Parallel Computing 57 (2016) 52–72

Contents lists available at ScienceDirect

Parallel Computing

journal homepage: www.elsevier.com/locate/parco

GPU-accelerated Hungarian algorithms for the Linear Assignment Problem

Ketan Date, Rakesh Nagi*

Department of Industrial and Enterprise Systems Engineering, 117 Transportation Building, University of Illinois at Urbana-Champaign, Urbana, IL 61801, USA

ARTICLE INFO

Article history:
Received 1 December 2014
Revised 8 April 2016
Accepted 19 May 2016
Available online 20 May 2016

Keywords:
Linear assignment problem
Parallel algorithm
Graphics processing unit
CUDA

ABSTRACT

In this paper, we describe parallel versions of two different variants (classical and alternating tree) of the Hungarian algorithm for solving the Linear Assignment Problem (LAP). We have chosen Compute Unified Device Architecture (CUDA) enabled NVIDIA Graphics Processing Units (GPU) as the parallel programming architecture because of its ability to perform intense computations on arrays and matrices. The main contribution of this paper is an efficient parallelization of the augmenting path search phase of the Hungarian algorithm. Computational experiments on problems with up to 25 million variables reveal that the GPU-accelerated versions are extremely efficient in solving large problems, as compared to their CPU counterparts. Tremendous parallel speedups are achieved for problems with up to 400 million variables, which are solved within 13 seconds on average. We also tested multi-GPU versions of the two variants on up to 16 GPUs, which show decent scaling behavior for problems with up to 1.6 billion variables and dense cost matrix structure.

1. Introduction

The objective of the linear assignment problem (LAP) is to assign n resources to n tasks such that the total cost of the assignment is minimized. The mathematical formulation for the LAP can be written as follows:

$$\min \quad \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij}; \tag{1}$$

$$\text{s.t.} \sum_{j=1}^{n} x_{ij} = 1 \quad \forall i = 1,\ldots,n; \tag{2}$$

$$\sum_{i=1}^{n} x_{ij} = 1 \quad \forall j = 1,\ldots,n; \tag{3}$$

$$x_{ij} \in \{0,1\} \quad \forall i,j = 1,\ldots,n. \tag{4}$$

* Corresponding author. Tel.: +1-217-244-3848; fax: +1-217-244-5705.
E-mail addresses: date2@illinois.edu (K. Date), nagi@illinois.edu (R. Nagi).

http://dx.doi.org/10.1016/j.parco.2016.05.012
0167-8191/

The decision variable $x_{ij} = 1$, if resource i is assigned to task j and 0 otherwise. Constraints (2) and (3) enforce that each resource should be assigned to exactly one task and each task should be assigned to exactly one resource. c_{ij} is the cost of assigning resource i to task j, and $\mathbf{C}_{n\times n} = \left[c_{ij}\right]$ is the cost matrix of the LAP.

LAP is one of the most well-studied optimization problems that can be solved in polynomial time. Until now, many efficient sequential algorithms have been proposed in the literature. These algorithms can be classified into three main classes (Burkard and Çela [8], Jonker and Volgenant [13]): (1) *Linear programming based algorithms*, which involve variants of the primal and dual simplex algorithms; (2) *Primal-dual algorithms* such as the famous Hungarian algorithm (Kuhn [16]) and the Auction algorithm (Bertsekas [4]); and (3) *Dual algorithms* such as the successive shortest path algorithm (Jonker and Volgenant [13]). Due to their polynomial worst-case complexity, the primal-dual and shortest path algorithms generally outperform the simplex-based algorithms. Several variations of the Hungarian and the shortest path algorithms have been proposed in the literature, for improving their execution time (Jonker and Volgenant [12], [14], Volgenant [30]). The theoretical complexity of the most efficient implementation of the primal-dual or shortest path algorithms is $O(n^3)$, where n is the number of resources or tasks.

