800GTX graphics chip with 768MB on-board, drivers v175.19, and using CUDA 1.1 under Microsoft Windows XP SP2. The algorithm is memory bound and there is no need of using CUDA shared memory, but even so GPU results seem promising for a deeper study in comparison to a CPU traditional implementation. For this edge-based approach, the number of evaluations per second (eps) for a graph with 800 nodes and 1600 edges is about 91000 in CPU and 690000 in GPU ($\sim x7.5$), for the same number of nodes and 8000 edges the GPU is $\sim x12$ times faster. For the first configuration, a CPU node based implementation would offer ~ 40000 eps as it is a sparse case in which the cocycle matrix is not so large in comparison to its corresponding adjacency matrix.

4 Conclusions

We present a promising work-in-progress GPU local search adaptation for the graph Max-Cut problem using the Nvidia CUDA architecture. Preliminary results show a good performance of the GPU approach for medium sized and sparse graphs. Even in CPU, the exploitation of the cocycle matrix instead of adjacency structures could be interesting compared to the node based implementation for sparse configurations. As future works we include a deeper study on different graph cases and some easy optimization of the GPU implementation that will lead to an improvement of about $\sim 15\%$.

References

- BLUM, C., AND ROLI, A. 2003. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Comput. Surv.* 35, 3, 268–308.
- DIXIT, N., KERIVEN, R., AND PARAGIOS, N. 2005. GPU-cuts: Combinatorial optimization, graphic processing units and adaptive object extraction. In *Technical Report, CERTIS 05-07*.
- HARISH, P., AND NARAYANAN, P. 2007. Accelerating large graph algorithms on the GPU using cuda. In *Proc. of High Performance Computing (HiPC)*.

*e-mail: antonio.sanz@urjc.es

⁰This research was partially supported by the Spanish Ministry of Education and Science CICYT TIN2005-08943-C02-02, URJC and CAM by URJC-CM-2007-CET-1724, and the Nvidia Professor Partnership Program.