Owing to their cubic worst-case complexity, sequential algorithms can prove to be a significant bottleneck for large instances of the LAP. This calls for the development of a parallel algorithm, which can take advantage of a specific architecture and divide the work among multiple processors, to alleviate the computational burden. Until now many parallel versions of the aforementioned sequential algorithms have been proposed which include parallel asynchronous version of the Hungarian algorithm (Bertsekas and Castañon [6]); parallel version of the shortest path algorithm (Balas et al. [2], Storøy and Sørevik [28]); and parallel synchronous and asynchronous versions of the Auction algorithm (Bertsekas and Castañon [5], Buš and Tvrdík [9], Naiem et al. [22], Sathe et al. [26], Wein and Zenios [31]). An empirical analysis of the sequential and parallel versions of the Auction and shortest path algorithms was performed by Kennington and Wang [15]. All the above parallel algorithms were designed for prevalent parallel computing architectures and they were shown to achieve significant speedups.

In recent years, there have been significant advancements in the graphics processing hardware. Since graphics processing tasks generally require high data parallelism, the GPUs are built as compute-intensive, massively parallel machines, which provide a cost-effective solution for high performance computing applications. Vasconcelos and Rosenhahn [29] developed a parallel version of the synchronous Auction algorithm for a single GPU. The authors tested the algorithm on problem instances with up to 16 million variables, which gets automatic scalability through CUDA with increasing number of GPU cores. Roverso et al. [25] developed a GPU implementation of the *deep greedy switching* (DGS) heuristic of Naiem and El-Beltagy [21], for solving the LAP under real-time constraints. It was shown that the heuristic sacrifices optimality in favor of significant speedup, on problem instances with up to 100 million variables.

In this paper, we are proposing parallel versions of two variants of the Hungarian algorithm, specifically designed for the CUDA enabled NVIDIA GPUs. We have chosen to parallelize the Hungarian algorithm, mainly because it operates on the cost matrix of the LAP and the GPUs are well suited for performing intense computations on arrays and matrices. Our main contribution is an efficient algorithm for the augmenting path search phase, which happens to be the most time intensive phase in the Hungarian algorithm. The prominent feature of our algorithm is that it takes advantage of the race condition to generate multiple vertex-disjoint augmenting paths, which can be used simultaneously to improve the current solution. We show that this parallelization leads to a dramatic reduction in the execution time, for both small and large sized problem instances. LAPs serve as sub-problems to many NP-hard optimization problems such as the Traveling Salesman Problem (TSP), the Quadratic Assignment Problem (QAP), and the Generalized Assignment Problem (GAP). Finding good solutions to these problems generally requires solving multiple LAPs in an iterative fashion. Therefore, having a fast, scalable, and cost effective LAP solver is extremely important. We believe that our GPU-accelerated algorithms stand true on all the three requirements.

The rest of the paper is organized as follows. In Section 2, we briefly describe the two variants of the sequential Hungarian algorithm. In Section 3, we describe some preliminaries, which will be useful in the development of the parallel algorithms. In Sections 4 and 5, we describe the various stages of our parallel algorithms, and their implementation on single and multi-GPU architectures. In Section 6, we present the experimental results on randomly generated problem instances. Finally in Section 7, we conclude the paper with a summary.

2. Sequential Hungarian algorithm

The Hungarian method developed by Kuhn [16] was the first systematic approach for finding the optimal solution to an LAP. Although, the algorithm is primarily based upon the works of Hungarian mathematicians König and Egerváry, the main idea behind the algorithm can be better explained with the help of linear programming duality (Bazaraa et al. [3], Nering and Tucker [23]). The dual of the assignment problem (1)–(4) can be written as follows:

$$\max \quad \sum_{i=1}^{n} u_i + \sum_{j=1}^{n} v_j; \tag{5}$$

$$\text{s.t.} \quad u_i + v_j \le c_{ij} \quad \forall i, j = 1, \ldots, n; \tag{6}$$

$$u_i, v_j \sim \text{unrestricted} \quad \forall i, j = 1, \ldots, n; \tag{7}$$